Name _____

Computer Architecture
EE 4720
Midterm Examination

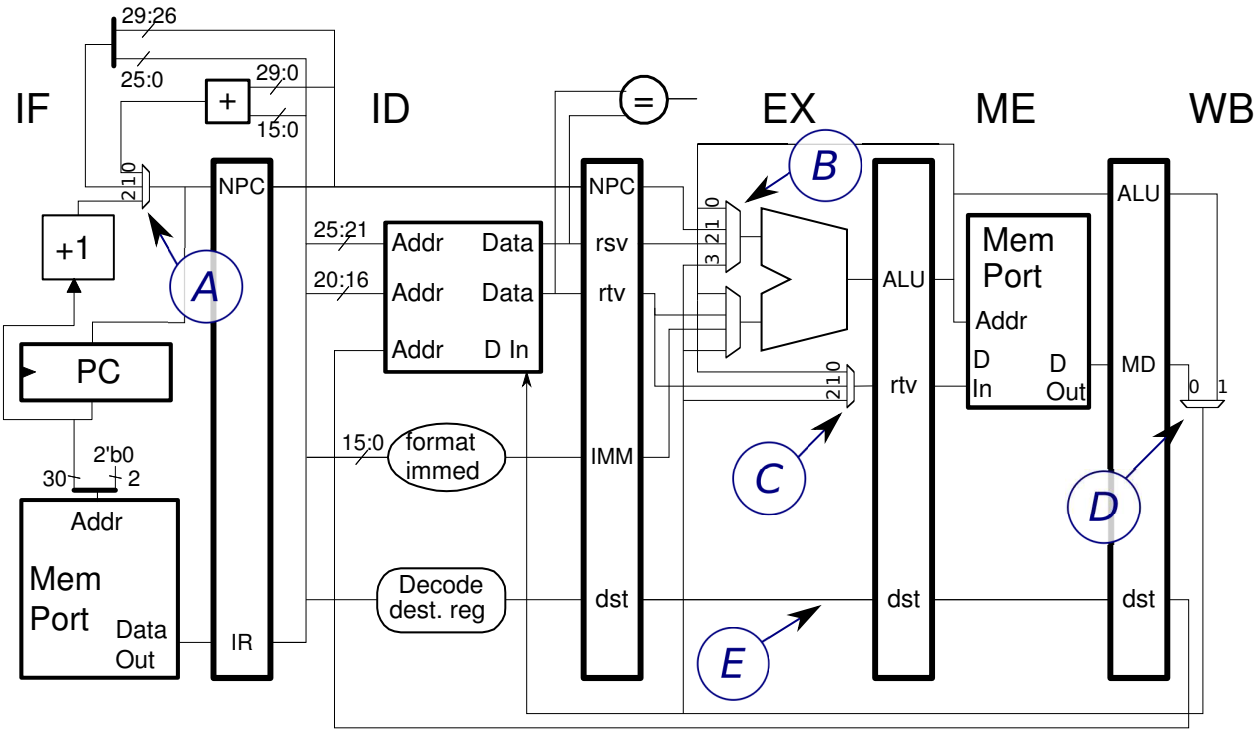
Monday, 31 March 2014, 9:30–10:20 CDT

Problem 1	_____	(20 pts)
Problem 2	_____	(20 pts)
Problem 3	_____	(18 pts)
Problem 4	_____	(12 pts)
Problem 5	_____	(6 pts)
Problem 6	_____	(12 pts)
Problem 7	_____	(12 pts)
Exam Total	_____	(100 pts)

Alias _____

Good Luck!

Problem 1: [20 pts] The MIPS code fragment below executes on our familiar five-stage scalar MIPS implementation.



(*a*) Show the execution of the code for enough iterations to determine CPI, and determine the CPI assuming a large number of iterations.

(*b*) There are five labels in the diagram, labels A through D point at multiplexor control inputs and label E points at the output of the `EX.dst` pipeline latch. Show the values of these signals up until the second execution of `lw r1`. For A show only values $\neq 2$, for E show only values $\neq 0$, for the others only show values at cycles in which the multiplexor is in use.

