CMSC 411 Computer Architecture

Final Exam
Spring, 1997

This exam comprises 4 problems, which may be done in any order as long as you start each problem on a new page. The point value of each problem is indicated with the problem, and sub-parts are equally weighted, unless otherwise indicated.

Remember, it is supposed to be HARD. Don't panic when you get stuck. Try something else and come back to it. If logic fails, try to think like a "dumb user". After that, thinking like a computer scientist will be a relief.

Ladies and Gentlemen.....

...................Let's get ready to R-U-MMMMMMM-B-LE.

You may turn the page and begin.

Problem 1: A Short Trip in Search of Truth (30 pts)

The following section requires short paragraph answers. Complete sentences are not necessary. However, make *sure* you address the questions I'm asking. Facts are nice, but your analysis of them is what I"m going for here. Counter-examples are acceptable as well. Oh, and each one is worth (5) points.

1.1 Why is the average execution time often preferred over MIPS and MFLOPS as metrics for evaluating the performance and throughput for general purpose computing platforms? (I want two or three well explained reasons.)

1.2 Why is it difficult to construct a benchmark that will aid you in purchasing a new computer for your company? Address briefly how you would construct an appropriate benchmark.

1.3 Why do you think that on the average, 90% of the program's execution time is spent running 10% of the code? What might cause this "rule of thumb" to be violated?

1.4 Explain the role of the compiler in reducing the potential for hazards in a pipelined architecture. That is, what problems does it cause, what problems can it solve, and what problems can't it solve.

1.5 Consider the CPU performance equation used throughout the book and address the following issues: 1) What quantity (or quantities) in the equation are the easiest to compute and why? 2) What quantity (or quantities) are difficult to compute, and why?

1.6 Explain the concepts of functional unit latency and the function repeat interval as they apply to pipelining in general. Why does pipelining functional units, such as the floating point adder and multiplier, tend to decrease the CPI, the functional unit latency, and the initiation interval.

Problem 2: Some Ugly Truths and Falsehoods (22)

In the first section, write the letter of any true statement on your answer sheet, with one point given for every correct answer. (No need to explain why it is false.)

A) Caches were designed specifically to supply pipelined architectures with rapid access to data and instructions, thus keeping the pipeline full.

B) Caches only incur miss penalties when the cache is full and some block must be replaced to makes space for the desired block of information.

C) A data cache should always be organized as a write back cache to minimize the penalty incurred when replacing a used cache block into main memory.

D) Virtual memory can only be used if the address word can be expanded to address the extended memory space–that is, the room for address expansion is designed into the original address length.

E) Main memory organization has no influence on the cache organization, the width of the data bus, or the handling of virtual memory.

F) The Alpha AXP cache hierarchy has separate caches for instructions and data, and prefetches instructions to minimize the time needed to start a new instruction.

G) The interaction between the memory and the operating system need not be considered in a good system design.

In this section, write the number of any true statement, with true next to it. If you believe the statement is false, please explain why. Each of these is worth 3 points.

2.1 Cache locations are marked with a dirty bit to indicate that the information has been used and should be returned to main memory when more room is needed in the cache.

2.2 The DLX pipeline with stages for floating-point computations uses a dynamic data structure to ensure that instructions do not execute out of order.

2.3 In VLIW systems, the programmer is required to write his code in chunks that can be processed independently but simultaneously by the CPU.

2.4 The principal difference between RISC and CISC architectures is the structure of their instruction words.

2.5 Regardless of the code being executed, multiple issue machines are faster than single-issue machines because more instructions can be executed during a given time period.

Problem 3: Exploiting In Line Parallelism. (25 pts)

This problem focuses on the execution of code in a pipelined architecture, and the changes needed to better exploit the pipeline.

3.1 Consider the following simple loop to be executed on the DLX pipeline shown in Figure 3.44 on page 190, assuming full forwarding and bypassing, and separate instruction and data memories.

For simplicity, assume that the array being operated on has its lowest address element at location zero, and that R1 starts with the address of the element with the highest address. A scalar is stored in location F2, and each element of the array is to be multiplied by F2 and then have F2 subtracted from it.

```
Loop:  LD      F0, 0(R1)     ; F0 is array element,
                             ; R1 has address of highest unused array element
       MULTD   F4, F0, F2    ; perform scalar multiplication
       SUBD    F4, F4, F2    ; perform scalar subtraction
       SD      0(R1),F4      ; store the result
       SUBI    R1, R1, #8    ; decrement pointer
       BNEZ    R1, Loop      ; branch while R1 is non-zero
```

3.1.1 (5) Compute the CPI of the loop on the specified DLX pipeline.

3.1.2 (10) Unroll the loop as many times as possible. Do whatever instruction re-ordering is needed to **minimize** the CPI and the effects of the loop overhead. Compute the new CPI.

3.2 (10) Compare and contrast the scoreboard algorithm with Tomasulo's algorithm regarding the pipelining problems they were designed to solve, their treatment of data hazards, any new hazards associated with them, the benefits of one over the other, and potential drawbacks of each. You may organize your results as a chart, if that makes it easier for you.

Problem 4: Miscellaneous (25 pts)

This is a hodge-podge of things that didn't fit elsewhere, but that I think you should know. Oh, and they're worth 5 points each; so, don't write a book, just address the basic question.

4.1 (5) Suppose you are using Amdahl's law or another formula to model performance of a real system under a proposed upgrade, and you get an answer that says the speed up is (-2), or that the execution time must be decreased by (-10) seconds to achieve a desired result. What can you conclude about the effect of the proposed changes upon the system?

4.2 (5) Explain the benefits of a split-cache organization for instruction and data memory. Indicate any potential drawbacks.

4.3 (5) Explain the difficulties associated with branching in pipelines? Why is accurate prediction of branch behavior important? What, if any is the best strategy to use in dealing with branches?

4.4 (5) Explain briefly the concept of stage slippage as implemented in the PowerPC 620 pipeline in **your own words**. Why is it so important to keep the instruction buffer full?

4.5 (5) In a distributed system of identical pipelined machines using a shared memory model, explain any problems associated with caches in assuring timely and correct computations. (Shared memory means that access of one processor's memory by another is expected to be a common occurrence.)