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

EE 382N, Spring 2002
Y. N. Patt, Instructor
D. N. Armstrong, TA
Exam 1, March 6, 2002

Name (2 points):_____

Problem 1 (14 points):_____

Problem 2 (14 points):_____

Problem 3 (14 points):_____

Problem 4 (14 points):_____

Problem 5 (14 points):_____

Problem 6 (14 points):_____

Problem 7 (14 points):_____

Problem 8 (14 points):_____

Problem 9 (14 points):_____

Total (100 points):_____

Directions: The first two problems of this exam are required problems. You may answer any 5 of the last 7 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 (14 points):

Part a (7 points): Circle the errors in the following Verilog code:

```
module TOP(z[3:0],  // output
           a[3:0],  // input
           b[3:0],  // input
           c[3:0]); // input

    output [3:0]  z;
    input [3:0]   a, b, c;
    reg           clock;

    nand2$ n2gate(t[0], a[0], b[0]);
    nand2$ n2gate(t[1], a[1], b[1]);
    nand2$ n2gate(t[2], a[2], b[2]);
    nand2$ n2gate(t[3], a[3], b[3]);

    nand3$ n3gate(z[0], t[0], c[0], 1'b1);
    nand3$ n3gate(z[1], t[1], 1'b1, c[1]);
    nand3$ n3gate(z[2], 1'b1, t[2], c[2]);
    nand3$ n3gate(1'b1, z[0], t[3], c[3]);

     nor3$  n3gate(z, 4'b0, 4'b0, 4'b0);

     always
       begin
         clock = ~clock;
       end

endmodule; // TOP
```

For each error that you circle, provide a 1 sentence explanation of the error.

Name:_____

Problem 1 - (continued) - Required (14 points):

Part b (7 points): When determining which instructions would be used in the project, it was decided that students needn't implement both the NOT and the XOR instruction. Which of the following x86 XOR instructions is the **best** instruction with which to implement the NOT of both 32-bit memory locations and also 32-bit registers. Why?

```
35 id           XOR EAX, imm32      EAX XOR imm32
81 /6 id        XOR r/m32, imm32    r/m32 XOR imm32
83 /6 ib        XOR r/m32, imm8     r/m32 XOR imm8 (sign-extended)
32 /r           XOR r/m32, r32      r/m32 XOR r/m32 XOR r32
33 /r           XOR r32, r/m32      r32 XOR r/m32
```

Name:_____

Problem 2 - Required (14 points):

Design a register file with a single write port and four read ports. 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  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
                     read_select2,  // input - select line for 3rd read port
                     read_select3,  // input - select line for 4th read port
                     write_select,  // input - select line for write port
                     write_enable,  // input - write enable for write port
                     write_data,    // input - data to write to regfile
                     read_port0,    // output - data from 1st read port
                     read_port1,    // output - data from 2nd read port
                     read_port2,    // output - data from 3rd read port
                     read_port3);   // output - data from 4th 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.

Note: you may use only one bit of storage to store each bit in the register file (i.e., designing a register file with a single read port and then duplicating that part is not satisfactory).

Name:_____

Problem 2 (continued) - Required (14 points):

Additional space for your register file design.

Name:_____

Problem 3 (14 points):

The first Alpha chip (21064) had the property that if a branch instruction predicted taken (i.e., the instruction stream was to be redirected rather than execute the fall through path), the target address would be computed (PC + offset) from the actual 0s and 1s of the branch instruction, and that target address would be used as the location in the instruction cache to perform the next fetch. What important consequence did this imply?

Was this a good consequence or a bad consequence? If good, explain why. If bad, explain what additional feature could have been added to the data path to remove the bad consequence.

In either case, feel free to illustrate the issue with a figure.

Name:_____

Problem 4 (14 points):

We have not talked a lot about SPEC yet. We will later in the semester. Suffice it to say for now that SPEC numbers are measures of the time it takes to execute certain representative benchmark programs. A higher number means the execution time of the corresponding benchmark(s) is smaller. Some have argued that this gives unfair advantage to processors that are designed using a faster clock, and have suggested that the SPEC numbers should be normalized with respect to the clock frequency, since faster clocks mean shorter execution time and therefore better SPEC numbers. Is this suggestion a good or a bad idea? Explain.
If you are told that your design will be evaluated on the basis of its SPEC/MHz number, what major design decision would you make?

Name:_____

Problem 5 (14 points):

The Pentium processor, vintage 1992, had a split-line first-level instruction cache. Why was that a good idea? Be brief, please, but specific.

Name:_____

Problem 6 (14 points):

Your job is to implement the x86 ISA, augmented with instructions that each write four results as part of the semantics of those instructions. Your design decision is to implement the machine with a ten-stage pipeline that writes results in stages 7, 8, 9 and 10. Is there any problem with your design decision? Discuss possible problems, and what you would do to make sure everything works.

Name:_____

Problem 7 (14 points):

Your job is to evaluate the potential performance of two processors, each implementing a different ISA. The evaluation is based on its performance on a particular benchmark. On the processor implementing ISA A, the best compiled code for this benchmark performs at the rate of 10 IPC. That processor has a 500 MHz clock. On the processor implementing ISA B, the best compiled code for this benchmark performs at the rate of 2 IPC. That processor has a 600 MHz clock.

What is the performance in MIPS (millions of instructions per second) of the processor implementing ISA A? _____.

What is the performance in MIPS (millions of instructions per second) of the processor implementing ISA B? _____.

Which is the higher performance processor? A___ B___ Don't know___.

Explain?

Name:_____

Problem 8 (14 points):

The Two-level branch predictor was first introduced on a commercially viable microprocessor by Intel in 1995 on their Pentium Pro. The processor had a twelve stage pipeline, and was capable of fetching and decoding three instructions each cycle. This meant that one could easily encounter several branch instructions to predict before the direction of older branch instructions were confirmed. That is, many newer branch instructions would come along while the direction of older branch instructions was still unknown. One can handle the problem in one of at least three ways:

1. Allow at most some small number of unconfirmed branch instructions in mid-stream at any one time, then stop fetching.
2. Allow as many as happen to occur, but use the confirmed Branch History Register entries (the first level of the two-level predictor) to index into the second-level Pattern History Table to perform the prediction.
3. Allow as many as happen to occur, but update the Branch History Register on the fly with the predicted values of the immediately preceding branches and use them to provide the index into the second-level Pattern History Table.

Which way do you think is the best way to handle the problem? Explain.

Name:_____

Problem 9 (14 points):

Your job is to improve the performance of the x86 ISA, and the corporate fathers decree that performance is so important that you don't have to worry about compatibility. [See, this is a university and we spend time on academic exercises. This one, most would argue, ain't going to happen! – Actually, it could within a certain context which we should discuss between now and the end of the semester. But, for now, let's deal with the problem at hand.] What is the first thing you would change in the x86 ISA to improve performance? Why? What is the second thing you would change in the x86 ISA? Why? Third thing?