Name ____________________________________

Computer Architecture
EE 4720
Final Examination

12 December 2002, 10:00–12:00 CST

Problem 1	__________	(10 pts)
Problem 2	__________	(17 pts)
Problem 3	__________	(11 pts)
Problem 4	__________	(17 pts)
Problem 5	__________	(45 pts)
Exam Total	__________	(100 pts)

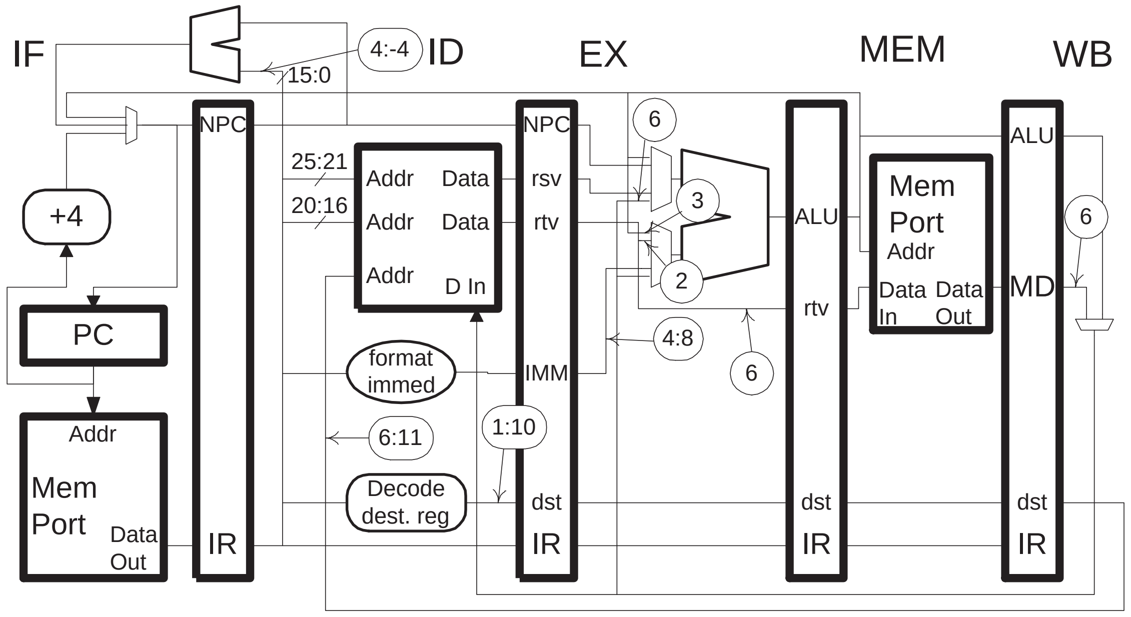
Alias ____________________________________

Good Luck!

Problem 1: In the diagram below certain wires are labeled with cycle numbers and, in some cases, values that will then be present, for example, $\boxed{2{:}9}$ indicates that at cycle 2 the wire will hold a 9. Other wires are labeled just with cycle numbers, indicating that the wire is used at that cycle. Write a program consistent with these labels. There are no stalls during the execution of the code. The code should use five instructions, use the PED to help solve the problem. (10 pts)

☐ Write a program consistent with these labels.

☐ Use labels for branch targets (if any) and label the target line.



```
Cycle                      0  1  2  3  4  5  6  7  8

                           IF ID EX ME WB

                              IF ID EX ME WB

                                 IF ID EX ME WB

                                    IF ID EX ME WB

                                       IF ID EX ME WB

Cycle                      0  1  2  3  4  5  6  7  8
```

Problem 2: The PED below shows execution on a defective 1-way dynamically scheduled machine using Method 3. A diagram appears on the next page. The circuitry that's supposed to restore the ID map after a branch misprediction is not working, the ID map is left unchanged. Everything else works correctly, in particular the free list is correctly restored to the exact state it was right after the `lw`. (17 pts)

(*a*) For this incorrect execution:

☐ Show where each instruction commits.

☐ Complete the ID map. (See the phys. reg. file for the free list.)

☐ Complete the commit map, include the initial state.

