

# Computer Science 252: Graduate Computer Architecture

University of California

Dept. of Electrical Engineering and Computer Sciences

David E. Culler  
TA: Steve Sorkin

Take Home due 5/20

Spring 2003

Name:

SID:

If you have questions, don't hesitate to address them to the instructor or TA. Please drop off your completed exam by 2:00 pm May 20, 2003 to Willa Walker in 626 Soda Hall. You are welcome to use any reading materials, but it should be your own work. Please sign below. Good luck.

The following work is my own. I have not received assistance in answering these questions from members of the class or other human sources.

X \_\_\_\_\_

## **Problem 1: Impact of Bus Design for Multiprocessors**

Suppose you are designing around a 32-bit microprocessor and that operates at 1 GHz. It has split 1<sup>st</sup> level instruction and data caches backed by a unified 2<sup>nd</sup> level cache. We will assume that the L2 cache has 128 byte blocks, write-back, write-allocate. It connects to memory over a cache-coherent system bus that operates at 128 MHz and is 128 bits wide. We will use a somewhat idealized bus, where each bus transaction involves an arbitration/address (A) phase followed by a data (D) phase. The A-phase consists of a single bus cycle for arbitration, a bus cycle for address, and two bus cycles waiting for caches to determine the snoop outcome and for memory to prepare to respond. It is the same pattern for read and write transactions; the write uses the D-phase.

1.a: What is the maximum possible bandwidth of the bus?

1.b. First we look at the performance of a single processor in this design. Consider a workload consisting of 16% loads and 8% stores and the remainder being other instructions. The global instruction miss rate is 0.5% and the global data miss rate (assumed the same for loads and stores) is 2%. With the caches in place, but an ideal memory system that responds immediately, the processor achieves an IPC of 2.0. With misses served by memory over this bus, what is the IPC? *Show your calculation in detail and explain your approach.*

.

1.c. If the bus is modified to have separate address and data lines, such that the A-phase can be performed in parallel with the previous D-phase (i.e., a pipelined or split-phase bus), what is the maximum bandwidth?

1.d. What is the speedup on the workload of part b?

1.e. Suppose the processor was enhanced to achieve an IPC of 4.0 with the ideal memory system. What would be the IPC on the workload of part b and d? What is the speedup relative to part d?

1.f. Now we want to add processors. In practice, you never want to design for 100% utilization of components of the machine; you need to leave head-room. How many processors of part d can we put on the bus and utilize about 80% of the bus bandwidth?

1.g. If the bus is 80% utilized, what is the expected number of processor cycles to service a cache miss? How does this compare to the single processor case?

1.h. We originally intended to give you a problem where you had an address trace for each of multiple processors and you would have to simulate the behavior of a cache coherence protocol. However, there is a serious and subtle problem with that. It is one that tripped up early SMP design studies. You may notice that most simulation studies on SMPs run the actual program on simulated processors which are connected to a memory simulator. This is quite different from uniprocessors, where we collect an address trace and feed it through cache organizations without regard to the processor. Why does the trace-driven methodology of analysis break down for multiprocessors? Explain and give an example.

## **Problem 2: Networks, Network Processors, and Multithreading.**

This problem will give you a chance to put together various aspects of networks and network processors from multiple reading sources. The IXP1200 can support 2 x 1 Gbps and 8 x 100 Mbps ports. It operates in chunks called MAC-Packets (MPs).

2.a. What portion of a IP packet (including what fields) is contained in the first MP seen by the router? How are the fields modified on the first MP before the second one arrives?

2.b. How many cycles do the microengines have to process an MP on a 100 Mbps link? Gbps link? At the very minimum, the input stage has to write the MP to DRAM and the output stage has to read the MP from DRAM. How many cycles does this leave for the rest of the processing on 100 Mbps links? 1 Gbps links?

2.c. With the software architecture described in Spalink's papers, what is the minimum delay introduced by a packet passing through a router? What is the maximum delay?

2.d If a 1024 byte packet were to go from a host, through 3 routers that are close to one another, to another host with no contention on 100 Mbps links, what would be the transport latency? If these routers were 10 kilometers apart, what would be the transport latency?

2.e If two hosts separated by three routers, as above, each send a packet to the other and whenever they receive a packet, immediately turn it around and send it back, how low would the sender and receiver overheads need to be to utilize 50% of the 100 Mbps link bandwidth in each direction? What about for 1 Gbps links?

2.f. To understand the value of multithreading, during the time to process an MP at 100 Mbps in the input stage, what fraction time is the context idle, assuming that it waits for each memory operation to complete? What about the output stage? How many contexts can a microengine usefully support.

### 3. Precise interrupts and dynamic instruction scheduling

3.a. After class when we discussed the Smith and Pleszkun paper, one student raised an excellent question about what happens in the history buffer scheme when a sequence of writes are performed on the same register. This touches on a subtle issue that is not clearly addressed in the paper, nor did I address it in lecture. Concretely, suppose a sequence of three instructions issue that all have the same destination register and they are all in progress, sitting in some reservation station, before the first completes. Assume the first completes correctly, but the second experiences an exception, causing a roll back for the exception.

The paper says, *“When an instruction issues, a buffer entry is loaded with control information, but the current value of the destination register (to be overwritten by the instruction) is also read from the register file and written to the buffer entry. Results on the result bus are written directly to the register file when an instruction completes....When an exception condition arrives at the head of the buffer, the buffer is held, instruction issue is immediately halted, and there is a wait until pipeline activity completes. The active buffer entries are then emptied from tail to head, and the history values are loaded back into their original registers.”*

I think there is a significant bug in this mechanism. But perhaps I am wrong. If you believe the approach as described works, explain why in the context of dynamic instruction scheduling that allows multiple outstanding operations on the same architected registers. If not, describe what needs to be done to fix the mechanism, without stalling on WAW hazards.

3.b. Give a detailed description and pseudo code for a register renaming mechanism at instruction issue. It should map 32 architected registers to 128 internal registers. How does the presence of this mechanism interact with reorder buffer or history buffer design?

**Extra Credit:** Milo Martin talked about protocol consistency. The paper just touches lightly on sequential consistency. Describe in a few paragraphs how you could adapt the token coherence scheme to provide sequential consistency. What constraints are placed on the processors? What modifications are made to the protocol? Reason through why your approach provides sequential consistency.