Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 382N, Spring 2004
Y. N. Patt, Instructor
Hyesoon Kim and Moinuddin Qureshi, TAs
Exam 1, March 10, 2004

Name :_____

Problem 1 (12 points):_____

Problem 2 (12 points):_____

Problem 3 (12 points):_____

Problem 4 (12 points):_____

Problem 5 (12 points):_____

Problem 6 (12 points):_____

Problem 7 (12 points):_____

Problem 8 (12 points):_____

Problem 9 (12 points):_____

Problem 10(12 points):_____

Bonus for legibility on

all anwers (4 points):_____

Total (100 points):_____

Directions: The first two problems of this exam are required problems. You may answer any 6 of the last 8 problems. Place an "X" in the 2 lines above for the 2 problems that you choose not to answer.

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

GOOD LUCK AND HAVE A GOOD SPRING BREAK!

Name:_____

Problem 1 - Required (12 points):

Design an architectural register file with two write ports and two read ports. Two write ports so two instructions with register writes can be retired in the same cycle. In this case, port 1 will write the older of the two destinations, port 2 will write to the younger of the two destinations. Note: make sure you take into account the case where both instructions write to the same register.
The register file should hold 4 registers where each register is 32 bits wide.

You may use the following library parts in your design:

```
module  reg32e$(clk, Din, Q, write_enable);  // 32-bit wide, write-enabled register
module  nand2$(out, in0, in1);               // two input nand gate
module  nor2$ (out, in0, in1);               // two input nor gate
module  mux2$(out, in0, in1, s0);            // 2 to 1 mux
module  decoder2_4$(s, y, ybar);             // 2 to 4 decoder
```

Use the following prototype for your register file module. It is up to you to decide the appropriate width of each signal.

```
module  regfile32_4 (clk, // input - the clock
           read_select0,  // input - select line for 1st read port
           read_select1,  // input - select line for 2nd read port
           write_select0, // input - select line for 1st write port
           write_enable0, // input - write enable for 1st writeport
           write_data0,   // input - data to write to register[write_select0]
           write_select1, // input - select line for 2nd write port
           write_enable1, // input - write enable for 2nd write port
           write_data1,   // input - data to write to register[write_select1]
           read_port0,    // output - data from 1st read port
           read_port1);   // output - data from 2nd read port
```

You may answer either in Verilog or with a block diagram, whichever you prefer. In either case, please use modular design in your solution. If you use a block diagram to answer, be sure to label all the signal names and the widths. Please make your solution as simple and clear as possible.

Name:_____

Problem 1 (continued) - Required (12 points):

Additional space for your register file design.

Name:_____

Problem 2 - Required (12 points):

Decode the x86 instruction stream shown below into its component instructions.

```
83 05 f4 04 03 0f 88 fc 0f a8 66 f7 90 01 02 03 01
```

Show each instruction in the following form :

```
prefix opcode mode r/m SIB disp imm dest src1 src2
```

Where a field does not apply for a particular instruction, just ignore that field for that instruction. (We have provided room for 6 instructions. Hint: There are not more than six instructions in the above code.)

inst #	prefix	opcode	mode	r/m	SIB	disp	imm	dest	src1	src2
1										
2										
3										
4										
5										
6										

Name:_____

Problem 3 (12 points):

UT382N is a single-issue, in-order, 5-stage RISC machine. The microarchitects have been told that feature size being what it is, their next generation implementation has a large increase in available transistor count. The first brainstorming session came up with the following two options:

	OPTION A	OPTION B
Branch Predictor	Twobc (90% accurate)	hybrid_gsh (98% accurate)
Data Cache	128KB (99% hit rate)	32KB (98% hit rate)
Cache Latency	1 cycle	1 cycle
Memory Latency	100 cycles	100 cycles

Which option makes more sense and why?

