Last (family) name: _____

First (given) name: _____

Student I.D. #: _____

Department of Computer Sciences
University of Wisconsin - Madison

CS/ECE 752 Advanced Computer Architecture I

Midterm Exam 1

Monday, February 17, 2014

Instructions:

1. Open book/open notes.
2. The exam is multiple choice and will be graded using a separate automatically read grading sheet. Please write your name, ID number, and answers on the grading sheet. Be sure to fill in the bubbles fully for each question.
3. Upon announcement of the end of the exam, stop writing on the exam paper immediately. Pass the exam to aisles to be picked up by the proctors. The instructor will announce when to leave the room.
4. Failure to follow instructions may result in forfeiture of your exam and will be handled according to UWS 14 Academic misconduct procedures.

Problem	*Type*	*Points*	*Score*
1-15	Multiple Choice	30	
16-18	Hierarchical Branch Predictor Performance	15	
19-23	Program Data Dependence Analysis	15	
24-27	Instruction Scheduling	20	
28-37	From the Readings	20	
Total		100	

Problems 1-20: (40 pts): Multiple choice; select the best answer for each question

Note: Some of the answers below are "list" answers, such as "All of the above" and "Both (a) and (b)." In these answers, the word "above" refers only to "real" answers with specific content, not other list answers.

1. A pipelined processor that does not have any WAR register hazards
    1) Must have an earlier register read stage and a later register write stage
    2) Must have an earlier register write stage and a later register read stage
    3) Must have two stages that write registers, one later than the other
    4) None of the above

2. A VLIW instruction set processor
    1) Packs multiple operations into a single instruction
    2) Usually relies on software to resolve pipeline hazards
    3) Can operate at much higher frequency than other approaches
    4) Exposes a lot more instruction-level parallelism than other approaches
    5) Both (a) and (b)
    6) None of the above

3. Local branch history in a dynamic branch predictor is used to:
    1) Predict a branch based on how neighboring branches were resolved recently
    2) Predict a branch based on how that same branch resolved recently
    3) Predict a branch based on the sign bit of its offset field
    4) Predict a branch based on a profiling run collected with a representative input set

4. A branch that is mispredicted as not-taken requires the processor control logic to:
    1) Restart fetching instructions from the not-taken path
    2) Clear out all instructions that were not tagged with the mispredicted branch's tag
    3) Fix up the rename table to correspond to the state following the branch
    4) None of the above
    5) Both (b) and (c)

5. A two-level dynamic branch prediction:
    1) Uses a second-level branch history table to capture the branch working sets of very large programs that contain thousands of branches
    2) Uses two levels of branch confidence to identify easy-to-predict and hard-to-predict branches
    3) Learns more than one possible prediction for a static branch by using branch outcome history as part of the lookup index into the pattern history table
    4) Accurately predicts exits from loops that have iteration counts in the hundreds
    5) None of the above

6. The return address stack (RAS):
    1) Pushes the return address whenever a call instruction is encountered
    2) Pops a return address for every return instruction
    3) Raises an exception when the stack overflows or underflows
    4) None of the above
    5) Both (a) and (b)
    6) All of the above

7. A branch target buffer:
    1) Uses full address tags to eliminate aliasing between branches
    2) Uses full address tags to differentiate branches and non-branches
    3) Uses partial address tags to reduce aliasing between branches
    4) Uses partial address tags to differentiate branches and non-branches
    5) Doesn't use address tags

8. A branch history table:
    1) Uses full address tags to eliminate aliasing between branches
    2) Uses full address tags to differentiate branches and non-branches
    3) Uses partial address tags to reduce aliasing between branches
    4) Uses partial address tags to differentiate branches and non-branches
    5) Doesn't use address tags

9. Longer branch histories
    1) Improve prediction of longer loops
    2) Increase aliasing in the prediction table
    3) Always improve prediction accuracy
    4) All of the above
    5) Both (a) and (b)
    6) Both (a) and (c)

10. Superblock scheduling is used:
    1) When branches are highly-biased taken
    2) When branches are highly-biased not-taken
    3) When branches are not highly-biassed
    4) All of the above
    5) Both (a) and (b)
    6) None of the above

11. Hyperblock scheduling is used:
    1) When branches are highly-biased taken
    2) When branches are highly-biased not-taken
    3) When branches are not highly-biassed
    4) All of the above
    5) Both (a) and (b)
    6) None of the above

12. Clustering an N-wide super scalar:
    1) Groups function units into K clusters
    2) Requires N/K bypass paths in each cluster
    3) Increases bypass latency between clusters
    4) All of the above
    5) Both (a) and (b)
    6) Both (a) and (c)