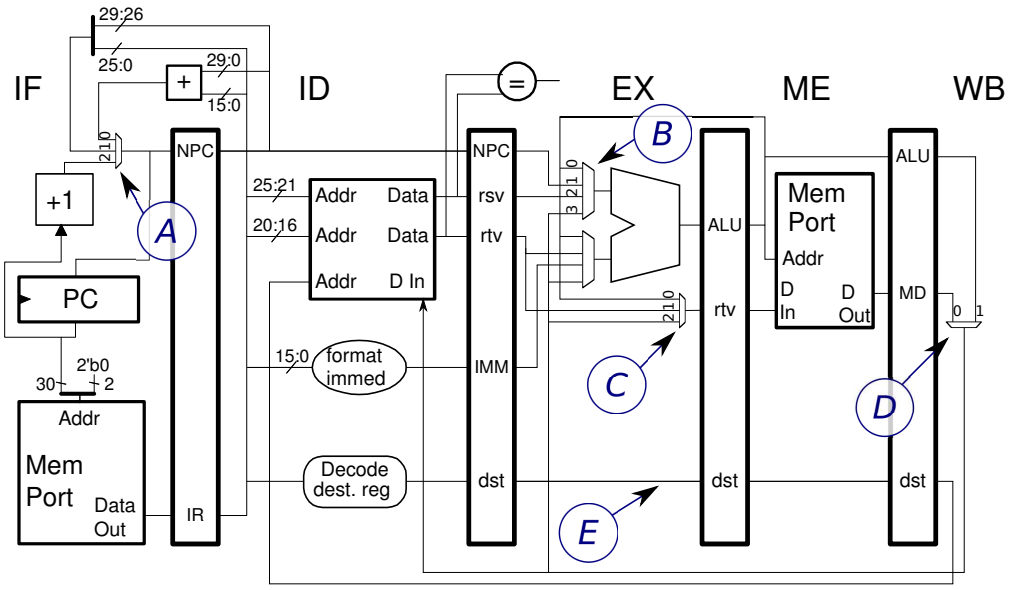
USE NEXT PAGE FOR SOLUTION

```
LOOP: Cycle        0
 A                 2
 B
 C
 D
 E                 0
 lw r2, 0(r5)      IF
LOOP:
 lw r1, 0(r2)
 sh r1, 16(r2)
 bne r1, r0, LOOP
 add r2, r2, r3
 xor r4, r5, r2
```

USE NEXT PAGE FOR SOLUTION

Problem 1, continued:

- ☐ Check code for dependencies.
- ☐ Show pipeline execution diagram for enough iterations to determine CPI.
- ☐ Determine the CPI.
- ☐ Show values for A through E.

