Name ______________________________

Computer Architecture
EE 4720
Midterm Examination

Wednesday, 30 March 2016, 9:30–10:20 CDT

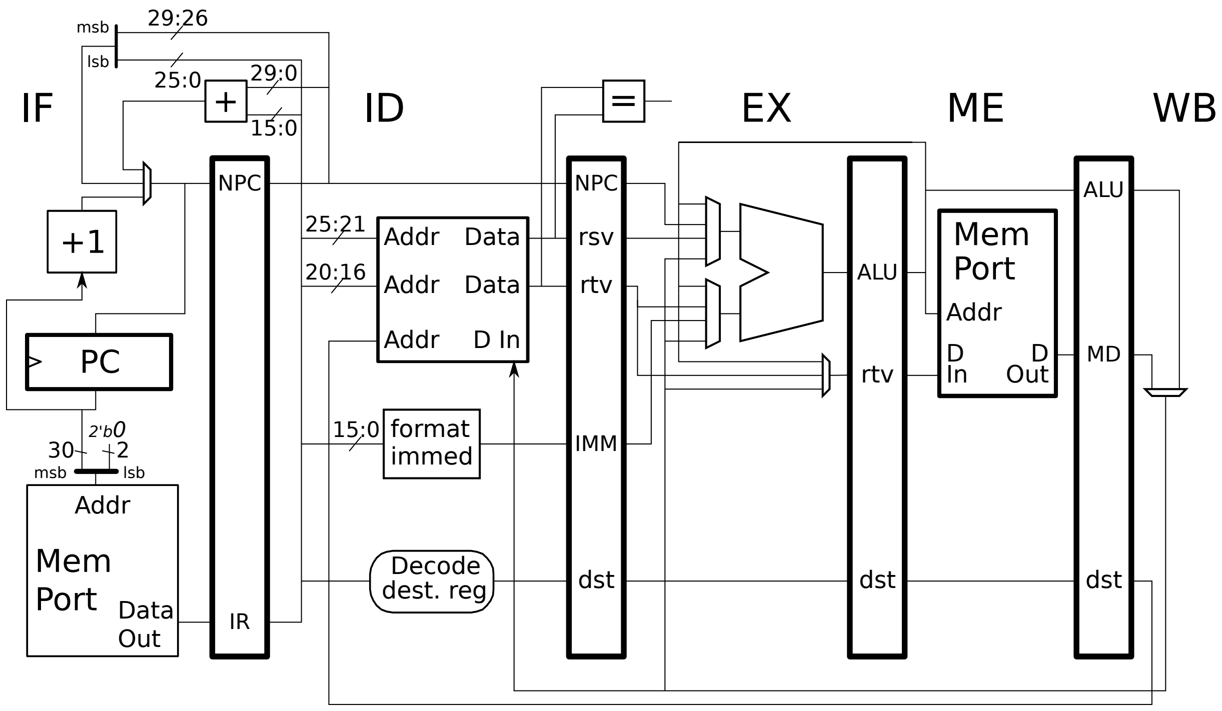
Problem 1	______	(25 pts)
Problem 2	______	(25 pts)
Problem 3	______	(12 pts)
Problem 4	______	(28 pts)
Problem 5	______	(10 pts)
Exam Total	______	(100 pts)

Alias ______________________________

Good Luck!

Problem 1: [25 pts] Appearing below are what are supposed to be pipeline execution diagrams (PEDs) of code fragments executing on the illustrated implementation. The PEDs are incorrect.

(*a*) Correct the PEDs.



☐ Correct the PED below.

```
add r1, r2, r3  IF ID EX ME WB
lw r3, 0(r1)       IF ID -> EX ME WB
```

☐ Correct the PED below.