13. Clustering an N-wide super scalar typically:
    1) Increases CPI and increases cycle time
    2) Increases CPI and decreases cycle time
    3) Decreases CPI and increases cycle time
    4) Decreases CPI and decreases cycle time

14. Dynamic power has a
    1) Linear relationship with voltage
    2) Quadratic relationship with voltage
    3) Cubic relationship with voltage
    4) None of the above

15. Reducing clock frequency
    1) Reduces dynamic power
    2) Reduces static power
    3) Reduces performance
    4) All of the above
    5) Both (a) and (b)
    6) Both (a) and (c)
    7) None of the above

Hierarchical Branch Predictors (15 points)

Some highly-pipelined processors use a hierarchical branch prediction scheme, similar to how most modern processors now use hierarchical caches. These systems typically have a small, simple level-1 (L1) predictor (e.g., a branch target buffer) that can return a prediction within a single cycle. The second level-2 (L2) predictor is typically a much larger multilevel predictor (such as a the Alpha EV-8 direction predictor) that makes a much more accurate prediction, but requires two or more cycles to make the prediction. Both predictors are accessed for each branch. The processor uses the L1 predictor to begin speculative execution of the branch, but checks this prediction using the L2 predictor. If the L2 predictor disagrees with the L1, the branch is aborted and restarted using the L2 prediction. Assuming the L2 prediction is correct, the L1 misprediction penalty is much smaller than a full misprediction penalty.

These hierarchical predictors can be better than a large single-level predictor because the latter may have a larger penalty on a correctly predicted branch.

Consider a hierarchical predictor with the following performance:

L1 Prediction	L2 Prediction	Stall Cycles
Correct	Correct	0
Correct	Incorrect	11
Incorrect	Correct	3
Incorrect	Incorrect	8

Assume that one in 6 instructions are branch instructions and that the L1 and L2 predictions are independent.

Assume that all branch prediction stalls directly impact performance (as they might in the MIPS 5-stage pipeline). Thus we want to know the contribution to the CPI caused by branch mispredictions (i.e., the stall cycles per instruction).

Assume the L1 predictor is right 80% of the time and the L2 predictor is right 95% of the time.

16. How many stall cycles per instruction are due to either L1 or L2 mispredictions?
    1) 0.65
    2) 1.01
    3) 1.09
    4) 1.12
    5) None of the above

17. Which case contributes the most to stall cycles per instruction?
    1) L1 Correct, L2 Correct
    2) L1 Correct, L2 Incorrect
    3) L1 Incorrect, L2 Correct
    4) L1 Incorrect, L2 Correct
    5) None of the above

18. Which is more important?
    1) Improving the L1 predictor from 80% to 90% accurate?
    2) Improving the L2 predictor from 95% to 97% accurate?
    3) Reducing the penalty on L1 mispredict, L2 correct predict from 3 to 2 cycles?
    4) Reducing the penalty on L1 correct predict, L2 mispredict from 11 to 8 cycles?

Program Data Dependence Analysis (15 pts)

The following questions concern the pseudo-assembly-language program below. All arithmetic instructions have the form:

```
#       opc     rd =    rs1 op rs2
```

Where # refers to the instruction number, opc is the pneumonic opcode, rd is the destination register and rs1 and rs2 are the first and second register source operands respectively. Memory instructions have the format

```
#       sw      rs2 → mem[rs1 + imm]      // Store contents of rs2 to memory
#       lw      rd =  mem[rs1 + imm]      // Load memory into register rd
```

The questions will refer to the code examples using the notation i#.rX. The rX notation can either refer to a specific register name or the generic register specifier. For example, in the code below: i1.r3 refers to the destination register of the first add instruction and i4.rs1 refers to register r1 in the shiftl instruction.

In the following straight-line (no branches, no loops) assembly-language program, the questions below concern data dependences that the program contains.

```
1       add     r3 =    r4 + r5
2       sub     r3 =    r3 - r7
3       mul     r8 =    r4 x r9
4       shiftl  r9 =    r1 << r3
5       andi    r1 =    r3 & 0xff
6       add     r3 =    r9 + r8
```

19. The specifiers i1.rd and i2.rs1 have the following relationship:
    1) RAW dependence
    2) WAR dependence
    3) WAW dependence
    4) No dependence
20. The specifiers i1.rd and i2.rd have the following relationship:
    1) RAW dependence
    2) WAR dependence
    3) WAW dependence
    4) No dependence
21. The specifiers i1.rd and i4.rs1 have the following relationship:
    1) RAW dependence
    2) WAR dependence
    3) WAW dependence
    4) No dependence
22. The specifiers i2.rd and i6.rd have the following relationship:
    1) RAW dependence
    2) WAR dependence
    3) WAW dependence
    4) No dependence
23. How many RAW dependences are there in the entire code sequence
    1) 3
    2) 4
    3) 5
    4) None of the above