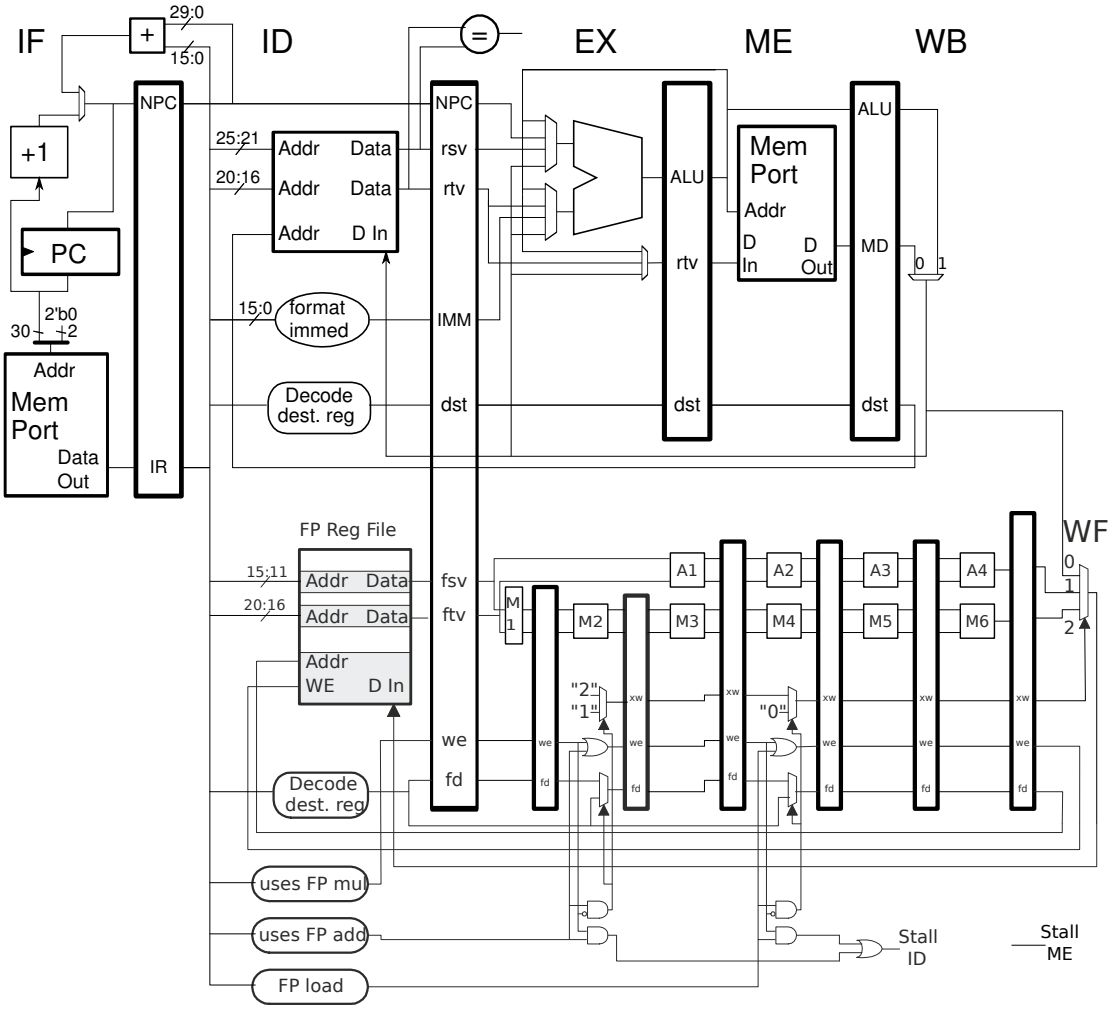



```
LOOP: Cycle        0
 A                 2
 B
 C
 D
 E                 0
 lw r2, 0(r5)      IF
LOOP:
 lw r1, 0(r2)
 sh r1, 16(r2)
 bne r1, r0, LOOP
 add r2, r2, r3
 xor r4, r5, r2
```

Problem 2: [20 pts] The pipeline execution diagram below shows the *incorrect* execution of the MIPS code for the illustrated pipeline. That is, in the pipeline below the stall to avoid the structural hazard would have occurred when `lwc1` was in the `ID` stage.

```
# Cycle              0  1  2  3  4  5  6  7
add.s f1, f2, f3     IF ID A1 A2 A3 A4 WF
addi r1, r1, 4          IF ID EX ME WB
lwc1  f4, 0(r1)            IF ID EX ME -> WF
```

USE NEXT PAGE FOR SOLUTION



USE NEXT PAGE FOR SOLUTION

As described below, modify the pipeline so that FP load instructions, such as `lwc1`, will stall when they are in `ME` rather than `ID` to avoid a `WF` structural hazard.

(*a*) Show logic to generate a signal `Stall ME` in the correct cycle. When `Stall ME` is logic `1` stages `ME` and earlier will stall.

(*b*) Cross out the logic generating the `Stall ID` for the `lwc1` that is no longer needed.

