Last (family) name: ___Solution_____

First (given) name: _____

Student I.D. #: _____

Department of Electrical and Computer Engineering
University of Wisconsin - Madison

ECE/CS 752 Advanced Computer Architecture I

Midterm Exam 1

Wednesday, March 12, 2008

Instructions:

1. Open book/open notes.
2. **You must show your complete work**. Points will be awarded only based on what appears in your answers.
3. No one should leave the room during the last 5 minutes of the examination.
4. 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.
5. 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	15	
16	Variable-length Instruction Decoder Design	30	
17	Cache Memory Simulation	25	
18	Branch Prediction Discussion Questions	10	
Total		80	

Problems 1-15: (15 pts): Multiple choice; circle the best answer for each question

1. For a latch-based pipeline with a single-phase clock, the maximum operating frequency (minimum clock period) is determined by:
    a. The skew constraint and the short-path constraint
    b. The latching constraint
    c. The long-path constraint
    d. Both (b) and (c)
2. The main advantage of a PC-relative encoding of a branch target address is
    a. It's easier to pipeline than a register branch target
    b. More dense instruction encoding
    c. Ability to implement function call and return semantics
    d. Branch target can be on a different page in virtual memory
3. Cache misses experienced by a program are often classified as:
    a. Cold, compensatory, and conflicted
    b. Cold, capacity, and conflict
    c. Compulsive, cardinal, and capacious
    d. Cutaneous, correlated, and conspicuous
4. A data cache that is:
    a. Fully-associative experiences no conflict misses
    b. Direct-mapped experiences only conflict misses
    c. Set-associative always experiences fewer misses than one that is direct-mapped
    d. All of the above
    e. Both (a) and (b)
5. The write policy of a cache:
    a. May require all writes to be applied to further lower levels of cache
    b. May depend on a dirty bit to mark blocks that have been changed
    c. Can force some evicted blocks to be written to the next level of cache, while others are simply dropped
    d. All of the above
6. A pipelined processor that has a WAR register hazard
    a. Must have an earlier register read stage and a later register write stage
    b. Must have an earlier register write stage and a later register read stage
    c. Must have two stages that write registers, one later than the other
    d. None of the above
7. Control dependences in programs
    a. Are caused only by function calls and returns
    b. Arise due to loops and conditional statements
    c. Always induce pipeline bubbles in a pipelined processor
    d. Can be resolved by adding a branch delay slot to the instruction set architecture
    e. All of the above
8. A superscalar processor implementation
    a. Provides better performance than a scalar pipeline
    b. Fetches multiple instructions per cycle
    c. Requires code compiled for a scalar pipeline to be recompiled
    d. Needs to have a dynamic branch predictor

