Name Solution

Computer Architecture
EE 4720
Midterm Examination

Friday, 1 April 2005, 11:40–12:30 CST

Problem 1		(25 pts)
Problem 2		(30 pts)
Problem 3		(25 pts)
Problem 4		(20 pts)
Exam Total		(100 pts)

Alias Yes

Good Luck!

Problem 1: In the diagram below some wires are labeled with cycle numbers, indicating that the wire is used at that cycle. If a value on any labeled wire is changed the code would execute incorrectly.

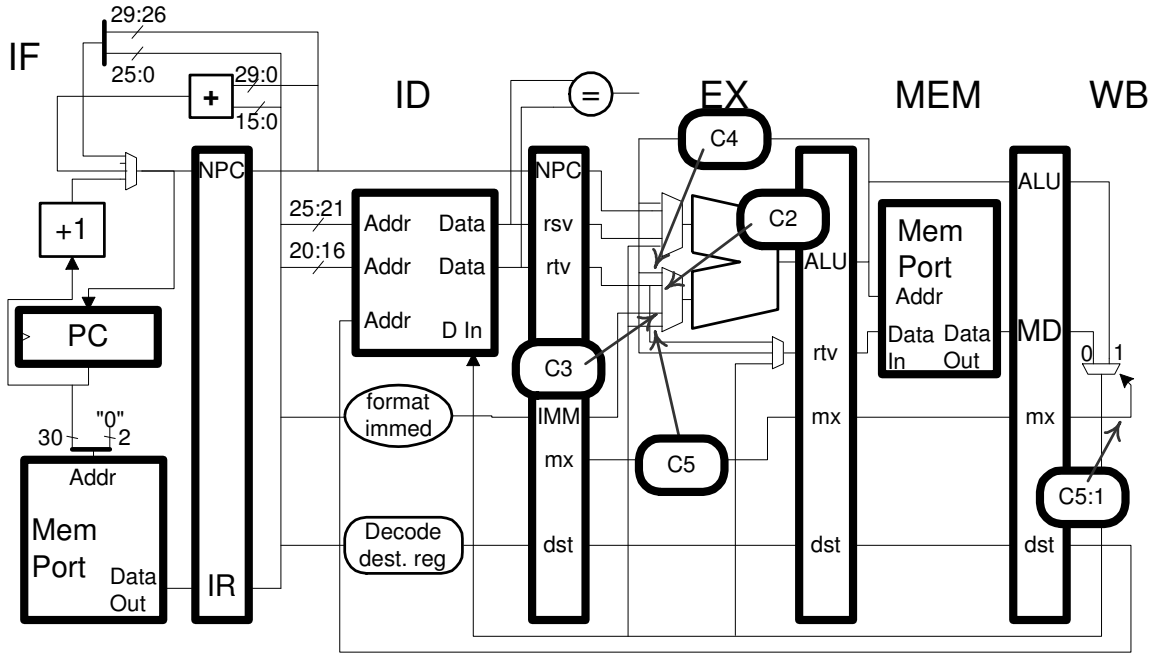
- There are no branches or other control-transfer instructions.

☑ [15 pts] Write a program consistent with these labels.

☑ All register numbers must be made up; **use as many different register numbers as possible** while still being consistent with the labels.

☑ [10 pts] Why would there be no solution to the problem if the C5:1 label in WB were changed to C5:0?

If the label were changed then the second instruction would have to be a load. The third instruction bypasses the result of the second instruction without a stall, and there's no way that can be done if the second instruction is a load.




```
# Solution
Cycle                0  1  2  3  4  5  6  7  8  9
add r1, r2, r3       IF ID EX ME WB
addi r4, r5, 6          IF ID EX ME WB
sub r6, r7, r4             IF ID EX ME WB
or  r8, r9, r4                IF ID EX ME WB
Cycle                0  1  2  3  4  5  6  7  8  9
```

Problem 2: In the diagram on the next page some wires are labeled with cycle numbers and corresponding values. For example, C1:8 indicates that at cycle 1 the pointed-to wire will hold an 8. Other wires are labeled just with cycle numbers, indicating that the wire is used at that cycle. If a value on any labeled wire is changed the code would execute incorrectly.