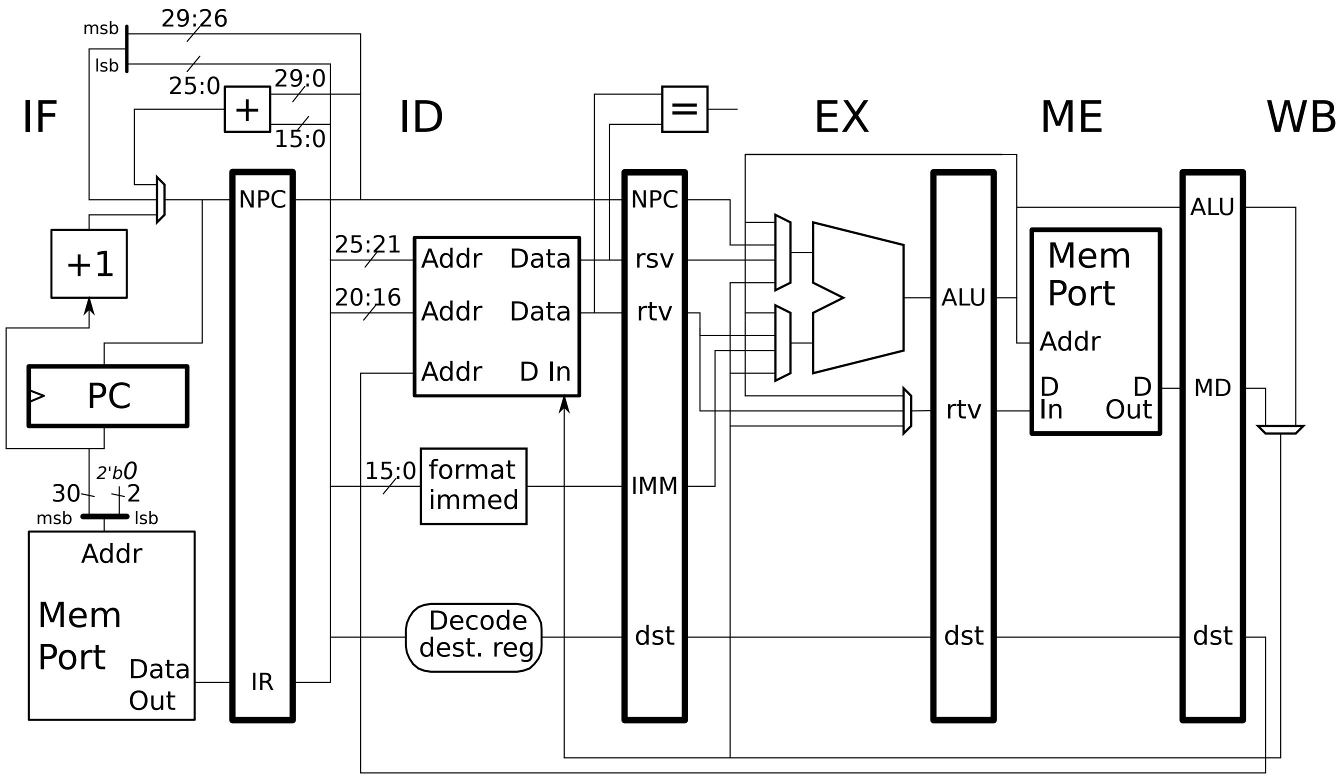
```
lw r3, 0(r1)    IF ID EX ME WB
add r4, r3, r5     IF ID -> EX ME WB
sub r6, r7, r8        IF ID EX ME WB
```

☐ Correct the PED below.

```
# Cycle            0  1  2  3  4  5  6  7
beq r1, r1  TARG   IF ID EX ME WB   # Branch is taken.
xor r5, r6, r7        IF IDx
add r8, r9, r10          IFx
TARG:
sub r2, r3, r4              IF ID EX ME WB
# Cycle            0  1  2  3  4  5  6  7
```

(*b*) Appearing below are more PEDs which are not correct for the illustrated implementation. This time **modify the implementation** so that the executions are correct. Only make necessary changes.

- Delete a bypass path by showing an × at the **mux input** where it ends.
- Do not delete or add more hardware than is necessary.



☐ Modify the implementation so that the execution below is correct.