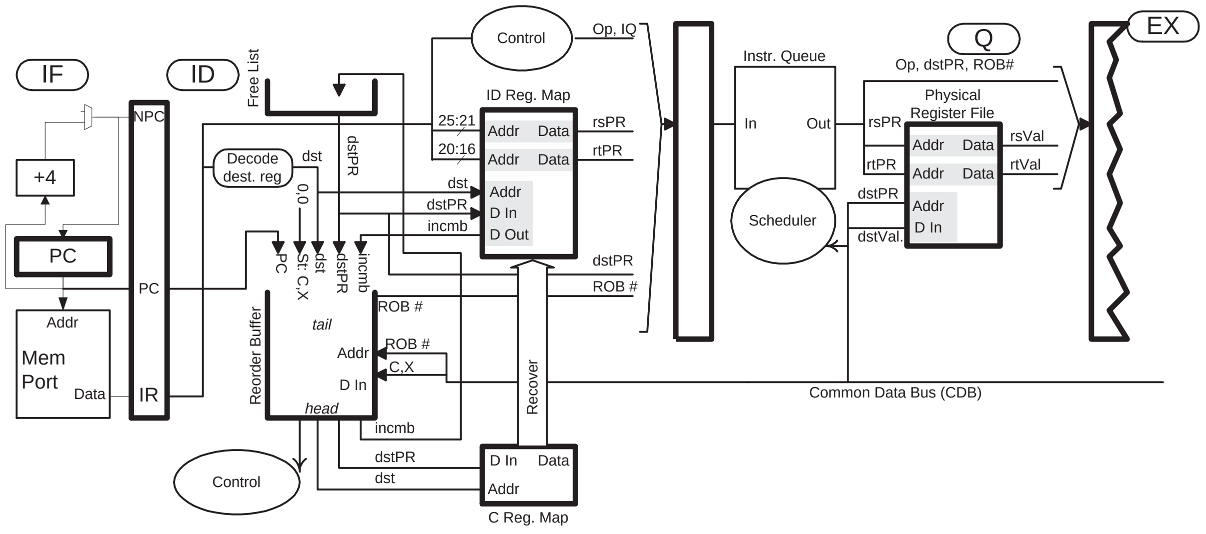
☐ Complete the physical register file.

☐ **In addition to other information** the physical register file should include a `[` in the cycle a register is removed from the free list and a `]` in the cycle it is returned to the free list.

☐ Circle incorrect entries (due to the defect) in the tables below.

```
 # lw loads a 0x200, lb loads a 0x202
 # Cycle            0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
 bne $s1, $s2  SKIP IF ID             Q  B  WB
 lw  $t1, 16($t1)      IF ID       Q  L1       L2 WB
 addi $t4, $t4, 3         IF ID Q  EX WB    x (squash)
 ori $t5, $t4, 0x30          IF ID Q  EX WB x (squash)
 nop ... (lots of nops)
SKIP:
 lb   $t1, 16($t1)                               IF ID Q  L1       L2 WB
 addi $t4, $t4, 3                                   IF ID Q  EX WB
 ori  $t5, $t4, 0xc0                                   IF ID Q  EX WB
 # Cycle            0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
ID Map
 t1   3
 t4   7
 t5  10
 # Cycle            0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
Commit Map
 t1
 t4
 t5
 # Cycle            0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
Physical Reg File.  All values in hex.  Free List: 18, 20, 23, 30, 37
 3  100
 7  101
 10 102
 18 103
 20 104
 23 105
 30 106
 37 107
 # Cycle            0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
```

Problem 2, continued: The illustration below is similar to one used in class; it is provided for reference.



(*b*) Suppose 1000 instructions later an instruction finds `0x4720` in register `t4` despite the fact that the code above wrote something else and no instruction wrote `t4` after that. (If the free list is used improperly the question might apply to `t5`, not `t4`. See the diagram for hints on proper use of the free list.)

Note: On the original exam register `t5` *was used instead of* `t4`. *Register* `t5` *can not change unexpectedly, though it still has an incorrect value.*

☐ How could that have happened? Be specific in how it's due to the defect.

(*c*) How could the defect have gone undetected?!? (That's not the question.)

☐ Suppose the `lw` raised an exception in `L2` on this defective hardware. Would the code above execute incorrectly? Explain. Remember, the only defect is with recovering the ID map on a branch misprediction. *Note: Original exam omitted "on a branch misprediction."*

☐ Now suppose the `lb` raised an exception in `L2` on this defective hardware. Would the code above execute incorrectly? Explain. Once again, the only defect is with recovering the ID map on a branch misprediction.

