Name _____

Computer Architecture
EE 4720
Final Examination

14 December 2006, 17:30–19:30 CST

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

Alias _____

Good Luck!

Problem 1: In the MIPS implementation on the next page some wires are labeled with cycle numbers and corresponding values. For example, c3:1 indicates that at cycle 3 the pointed-to wire will hold a 1. 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. The ALU label shows the arithmetic operation performed at the indicated cycle.(20 pts)

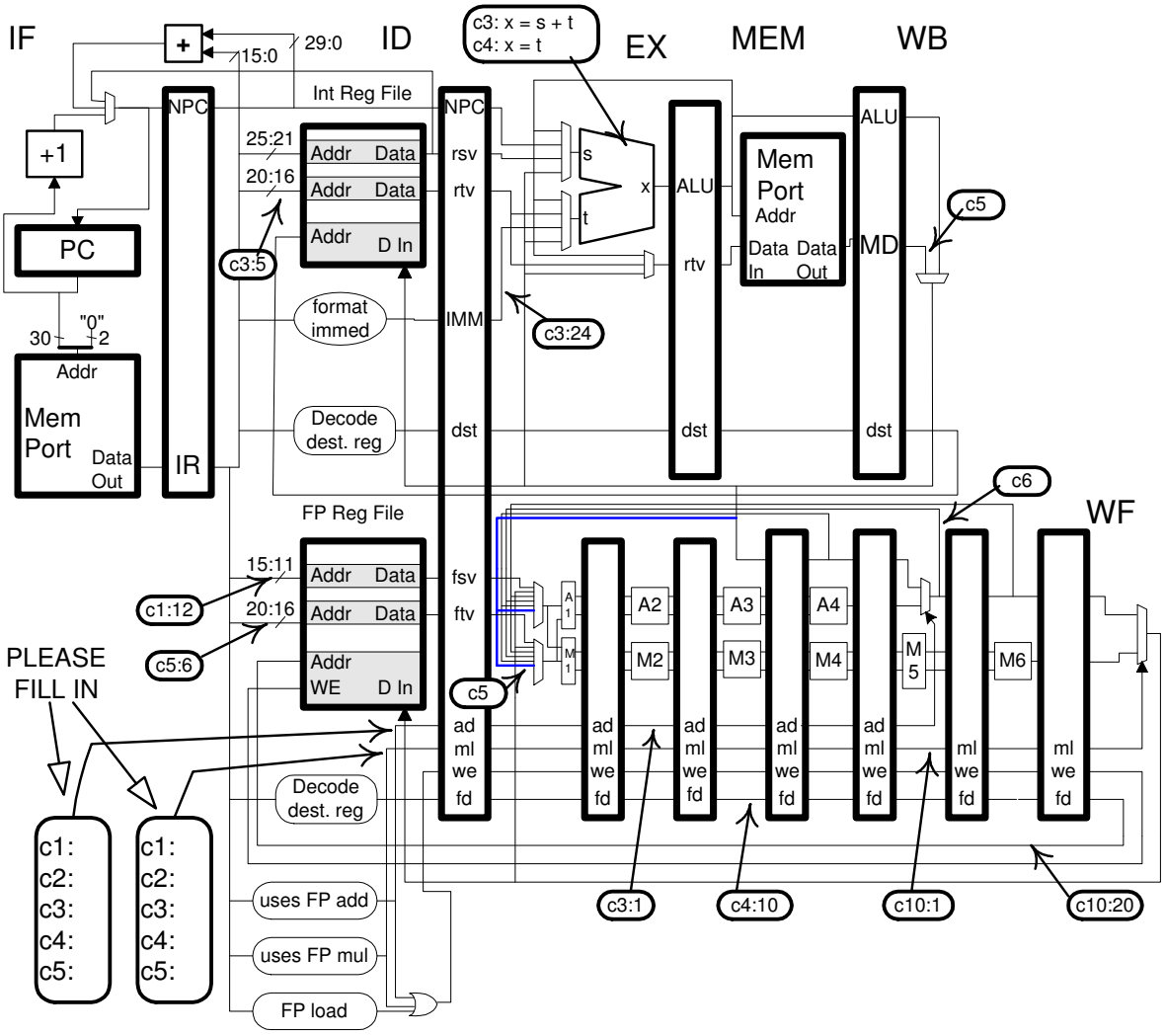
- There are no branches or other control-transfer instructions.
- There are no stalls.
- Every instruction writes the floating-point register file.
- Some instruction(s) read the integer register file.
- One instruction has only briefly been covered, make up a reasonable name for it if you don't remember it.