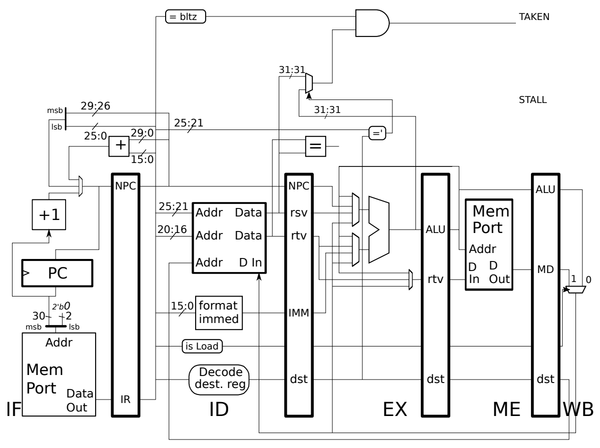
```
add r1, r2, r3  IF ID EX ME WB
sub r3, r1, r5     IF ID ----> EX ME WB
```

☐ Modify the implementation so that the execution below is correct.

```
lw r1, 0(r2)  IF ID EX ME WB
sw r1, 0(r3)     IF ID EX ME WB
```

Problem 2: [25 pts] The implementation below is based on the solution to Homework 2 Problem 2 in which a bypass was added for `bltz` instructions.

Use Next Page for Solution



Use Next Page for Solution

(*a*) The implementation can only bypass values in `EX` to a `bltz`. Modify the implementation on the next page so that values can be bypassed from both `EX` and `ME`. With these changes the two fragments below should run without a stall and of course bypass the correct value.

```
# Example 1
add r1, r2, r3
sub r4, r5, r6
bltz r1, TARG

# Example 2.
add r1, r2, r3
sub r1, r1, r6
bltz r1, TARG
```

(*b*) A bypass from EX isn't possible for the code fragment below, and a bypass from `ME` is problematic too. On the next page add logic to generate a stall signal for these situations (load/`bltz` dependencies) and connect it to the word `STALL` in the upper-right of the diagram. Notice that there is an $\boxed{\text{is Load}}$ logic block in `ID`.