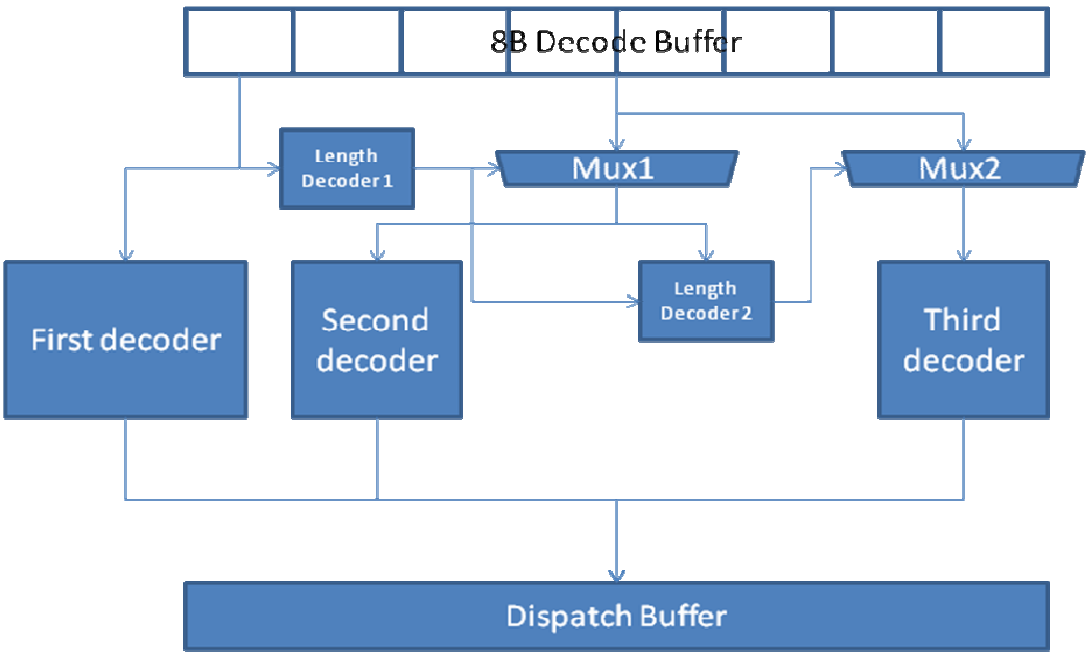
9. A static branch predictor cannot:
    a. Predict all branches not-taken
    b. Predict forward branches not-taken and backwards branches taken
    c. Predict a branch at PC location 0xab04 always taken and a branch at PC location 0xbc24 as always not-taken
    d. Predict a branch at PC location 0xbc48 as taken the first five times it executes, then not-taken the sixth time it executes
10. A branch that is mispredicted as taken requires the processor control logic to:
    a. Restart fetching instructions from the not-taken path
    b. Clear out all instructions that were tagged with the mispredicted branch's tag
    c. Flush the fetch and dispatch buffers
    d. All of the above
    e. Both (a) and (b)
11. An out-of-order processor resolves memory WAR/WAW hazards by
    a. Delaying all load instructions until the commit stage
    b. Delaying all store instructions until the commit stage
    c. Setting the dirty bit in the cache speculatively
    d. All of the above
12. An out-of-order processor handles memory RAW hazards by
    a. Satisfying all loads with data from the store queue
    b. Setting the dirty bit in the cache speculatively
    c. Satisfying some loads with data from the store queue
    d. Delaying all loads until the commit stage
13. The MIPS R10000 processor
    a. Uses a separate architected register file and rename register file
    b. Uses a single physical register file
    c. Allows multiple unresolved predicted branches to coexist in the machine
    d. Both (b) and (c)
14. The Distributed Register Algorithm (DRA) proposed in the loose loops paper from the reading list
    a. Reduces the number of physical registers in the processor
    b. Increases the number of pipeline stages between issue and execute
    c. Relies on a history file to guarantee precise exceptions
    d. Adds a register cache that keeps only a subset of register values near the execution units
15. The process of decoding a single programmer-visible instruction into multiple internal micro-ops:
    a. Simplifies the control logic in an out-of-order superscalar execution core
    b. Was first implemented in the Intel Pentium Pro in 1995
    c. Was first proposed by Hwu, Patt, and Shebanow in their HPS paper in 1985
    d. All of the above
    e. Both (a) and (b)

16. Variable-length Instruction Decoder Design (30 pts)

As discussed in lecture, the Pentium Pro instruction decoder has a 4-1-1 configuration, where the first decoder can generate up to four internal uops from the first macro-instruction in the decode buffer, and the second and third decoders can generate one uop each for the second and third macro-instructions. Unfortunately, the macro-instructions can vary in length, hence complicating the task of the second and third decoders, since they must either wait for the earlier decoder(s) to determine their instruction length so they know which bytes to decode, or they must decode all possible bytes in parallel and then choose the correct one once the earlier decoders have determined the length(s) of their instruction(s).

In this problem, you are to assume a simplified ROM lookup-table-based decoder implementation, and then reason about the design of a 4-1-1 decoder similar to the one in the Pentium Pro. Rely on the following assumptions:

- The fetch unit presents 8 bytes of instructions to the decoder in the decode buffer
- Individual instructions can be 1, 2, or 4 bytes in length.
- All 1- and 2-byte instructions are decoded into a single uop, and can be handled by the secondary decoders.
- All 4-byte instructions are decoded into two, three, or four uops, and can only be handled by the first decoder.
- The instruction set contains 60 instructions, encoded using a 6-bit opcode field in the first byte of each instruction.
- The 6 bits of opcode uniquely determine the length of the instruction (expressed as 2 bits that encode 1, 2, or 4 bytes as 00, 01, or 10). The length is determined by indexing into a ROM lookup table using the opcode bits and reading out the corresponding 2-bit entry.
- The same 6 bits of opcode uniquely determine the 1-4 uops that are dispatched into the execution core. The uops are determined by indexing into a lookup table using the opcode bits and reading out the corresponding uop bits.
- Each uop is 64 bits in length.
- All decoder blocks are implemented as ROM lookup tables.