- Unlike other problems of this type all registers are different and there are no dependencies.
- The code contains at least one floating-point add and one floating-point multiply.

☑ [10 pts] Write a program consistent with these labels; if a register number cannot be determined use rX or fX.

☑ [5 pts] Complete the pipeline execution diagram.

☑ [5 pts] Fill in the box on the lower-left of the diagram.

☑ [5 pts] Your answer should have a FP multiply instruction. Explain how you knew it was a multiply. (Don't answer "because I already found the add.")

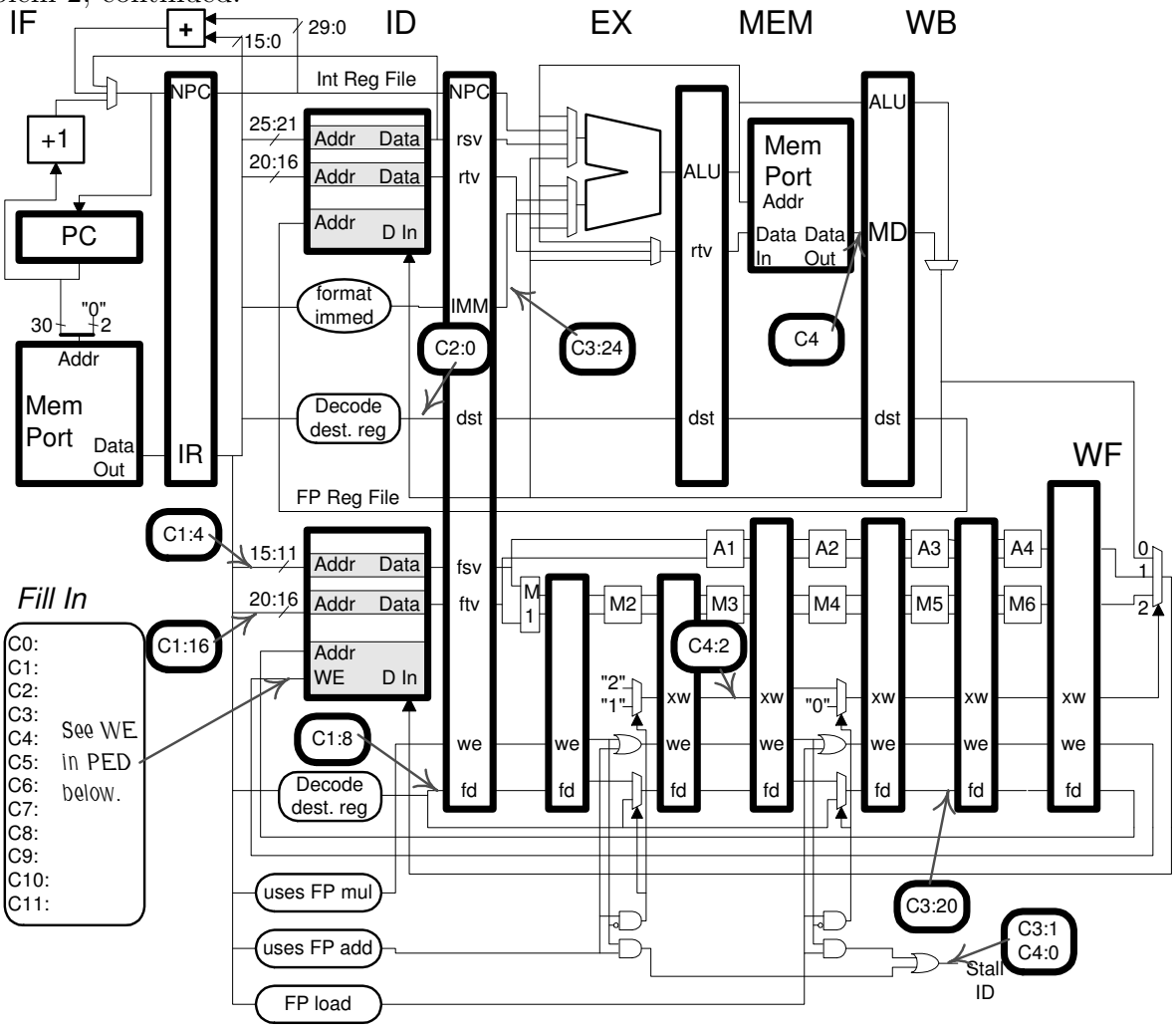
The C4:2 in A1/M3 indicates that in cycle 4 there is a multiply in M3. Working backward, that multiply had to be in ID in cycle 1, and so it's the first instruction.

