CMSC 411 Computer Architecture

Midterm Exam Solution

Exam Number 800

Summer, 1997

Problem 1: Truth With Consequences (22)

Most are false–offending words are in bold.

A) The PowerPC uses a (0,3) **CISC** ISA. Should be RISC.

B) The terms MIPS and MFLOPS indicate the average **throughput** of the systems they describe.

Throughput really isn't in terms of instructions per second, but really more like tasks per unit time from a system manager standpoint. So, the actual behavior may never actually match the term MIPS; it just means that they have a program, or set of programs, which achieve this rating. The MFLOPS term fails for that reason too, but more basically because NO normal CODE can possibly be all floating point operations. Some fardling hacker could come up with an outlandish example, but, in general, other factors besides MFLOPS contribute to the computer's throughput–such as non-FLOPS. The MFLOPS rating tells you zilch about the average number of operations in a program with minimal MFLOPS. So, it really doesn't match up with throughput as an appropriate measure.

C) **All** GPR machines are RISC architectures. *Nope. Nice, big fat CISC Vaxes are GPR machines.*

D) The rule of thumb "Make the common case fast" implies that the registers in the CPU should be as fast as possible because all instructions use them.

Unless I see a dynamite counterexample, this one gets to be true for a while. However, it wasn't one of my better T?F attempts.

E) An accumulator-based computer is a **(1,1)** machine.

Yeah, right. A machine with one user-addressable register thinks it can be called a "general-purpose register machine". That arrogant beast!

F) Amdahl's law implies that the amount of time saved by using a given enhancement to a machine is inversely proportional to the fraction of the original code that was enhanced.

G) The instruction set architecture plays *little or no role* in determing the average number of clocks per instruction. *See page 34 of H&P again.*

H) Caches were developed to increase the rate at which the CPU could **access main memory**. *This may get arguments, but just because you access main memory, it doesn't imply to me that you've picked up the data. A better statement would be: ....increase the rate at which the CPU receives data from memory.*

I) Stack machines require **one** operand in their ALU instructions. *Zero, duh.*

J) The principal difference between RISC and CISC architectures is the hardware organization.

K) Regardless of the code being executed, pipelined architectures are faster than sequential machines because more instructions can be executed during a given time period.

L) Pipeline architectures were originally designed to make memory-intensive computations faster.

Problem 2: I'd Turn Back If I Were You (25 pts)

The following section requires short paragraph-like 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.

2.1 Why is the average CPU execution time often preferred over MIPS and MFLOPS as a metric for expressing the speed of general purpose computing platforms? (I want two well explained reasons.)

2.2 Suppose that my code runs on a 5 stage DLX pipeline that has only one memory port. Furthermore, suppose I choose to ignore hazards, because I don't believe they are a problem. After all, no computer course I took in college ever discussed such things; so, I should know!

Define the potential hazards, and for each class of hazard, explain the potential consequences of this brash decision while I, er, update my resume.

2.3 I just had to buy braces to realign my computer's over-byte! Explain what byte alignment is and what happens if you mistakenly commit a byte alignment violation (after they sew the ear back on, of course).

2.4 Consider the CPU execution time equation and address the following issues: What quantity (or quantities) in the equation are the easiest to calculate, and why? What makes the remaining item(s) harder to compute?

2.5 If I increase the average CPI by 5%, decrease the instruction count by 20% and exchange the clock crystal for one that has twice the old clock rate, what speedup should I expect, if any, and why?

Problem 3: Enter at Your Own Risk (20 pts)

See how much time you can save, or lose, by modifying the hardware and software in the following scenarios. **Make sure you clearly state any assumptions you make that are not explicitly provided in the problem.**

3.1 Suppose you secretly increase the clock cycle time by 10% on your roommate's brand new computer. Will your roommate definitely notice, or is there a possibility that you may get away with yet another outrageous stunt?

Explain your answer based on the potential impact, if any, upon the average instruction count and the average clocks per instruction. Clearly state any other assumptions that you make in your reasoning. (Even if I don't agree with your answer, I'll be gracious in the face of a good argument.)

3.2 Suppose that 75% of the code, a routine called Fu, is executed 25% of the time, and that you have purchased a new optimizing compiler, Bar, that will decrease the instruction count significantly, essentially cutting it by a factor of 4 for the entire program. Furthermore, after the optimization, the computer spends 50% of time executing the new routine, now called FuBar. Compute the speedup, if any after the optimization, and determine the new percentage of the code that corresponds to the FuBar routine.

Problem 4: A Night on Code Mountain (36 pts)

As usual, I've saved the worst for last. A little fragment music, maestro! **Make sure you clearly state any assumptions you make that are not explicitly provided in the problem and document your code.** And, this problem is really worth 36 points; so, don't wait too long to start it.

4.1 (10) Your job is to implement portions of a stack in the memory of a DLX machine. Assume that the stack pointer is register R31, and that you may the abbreviation SP for it as a reminder. We assume 0(SP) refers to the contents of the top of the stack, -4(SP) the next word on the stack, and so on.

4.1.1 (4) Write the DLX code fragments needed to initialize the stack.

4.1.2 (4) Write the code fragments needed to implement the PUSH operation in DLX, where the value to be pushed is stored in location 0(R30) prior to calling the push routine.

4.1.3 (2) Explain why you might want to implement a stack in DLX, even though it is not a stack architecture.

4.2 (10) Write the DLX code fragment that will compute the function Power(X,N), which raises the real number X to the **integer power N**. For my convenience, assume that X and N are stored in consecutive memory locations 32 and 36. Make sure you fully document your code, and give the code fragment needed to call this routine as well.

4.3 (16) Consider the following DLX code fragment to be executed on a a simple 5-stage *pipelined* machine separate instruction and data memories.

```
       ADD    R1, R0, R0     ; init counter to 0
Loop:  LW     R2,100(R1)     ; R2 is array element,
                             ; R1 has offset of highest unused array element
       LW     R3, 500(R1)    ; R3 holds array element
       SUBI   R5, R3, R2     ; perform subtraction
       ADD    R5, R5, R4     ; perform addition of a constant
       SW     1000(R1),R5    ; store the result
       ADDI   R1, R1, #4     ; increment pointer
       SUBI   R5, R1, #400   ; check pointer
       BNEZ   R5, Loop       ; branch while not done
       STOP
```

4.3.1 Compute the CPI for one iteration of the loop, ignoring the BNEZ command in your analysis. Use this to estimate the total CPI for the loop again, ignoring the contribution of the branch.

4.3.2 Suppose you were to increase the number of iterations performed inside the loop. Rewrite the code to operate on 5 pairs of array operands during the loop body. How will this change the average CPI and the overall execution time of the code? Justify your answer for full credit.