LSU EE 4720 **Homework 2** Solution **Due: 17 March 2010**

Problem 1: The SPARC `jmpl` (jump and link) instruction adds the contents of two source registers or a register and an immediate, and jumps to that address. It also puts the address of the instruction in the destination register (usually to be used to compute a return address). For more information, find the description of `jmpl` in the SPARC V8 ISA description from the references linked to the course home page.

In this problem a similar instruction (or instructions) will be added to MIPS. Like the SPARC `jmpl`, the MIPS variant can jump to a target determined by the sum of two registers or a register and an immediate, while the address of the instruction is saved in the destination register. (Note that the saved address is different than the address saved by MIPS' `jalr` and `jal` instructions. Be sure to save the address indicated by the SPARC definition.)

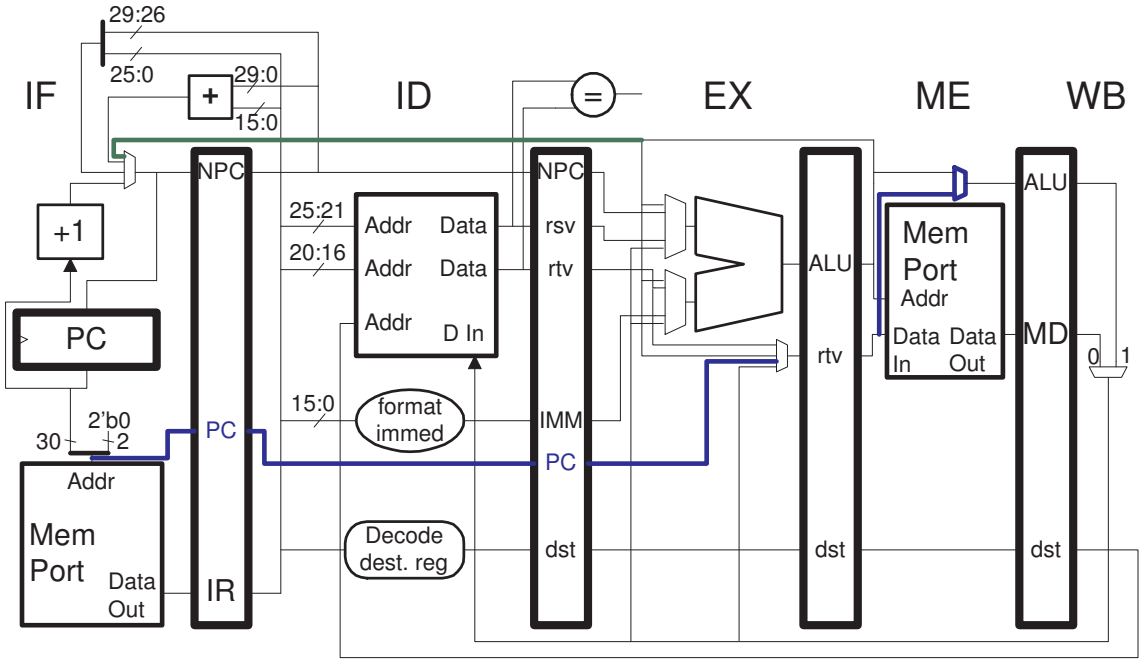
(*a*) Show how the MIPS version of these instruction(s) can be encoded. Show a format for the instruction, using the descriptions in the MIPS32 Architecture Volume II (linked to the references page) as an example. The format should show which instruction fields indicate each part of the instruction.

Two instructions are needed, `jmpl` for the two-source-register form and `jmpli` for the source-register-plus-immediate form. Assembly syntax:

```
jmpl rd, rs, rt
Description:
 pc_cpy <- PC
 PC <- rs + rt
 rd <- pc_cpy

jmpli rt, rs, immed
Description:
 pc_cpy <- PC
 PC <- rt + sign_extend(immed)
 rt <- pc_cpy
```

(*b*) Show datapath changes (that is, omit control) to the implementation below needed to implement this (these) instruction(s). The changes must fit in naturally with what is present and should not risk lowering clock frequency. Do not forget about any changes needed to save a return address.



Solution appears above. The changes needed for the target address appear in green, the changes needed to save the return address appear in blue. The target address is computed by the ALU, it is sent to the IF-stage PC mux from the ME stage to preserve clock frequency. If the output of the ALU connected directly to the IF-stage PC mux the clock frequency might have to be lowered because the ALU output would not be ready until the end of the cycle (based on the pre-change clock frequency).

The SPARC jmpl instruction saves the address of the instruction itself, new pipeline latches were added to carry that. Note that this is probably wasteful, since there are already latches to carry NPC needed by the existing `jal` and `jalr` instructions. The alternative would be to subtract four from the NPC value.

To save some hardware, the PC value joins the rtv in EX, making use of a new ME-stage mux.

(*c*) As discussed in class, a SPARC-style `jmpl` on something like our 5-stage pipeline would have to be resolved in EX. However, a higher-cost implementation might resolve a `jmpl` in ID if no addition were necessary.

Identify which of the following cases is the least trouble to detect (shown with SPARC assembler), and explain why it is the least trouble:

```
jmpl %g1, %g0, %o7    ! g0 is the zero register.
jmpl %g1, 0, %o7      ! The immediate is zero.
jmpl %g1, %g2, %o7    ! Contents of g2 is zero.
```

Trouble in this context is time and hardware cost. Both hardware cost and time are determined by how many bits need to be checked for a zero value. Another factor for time is when during a clock cycle the check can start.

For the first case we need to check for the zero register in the `rt` field. That entails checking five bits, and these bits are available at the beginning of the clock cycle.

For the second case we need to check whether the immediate is zero. That means checking 16 bits, also starting at the start of the clock cycle.

For the last case we need to check 32 bits (the rt value at the output of the register file), and those bits are not available until near the end of the clock cycle.

Therefore the first case is the best (least trouble), and the last case is worst, by far.

Problem 2: **Without looking at the solution,** do Fall (November) 2007 Midterm exam Problem 1. Use the Statically Scheduled MIPS study guide, `http://www.ece.lsu.edu/ee4720/guides/ssched.pdf`, for tips on how to solve this interesting, understanding-building, and fun-to-solve (if one is prepared and not under intense time pressure) problem. Only use the solution if you must. **Warning:** *The test problems will be chosen under the assumption that students really solved this problem.*