FALL 2008
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, September 15, 2008

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.

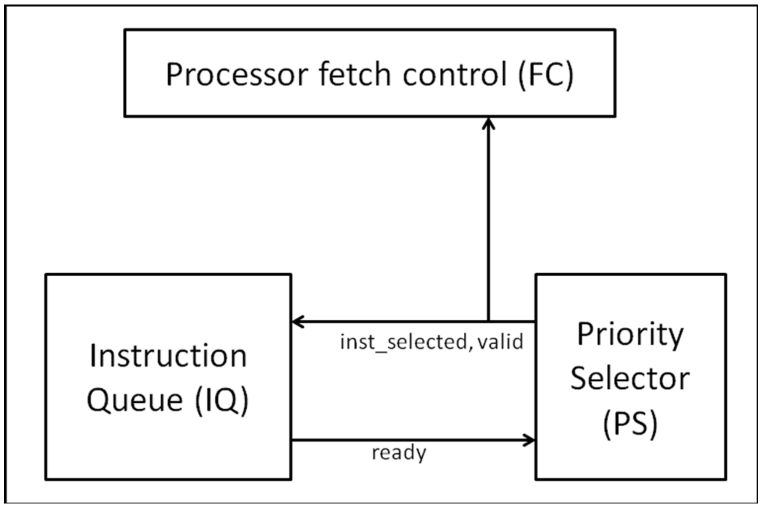
POLICY ON MISPRINTS AND AMBIGUITIES:

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. Designing priority selectors

An important component of the instruction issue logic is a priority selector that selects which instruction to issue from the set of available and ready instructions. This priority selector is connected to an instruction queue which includes status information like *ready, valid, issued,* etc.

For this question, you must design just the priority selector for a 16-entry instruction window. The main input to this module is one 16 bit-vector called "ready" that indicates which instructions are ready to be issued. The output of this module is an instruction number "inst_selected" pointing to the selected instruction which can be issued. A "valid" signal should indicate if any such instruction is available. See detailed interface below. The figure shows the context of this priority selector in the processor's hierarchy.



Inputs: ready[15:0], reset, clk
Outputs: inst_selected[3:0], valid

a) Implement the logic design for the priority selector (PS). This is a "static" priority selector – i.e the earliest instruction is always instruction 0 in the ready vector. You may use basic logic gates (AND, OR, NOT, NAND, XOR) and flip-flops. Use hierarchy where you think it is necessary. You must submit a drawn schematic.

b) In a "dynamic priority selector", the earliest instruction can change because the instruction queue is maintained as circular buffer. In a processor, the instruction queue is maintained as a circular buffer as instructions are fetched and retired and hence you need this dynamic priority. Describe how you would use your module as a building block for such a "dynamic priority selector" (DPS). Also discuss the interface signals to the rest of the processor for this "dynamic priority selector" if you feel they need to be different from the static priority selector.

c) Discuss why this design may not scale to very large instruction windows like 1024 instructions.

2. ILP and Power

For the past decade or so, most high-performance processors have aggressively used speculation and deep out-of-order pipelines to exploit instruction-level parallelism (e.g., the Intel Pentium IV). More recently, power concerns have lead to multicores with shorter out-of-order pipelines (e.g., Intel Core 2) and even simple in-order pipelines (e.g., Sun Niagara) that resemble the classic 5-stage MIPS pipeline. For a given chip area, because of the design complexity that makes an out-of-order core larger than an in-order core, you will be able to fit fewer out-of-order cores than in-order cores.

a) Argue why future chips will have hundreds of simple in-order cores.

b) Argue why future chips will have many fewer out-of-order cores.

c) If you were the lead architect on a multicore intended to ship in 2013, describe what kind of cores you wo uld want to use and why.

3. Vector processing

a) Explain the motivation for vector processors as classically used in super-computers (e.g., the CRAY-1).

b) Recently the vector processing technique has been adopted in the form of short-vector ISA extensions (e.g., four 16-bit elements in a 64-bit word) in general purpose microprocessors. Explain the rationale for this recent trend and what are the architectural and microarchitectural enhancements required to implement such short-vector extensions.

c) Graphics processing units (GPU) implemented as stream processors are also extensions of the same idea. Discuss the differences between the architectural implementations of these processors compared to short-vector extensions in general purpose processors.

4. Multicore Coherence

Most future processor chips will contain multiple cores whose references to shared memory are accelerated by caches. These caches will usually be kept transparent to software via a coherence protocol. Many protocols are based on snooping or directories. Nevertheless, multicore chips will differ regarding interconect, small or large number of cores, flat vs. hierarchical, etc.

a) Discuss what factors that favor snooping coherence and why.

b) Discuss what factors that favor directory coherence and why.

c) What coherence protocols do you expect 2013 multicore chips to use and why?

5. Prefetching in Multicore Caches

Most single-core processors support some form of cache prefetching, either using ISA extensions to allow software to direct which data to prefetch or by using hardware structures to detect and predict memory access patterns. Either prefetching approach can be designed to prefetch to different levels of the memory hierarch (e.g., L1 or L2) or to a dedicated prefetch buffer. With the move to multicore processors, caches and off-chip bandwidth become shared resources, which introduce additional issues that architects must consider.

a) Discuss the trade-offs between different destinations for prefetched data: a shared-L2, a private-L1, and a private prefetch buffer.

b) Discuss the advantages and disadvantages between using hardware-directed and software-directed prefetching when shared off-chip bandwidth is a limited resource.

6. Non-Volatile Memory

Over the years, there have been many non-volatile memory technologies that provided random memory access, promised cost per bit that is between DRAM and disk, and required no power to retain previously stored bits.

Consider a new technology NVM, inspired by Flash Memory, that achieves the above, but also requires block writes (e.g., of 8KB) and limits the number of reliable writes to a given block (say, 1 million times).

a) What are the opportunities and challenges of using NVM to replace/ augment disks?

b) How might the presence of NVM at the disk level affect other system components (e.g., DRAM size)?

c) What are the opportunities and challenges of using NVM to replace/ augment DRAM?