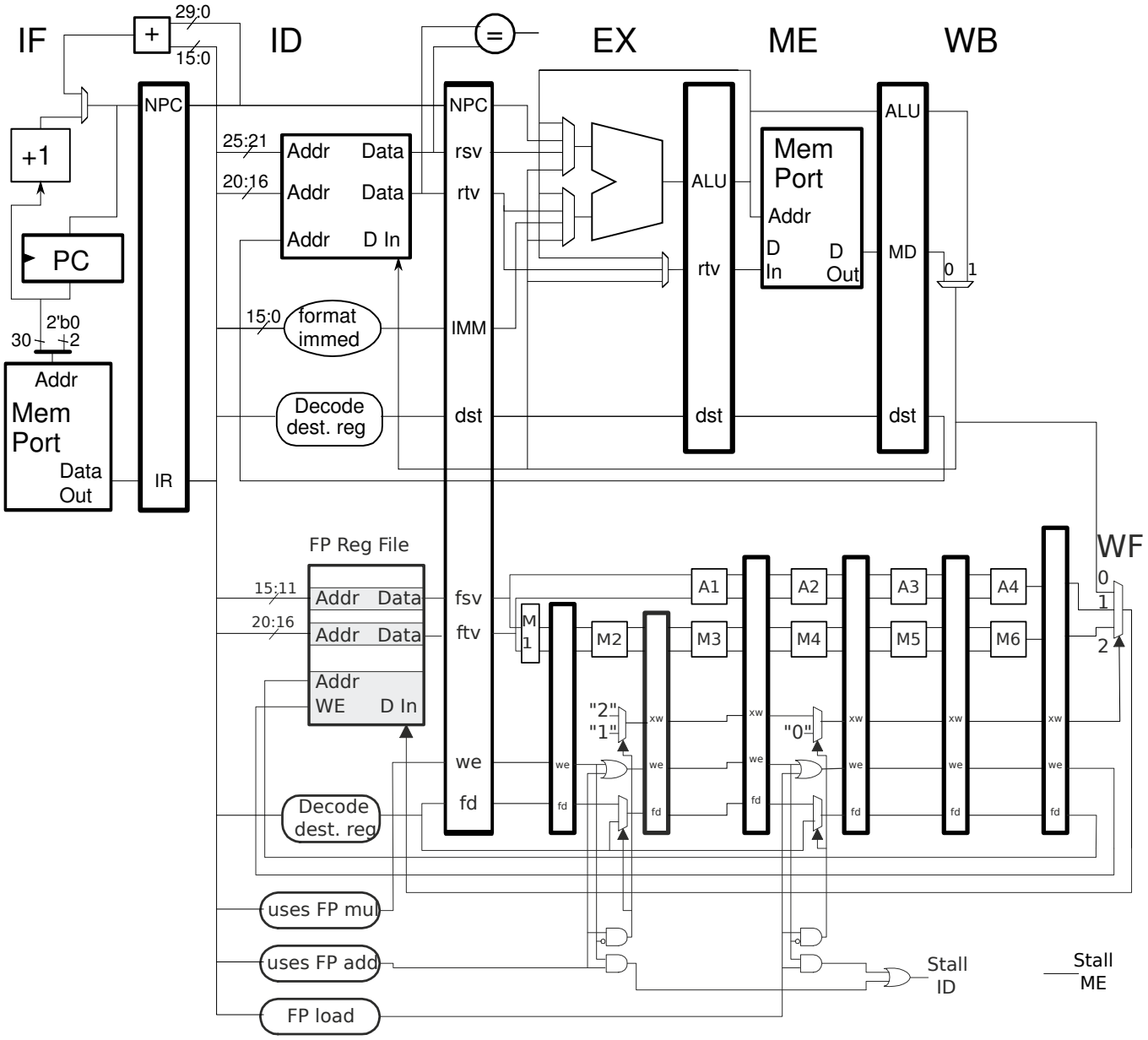
(*c*) Make other changes, including control logic, so that the correct destination register number, destination value, and write-enable signal are delivered to the FP register file. *Hint: Some solutions are simpler than others.*

Problem 2, continued:

☐ Generate `Stall ME` to avoid `lwc1 WF` structural hazard. ☐ Make sure signal is `1` in the right cycle.

☐ Cross out `Stall ID` logic for FP loads, but leave logic for other instructions.

☐ Make sure that correct `fd`, `we`, and value delivered to FP register file.



Problem 3: [18 pts] Answer each question below.

(*a*) Re-write the SPARC code fragment below in MIPS. Pay attention to the load instruction and the instructions related to the branch. Don't forget that in SPARC assembly language the last register is the destination.

```
!
 addcc %g1, %g2, %g3
 ld [%g5+%g3], %g4
 bg TARGET          ! Branch greater than zero.
 add %g5, 22, %g6   ! g6 = g5 + 22
```

☐ MIPS equivalent of SPARC code. ☐ Pay attention to load and branch-related insn.

(*b*) Show a MIPS equivalent of the ARM A32 code below. The MIPS code will use more than one instruction. (This is based on Homework 3.) *Note: A32 added in 2017 to avoid confusion with the A64 instruction set.*

```
 add r1, r2, r3, LSL #4
```

☐ MIPS equivalent of ARM code above.

