Name ______________________________

Computer Architecture
EE 4720
Final Examination

9 December 2008, 17:30–19:30 CST

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

Alias ______________________________

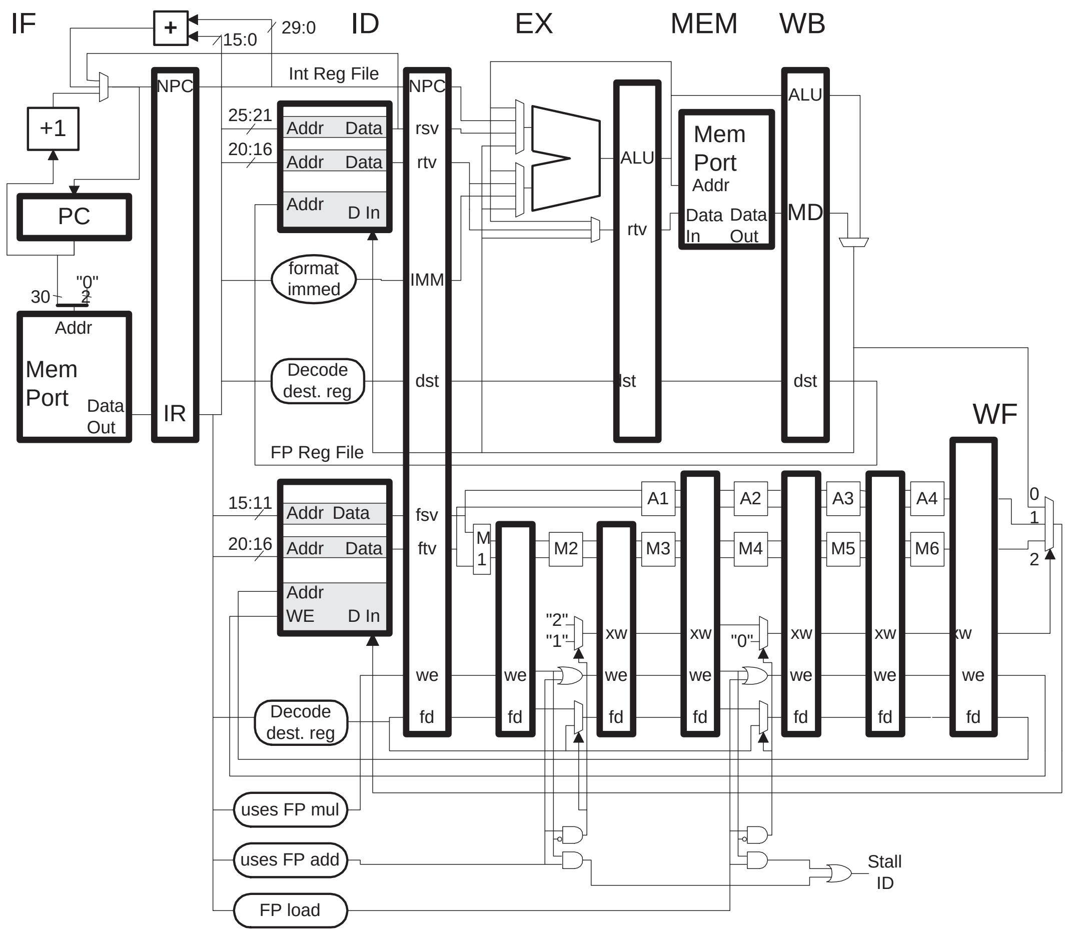
Good Luck!

Problem 1: (10 pts) Consider a new floating point instruction `nin.d` which uses a 5-stage computation unit, N1 - N5 (in contrast to FP add's A1-A4). The format and register usage for `nin.d` is the same as the other FP arithmetic instructions. Modify the implementation below so that it can execute `nin.d`. *Hint: This is an easy question, `nin.d` is just like the FP add and multiply, except it's 5 stages.*

☐ Add the datapath components, including the functional unit stages (N1 to N5).

☐ Add control logic for proper write-float and structural hazard detection (as is already present for add and multiply). Be sure not to break existing instructions.

☐ The changes must fit in efficiently with what is already present.