☐ Write a program consistent with these labels.

☐ Some registers can be determined exactly, others must be made up. **Use as many different register numbers as possible** while still being consistent with the labels.

☐ Fill in the block in the lower-left of the diagram.

Problem 1, continued:




```
Cycle:  0   1   2   3   4   5   6   7   8   9   10  11  12
```

```
                    IF  ID  _1  _2  _3  _4  _5  _6  WF
```

```
                        IF  ID  _1  _2  _3  _4  _5  _6  WF
```

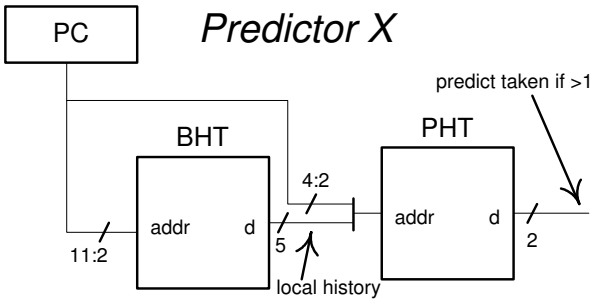
```
                            IF  ID  _1  _2  _3  _4  _5  _6  WF
```

```
                                IF  ID  _1  _2  _3  _4  _5  _6  WF
```

```
                                    IF  ID  _1  _2  _3  _4  _5  _6  WF
```

```
Cycle:  0   1   2   3   4   5   6   7   8   9   10  11  12
```

Problem 2: The code below runs on three systems which are identical except for the branch predictors. One system uses a bimodal predictor with a 1024-entry BHT, one uses a local history predictor with a 1024-entry BHT and a five-outcome history, and one uses Predictor X, illustrated below. (The PHT input is a concatenation of the local history and three branch PC bits.) The code below has two branches, B1 and B2, which execute in a repeating pattern as shown. There are no other branches in the code.(20 pts)



Note: In the original problem the input to Predictor X used an exclusive or rather than a concatenation.

```
LOOP:
 ..
0x1000: B1: bne r1,r2 SKIP1   N N N T T N N N T T N N N T T N N N T T ...
 ..
SKIP1:
 ..
0x1124: B2: bne r3,r4, SKIP2   N N N T T T N N N T T T N N N T T T ...
 ..
SKIP2
..
     j LOOP
```

☐ What is the accuracy of the bimodal predictor on branch B1 after warmup?

Note: In the original exam the questions below asked about B1 rather than B2.

☐ What is the accuracy of the local predictor on branch B2 after warmup? (Do not ignore branch B1 when answering this part.)

☐ What is the minimum local history size for the local predictor to achieve 100% accuracy on branch B2 (without ignoring B1)?

☐ What is the accuracy of Predictor X on branch B2?

☐ What is the minimum local history size for the Predictor X to achieve 100% accuracy on branch B2 (without ignoring B1)?

☐ Explain why predictor X has lower or higher accuracy than the local predictor on branch B2.

☐ As indicated above, B1 is at address `0x1000` and B2 is at address `0x1124`. How would different branch addresses affect the answers above?

Problem 3: Consider the execution of MIPS code below. The code follows a large number of `nop` instructions. As can be seen the system below is statically scheduled. In the questions below "relatively simple" means simple compared to dynamic scheduling.

