Name ______________________________

Computer Architecture
EE 4720
Midterm Examination

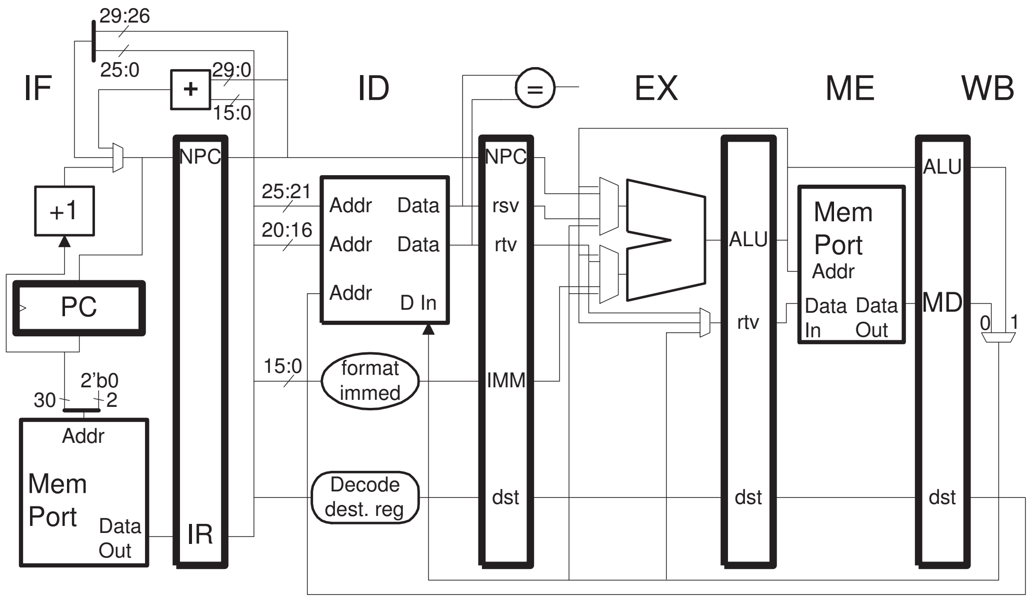
Wednesday, 30 March 2011, 9:40–10:30 CDT

Problem 1	______	(15 pts)
Problem 2	______	(20 pts)
Problem 3	______	(15 pts)
Problem 4	______	(10 pts)
Problem 5	______	(10 pts)
Problem 6	______	(15 pts)
Problem 7	______	(15 pts)
Exam Total	______	(100 pts)

Alias ______________________________

Good Luck!

Problem 1: [15 pts] The MIPS code below executes on the illustrated hopefully familiar implementation.



```
LOOP:
lh r1, 0(r2)

addi r4, r4, 4

andi r3, r1, 0xf0f0

bne r3, r0 LOOP

lw r2, 4(r2)
```

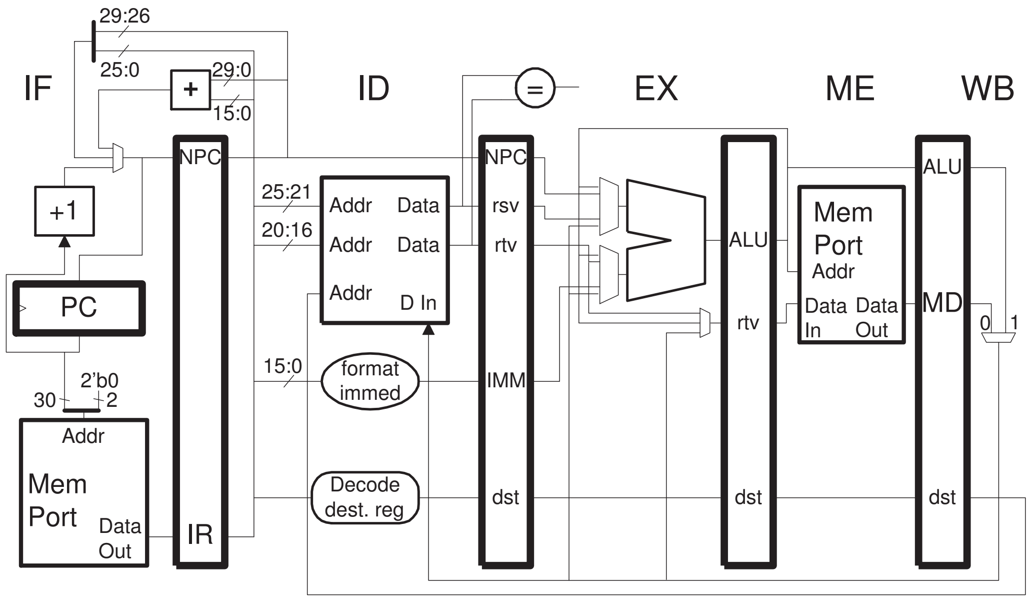
(*a*) Show a pipeline execution diagram for the code below running on the illustrated MIPS implementation for enough iterations to determine CPI.