Problem 2: (15 pts) With the MIPS implementation illustrated below floating-point add and multiply instructions do not raise precise exceptions. If a programmer needs a precise exception for a particular FP instruction he or she can follow it with a `test` (test for exception) instruction. In the code below `test` is used to provide a precise exception for `mul.d`.

```
mul.d  f2, f2, f6
test
sub.d  f16, f14, f20
```

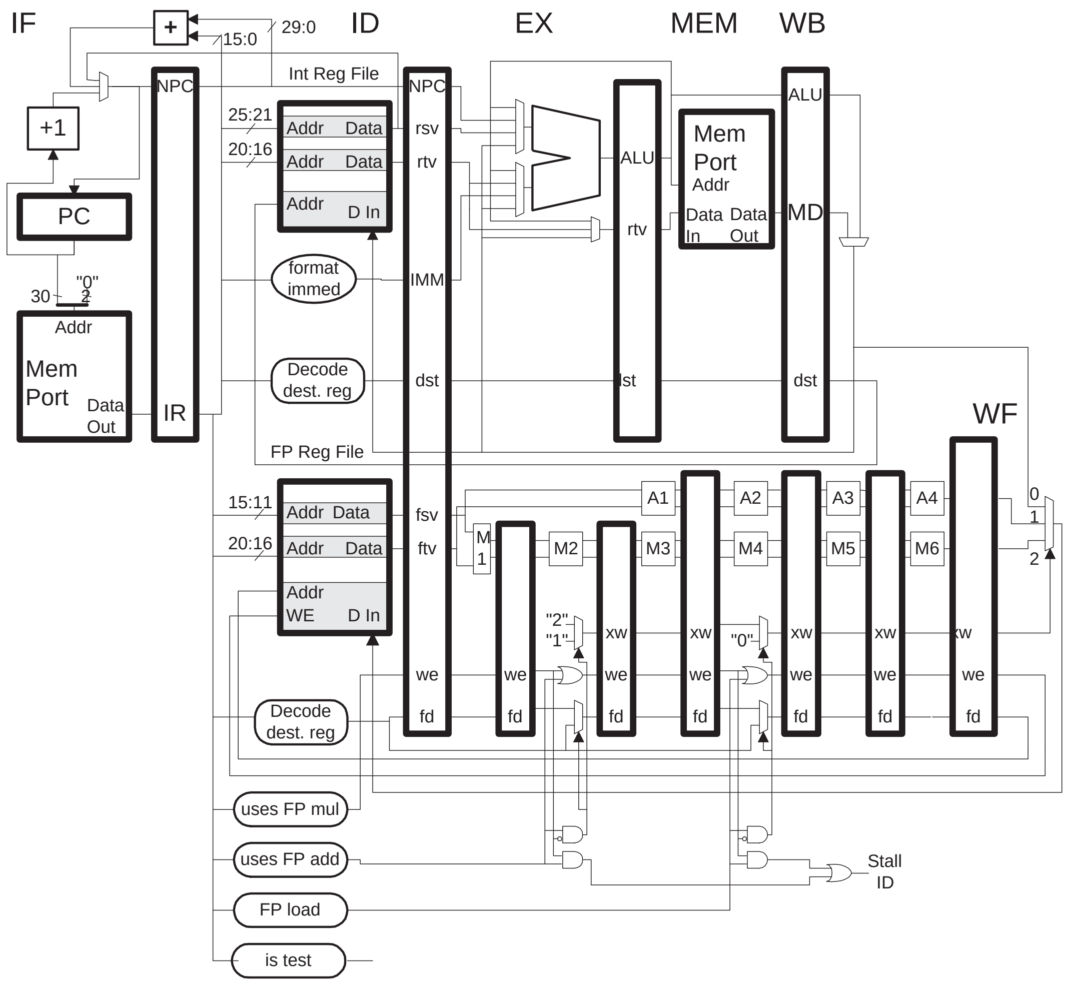
The `test` instruction will stall in ID for **the minimum number of cycles necessary** to ensure a precise exception and will suppress a WF if necessary.

(*a*) Add hardware to implement the `test` instruction. The output of $\boxed{\text{is test}}$ is `1` if a `test` instruction is in `ID`. Assume there are A4 and M6 outputs that indicate whether an exception was raised. These can be checked in the middle of the cycle.