(*c*) Explain why the MIPS code fragment below is certain to execute with an error.

```
 lw r1, 0(r2)
 lw r3, 1(r2)
```

☐ Error is certain because:

Problem 4: [12 pts] Answer each question below.

(*a*) In the execution below the **ant** instruction raises an illegal instruction exception and the handler is fetched, note that the handler starts execution in cycle 4. Explain why this exception (as executed) cannot be precise and show how execution should proceed for our five-stage pipeline.

```
# Cycle             0  1  2  3  4  5  6
 add r10, r11, r12  IF ID EX ME WB
 sw r3, 4(r5)          IF ID EX x
 ant r1, r2, r5           IF*ID*x
 or  r5, r6, r7              IF x

HANDLER:
 sw                             IF ID ..
```

☐ Reason why this execution not for a precise exception.

☐ Show execution that could be precise when ID stage discovers the bad **ant** opcode.

```
# Cycle
 add r10, r11, r12

 sw r3, 4(r5)

 ant r1, r2, r5

 or  r5, r6, r7

HANDLER:
 sw
```

(*b*) An interrupt mechanism will switch the processor from user mode to privileged mode when the handler starts. What is the difference between user and privileged mode and why are they necessary to implement an operating system?

☐ Difference between user and privileged mode? ☐ Importance for OS?

Problem 5: [6 pts] In class we described three broad families of ISAs: CISC, RISC, and VLIW.

(*a*) In which ISA family is it easy to have instructions with long (say, 32 bit) immediates? Explain how.

☐ ISA family with long immediates is:

☐ What about the ISA family enables this?

(*b*) In which ISA family are dependencies between instructions explicitly given? How is the dependence information supposed to help?

☐ ISA family in which dependence information is part of program is:

☐ Reason for providing dependence information.

(*c*) Which ISA family tends to have the most compact programs? (That is, the size in bytes of the program is smallest.) What makes them compact?

☐ ISA family with most compact programs is:

☐ Program sizes are small because.

Problem 6: [12 pts] Answer each question below.

(*a*) Consider a 2-way superscalar implementation and a 10-stage implementation, both derived from our 5-stage MIPS used in class. All implementations include reasonable bypass paths.

Explain how the instruction sequence below would always result in a stall on the 10-stage pipeline but might not result in a stall on the 2-way system. (The compiler cannot separate the two instructions.) Illustrate your answer with a pipeline execution diagram.

```
add r1, r2, r3
sub r4, r1, r5
```

☐ Reason for maybe stall on superscalar and certain stall on 10-stage.

☐ Illustrate using a pipeline diagram.

(*b*) Consider two MIPS implementation, one is an n-way superscalar of MIPS-I, with n sets of FP units, the other is a scalar implementation of a hypothetical MIPS-I-vec, which includes an n-lane vector unit for the vector instructions in MIPS-I-vec. Both the superscalar and vector MIPS implementations can sustain execution at n FP operations per cycle. *Note: The phrase "In terms of n" did not appear in the original exam.*

☐ In terms of n how does the cost of bypass paths compare in the two systems?

☐ In terms of n how does the cost of registers compare in the two systems?

Problem 7: [12 pts] Answer each question below.

(*a*) The SPECcpu benchmarks are designed to evaluate the CPU and memory system in new implementations and ISAs. To realize this testers are responsible for providing a compiler and for compiling the benchmarks.

Suppose SPEC required testers to use a compiler that they provide. How much would this impact the goal of testing new ISAs? New implementations of existing ISas?

☐ Degree of impact of SPEC-provided compilers for testing new ISAs. ☐ Explain.

☐ Degree of impact of SPEC-provided compilers for testing new implementations of old ISAs. ☐ Explain.

(*b*) A company releases a SPEC disclosure in which they have substituted the bzip2 benchmark with another version of bzip2 which does the same thing but runs faster. Since it does the same thing it provides the correct output to the SPEC verification script. As a result they get very good scores. How will they get caught? How is it against the goal of SPECcpu?

☐ How will the benchmark substitution get caught?

☐ How does the substitution undermine what SPECcpu is supposed to measure?