☐ Show a pipeline execution diagram.

(*b*) Determine the CPI.

☐ Find the CPI.

Problem 2: [20 pts] In the implementation below the datapath for the `jalr` and `jr` instructions is not shown.



(*a*) Add datapath (but not control logic) needed for the `jr` instruction to the diagram above.

☐ Add datapath (but not control logic) for `jr`.

(*b*) Show the execution of the code below based upon the answer above. Show execution starting from the first instruction (as usual) and continue until `ori` executes or eight instructions have executed whichever is longer. (In the correct answer `ori` is the eighth instruction to execute.) Note that the `jalr` will jump to `ROUTINE` the first time it executes.

☐ Show execution consistent with previous answer.

```
 addi r1, r1, 16
 # r1 = ROUTINE
 jalr r31, r1

 addi r4, r31, -8

 ori r6, r6, 0xff

ROUTINE:
 addi r4, r4, 4

 jr r4

 lw r31, 0(r7)
```

(*c*) If your answer to the previous part was correct the code above would suffer stall(s). Add bypasses to avoid as many of the stalls as possible without significant critical path impact.

☐ Bypass paths to eliminate stalls.

Problem 3: [15 pts] In the code fragment below the `lw` raises an exception due to a TLB miss. Remember that a TLB miss is something that happens all the time to almost any load.

```
 # Cycle          0   1   2   3   4   5   6   7
 lw r1, 0(r2)     IF
 add r3, r5, r6
 xor r4, r3, r9
...
HANDLER:
 sw r10, ..
```

(*a*) Based on the exception hardware presented in class, at which cycle will the first instruction of the handler be in `IF`?

☐ Handler in `IF` in cycle:

For the next parts of this problem don't consider the exception hardware presented in class, instead the `lw` will trigger a deferred exception. That is, the `lw` will raise a TLB miss exception but `add` will successfully complete `WB` before the handler is called. Register `r1` will have some garbage value but the `add` will execute correctly.

(*b*) This is **not** a good example for why precise exceptions are needed for loads. Why not?

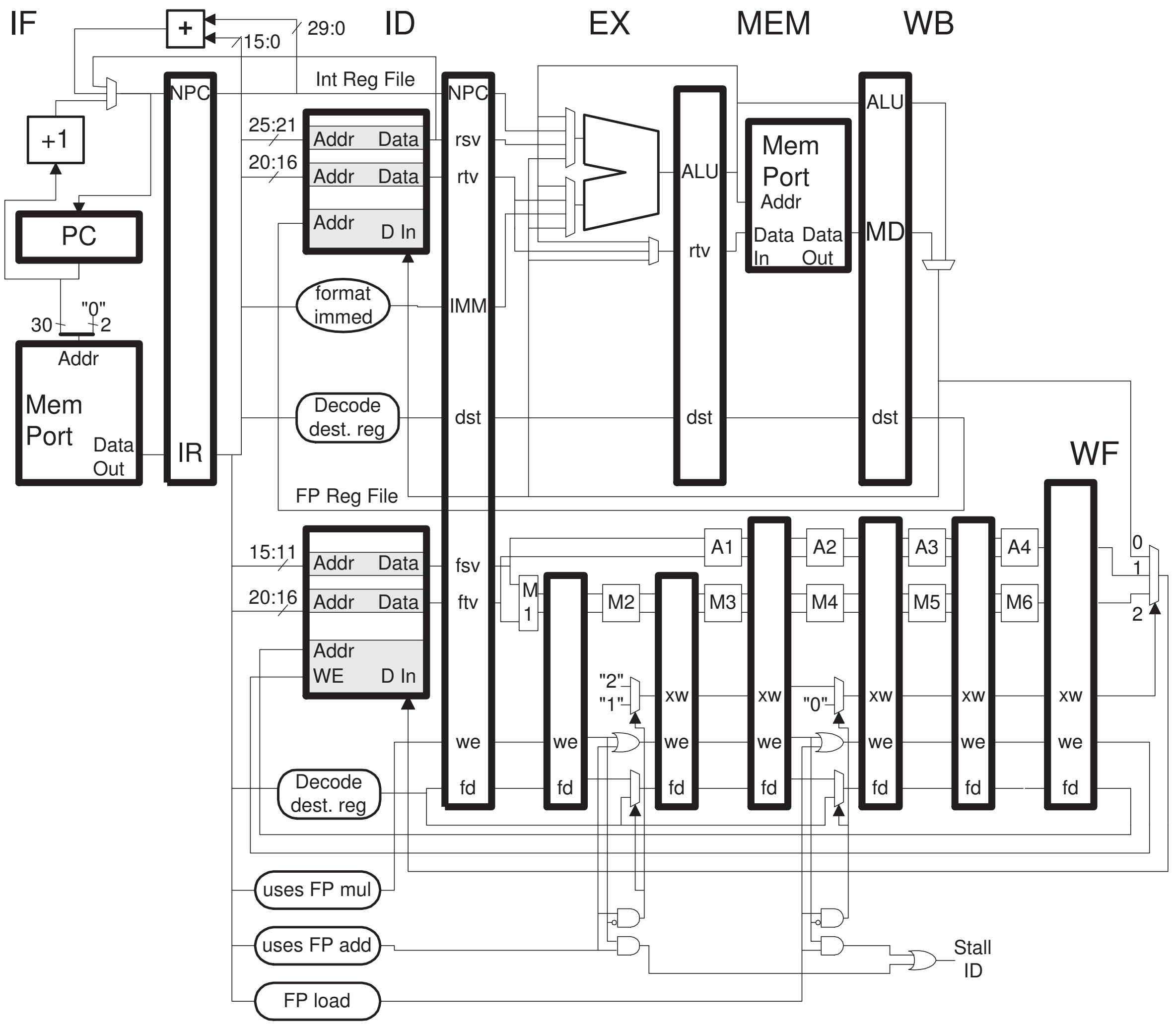
☐ Not a good example because:

(*c*) Modify the code after the `lw` so that this *is* a good example of why precise exceptions are needed. Briefly explain why.

☐ Modify code.

☐ Explain why it is a better example.

Problem 4: [10 pts] Show the execution of the MIPS code below on the illustrated implementation. Assume that all needed bypass paths are available.



☐ Execution of code.

```
add.d f2, f4, f6

sub.d f8, f4, f10

mul.d f12, f2, f14

addi r1, r1, 8
```

Problem 5: [10 pts] Answer each question about the SPECcpu suite.

(*a*) With SPECcpu the tester is responsible for providing compilation tools and choosing optimization switches. Which feature or goal of the SPEC benchmarks does that support?

☐ Compilation tools and optimization choices supports:

(*b*) Part of the SPEC rules requires that compilation tools be actively marketed as products, not just available to someone who knows exactly how to ask for them. What sort of abuse does that prevent?

☐ Actively marketed compilation tools prevents:

Problem 6: [15 pts] Answer each question below.

(*a*) ISAs are frequently updated, examples mentioned in class include MIPS I, MIPS II, SPARC v8, SPARC v9, and the multimedia extensions such as MMX, SSE, SSE 2, etc. for IA-32. With all these updates it sounds like an ISA is not really frozen for All Time. Does that mean the main advantage in separating ISA design from implementation design has been lost?

☐ Has main advantage of ISA/implementation separation been lost? Explain.

(*b*) Why is the typical RISC program larger than an equivalent CISC program?

☐ RISC larger than CISC because:

(*c*) Explain how the approaches to encoding immediate arithmetic instructions differ in the SPARC and MIPS ISAs.

☐ How approaches differ.

Problem 7: [15 pts] Answer each question below.

(*a*) Provide the following optimization examples:

☐ An optimization that can be performed well without knowing the particular implementation being targeted.

☐ An optimization that can be performed well only if the compiler knows the particular implementation being targeted.

(*b*) In an alternate universe, when designing the ALT-MIPS ISA, computer engineers insisted, perhaps for cost reasons, that the data memory port and ALU had to be in the same stage (that is, the `EX` and `ME` stages would be combined), resulting in a four-stage pipeline. How would the matching ALT-MIPS be different from our MIPS ISA? Note that elegance and performance are important for both ALT-MIPS and MIPS. Provide two code samples, one for which the four-stage ALT-MIPS is better and one for which the five-stage MIPS is better.

☐ ALT-MIPS differences with MIPS.

☐ Code sample better for ALT-MIPS.

☐ Explain.

☐ Code sample better for MIPS.

☐ Explain.