☑ [5 pts] Your answer should have a FP add instruction. Explain how you knew it was an add. (Don't answer, "because I already found the multiply.")

The C3:1 in the lower right indicates a stall in cycle 3. Such a stall could be because the decoding instruction (the third one, since it's cycle 3) is an add or a load. If it were an add then it would indeed have a structural hazard (thus requiring a stall) with the first instruction, a multiply. If it were a load then the only instruction it could have a structural hazard with is the second one, but there is no instruction type that would need the WF register at the same time, so it has to be an add.

The second instruction is a FP load. We know that because it's using the Data Out of the memory port, and when it reaches WB/F the WB register is zero (from the C2:0) and the WF register is 20 (from the C3:20). Many realized it was a load, few got the correct destination register.

Problem 2, continued:




```
# Solution
Cycle                0  1  2  3  4  5  6  7  8  9
mul.d f8, f4, f16    IF ID M1 M2 M3 M4 M5 M6 WF
lwc1 f20, 24(rX)        IF ID EX ME WF
add.d fX, fX, fX           IF ID -> A1 A2 A3 A4 WF
WE:                  0  0  0  0  0  1  0  0  1  1  0  0  <- Values for Fill-In box.
Cycle                0  1  2  3  4  5  6  7  8  9  10 11
```

Be sure to complete all parts, including the justification for add and multiply.

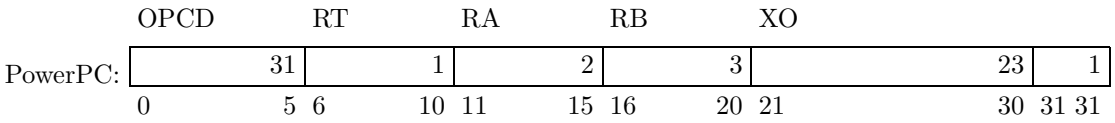
Problem 3: Answer each question below.

(*a*) The encoding of the PowerPC `lwzx r1, r2, r3` instruction is illustrated below, followed by the Type-R MIPS instruction format. [6 pts]

☑ If a field in the illustrated PowerPC instruction format has a counterpart in the MIPS R format draw an arrow between the fields. For example, an arrow should be drawn from OPCD to Opcode. Pay close attention to the register fields.

☑ If fields connected by an arrow are different sizes explain the significance of the difference. (What advantage or disadvantage would the format with the larger field have, if any.)

The smaller function field means that Type R can specify fewer functions than the PowerPC field. Having more opcode space is an advantage if those opcodes are needed.



OPCD ⟶ Opcode, RT ⟶ RD, RA ⟶ RS, RB ⟶ RT, XO ⟶ Function,

(*b*) Suppose that your favorite program is both in the SPEC CPU2000 suite and in the suite used by a popular personal computing magazine and that the input data used for testing the program are also identical. You are considering buying two computers, A, and B. The spec testing shows that your favorite program is faster on A but the magazine's test shows that it is faster on B. The exact same model of machine A is used for running the spec test and the magazine's test, likewise for B. [6 pts]

☑ Provide a possible reason for the discrepancy, the reason should help explain why SPEC is used by the CPU research and development community. *Note: In the original exam the text after discrepancy was offered as a hint.*

Short answer: unlike the magazine's, the spec test would be specially compiled for each machine and might have taken advantage of some unique features of A.

When doing the spec testing the program can be compiled just for the tested machine. Personal computer magazines test programs in the form they are sold to customers, which is already compiled (and so in binary form). Therefore the magazine would test the same binary on both machines whereas those doing the spec testing would specially compile it for each machine, possible taking advantage of special features of A.

Grading Notes: Most missed the key point, that spec benchmarks are compiled for each test whereas end-user benchmark suites, as used by magazines, use just one binary.