☐ Add control logic to stall the pipeline using a new input to the *Stall ID* or gate. *Hint: Very little logic is needed.*

☐ Add logic to suppress WF when necessary.

☐ The hardware should work correctly for floating-point multiplies and adds.



Problem 2, continued:

(*b*) Show an example in which, in a faulty implementation, the test instruction stalls one less than the minimum number of cycles and as a result an exception is not precise.

☐ Example code and pipeline execution diagram, including fetch of first handler instruction.

☐ Explain why the exception is not precise.

Problem 3: (20 pts) The code below has two branches, branch `B1` implements a 3-iteration loop with outcome pattern `T T N T T N`⋯. The outcome pattern for branch `B2` is shown below. **Note that `B1` executes three times for each execution of `B2`.**

The code runs on three systems, the branch predictor on all systems uses a 2^{14}-entry BHT. One system has a bimodal predictor, one system uses a local history predictor with a 10-outcome local history, and one system uses a global predictor with a 10-outcome global history.

```
BIGLOOP:    Note: B1 and B2 are the only branches in this code.
T1:
B1:  bne r3, r4 T1   .. 3 iteration loop ..

# .. no CTIs ..
B2:  beq r1, r2 T2   N N N T N N T N N N T N N T N N N T N N T N N N T N N T ...
T2:

j BIGLOOP
```

☐ What is the accuracy of the bimodal predictor on branch `B2` after warmup?

☐ What is the accuracy of the local predictor on branch `B2` after warmup?

☐ What is the size of the BHT and PHT, in bits, needed to implement the local predictor? Only take into account the storage need for the branch predictor, omit things like CTI type.

☐ What is the minimum local history size needed to achieve 100% accuracy on branch `B2` using the local predictor? Explain.

☐ What is the minimum global history size needed to achieve 100% accuracy on branch `B2` using the global predictor? Explain.

Problem 4: (15 pts) Consider two alternatives for improving the performance of our familiar scalar 5-stage implementation: an n-way superscalar implementation or a $5n$-stage superpipelined implementation.

(a) Both systems can *potentially* reduce execution time by a factor of n. Explain how this is achieved for each system in terms of CPI (cycles per instructions), clock frequency, and other relevant factors. The answers might read, "Because of ... the CPI is x^2 times larger which causes ... but ... **and so overall execution is** n **times faster**."

☐ Explain how the superscalar system is potentially n times faster.

☐ Explain how the superpipelined system is potentially n times faster.

(b) Bypass paths add substantially to the cost of sufficiently large superscalar systems. Provide expressions in terms of n for the cost of bypass paths in both systems. Briefly justify your answers, using a diagram if necessary.

☐ Expression for the cost of bypass paths in superscalar systems, with quick diagram and brief description.

☐ Expression for the cost of bypass paths in superpipelined systems, with quick diagram and brief description.

(c) Ideally the $5n$-stage superpipelined system would have a speedup of n (the scalar system would take n times longer than the superpipelined system to run a program). Consider a program that has no nearby dependencies so that the superpipelined system does not have to stall.