a. Given 8 bytes numbered B0-B7 in the decode buffer, with the opcode of the first instruction at byte B0, what are all the possible locations for the opcode byte of the 2nd instruction? What about the 3rd instruction? I.e. which bytes are the inputs to each of the two muxes shown in the diagram? (5 pts)

2nd instruction (inputs to Mux1): B1, B2, B4

3rd instruction (inputs to Mux2): B2, B3, B4, B5, B6

b. Determine the total number of bits in each of the ROM lookup tables in the diagram as the product of the number of rows and the bits per row (10 pts):

First Decoder: 2^6 x (4 x 64) = 16K

Second Decoder: 2^6 x 64 = 4K

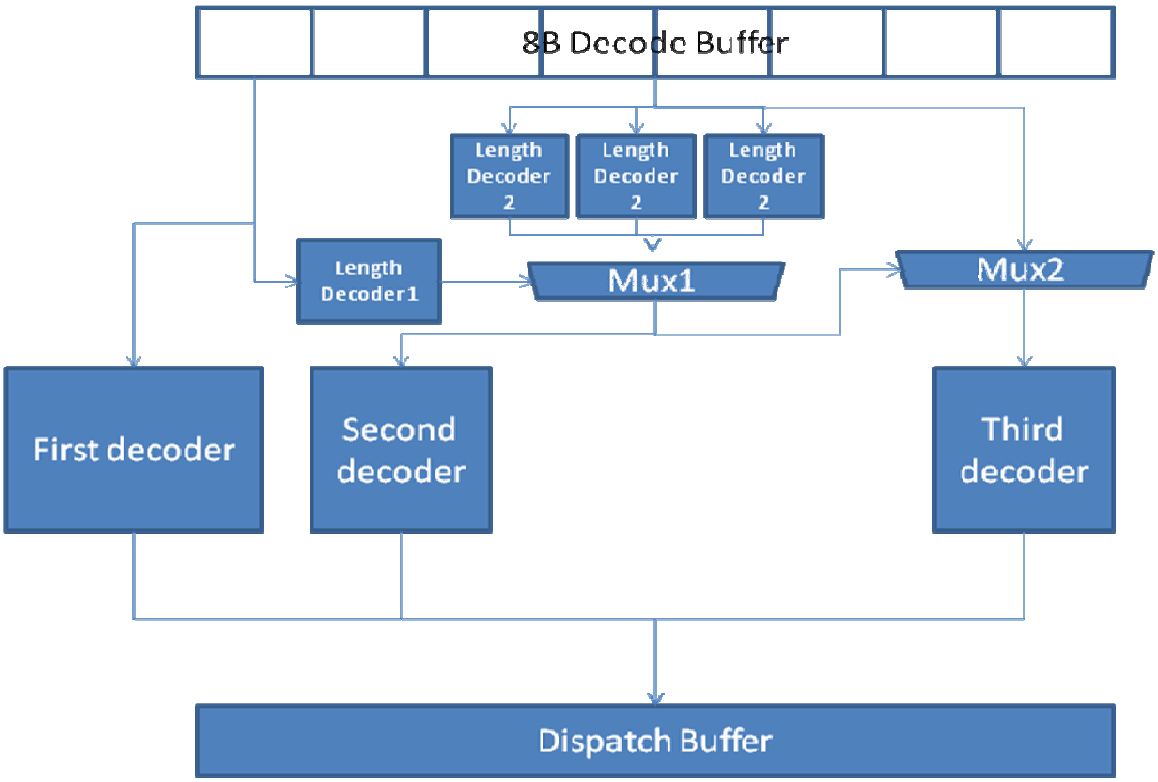
Third Decoder: 2^6 x 64 = 4K

Length Decoder 1: 2^6 x 2 = 128

Length Decoder 2: (3 x 2^6) x 3 bits/entry = 576 bits (or 768 if restricted to power of two)

c. Assume the critical delay path is <Length Decoder1, Length Decoder 2, Mux2, Decoder3>. To shorten the critical path, *Length Decoder 2* could be replicated (once for each possible location of the opcode byte of the second instruction) and moved above Mux1. Draw a diagram for this new organization, compute the total size of the lookup tables needed to replace Length Decoder 2 in this new arrangement, and identify what is likely to be the new critical path.