Another common misconception is that spec does the testing. They don't, they provide the suite and specify the rules, but they don't actually test.

(*c*) SPEC members and associates are drawn from computer manufacturers, other industries, academia, and elsewhere. [6 pts]

☑ Would you trust SPEC benchmark results more (or less) if computer manufacturers were not allowed to join SPEC? Explain.

Both answers would be considered correct (that includes the reasoning):

I would trust them less because manufacturers have a very strong incentive to keep each other honest.

I would trust them more because manufacturers might want to make it look like future machines are faster by putting in benchmarks which would be faster on future processors but which are not the kinds of programs that many users would run.

(*d*) In class we saw that the optimized π program (reproduced below) ran faster than the unoptimized version (good) but also had higher CPI than the unoptimized version (bad). *Note: the words "than the unoptimized version" was not included in the original exam.* [7 pts]

☑ Why was the CPI higher?

Hint: It has to do with something you learned early in grade school.

```
double i, sum = 0;
for(i=1; i<5000; )
  {
    sum = sum + 4.0 / i;
    i += 2;
    sum = sum - 4.0 / i;
    i += 2;
  }
```

Optimization eliminated many instructions though not divides. Since divides take the longest the CPI went up, performance improved because there were much fewer instructions.

Grading Note: A surprising number of people answered that the optimizations included loop unrolling. The optimized example used in class did not, perhaps because there would be no performance improvement because execution time is dominated by the division dependencies and loop unrolling won't help that. What the optimizer did do is eliminate unneeded loads and stores and other instructions (reducing instruction count) and scheduled (re-arranged) the code to reduce stalls.

Problem 4: Answer each question below.

(*a*) MMX and VIS are examples of ISA extensions that add packed-operand data types and instructions.[7 pts]

☑ How do packed-operand data types and instructions improve performance?

They allow simultaneous operations on several pairs of operands, a set of operands are packed together in a register and operated on by special instructions.

☑ Show a code example that can use such instructions and explain how they would be used.

```
 // Before
 extern char *a, *b, *c;
 for(i=0; i<1024; i++) c[i] = a[i] + b[i];

 // Before, assembler.
LOOP:   // Iterates 1024 times.
 lb t0, 0(t1)
 lb t2, 0(t3)
 add t4, t0, t2
 sb t4, 0(t5)
 addi t1, t1, 1
 addi t3, t3, 1
 bne t5, t6, LOOP
 addi t5, t5, 1

 // After, assembler.
LOOP:   // Iterates 256 times.
 lw t0, 0(t1)
 lw t2, 0(t3)
 add.pack8 t4, t0, t2  // Add assuming 8-bit quantities packed into register.
 sw t4, 0(t5)
 addi t1, t1, 4
 addi t3, t3, 4
 bne t5, t6, LOOP
 addi t5, t5, 4
```

(*b*) Name an advantage and a disadvantage that RISC's fixed-size instructions have over CISC's variable length instructions.[6 pts]

☑ Advantage:

Branch displacements can be in instructions rather than bytes, allowing for further jumps for the same instruction size. Simplified fetch and decoding.

☑ Disadvantage:

Some instructions are larger than they have to be. Another disadvantage, what could be done with one instruction in CISC, say loading a 32-bit constant, needs at least two instructions in RISC because of space limitations.

(*c*) A trap instruction is something like a subroutine call to the operating system. [7 pts]

☑ So why couldn't it just specify the address of the trap handler?

Because then it might bypass protections that the operating system tries to impose. For example, suppose the address of a file read system call were 0x1000, and that the first part of this routine checked if the user had permission to read that file, followed at address 0x1800 by the code that does the read. The user might trap to address 0x1800 and bypass the protection.

Grading Note: Many answered that a user program would not be allowed to execute in the system area. That's not an issue here because the trap instruction already changes the processor from user to system mode.

☑ What does the trap instruction specify (in SPARC but not MIPS) in place of an address and how does execution actually reach the trap handler.

It specifies an index into (entry number in) the trap table. The hardware will combine this index with the contents of the trap base register (address of the beginning of the trap table) obtaining the address of the handler itself (within the trap table). This address will make it to the PC, and so execution will proceed in the handler.