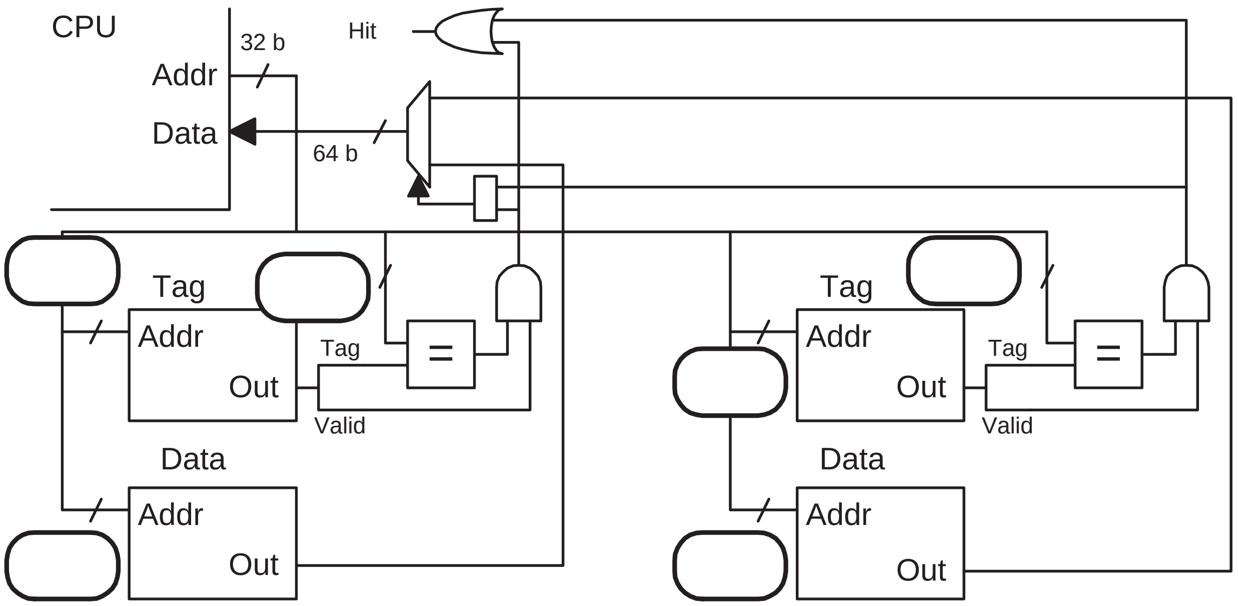
☐ Explain why the speedup of the superpipelined system might be $\frac{t_0+t_1}{t_0+\frac{t_1}{n}}$, where the clock frequency of our familiar scalar system is $\frac{1}{t_0+t_1}$.

☐ Explain what t_0 is likely to be.

Problem 5: (20 pts) The diagram below is for a 4-MiB (2^{22}-character) set-associative cache with a line size of 512 (2^9) characters, with the usual 8-bit characters.

(*a*) Answer the following, formulæ are fine as long as they consist of grade-time constants.

☐ Fill in the blanks in the diagram.



☐ Show the address bit categorization. Label the sections appropriately. (Alignment, Index, Offset, Tag.)

Address:

☐ Associativity:

☐ Memory Needed to Implement (Indicate Unit!!):

☐ Show the bit categorization for a direct mapped cache with the same capacity and line size.

Address:

Problem 5, continued:

(*b*) The code below runs on the cache from the first part of this problem. Initially the cache is empty; consider only accesses to the array.

☐ What is the hit ratio running the code below? Explain

```
double sum = 0.0;
short *a = 0x2000000;    // sizeof(short) = 2 characters.
int i;
int ILIMIT = 1 << 10;    // = 2^10

for (i=0; i<ILIMIT; i++) sum += a[ i ];
```

(*c*) The code below runs on a direct mapped (not set-associative) cache with a line size of 512 characters and a capacity of 4 MiB. *Grading note: The cache size was omitted from the original problem.*

☐ Determine an address for `b` that will result in a 0% hit ratio when running the code below.

☐ Briefly explain.

```
double sum = 0.0;
short *a = 0x2000000;    // sizeof(short) = 2 characters.
short *b =                                          <-- FILL IN
int i;
int ILIMIT = 1 << 10;    // = 2^10

for (i=0; i<ILIMIT; i++) sum += a[ i ] + b[ i ];
```

Problem 6: (20 pts) Answer each question below.

(*a*) Describe restrictions (or lack of) on instruction addresses and the placement of instructions that distinguish RISC, CISC, and VLIW ISAs.

☐ RISC address and placement restrictions.

☐ Reason for each restriction (if any).

☐ CISC address and placement restrictions.

☐ Reason for each restriction (if any).

☐ VLIW address and placement restrictions.

☐ Reason for each restriction (if any).

(*b*) A feature of the SPECcpu benchmarks is that they come with **source code** and it is the tester's responsibility to **compile** (and run) them. Note: "are these" in the questions below refers to source code and compiling.

☐ Are these necessary, important, or irrelevant for testing an implementation of a new ISA? (This is not a yes-or-no question.) Explain.

☐ Are these necessary, important, or irrelevant for testing a new implementation of an existing ISA? Explain.

(*c*) A corrupt member of the committee choosing benchmarks for the next SPECcpu suite has successfully bribed the other committee members so that the benchmarks that were selected favor the corrupt member's company.

☐ How might this be discovered? Indicate who would discover the problem and what public information draw their suspicion?

☐ Can we expect those involved to be sufficiently motivated to correct the situation? Explain.