Instruction Scheduling (20 points)
Using the same pseudo-assembly language, but adding floating point (i.e., the ".d" suffix means double precision), the questions below involve instruction schedules.

```
1     lw.d       F0 = mem[R2 + 0]
2     mult.d     F3 = F0 * F2
3     sw.d       F3 → mem[R2 + 0]
4     lw.d       F4 = mem[R3 + 0]
5     mult.d     F5 = F4 * F2
6     sw.d       F5 → mem[R3 + 0]
7     add.d      F6 = F3 + F5
8     sw.d       F6 → mem[R4 + 0]
```

Assume the standard MIPS 5-stage (single-issue) pipeline plus separate pipes for floating point multiply and floating point add. All loads and stores use the integer pipeline and the floating-point pipeline does not have an "M" stage. Load-use delay is one cycle, regardless of which pipeline uses the result. Floating point multiply has 3 execute cycles and floating point add takes two execute cycles, and both are fully pipelined.

24. On what cycle does instruction 8 store its value to memory?
    1) Cycle 13
    2) Cycle 14
    3) Cycle 15
    4) Cycle 16
    5) Cycle 17
    6) None of the above
25. A compiler scheduler that is trying to reduce stalls might generate the following instruction schedule (using the instruction numbers):
    1) 1, 2, 3, 4, 5, 6, 7, 8
    2) 1, 2, 3, 4, 5, 7, 6, 8
    3) 1, 2, 3, 4, 7, 5, 6, 8
    4) 1, 4, 2, 5, 3, 6, 7, 8
    5) 1, 4, 7, 8, 2, 5, 3, 6
    6) None of the above
26. The key source of stalls in this code sequence is:
    1) The depth of the floating point multiply pipeline
    2) The depth of the floating point add pipeline
    3) The load-use delay
    4) A maybe dependence
    5) None of the above
27. If the instruction set were IA-64 instead of MIPS, we could reduce stalls using:
    1) An advanced load for instruction 1
    2) An advanced load for instruction 2
    3) A speculative load for instruction 1
    4) A speculative load for instruction 2
    5) None of the above

From the readings (20 points)

28. In Moore's classic paper semiconductor scaling, he predicted which of the following technologies might be made possible:
    1) Personal computers
    2) Cell phones
    3) Self-driving cars
    4) None of the above
    5) All of the above

29. Moore also predicted that the maximum number of transistors per chip would double:
    1) Every 12 months
    2) Every 18 months
    3) Every 24 months
    4) None of the above

30. In Wulf's paper on Compilers and Com[uter Architecture, he argues that "the failure general-register machines to treat all their registers alike" violates the following property:
    1) Regularity
    2) Orthogonality
    3) Composability
    4) None of the above
    5) All of the above

31. In Srinivasan, et al.'s paper on optimal pipeline depth, they argue that as the pipeline depth increases, the following changes also occur:
    1) FO4 per stage increases, average glitch factor increases
    2) FO4 per stage increases, average glitch factor decreases
    3) FO4 per stage decreases, average glitch factor increases
    4) FO4 per stage in decreases, average glitch factor decreases
    5) None of the above

32. In Seznec, et al.'s paper on the Alpha EV-8 branch predictor, the authors argue that partial update is superior to full update because:
    1) It limits the number of strengthened counters on a correct prediction.
    2) It doesn't steal a table entry if it can be avoided
    3) It utilizes space better.
    4) None of the above
    5) All of the above

33. The Intel IA-64 instruction set includes the following features:
    1) A register stack to pass arguments to and return values from subroutines
    2) Support for full predication
    3) Register renaming to support software pipelined loops
    4) None of the above
    5) All of the above

34. Mahlke, et al define *Hyperblocks* to be a collection of connected basic blocks with:

1) A single entry point and a single exit point
2) A single entry point and multiple exit points
3) Multiple entry points and a single exit point
4) Multiple entry points and multiple exit points
5) None of the above

35. In Mudge's paper on power, he says that clock gating

1) Is a poor design practice because of the effect on clock skew
2) Can eliminate significant dynamic power
3) Can eliminate significant static power
4) All of the above
5) None of the above

36. In Emer and Clark's classic paper on the VAX 11/780, they found that

1) Over 80% of the operand specifiers were "simple" specifiers
2) The average instruction size was 3.8 bytes
3) All of the above
4) None of the above

37. Iwai predicts that transistor scaling will

1) Likely end at the 7nm node due to high lithography costs
2) Likely end at the 5nm node due to high lithography costs
3) Likely end at the 7nm node due to high leakage current
4) Likely end at the 5nm node due to high leakage current
5) Never end

(blank page for additional work)