Name _____

Computer Architecture
EE 4720
Midterm Examination

Friday, 26 March 2010, 10:40–11:30 CDT

Problem 1	_____	(40 pts)
Problem 2	_____	(12 pts)
Problem 3	_____	(14 pts)
Problem 4	_____	(10 pts)
Problem 5	_____	(24 pts)
Exam Total	_____	(100 pts)

Alias _____

Good Luck!

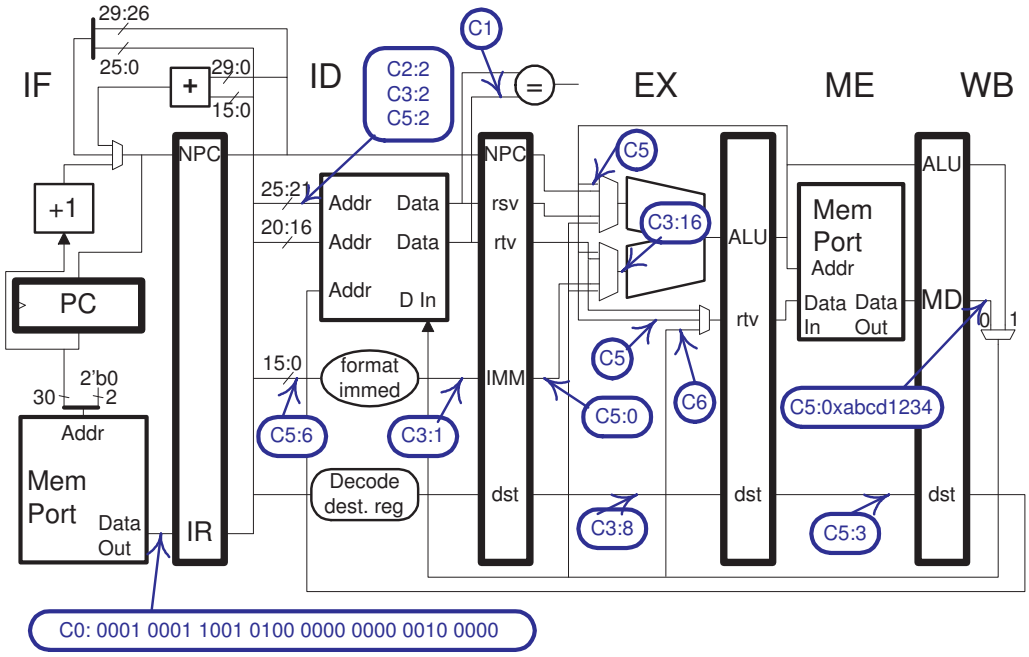
Problem 1: [40 pts] In the MIPS implementation below some wires are labeled with cycle numbers and values that will then be present. For example, c5:6 indicates that at cycle 5 the wire will hold a 6. Other wires are labeled just with cycle numbers, indicating that the wire is used at that cycle. If a value on any labeled wire is changed the code would execute incorrectly. Write a program consistent with these labels.

☐ All register numbers and immediate values can be determined.

☐ The first instruction address has been provided, **show the addresses of the remaining four instructions**.

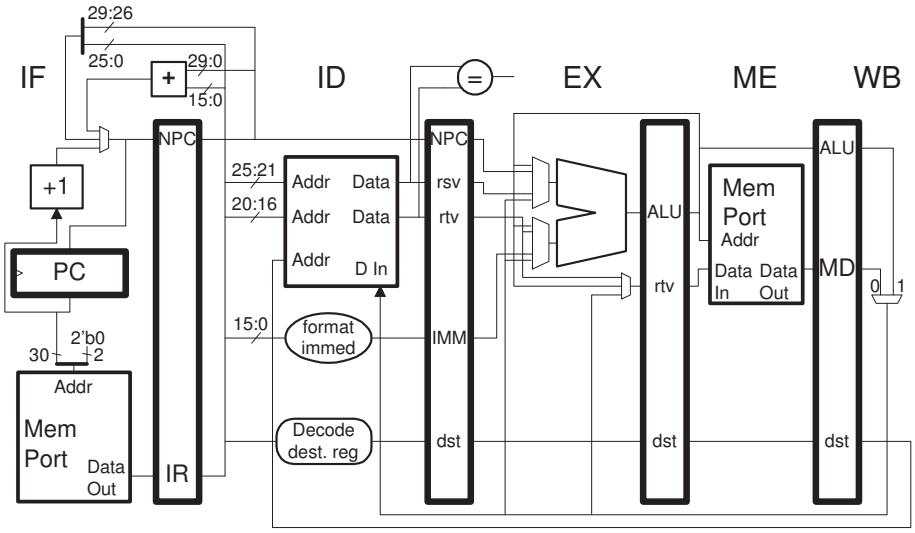
☐ The third instruction is an `addi`, don't forget to show its registers and immediates.

