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

EE 382N, Spring 2010
Y. N. Patt, Instructor
Eiman Ebrahimi and Khubaib, TAs
Exam 1, March 31, 2010

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 answers (4 points):_________

Total (100 points):____________

Directions: The first problem of this exam is a required problem. You may answer any 7 of the last 9 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!

Name:________________________________

Problem 1 - Required (12 points): You are part of a team designing a GAg branch predictor. Specifically, you are responsible for the module that computes the branch history used to index the pattern history table. The branch history register is updated on each prediction, and bit 0 of this register corresponds to the most recent branch.

Question a:

The 64-entry PHT requires a 6 bit index. The in-order processor pipeline allows for up to 3 in-flight branches. The module you are designing must provide the correct branch history even after a processor pipeline flush. The correct branch history to predict branch B consists of the directions of the branches immediately preceding branch B in program order. If a pipeline flush is caused by a branch misprediction, the module will see the correct outcome of the mispredicted branch on the `br_dir` input in the next cycle.

How many bits of branch history need to be stored in order to satisfy the requirements?

Answer: N =

Question b:

Design the combinational logic circuit that computes the branch history given the branch history from the previous cycle and the other inputs listed below. You may use any parts from the project library.

```
module bp_hist(new_hist[N-1:0], // output -- branch history at the end of the cycle
               old_hist[N-1:0],  // input -- branch history from the previous cycle,
               br_dir_is_valid,  // input -- the br_dir input has a valid value this cycle
               br_dir,           // input -- direction of the predicted branch
               flush,            // input -- the entire processor pipeline is flushed this cycle
               num_br[1:0]       // input -- number of branches in the processor pipeline
               );
```

Assume that a "`flush`" assertion should take precedence over a concurrent "`br_dir_is_valid`" signal.

Show both a block diagram of your design and structural verilog code. Be sure to label all the signal names and the widths. Please make sure your solution is as simple and clear as possible.

Name:________________________________

Name:________________________________

Problem 2 (12 points): Latency and bandwidth (both on-chip and off-chip) are important issues in today's and future microprocessors. Of the four cases (on-chip latency, on-chip bandwidth, off-chip latency, off-chip bandwidth), only one is not problematic. Explain.

Problem 3 (12 points): As I pointed out in class, our version of the trace cache provided for partial matching, Jim Smith's version did not. What is partial matching? Given an example. What are the pros/cons of providing partial matching in the microarchitecture?

Name:________________________________

Problem 4 (12 points): The RISC mania of the 1980s insisted on ISAs being Load/Store. What does that mean? How did Load/Store ISAs provide a temporary benefit over non- Load/Store ISAs during the late 1980s? What microarchitectural feature surfaced in the 1990s to remove that benefit? Explain.

Problem 5 (12 points): The Pentium implementation did not take a one-cycle pipeline bubble on a taken branch, while the first Alpha chip did take at least a one-cycle bubble on a taken branch. What structure on the Pentium chip provided that benefit. Show (with a drawing) the data path required to enable that benefit.

Name:________________________________

Problem 6 (12 points): Classical VLIW ISAs suffer from two constraints. One is that all instructions in a VLIW word must be independent. What is the other? What paradigm removes this "other" constraint? Explain how?

Problem 7 (12 points): Hybrid branch predictors combine predictors having different good qualities, the intent being that the predictor used at any point in time is the one that optimizes the performance at that particular time. An example could be a predictor that consists of PAp, GAg, and the 2-bit counter. Explain the pros and cons of each of these three component predictors that would make the three a viable collection for a hybrid predictor.

Name:________________________________

Problem 8 (12 points): A continual question we ask is whether to do something at compile time or at run time. We say that one of the good things about compile time is that the compiler can predict the future, whereas the hardware at run-time can not. What do we mean by this? Give two caveats that must be satisfied if the compiler is to have a chance at "predicting the future".

Problem 9 (12 points): Last week, I gave you two position papers I wrote, one for IEEE Computer on the billion transistor chip, and one as an introduction to a special issue of Proceedings of the IEEE devoted to the microprocessor. Pick one of those articles (your choice), select a point that I made that you either agree with or disagree with (again, your choice). If you agree with my point, give an example different from anything I wrote that further supports my point. If you disagree with my point, explain why.

Name:________________________________

Problem 10 (12 points): The VAX ISA, at its initial announcement, contained 244 distinct opcodes, encoded in the first byte of the instruction. Each opcode required a number of operands, specific to that opcode. For example, the "change priviledge mode" instruction required 0 operands. The "move string" opcode required five operands: the starting address of the source, the starting address of the destination, the length (in bytes) of the source, the space (in bytes) allocated at the destination, and a "filler" byte to insert in every unused location in the destination if the length of the source is smaller than the space allocated at the destination. Each operand was determined by a single byte operand specifier that specified the addressing mode to use to compute the address. If the addressing mode required extra registers or displacements, that information followed the operand specifier in the instruction stream. The operand specifiers were orthogonal to the opcodes; that is, for each operand, the ISA allowed any operand specifier to be used for computing that operand address.

Example of a VAX istream highlighting a VAX opcode with three operands:

Byte 1: opcode

Byte 2: operand specifier designating an extra register and a 2 byte offset

Byte 3: extra register needed to compute the address of first operand

Byte 4,5: the 2 byte offset

Byte 6: operand specifier designating the requirement for just a 4 byte offset

Byte 7,8,9,10: the 4 byte offset

Byte 11: operand specifier requiring no extra information

Byte 12: the next opcode

Your job: compare the decode problem for VAX with the decode problem for x86. Which is easier? Explain.