(15 pts)

```
# PC             Cycle:  0  1  2  3  4  5  6  7
0x1ff0: lh r1, 0(r2)     IF ID EX ME WB
0x1ff4: lw r4, 8(r2)     IF ID -> EX ME WB
0x1ff8: addi r6, r6, 1   IF ID -> EX ME WB
0x1ffc: xor r8, r9, r10  IF ID -> EX ME WB
0x2000: sub r11, r8, r13    IF -> ID EX ME WB
0x2004: and r14, r11, r16   IF -> ID -> EX ME WB
# PC             Cycle:  0  1  2  3  4  5  6  7
```

☐ What is the minimum fetch/decode width of a system that could produce that execution? (The x in x-way superscalar)

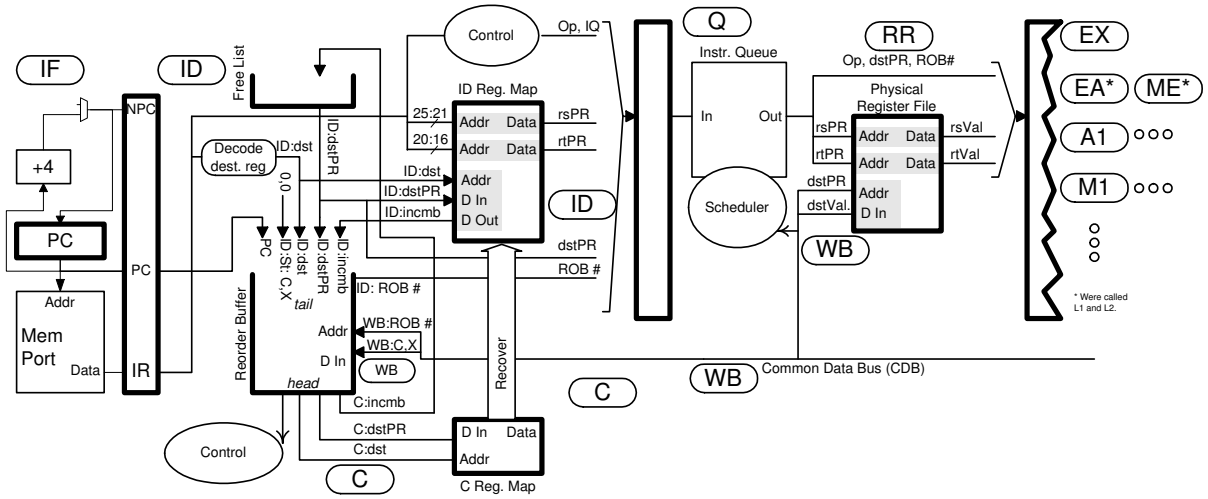
☐ Why is the fetch/decode width above a minimum and not an exact number?

☐ What caused the `and` to stall in cycle 4? Could the stall be avoided?

☐ What might have caused `lw` to stall? Suggest a relatively simple change to the implementation to avoid such stalls.

☐ What might have caused `addi` and `xor` to stall? *Note: The original question also asked for a "relatively simple solution."*

Problem 4: Consider the dynamically scheduled implementation below. (15 pts)



(*a*) Where is the logic for finding the (data) dependencies that requiring bypassing most likely to be?

☐ Indicate on diagram.

(*b*) Suppose due to a manufacturing error every entry in the ID register map is initialized to 12 after each reset, but the commit register map was properly initialized. Initial register values are not defined, so getting the wrong value for a register that was never written is not a problem here.

☐ Which code fragment below is more likely to encounter a problem? *Hint: It has something to do with the connection from the ID Register Map to the ROB.*

```
# Fragment A
 add r1, r2, r3
 add r2, r1, r5
 add r1, r6, r7
 add r2, r8, r9
 nop
 ..

# Fragment B
 add r0, r2, r3
 add r0, r1, r5
 nop
 ...
```