☐ If an instruction is a load or store, show all possible size and sign possibilities. For example, (lw,lh)



	Cycle:	0	1	2	3	4	5	6	7	8
0x1000		IF	ID	EX	ME	WB				
			IF	ID	EX	ME	WB			
addi				IF	ID	EX	ME	WB		
					IF	ID	EX	ME	WB	
						IF	ID	EX	ME	WB
	Cycle:	0	1	2	3	4	5	6	7	8

Problem 2: [12 pts] Consider the following cost-reducing design options for the MIPS implementation shown below. Performance is still important, and a compiler will be able to optimize for this lower-cost implementation, but even with skillful optimization there may be some performance loss.



(*a*) Suppose one had to choose between eliminating all upper-ALU-input bypass paths or all lower-ALU-input bypass paths. Which would you choose? *Note: The following sentence did not appear on the original exam.* (Only bypass paths are eliminated, other mux inputs remain.)

☐ Eliminate: upper or lower. (Circle one.)

☐ Briefly explain why.

(*b*) Suppose one had to choose between eliminating all upper-ALU-input bypass paths or all rtv (not to the ALU) bypass paths. Which would you choose (Note: Only bypass paths are eliminated, other mux inputs remain.)

☐ Eliminate: upper or rtv. (Circle one.)

☐ Briefly explain why.

(*c*) Suppose that the mux in the WB stage was moved to the ME stage.

☐ How would that reduce cost?

☐ How might that affect performance?

Problem 3: [14 pts] The branch delay slot ISA feature eliminates the need for the stall or squash following a branch that occurs in implementations such as our 5-stage pipeline.

(*a*) The two MIPS code fragments below do not exactly make a strong case for delay slots. In both cases there is no squash or stall, which sounds good. For each code fragment indicate whether performance is slower, equal to, or faster than corresponding non-delay-slot ISA code on a corresponding implementation (one that will suffer a squash after every branch). If faster indicate if the improvement is full (based on the eliminated squash) or partial. *Hint: it won't be full.*

```
# -------------------- Code Fragment 1 --------------------
beq r1, r2, TARG
nop
lw r3, 0(r1)
```

☐ Delay slot in code above results in: slower, equal, partial improvement, full improvement. (CIRCLE ONE).

☐ Explain.

```
# -------------------- Code Fragment 2 --------------------
beq r1, r2 SKIP
add r9, r4, r5
or r6, r9, r8
SKIP:
sub r3, r6, r5
lw r9, 0(r3)
```

☐ Delay slot in code above results in: slower, equal, partial improvement, full improvement. (CIRCLE ONE).

☐ Explain.

(*b*) How can Code Fragment 2 (above) be improved by profiling? Show the optimized fragment and how profiling led to the optimized fragment.

☐ Optimized version of Code Fragment 2

☐ Specifically, how did profiling help?

Problem 4: [10 pts] The SPECcpu2006 integer suite has 12 benchmarks. Suppose instead it included only four benchmarks, but those benchmarks were chosen fairly, given that only four could be chosen.

(*a*) Considering just the base score (or ignoring the base/peak distinction altogether), what is the disadvantage of having just four benchmarks?

☐ Disadvantage of having four benchmarks.

(*b*) Suppose that with this smaller set of benchmarks the difference between the base and peak scores was smaller than if 12 benchmarks were used. Assume that in both cases the testers were skilled and followed the rules. Also assume that the base score values were about the same with 4 or 12 benchmarks.

☐ Give a reason for the smaller difference that has to do with the different rules for base and peak tests.

☐ Does the smaller difference indicate that base and peak are not measuring what they should?

Problem 5: Answer the questions below.

(*a*) [14 pts] Write the following MIPS code fragments with the minimum number of instructions. If you don't know the exact mnemonic, such as `mtc1` or `cvt.w.f`, make up something that sounds right.

☐ Put constant `0x12345678` into register `r1`.

☐ Jump to address `0x72345678` from address `0x1000`.

☐ Jump to address `0x72345678` from address `0x1000` using a SPARC-like `jmpl` (but in MIPS, like Homework 2).

☐ Registers `r1` and `r2` contain integers. Write register `f4` with the sum as a single-precision floating point number.

(*b*) [10 pts] Typical CISC ISAs have a range of immediate sizes for use in ALU instructions, up to the data size limit. (Say, 8, 16, 24, and 32 bits). MIPS, and other RISC ISAs, have just one immediate size for ALU instructions.

☐ Why can't MIPS have a 32-bit immediate?

☐ Why can a CISC ISA have a 32-bit immediate?

☐ Why would there be no benefit for MIPS to have both 8- and 16-bit immediate sizes for arithmetic instructions?

☐ What is the benefit for CISC ISAs in having 8-, 16-bit, (and more) immediate sizes for arithmetic instructions?