Name:_____

Problem 4 (12 points):

CacheSim is a trace driven simulator for evaluating cache miss rates. The input to CacheSim is a trace file containing data addresses generated by correct path instructions. Simulation of gcc, perl, bzip, and mcf (four popular benchmarks that we will talk about soon) were carried out using CacheSim, in order to test the effect of increasing cache associativity from four to eight. The following was the result.

```
CacheSim parameters:
CACHE_SIZE  = 256KB
LINESIZE    = 32 B
REPLACEMENT = PERFECT_LRU

Starting simulation : ..................
Simulation finished.
Reporting D-cache miss rates:
-----------------------------------------------
CACHE_ASSOC     gcc     perl    bzip    mcf
-----------------------------------------------
4 Way           0.20%   0.05%   1.62%   15.50%
8 Way           0.18%   0.05%   1.61%   15.58%
-----------------------------------------------

Thank you for using CacheSim.
```

A. Is there anything about these results that immediately suggest that there is a bug in the Simulator? Explain.

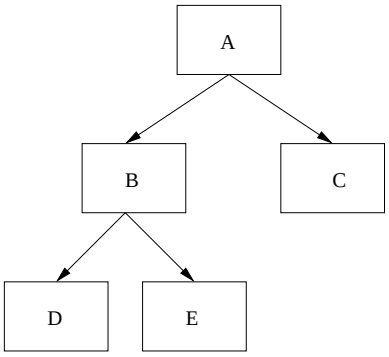
B. Upon further exploration, you discover that the Simulator has no bugs in it. Show by an example what could be the root of the counter-intuitive behavior.

Name:_____

Problem 5 (12 points):

The block-structured ISA approach suggests combining more than one block into an enlarged block. Why is that a good idea?

There are two counter-arguments to generating enlarged blocks that suggest it is a bad idea? What are they? Discuss it in the context of the figure below. Be explicit, but not unnecessarily wordy.
Hint: Consider combining Block B with its successors.



Name:_____

Problem 6 (12 points):

What does branch promotion allow that enhances the performance of a trace cache? How is the misprediction of a promoted branch handled? Please be explicit.

What does inactive issue provide? Why is it a good idea? Why is it a bad idea?

Name:_____

Problem 7 (12 points):

What does a dataflow implementation provide that is very difficult to provide any other way? Explain.

Name:_____

Problem 8 (12 points):

The pipelines of today's out-of-order implementations can be thought of as having three phases. What structure separates phase 1 from phase 2? Why is that structure needed?

What structure separates phase 2 from phase 3? Why is that structure needed?

How are the instructions processed during phase 1? Why?

Do all three phases have to operate as fast as possible in order to obtain near optimal performance? Explain.

Name:_____

Problem 9 (12 points):

An aggressive out-of-order execution implementation usually has a load/store queue. In it we store information about every load and every store that is in-flight (i.e., in the active window).

a.(4 points) There are two reasons for storing ST values(data) in this queue. What are they?

b.(8 points) Suppose instruction LD R2,A enters the queue, where the address A is known. One of the following four cases must be true. For each case, how do we process LD R2,A ?

i. There is no older ST in the queue.

ii. An older ST in the queue is to the same address as the LD.

iii. All older ST in the queue are to addresses different from the LD.

iv. There is an older ST in the queue whose address has not yet been evaluated.
(Note: There are two reasonable possibilities for handling case iv. Explain.)

Name:_____

Problem 10 (12 points):

The first two-level branch predictor was published in Micro-24 in 1991. Since then, it has seen many variations. Each variation has been proposed to provide some additional benefit. Four such variations are described below.
Select three of the four variations. For each, describe the intended additional benefit over the baseline two-level predictor that we came up with in 1991. It is important to be explicit in your answers without being unnecessarily wordy.

a. Instead of using the global history register to index into the pattern history table, use the XOR of the global history register combined with some bits of the branch's address.

b. Combine the two-level predictor with a 2-bit counter or a last time taken predictor.

c. Do not increment/decrement the 2-bit counter in the pattern history register on the basis of T/NT. Instead store a direction bit in the tag store entry for each branch, and increment/decrement on the basis of whether the prediction agrees or does not agree with the stored direction bit.

d. Instead of using the directions of the last several branches leading up to a branch as the index into the pattern history table, combine bits of the addresses of each of the intervening branches into an index.