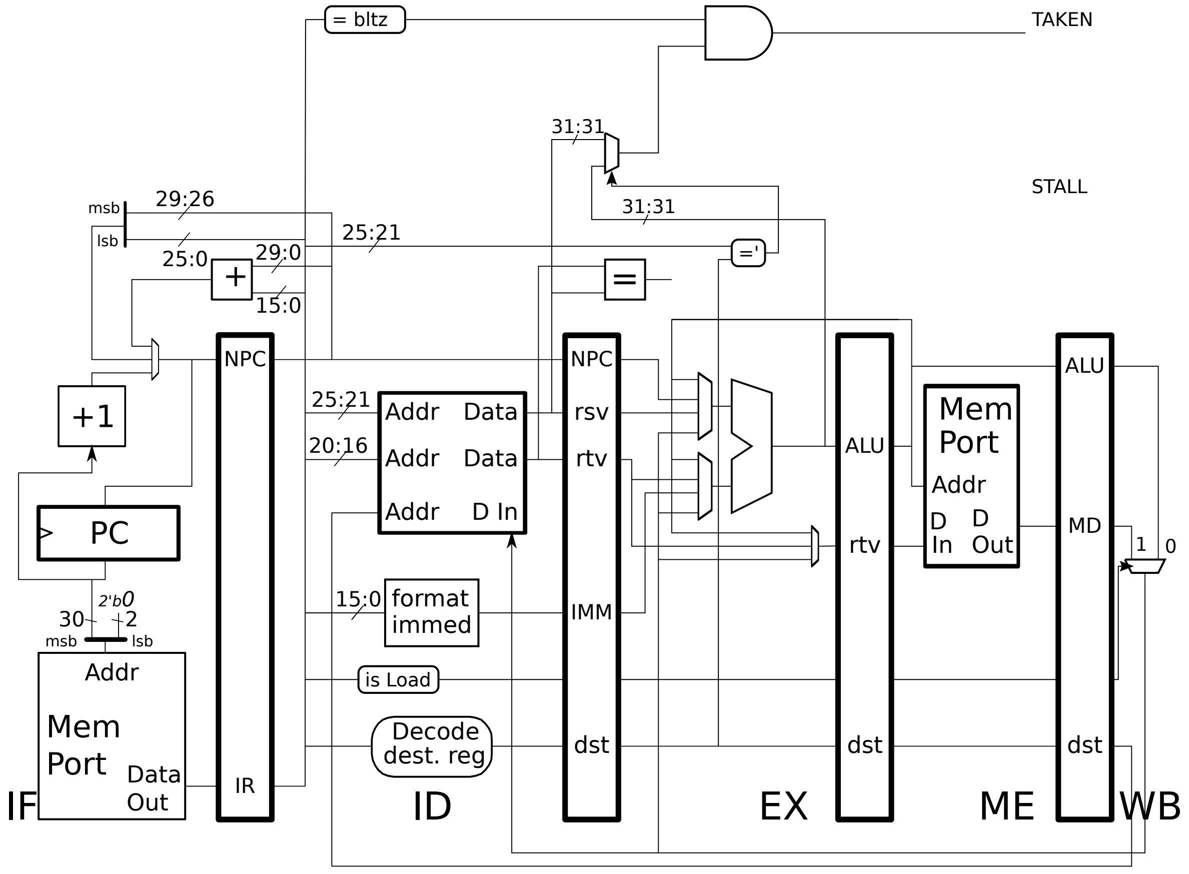
```
lw r1, 0(r2)
bltz r1, TARG
```

(*c*) Explain why it would not be a good idea to bypass the load value to the `bltz` when the load is in `ME`.

☐ Bypassing load from `ME` not a good idea because:

Problem 2, continued:

- ☐ Modify implementation so `bltz` can bypass from `EX` and `ME`.
- ☐ Logic to generate stall signal for `bltz` dependent on load.
- ☐ Answer part c.



Problem 3: [12 pts] Answer each question below.

(*a*) Each code fragment below writes register `f30` with the sum `f2 + 4720`.

```
# Plan A
        addi $t0, $0, 4720
        mtc1 $t0, $f17
        cvt.s.w $f16, $f17
        add.s $f30, $f2, $f16


# Plan B
        lui $t0, 0x4593
        ori $t0, $t0, 0x8000
        mtc1 $t0, $f16
        add.s $f30, $f2, $f16
```

☐ What is the difference between `mtc1` and `cvt`?

☐ Why doesn't Plan B need a `cvt`?

(*b*) All MIPS integer instructions have their source register numbers in the `rs` and, if needed, `rt` fields. But the destination register number can be found in either the `rt` or `rd` fields.

☐ How does limiting integer sources to `rs` and `rt` reduce cost and improve performance?

☐ Why isn't performance hurt by having the destination in either `rt` or `rd`?

Problem 4: [28 pts] Answer each question below.

(*a*) The statement below omits an important reason why customers can be kept by companies that manage an ISA and implementation as two different things.

> *By separating the ISA from the implementation we can keep our customers by offering them a faster implementation when they are ready to buy a new system.*

☐ What is the important reason that has been omitted?

(*b*) To use profiling to improve performance a program is compiled twice.

☐ What is done between the first and second compilation?

☐ Why does the program need to be compiled a second time?

☐ Suppose that taken branches have a penalty. Show how profiling helps.

Problem 4, continued:

(*c*) Consider an instruction such as `add (r1), r2, 4(r3)`. What about it makes it unsuitable for a RISC ISA? Explain why it would be difficult to implement in our pipelined design.

☐ `add (r1), r2, 4(r3)` unsuitable for RISC because:

☐ It would be difficult to implement because:

(*d*) When we compared the un-optimized and optimized versions of the π program we found that the optimized version had many fewer load and store instructions. Why?

☐ The optimized π program had fewer loads and stores because:

(*e*) A tester preparing a run of the SPECcpu suite is responsible for compiling the benchmarks. Why does that make SPECcpu results interesting to computer engineers?

☐ Tester compilation makes SPECcpu interesting to computer engineers because:

Problem 5: [10 pts] Answer the following questions about bypass paths.

(*a*) Consider the two statements below about bypasses in implementations like our five-stage MIPS **running typical programs**.

A: Compiler scheduling makes bypass paths unnecessary.

☐ Explain why the statement above is wrong.

B: Bypass paths make compiler scheduling unnecessary.

☐ Explain why the statement above is wrong.

(*b*) Consider the two above statements (about bypass paths) again as it applies to our MIPS implementation, but this time **running a special set of programs**. We plan to design an implementation for this set of programs. For these programs the two statements are true! *Note: The original exam did not mention the new implementation, and it had an "or both" option below.*

☐ For such programs should we eliminate bypass paths or should we eliminate compiler scheduling?

☐ Explain.