.

UNIVERSITY OF WISCONSIN

ECE 752 Adv. Computer Architecture I

Midterm Exam 1

Held in class Wednesday, March 9, 2005

Name: _____

This exam is **open books**, **open notes**, and **open all handouts** (including previous homeworks and exams, project descriptions, textbook, and readings). **However, it must be your own work--do not discuss any aspects of the exam problems with other students until after the exams have all been turned in.**

Read each problem statement carefully. Failure to follow directions will cost you points.

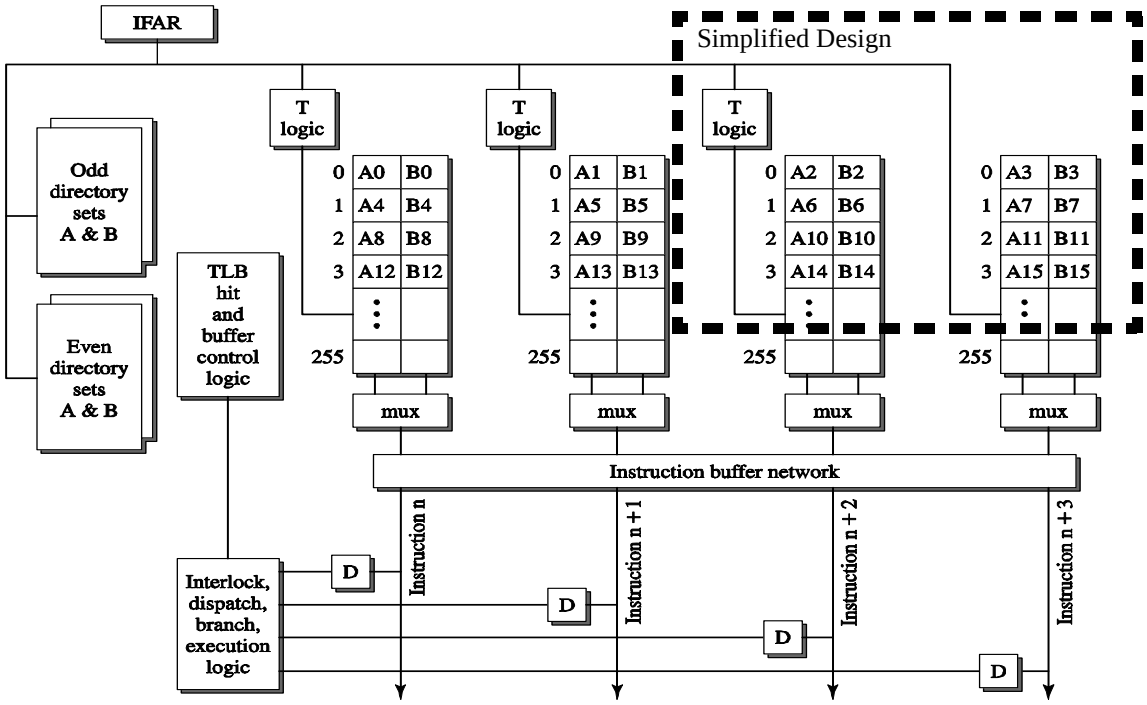
Write all your answers in the space provided. If you use the back of any page, please clearly indicate on the front of the page. If you use additional sheets, label them with your name and problem number.

If you feel there is ambiguity in the problem statement, please contact the TA or the professor. When in doubt, state your assumptions.

Problem 1.	[20 points]	_____	Instruction Fetch Alignment Logic
Problem 2.	[20 points]	_____	Global Branch Prediction
Problem 3.	[15 points]	_____	Data-Capture vs. Non-Data-Capture Schedulers
Problem 4.	[10 points]	_____	Discussion Questions
Problem 5.	[15 points]	_____	Questions from Readings
TOTAL:	[80 points]	_____	

1. Instruction Fetch Alignment [20 points]

Given the IBM RIOS-I instruction fetch architecture discussed in lecture and in the textbook, design the "T Logic" (specialized word-line decoder) for a simplified version of this cache architecture.



In the actual design, the T logic activates one of 256 word lines in each instruction cache subarray, and increments to the next word line whenever the 4-instruction fetch is not aligned on a natural 16-byte boundary. In your simplified design, the T logic will select one of 8 word lines instead of 256, and there are only two subarrays to support a peak fetch rate of two instructions per cycle. The cache blocks are 32 bytes (8 4-byte instructions each), resulting in 2 sets that are each 2-way associative, for a total cache size of 2 sets x 2 blocks/set x 32 bytes/block = 128 bytes. As in the RIOS-I design, the tag comparators can only check one set of tags at a time. The 32-bit instruction fetch address breaks down as follows:

	Tag	Set index	Row	Column	Byte
Address bits	b31...b6	b5	b5...b3	b2	b1 b0
Purpose	Tag compare	Select top (rows 0-3) or bottom (rows 4-7) set	Select one of 8 rows	Select one of 2 columns	Not used (instructions are 4B each)

A conventional row decoder would simply decode b5...b3 to activate one of 8 word lines. Your task is to design the "T logic" row decoder for the left column in your simplified cache (the right column will simply use a standard decoder, as is the case in the figure above).

a) [2 pts] Assuming 100% cache hits and uniform distribution of fetch addresses across all 8 instructions in a block, on average how many instructions per cycle can the simplified scheme supply?

b) [10 pts] Express the "T logic" boolean logic function using a truth table that enumerates all relevant input conditions along with their corresponding word-line outputs. Hint: you may not need the whole table.

