LSU EE 4720 **Homework 4** **Due: 31 March 2003**

Problem 1: The two code fragments below call trap number 7. How do the respective handlers determine that trap 7 was called?

```
! SPARC V8
ta %g0,7

# MIPS
teq $0, $0, 7
```

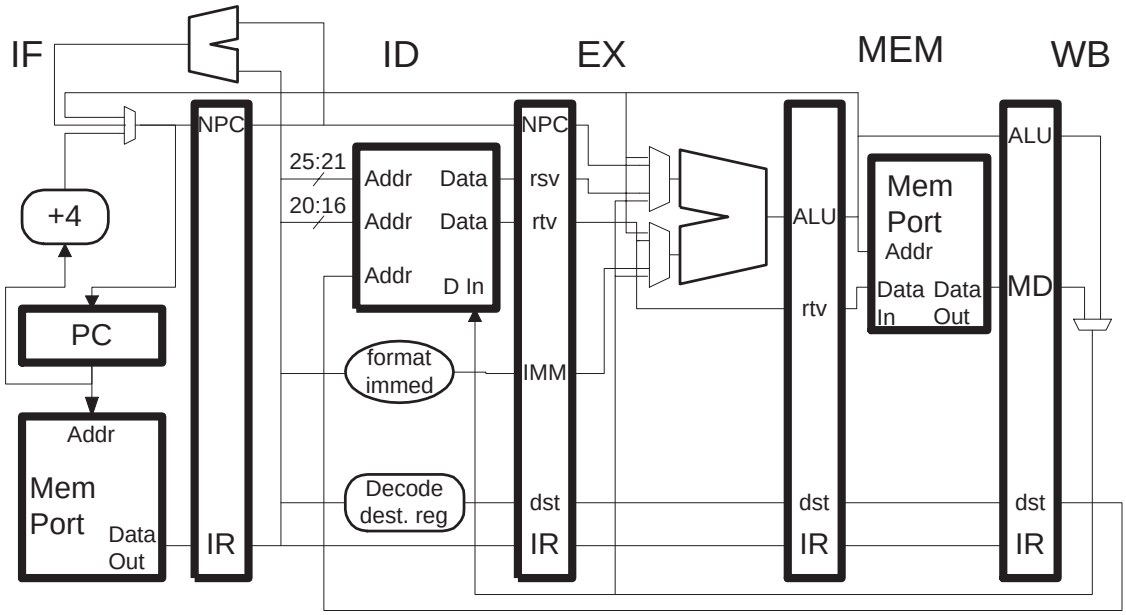
Problem 2: There is a difference between the software emulation of unimplemented SPARC V8 instructions triggered by an illegal opcode exception, such as `faddq`, and Alpha's use of PALcode for certain instructions. (See the respective ISA manuals on the references Web page. For SPARC, see Appendix G, it should not be difficult to find the PALcode information for Alpha.)

(*a*) What is similar about the two?

(*b*) What is the difference between the kinds of instructions emulated using the two techniques? Why would it not make sense to use PALcode for quad-precision arithmetic instructions?

Problem 3: In both SPARC and MIPS each trap table entry contains the first few instructions of the respective trap handler. On some ISAs a *vector table* is used instead, each vector table entry holds the **address** of the respective handler.

Why would the use of a vector table (rather than a trap table) be difficult for the MIPS implementation below?



Problem 4: One way of implementing a vector table interrupt system on the MIPS implementation above would be by injecting hardware-generated instructions into the pipeline to initiate the handler. These instructions would be existing ISA instructions or new instructions similar to existing instructions.

What sort of instructions would be injected and how would they be generated? Show changes needed to the hardware, including the injection of instructions. In the hardware diagram the instructions can be generated by a magic cloud [tm] but the cloud must have all the inputs for information it needs.

Include a program and pipeline execution diagram to show how your scheme works.

- Assume an exception code is available in the MEM stage.
- Include a vector base register, (VBR), which holds the address of the first table entry.