LSU EE 4720 **Homework 2** Solution **Due: 9 March 2005**

For answers to the questions below refer to the PowerPC description Book I which can be found on the class references page, `http://www.ece.lsu.edu/ee4720/reference.html`.

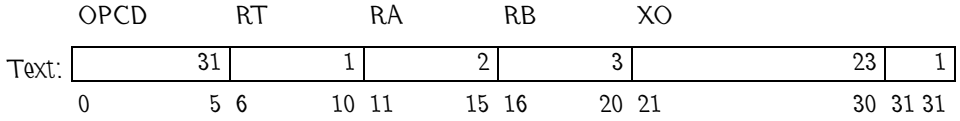
Problem 1: One instruction that MIPS lacks but many RISC ISAs have is an indexed load. Find the closest equivalent PowerPC instruction to SPARC's `lw [%r2+%r3],%r1`.

(*a*) Show the instruction in PowerPC assembly language.

```
# Solution:
lwzx r1, r2, r3
```

Note: Instruction `ldx`, which loads 64 bits) would also be graded correct, but since SPARC's `lw` is a 32-bit unsigned load (in SPARC V9), PPC's 32-bit unsigned indexed load, `lwzx`, is more correct.

(*b*) Show how the instruction is coded, include the register numbers.



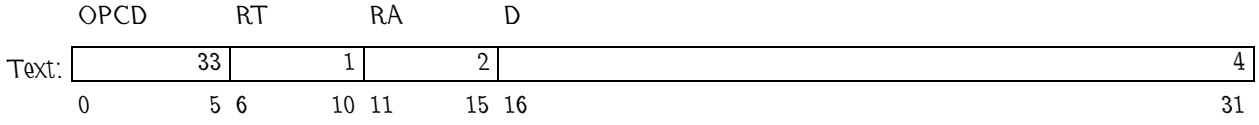
Problem 2: One instruction that MIPS lacks but that a few other RISC ISAs have is autoincrement addressing. PowerPC has an instruction that can be used for autoincrement addressing but is more powerful than the autoincrement addressing described in class. Find the PowerPC instruction.

(*a*) Show the assembly language for the PowerPC instruction doing the same thing as the following autoincrement instruction: `lw r1, (r2)+`.

```
# Solution
lwzu r1,4(r2)
```

The PowerPC instruction above is not 100% equivalent because it uses `r2+4` as the effective address whereas a typical autoincement would use `r2` as the effective address. This is not a practical problem because `r2` could be initialized to four less that the first address to be loaded.

(*b*) Show the coding for the instruction above.



(*c*) The PowerPC instruction is more powerful than an ordinary autoincrement instruction. Show a code sample using the PowerPC instruction for which an ordinary autoincrement would not be suitable. Briefly explain why an ordinary autoincrement would not do.

```
LOOP:
 lwzu r1, 16(r2)
 cmpdi r1,0
 bne LOOP
```

The loop above loads words separated by 16 bytes. An ordinary autoincrement would require an extra add instruction (otherwise it would load words separated by 4 bytes (that is, consecutive words).

Problem 3: PowerPC has a wide variety of load and store instructions. Find the load instruction that is least suitable for a RISC ISA based upon the criteria discussed in class. Explain why it's least suitable.

Instruction `lswx` is un RISC like because it can load several registers and so an implementation would have to send the instruction through part of the pipeline several times, unlike conventional RISC instructions. This would make the pipeline control much more complicated, but not too complicated since PPC is a real ISA with fast implementations.

Problem 4: Some instructions are more difficult to implement than others, one reason is that the difficult instruction does something very different from normal instructions, requiring at least a moderate amount of additional hardware. Some difficult-to-implement instructions are listed below. Explain what the difficulty is (what extra hardware or control complications would be needed).

(*a*) An indexed store instruction. (An indexed load instruction would **not** be considered difficult.)

An indexed store would have three source operands, two for the effective address and one for the store value. That would require three read ports from the register file, it no other instruction has three source register operands then implementing the indexed store would increase the cost by a significant amount.

(*b*) Autoincrement (or PowerPC's version) load instructions. (The autoincrement or PowerPC version of the store instructions are not difficult.)

An autoincrement load writes two register values, the loaded value and the incremented address. That would require two register write ports, if no other instruction wrote two or more operands than that would increase the cost by a significant amount.