Problem 3: The code below runs on three systems: one using a bimodal (one-level) predictor, I; one using a two-level local history predictor, L; and one using a two-level gshare predictor, G. The global history register is 16 bits and the local history is 16 bits. The branch history tables have 256 entries. Assume that each branch below has its own BHT entry. The branch outcomes for B2 and B3 are provided for your solving convenience. Note that B3 has the same pattern as B2 but at a different phase, the phase difference is important. (11 pts)

```
BIGLOOP: LOOP:
 B1: bne $t1, $0, SKIP  # Random, independent, taken 25% of time.
 ...
 SKIP:
 B2: bne $t2, $t5, SKIP2 # Pattern: N N N N N N T T N N N N N N T T N N N N N N T T ....
 ...
 SKIP2:
 B3: bne $t7, $t8, SKIP3 # Pattern:  N N N N N T T N N N N N N T T N N N N N N T T N ....
 ...
 SKIP3:
 B4: bne $t9, $t10, LOOP # Iterates 50 times.
...
 j BIGLOOP
 nop
```

(*a*) Determine the prediction accuracy for each branch and each predictor. The accuracies should be after warmup. **Do not** compute the number of entries. Approximate the accuracy of B1 predictions.

☐ B1 on I: Accuracy:

☐ B1 on L: Accuracy:

☐ B1 on G: Accuracy:

☐ B2 on I: Accuracy:

☐ B2 on L: Accuracy:

☐ B2 on G: Accuracy:

☐ B3 on I: Accuracy:

☐ B3 on L: Accuracy:

☐ B3 on G: Accuracy:

☐ B4 on I: Accuracy:

☐ B4 on L: Accuracy:

☐ B4 on G: Accuracy:

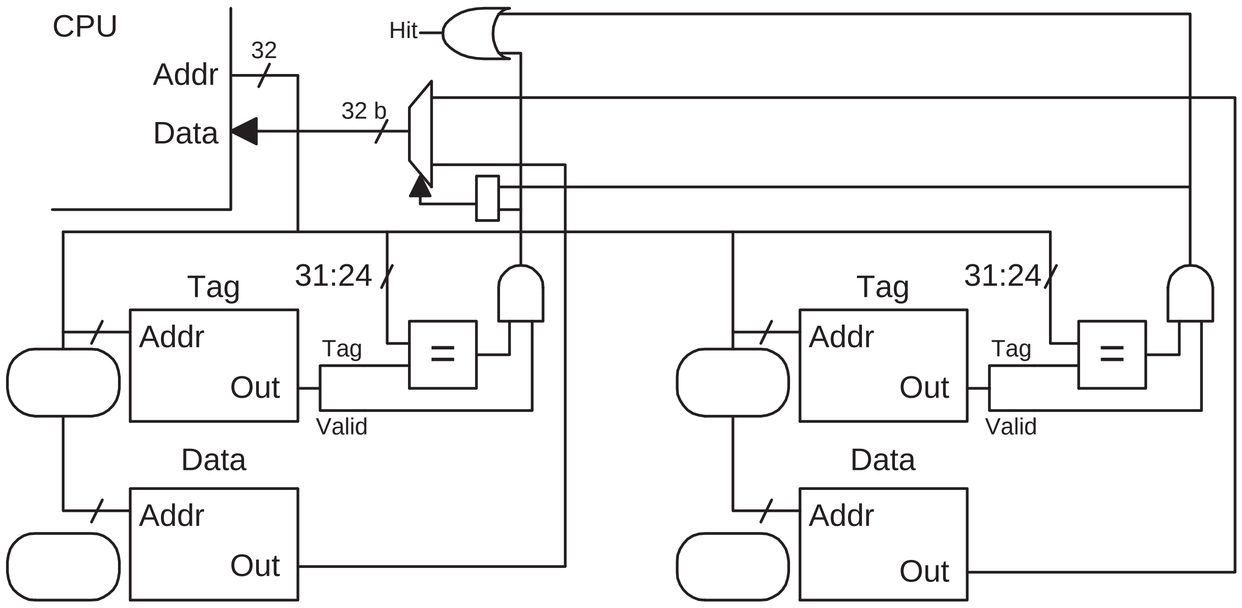
(*b*) A gshare predictor using a 15-bit GHR (instead of the 16-bit GHR used above) should give the same prediction accuracy for branch B2. How small can the GHR be made without changing the prediction accuracy of B2? Assume there are no collisions. *Note: The original question asked about B3.*

(*c*) For the local predictor, how small can the local history be made without affecting the prediction accuracy of B2 and B3?