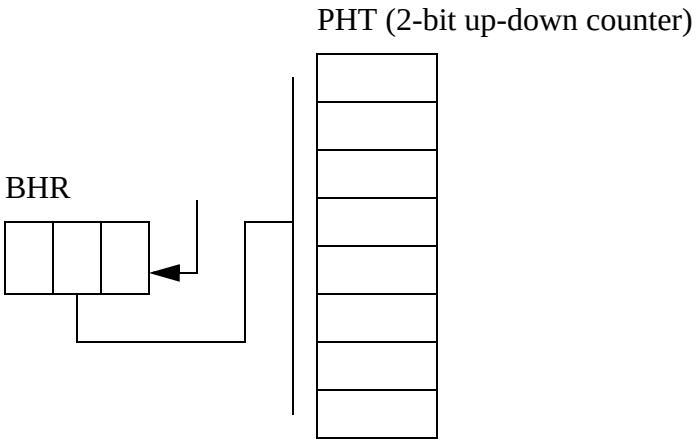
Enumerated Input Combinations	Word-line outputs (one hot) for left subarray							
	W0	W1	W2	W3	W4	W5	W6	W7

c) [8 pts] Draw a circuit diagram using simple gates (and, or, xor, with any number of inputs, show inverters as bubbles).

2. Global Branch Prediction [20 points]

Given a global branch predictor with a 3-bit branch history register (BHR) and 8 PHT entries (PHT0-PHT7), executing the following code with two branches A and B, fill in the table below to reflect the steady-state contents of each entry in the PHT after several hundred iterations of the outer while loop. Use the simulation table on page 6 to help you solve the problem, and to show your work for partial credit.

```
j = 0;
repeat {
    for(j=0;j<3;++j)              // conditional branch B, NT 3 times, then T one time
        sum += x[i] + y[j];
    ++j;
} until (j == 10000)              // conditional branch A, T for 10000 iterations
```



- Each predictor entry is a saturating up-down 2-bit Smith counter.
- A taken branch (T) increments the predictor entry; a not-taken branch (N) decrements the predictor entry.
- A taken branch (T) shifts a '1' into the LSB of the BHR, while a not-taken branch (N) shifts a '0' into the LSB of the BHR.
- A predictor entry less than 2 (0 or 1) results in a not-taken (N) prediction, otherwise taken (T).
- The predictor is updated for a branch before the next branch is predicted (no speculative updates).

a) [10 pts] Fill in the table below with the steady state contents of the PHT after several hundred iterations of the outer loop in the code example.

Table 1: Global Branch Predictor Steady State Contents

PHT Entries							
PHT0	**PHT1**	**PHT2**	**PHT3**	**PHT4**	**PHT5**	**PHT6**	**PHT7**

b) [2 pts] In the steady state, what is each branch's prediction rate? A: _____ B: _____

```
 j = 0;
repeat {
      for(j=0;j<4;++j)                  // conditional branch B, NT 4 times, then T one time
            sum += x[i] + y[j];
      ++j;
} until (j == 10000)                    // conditional branch A, T for 10000 iterations
```

c) [4 pts] Given the above--very slightly modified--program, what happens to the prediction rates for each branch? You should be able to answer without "simulating" the predictor.

Branch A: _____ Branch B: _____

```
j = 0;
repeat {
      for(j=0;j<30;++j)                 // conditional branch B, NT 30 times, then T one time
            sum += x[i] + y[j];
      ++j;
} until (j == 10000)                    // conditional branch A, T for 10000 iterations
```

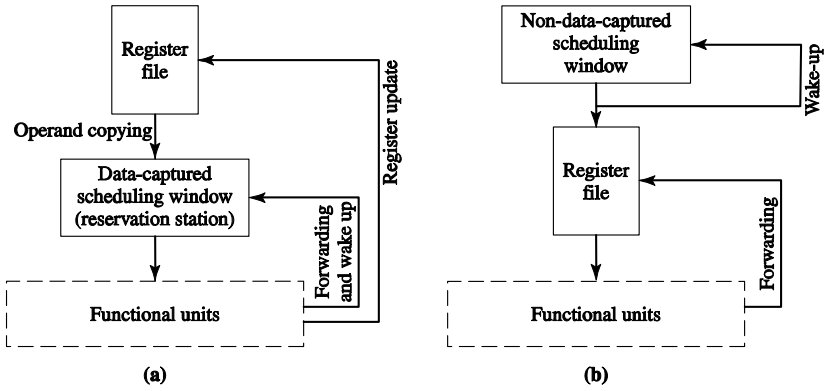
d) [4 pts] Finally, given the third version of the program shown above, name and describe the best choice of an advanced branch predictor from Chapter 9 in the textbook that would provide near 100% prediction rates for branch B with very reasonable storage overhead.