Diagram (10 pts):



Total size of Length Decoder 2 lookup tables (show your work) (4 pts):

$3 \times 2^6 \times 3$ bits/row = 3 x 204 = 612 bits

Likely new critical path or paths through entire decoder (1 pt):

LD1 -> Mux1 -> Mux2 -> D3 or
LD2 -> Mux1 -> Mux2 -> D3

17. Cache Memory Behavior (25 pts)

Given the following code, assume that all variables except array locations reside in registers, and that array A begins at address 0x0, arrays B and C are placed consecutively after A in memory, and that each "double" array entry consumes 8 bytes (64 bits). All caches are initially empty.

```
double A[1024], B[1024], C[1024];
for(int i=0; i<1000; i++) {
    A[i] = 35.0 * B[i] + C[i];
}
```

There are 3000 memory references (1000 each to A, B, C). For (a)-(c), assume an in-order processor where all references are performed in strict program order following total memory ordering rules (load B[i]; load C[i]; store A[i]; then load B[i+2]; load C[i+2]; store A[i+2]; and so on).

a. Assuming a virtually-addressed fully-associative data cache with 8KB capacity, 32-byte blocks, and an LRU replacement policy, count the number of cache misses.

Number of cache misses (5 pts)**:**

All cold misses. 4 refs/line, one miss per line, 25% miss rate => .25 x 3000 = 750 misses

b. Assuming a virtually-addressed direct-mapped cache with 8KB capacity and 32-byte blocks, compute the number of cache misses.

Number of cache misses (5 pts)**:**

Cold misses (as above), but no temporal reuse, since A/B/C conflict in the cache. 100% miss rate => 3000 misses

c. Assuming a virtually-addressed two-way set associative cache with 8KB capacity, 32-byte blocks, and an LRU replacement policy, compute the number of cache misses.

Number of cache misses (5 pts)**:**

Cold misses (as above), but no temporal reuse, since A/B/C conflict in the cache. 100% miss rate => 3000 misses

d. Repeat (c) assuming an out-of-order execution window with 4 store buffer entries and 8 load buffer entries where stores don't access the cache until they commit. Further assume that loads always receive issue priority over stores, so any load currently in the instruction window will issue before the oldest store commits.

Number of cache misses (10 pts)**:**

Cold misses as in part (a). But all 8 loads from first 4 iterations issue before conflicting store issues from store buffer, so 6/8 are hits. Also, trailing 3 stores in store buffer do not conflict with new loads from following iterations, so ¾ stores also hit. Overall hit rate is 75%, so .25 x 3000 = 750 total misses

Branch Prediction Discussion Questions (10 pts)

a. Name the different types of branch history used in dynamic branch predictors and explain what branch behaviors or types of branches each history can reliably predict (4 pts).

Bimodal – captures branches that are predominantly taken (or not taken)
Local history – detects patterns for a single branch and given context can predict some instances taken, others not (e.g. loop closing)
Global history – detects patterns across all branches and can find correlations between e.g. two if statements checking for the same or similar condition
Path history – can differentiate histories in some cases, providing additional context based on path used to reach current branch.

b. Explain the principle of operation of the loop count predictor, and describe a scenario where it will be useful (2 pts).

Converts BHR to counting mode when all entries are 0 or 1. Logarithmic loop count enables capturing loops with modest number of iterations by pointing to a different second-level PHT entry for the loop closing branch.

c. The agree, bi-mode, and YAGS predictors all attack the same problematic issue in branch prediction. Identify this issue and briefly describe how each predictor can reduce its impact (4 pts).

All three tackle negative aliasing/interference. Agree uses static bias prediction, and since most branches "agree" with it, most PHT entries saturate in same direction. Bi-mode separates mostly-taken and mostly-NT into separate tables, which YAGS records only the exceptions in small tagged tables.