Machine Organization and Assembly Language Programming

Problem Set #4

Due: **Friday**, February 6

This assignment is on the material covered in Chapter 2 and Chapter 5 Sections 1-3 of your book.

Problem Set #5 will be the continuation of the JVM simulation. A hand-out will be given shortly. Problem Set #5 will be due **Wednesday, February 11**

1. Problems 2.2 (Note the typo, "106" is in fact 10^6 in the "number of instructions executed by M2"); 2.3; 2.4.

2. Problems 2.10 and 2.11

3. After graduating, you are asked to become the lead computer designer at Super Duper Computer Company Inc. (SDCC). After taking numerous courses in high-level languages, compilers, and machine organization, you know that procedure calls are expensive operations. You have invented a scheme that reduces the loads and stores normally associated with procedure calls and returns. You run experiments with the same compiler on the original machine Mbase and the machine that you propose Mpro. You find out that:

- The cycle time of Mpro is 8% higher than that of Mbase (because your clever scheme is on the critical path of timing)
- 36% of the instructions when running on Mbase are loads and stores.
- Mpro executes 1/3 fewer loads and stores than Mbase (because of your clever scheme)
- All instructions (including load and stores) take one cycle.

Is the technical department (the one that looks at execution time) happy? In other words, which of Mbase or Mpro is faster. Justify quantitatively.

Is the marketing department (the one that sells machines depending on their MIPS rate) happy? Justify quantitatively.

What do you think your future is with SDCC (just kidding...)

4. Well undaunted by your previous attempts in computer design and not intimidated by the marketing hype, you now attack the problem through compiler optimizations. Because of your influence at the company, SDCC is now manufacturing a more realistic RISC (load-store) machine Mfoo. Mfoo runs as fast as all well-known current microprocessors with a 500 MHZ clock rate (i.e., the clock cycle time is 2ns). Measurements of various benchmarks running on Mfoo have yielded the results in the table next page.

Instruction type	Frequency	Clock cycle count
ALU ops	40%	1
Loads	23%	2
Stores	13%	2
Branches	24%	2

What is the CPI of Mfoo? What is the MIPS rating of Mfoo?

Now your compiler optimizations are such that you can dispense with half of the loads and stores.

What is the new CPI? How much better is the *performance* of Mfoo with optimized code. Do the MIPS ratings for Mfoo with and without optimizations give an adequate measure of the improved performance?

Is your future with SDCC brighter (still kidding...)?

5. Read the book page 101 explaining Amdahl's law. Then answer the following question:

Implementations of floating-point square root (FPSQR) vary widely. At SDCC (again!) in a benchmark very important for a large customer FPSQR is taking 15% of the execution time on Mfoo. A friend of yours, Joe Nerd, proposes to make FPSQR hardware that will speed up its operation by a factor of 10. You, with your vast experience, know that floating-point instructions, including FPSQR, are responsible for 50% of the execution time of this benchmark. You propose a scheme that will make all floating-point instructions run 1.8 times faster with no more effort than Joe Nerd's design.

What is the speed-up that will result from Joe Nerd's enhancements?

What is the speed-up that will result from your enhancements?

Will you become Joe Nerd's manager or will you be fired?

6. Problems 5.5 and 5.12