Table 2: Global Branch Predictor Simulation (show work here)

Branch Address	Branch Outcome (T/N)	Prediction (T/N)	Predictor State After Branch is Resolved								
				PHT Entries							
			BHR	PHT0	PHT1	PHT2	PHT3	PHT4	PHT5	PHT6	PHT7
Initial	**N/A**	**N/A**	**111**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
B	**N**	**N**	**110**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	N										
B	N										
B	T										
A	T										
B	N										
B	N										
B	N										
B	T										
A	T										
B	N										
B	N										
B	N										
B	T										
A	T										
B	N										
B	N										
B	N										
B	N										
B	T										
A	T										
B	N										
B	N										
B	N										
B	T										
A	T										

3. Data-Capture vs. Non-Data-Capture Schedulers [15 points]

As discussed in lecture, data-capture schedulers that implement Tomasulo's algorithm suffer from less competitive cycle times compared to non-data-capture schedulers.



In this problem, you are to develop a simplified analytical delay model for both scheduler types, based on the following parameters and assumptions:

- Assume that each scheduler entry contains flip-flops used as pipeline registers. Delay will be measured from the output of these flip-flops back to their inputs.
- Your simplified model can ignore flip-flop setup and hold times and clock skew (i.e. assume they are zero)
- Assume **n** entries in the scheduler or issue queue
- Assume **p** bits of non-operand payload per entry for both scheduler types (not including operands)
- Assume **128** bits of source operand payload per entry in the data-capture scheduler
- Assume **s** nanoseconds of delay for issue selection logic (not including wire delay), which selects instructions that will issue in a cycle out of the pool that are data-ready and have asserted their select request line.
- Assume **t** nanoseconds of wakeup delay (tag match and ready signal generation)
- Assume **a** nanoseconds of ALU delay, which includes wire delay for transferring operands to/from the perimeter of the scheduler to/from the ALU input/output.
- Accounting for **wire delay:** assume that all signals that have to span the entire scheduler (this includes tag broadcast, data broadcast, select request, and select grant) will have wire delay that is **w** nanoseconds of delay per unit distance, where distance is defined as twice the square root of scheduler capacity in bits. This assumes a square layout for the bits stored in the scheduler with a worst-case diagonal traversal for each signal that must span the entire array.

a) [5 pts] Construct an arithmetic delay equation for a data-capture scheduler using the terms above (**n, p, s, t, a,** and **w**), assuming a round-trip combinational path from scheduler entry back to scheduler entry.

b) [5 pts] Construct an arithmetic delay equation for the wakeup-select loop of a non-data-capture scheduler using the terms above (**n, p, s, t, a,** and **w**), assuming a round-trip combinational path from scheduler entry back to scheduler entry.

c) [3 pts] Substitute values {p=10, n=32} and comment on the relative delays for your equations from (a) and (b).

d) [2 pts] Identify and discuss at least one weakness in this simplified analytical delay model.

4. Short Discussion Questions [10 points]

a) [4 pts] Name and describe the key attributes used to describe instruction-level parallel machines in Jouppi's classification scheme, as discussed in lecture. In light of modern processors, name one or more important attributes that are missing from this classification scheme?

b) [2 pts] Given a future file-based out-of-order processor that incorporates an architected register file and a rename register file and uses a rename map table for storing rename register mappings, identify and discuss all pipeline hazards affecting the rename map table.

c) [4 pts] Given the Iron Law of processor performance, describe two examples of instruction set changes that reduce the first term but can still degrade performance due to their effects on the second and/or third term. Describe the examples and their effects on each term.

5. Questions from Readings [15 points]

a) [5 pts] Draw a diagram of a two-level branch predictor as presented in the Yeh and Patt paper and describe its operation. What are the key benefits of two-level predictors over previous branch prediction schemes?

b) [2 pts] Identify and discuss the main stated purpose of the International Technology Roadmap for Semiconductors.

c) [3 pts] What is the difference between a future file and a history buffer, as described in the Smith and Pleszkun paper? What is the purpose of these structures? Describe their operation.

d) [3 pts] Name and describe the “loose loops” described in the Borch et al. paper. According to the paper, which loop or loops are most critical for performance? Do you agree? Why or why not?

e) [2 pts] What was the main battle described in "The Microprocessor Today?" Does this “battle” continue to be as important as it was when the paper was written? Why or why not?