COMP206 **Homework 4** Assigned: November 6, 2003
Prof. Montek Singh Due: November 17, 2003

Note: This homework assignment is in two parts, Part I and Part II. Part I consists of problems from the textbook (Hennessy/Patterson 3rd ed., or "HP3") whose solutions appear at the end of the book. These problems are for self-study only, and will not be graded, and therefore need not be submitted as part of your written work. Part II consists of problems that will count towards your grade for this homework.

Part I (not graded):

1. **Superscalar MIPS.** Do HP3 Problem 3.17.
2. **Tomasulo Algorithm with Speculation.** Do HP3 Problem 3.21.
3. **Loop-Carried Dependences.** Do HP3 Problem 4.5.
4. **Loop Unrolling.** Do HP3 Problem 4.7.
5. **Benchmarking Cached Systems.** Do HP3 Problem 5.1.

Part II (to be graded):

1. (25 points) **Dynamic Scheduling vs. Speculation.** Do HP3 Problem 3.18, but with the following clarifications. For the first part (*i.e.,* dynamic scheduling), assume the original Tomasulo's algorithm, *without* speculation, and present your answer in the same format as the table in Figure 3.33. For the second part, assume Tomasulo's *with* reorder buffer, and use the format of the table in Figure 3.34.

2. (25 points) **Multiple-Issue: VLIW.** Suppose we have a VLIW processor that could issue two memory references, one floating-point operation, and one integer operation or branch in every clock cycle. Assume the following:

   - *Load to add latency:* If a L.D instruction is issued in cycle 1, then a dependent ADD.D instruction, which needs the value just loaded, must be issued in cycle 3 or later.
   - *Add to store latency:* If an ADD.D instruction is issued in cycle 1, then a dependent S.D instruction, which saves the result to memory, must be issued in cycle 4 or later.
   - There is one branch-delay slot.
   - An unlimited number of registers are available.

   (a) Show an unrolled version of the following loop for such a processor. and schedule it to achieve the shortest execution time. Unroll the loop the minimum number of times necessary to eliminate all stalls, but no more than *six* times. *For full credit, your unrolled code must execute as fast as possible.*

   ```
   Loop:  L.D    F0, 0(R1)
          ADD.D  F4, F0, F2
          S.D    0(R1), F4
          SUBI   R1, R1, #8
          BNEZ   R1, Loop
   ```

   You may enter your answer in the following table.

Mem Ref 1	Mem Ref 2	FP op	Integer/branch

(b) How many clock cycles does your unrolled loop require for each iteration of the original loop?

3. (10 points) **Loop-Carried Dependences.** Do HP3 Problem 4.6. Note that this problem simply asks you to rewrite the loop so as to eliminate all loop-carried dependences (*i.e.,* dependences between instruction instances across loop iterations). *Hint:* This is an easy question!

4. (20 points) **Software Pipelining.** Do HP3 Problem 4.11. This problem refers to the example in HP3 on page 330 and/or the example on slide 19 of Lecture 16 (October 29, 2003). Note that the phrase "latency of the ADD.D was 5 cycles" means the following: if ADD.D is issued at clock cycle 1, then the subsequent consuming S.D must be issued at clock cycle 7 or later. Due to such a long latency, software pipelining is not enough to ensure that dependent instructions are placed far enough apart so as to not require stalls. In this question, you are to first apply software pipelining to expose loop-level parallelism, and then unroll the resulting loop and schedule it to eliminate stalls. Be sure to show the loop start-up and clean-up code.

5. (20 points) **Caches: Block Replacement Policy.** Do HP3 Problem 5.12, *parts (a)–(c) only.*