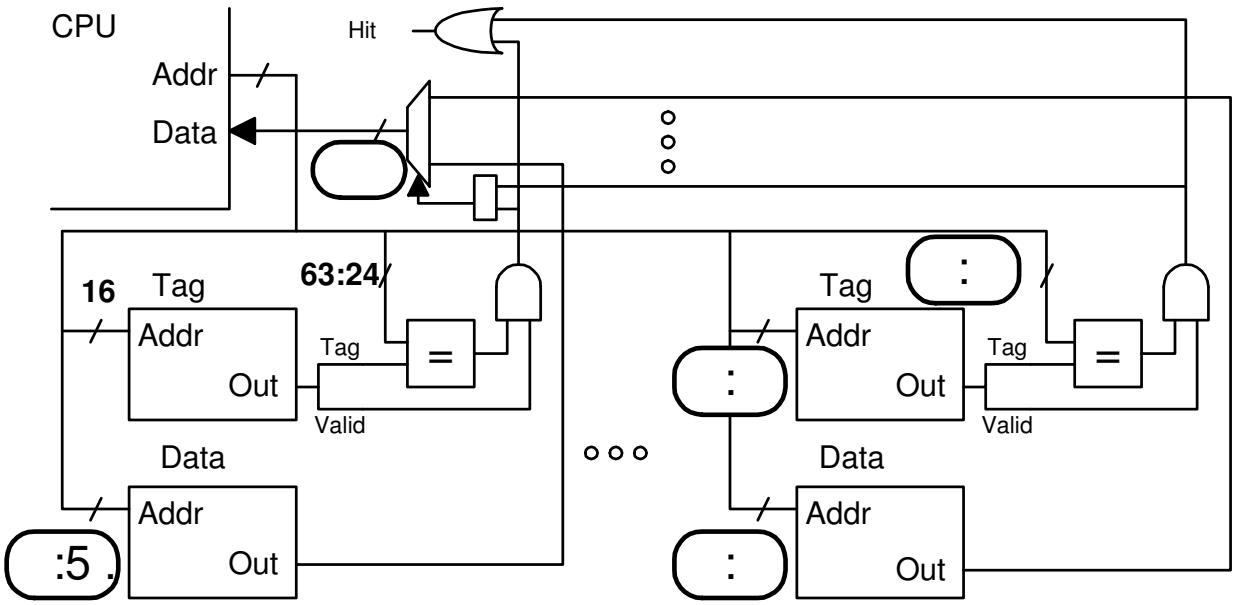
☐ Explain what goes wrong.

☐ Suppose millions of these defective implementations have already been manufactured. Suppose when turned on the processor starts executing code at address `0x1000`. What code could be put there to fix the problem? (It's not one of the fragments above because one of them only avoids it, later code could still trigger it.) *Hint: There's enough room for the answer below.*

Problem 5: The diagram below is for a 64-MiB (2^{26}-character) set-associative cache on a system with the usual 8-bit characters. (15 pts)

(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!!):

☐ Line Size (Indicate Unit!!):

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

Address:

Problem 5, continued: For the problems on this page use the cache from the previous page.

(*b*) The code below runs on the same cache as 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;
char *a = 0x2000000;    // sizeof(char) = 1 character.
int i;
int ILIMIT = 1 << 10;  // = 2^10

for(i=0; i<ILIMIT; i++) sum += a[ i * 4 ];
```

(*c*) The code below also runs in the cache from part a. Find the minimum values of `JLIMIT` and `JSTRIDE` that will result in the code below having a 0% hit ratio.

☐ `JSTRIDE` and `JLIMIT`

```
double sum = 0.0;
char *a = 0x2000000;  // sizeof(char) = 1 character.
int i, j;
int ILIMIT = 1 << 10;

int JLIMIT =                                          // FILL IN

int JSTRIDE =                                         // FILL IN

for(i=0; i<ILIMIT; i++)
 for(j=0; j<JLIMIT; j++)
  sum += a[i + j * JSTRIDE ];
```

Problem 6: Answer each question below.

(*a*) In MIPS, SPARC, and many other RISC ISAs memory accesses are aligned and so any instruction that uses an un-aligned address, such as `0x1001` for a MIPS word, will raise an exception usually resulting in the program exiting with a Bus Error.

Perhaps due to the stress of an upcoming code freeze for a product release, a mysterious programmer at Software Company X secretly hacked the operating system of their SPARC computers so that loads and stores to un-aligned addresses would complete correctly, as though un-aligned accesses were not forbidden. There would be no more bus errors. (5 pts)

☐ How might the mysterious programmer have done it?

☐ Should Software Company X reward or punish the mysterious programmer? Explain.

(*b*) The SRAM used to implement caches is costly but in many cache designs can provide 1- or 2-cycle hit latencies. If cost were not an issue, could one use SRAM for the entire memory system and get 1- or 2-cycle latencies on *all* memory accesses?(5 pts)

☐ Explain.

(*c*) Manufacturers have a great interest in having their processors score high in SPECcpu. Given this strong interest how can we be sure that benchmark selection and the run & reporting rules have not been chosen to favor a particular manufacturer? (This isn't kindergarten, so "because it's not allowed" is not a good answer.)(5 pts)

☐ Explain.