LSU EE 4720 **Homework 4** **Due: 28 April 2010**

Problem 1: A deeply pipelined MIPS implementation is constructed from our familiar five-stage pipeline by splitting `IF`, `ID`, and `ME` each into two stages, but leaving `EX` and `WB` as one stage. The total number of stages will be eight, call them `F1`, `F2`, `D1`, `D2`, `EX`, `Y1`, `Y2`, and `WB`. In this system branches are resolved at the end of `D2` (rather than at the end of `ID`). Assume that all reasonable bypass paths are present.

(*a*) Provide a pipeline execution diagram of the code below for both the 5-stage and this new implementation, for enough iterations to compute the IPCs.

```
LOOP:
 addi r2, r2, 4
 lw r1, 0(r2)
 add r3, r3, r1
 bne r5, r4 LOOP
 addi r5, r5, 1
```

(*b*) Suppose the 5-stage MIPS runs at 1 GHz. Choose a clock frequency for the 8-stage system for which the time to execute the code above is the same as for the 5-stage MIPS.

(*c*) Consider two ways to make a 7-stage system from the 8-stage system. In method ID, the two ID stages (`D1` and `D2`) are merged back into one (or if you prefer, the ID stage was never split in the first place). In method ME, the two ME stages (`Y1` and `Y2`) are merged back into one (or were never split).

Which method is better, and why? Assume that the eight-stage system runs at 1.8 GHz. Consider both the likely impact on clock frequency (remembering that you are at least senior-level computer engineering students) and the benefit for code execution (don't just consider the code above, argue for what might be typical code).

Problem 2: Itanium is a VLIW ISA designed for general-purpose use. Being a VLIW ISA (as defined in class) its features were chosen to simplify superscalar implementations. The questions below are about such features, read the Intel Itanium Architecture Software Developer's Manual Volume 1, Section 3 for details and concentrate on Sections 3.3 and 3.4. The manual is linked to the course references page, `http://www.ece.lsu.edu/ee4720/reference.html`. Use this copy to be sure that section and table numbering used here match.

(*a*) Section 3.3 mentions four types of functional unit, and where an instruction using a particular unit can be placed in a bundle (see Table 3-9 and 3-10).

Suppose an Itanium implementation fetches one bundle per cycle. Indicate the maximum number of execution units of each type needed. (That is, there would be no advantage of having more than this number.) Assume that the units all have latency 1 or else are fully pipelined.

(*b*) For this problem suppose the implementation had the minimum number of units of each type, one. Sketch the pipeline execute stages, and show connections to each of the FU inputs. There should be three sets of source operands flowing down the pipeline. Some (or all) of the execute units should have multiplexors at their inputs to select operands from one of the three instructions in a bundle. Show the multiplexors, and based on the slot restrictions show the minimum number of inputs.

(*c*) Notice in Table 3-10 that there is no template with a stop right after Slot 0 and right after Slot 1. Provide a possible reason for this.

Suppose there was a template with two such stops (as described above), call this ISA Itanium-stop-stop. Why might code compiled for Itanium-stop-stop be smaller than code compiled for Itanium?

Consider Itanium and Itanium-stop-stop implementations that fetch one bundle per cycle (same as in the prior problems). Explain why Itanium-stop-stop might be no faster than Itanium.