Problem 4: The diagram below is for a cache with 256-character lines on a system with 8-bit characters. (17 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:

☐ Cache Capacity (Indicate Unit!):

☐ Memory Needed to Implement (Indicate Unit!):

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

Address:

Problem 4, continued:

(*b*) The code below runs on the cache from the previous part. When the code below starts running the cache is empty. Consider only accesses to the array.

```
int *a = 0x100000;  // sizeof(int) = 4 characters
int sum, i, j;

for(j=0; j<2; j++)
  for(i=0; i<1024; i++)
    sum += a[ i ];
```

☐ What is the hit ratio for the program above?

(*c*) In the problem below, only consider accesses to the arrays.

```
int *a = 0x10000;    // sizeof(int) = 4 characters
int *b = 0x20000;
int sum, i, j;

for(j=0; j<2; j++)
  for(i=0; i<1024; i++)
    sum += a[ i ] + b[ i ];
```

☐ What is the minimum size of a direct mapped cache for which the program above will have a 100% hit ratio on the second j iteration? Explain.

☐ What is the minimum size of a two-way, set-associative cache for which the program above will have a 100% hit ratio on the second j iteration?

☐ What is the minimum size of a three-way, set-associative cache for which the program above will have a 100% hit ratio on the second j iteration?

Problem 5: Answer each question below.

(*a*) In the PED below for a statically scheduled 2-way superscalar machine the `xor` instruction stalls in IF even though there is an empty space in ID. (5 pts)

```
add s1, s2, s3    IF ID EX ME WB
or  s4, s1, s5    IF ID -> EX ME WB
xor s6, s7, s8       IF -> ID EX ME WB
and s9, s10, s11     IF -> ID EX ME WB
```

☐ Why?

☐ Would it be difficult or would it be impossible to modify the implementation (but keeping it statically scheduled) so that such an instruction could move to ID? Explain.

(*b*) In a dynamically scheduled system why should store instructions wait until they are ready to commit before storing the data? (6 pts) (*c*) The same code fragment executes on 1-way, statically scheduled systems with the specified FP add functional unit(s). For each show a pipeline execution diagram. Don't forget to consider *all* structural hazards and check carefully for dependencies. All adders take a total of four cycles to compute a result (latency is 3). (5 pts)

☐ One adder, `A`, initiation interval: 1.

```
add.d f0, f2, f4

add.d f6, f0, f8

add.d f10, f0, f12
```

☐ One adder, `A`, initiation interval: 2.

```
add.d f0, f2, f4

add.d f6, f0, f8

add.d f10, f0, f12
```

☐ One adder, `A`, initiation interval: 4.

```
add.d f0, f2, f4

add.d f6, f0, f8

add.d f10, f0, f12
```

☐ Two adders, `A` and `B`, each has initiation interval: 4.

```
add.d f0, f2, f4

add.d f6, f0, f8

add.d f10, f0, f12
```

(*d*) One problem in Homework 5 was to analyze the execution of an unoptimized program to compute π. Many people got the question below wrong, here's your chance to get it right! (5 pts)

☐ Should computer engineers analyze the performance of processors running unoptimized code? Explain.

(*e*) Consider a new data type, binary coded trianary. This 32-bit data type consists of 16 radix-3 digits, each coded with two bits, in the same way a BCD data type would consist of 8 radix-10 digits. (8 pts)

☐ Why might 3- and 9-fingered aliens find this data type useful?

☐ Explain how the data type would be useful in accessing arrays in which the element size was a power of three.

In a meeting next week you plan to argue that this data type should be included in the next revision of the ISA, one of several proposed new data types. Only one will be chosen. Three-fingered aliens are **not** expected to make up a large portion of your customer base.

☐ How will you argue that the data type should be included?

☐ What will you do in the next week to prepare your argument?

(*f*) In Homework 6 the performance of a linear search of an array and linked list were compared. On the dynamically scheduled systems the search was much faster on the array. Now consider the array program on a statically scheduled system. (8 pts)

```
\scoreable Would the array program perform as well?  Consider
the effect of memory latency and line size.\par

\solution{The array program would still perform better than the linked
list program, but with longer memory latency or shorter line size the
array program on the dynamically scheduled system would run
substantially faster.  }
```

☐ Explain. *Hint: the solution above was intentionally included.*

(*g*) Predicated instructions can be used to avoid branches. (8 pts)

☐ Why do predicated instructions have maximum benefit over branches that skip over one instruction (compared to predicated instructions used to avoid branches that skip over more than one instruction)?

☐ Suppose the compiler is considering whether to replace a branch that skips n instructions with predicated instructions. How could the compiler use an estimate of prediction accuracy, f_{correct}, and resolution time, t_{resolve}, as well as implementation details to make its decision?