EECS 370

Exam 2

Name:

Uniqname:

Open book, open notes, calculators permitted. No laptops, cellphones, PDA's, etc. This exam has 9 questions, 11 pages, and 100 points.

Extra copies of the pipeline diagram from page 8 are available in case you feel you have hopelessly ruined yours. If you use an extra sheet, please clearly put your uniqname on it, and please cross out that page in the stapled exam.

1. *(16 points, 2 points each)* Indicate (by circling TRUE or FALSE) whether or not the following statements are true or false:

   (a) TRUE / FALSE : Out-of-order execution is better at tolerating data cache misses than instruction cache misses.

   (b) TRUE / FALSE : In LC2K3, resolving branches in the EX stage rather than the MEM stage will increase CPI.

   (c) TRUE / FALSE : As branch prediction accuracy increases, the time to access the branch predictor increases while CPI decreases.

   (d) TRUE / FALSE : Pipelines with more stages make predication more beneficial.

   (e) TRUE / FALSE : Spatial locality states that recently accessed items are likely to be accessed in the near future.

   (f) TRUE / FALSE : An LRU replacement policy will always produce a lower or equal miss rate than a random replacement policy.

   (g) TRUE / FALSE : In an N-way set-associative cache, there are N sets of cache lines.

   (h) TRUE / FALSE : Larger cache line sizes take better advantage of spatial locality.

2. *(12 points, 2 points each)* For each question, circle the BEST answer out of **F**, **D**, **S**, or **N** as follows:

   - **F:** Fully associative cache
   - **D:** Direct mapped cache
   - **S:** Set associative cache
   - **N:** None of the above; there is no significant difference between the above three cache types (or the question makes no sense)

   In making comparisons, assume that all caches have the same amount of data storage.

   F **D** **S** **N** (a) Needs the fewest tag bits.

   F **D** **S** **N** (b) Does not support writeback caching modes.

   F **D** **S** **N** (c) Has the fewest conflict misses.

   F **D** **S** **N** (d) Has the fewest compulsory misses.

   F **D** **S** **N** (e) Has the fewest capacity misses.

   F **D** **S** **N** (f) Needs the least amount of hardware to implement.

3. Please answer the following questions about a cache organized as follows:

   - Block size: 2 bytes
   - Total cache data size: 8 bytes
   - 2-way set associative
   - LRU replacement policy

   (a) *(5 points)* Given the word references below, mark each reference as a hit (H) or miss (M). Each reference is a load of a single byte.

Address	2	5	3	0	7	13	12	9	1	9	2	6
H/M	M											

   (b) *(3 points)* What is the hit rate of the reference sequence?

   (c) *(3 points)* What is the average access latency (measured in cycles) of this reference stream if hits take 1 cycle and the miss penalty is 8 cycles?

4. Consider a cache with the following characteristics:

   - 32-bit memory addresses
   - Byte addressable
   - 64 kilobyte cache
   - 64 byte cache block size
   - Write-allocate, write-back cache

   This cache will need 512 kilobits for the data area (64 kilobytes times 8 bits per byte). Note that in this context, 1 kilobyte = 1024 bytes (NOT 1000 bytes!)

   Besides the actual cached data, this cache will need other storage. Consider tags, valid bits, dirty bits, bits to keep track of LRU, and anything else that you think is necessary. Show your work if you wish to be eligible for partial credit.

   (a) *(4 points)* How many additional bits (not counting the data) will be needed to implement this cache if it is direct mapped?

   (b) *(5 points)* How many additional bits (not counting the data) will be needed to implement this cache if it is 2-way set associative?

5. Consider the following fragment of code running on the pipelined implementation of the LC2K3. Assume everything is exactly as it was in project 3; the same rules apply for forwarding and stalling as in the project.

```
start   nand  1  1  2        _____________________
        beq   1  2  target   _____________________
        lw    2  5  data     _____________________
        add   2  5  6        _____________________
target  sw    0  6  data     _____________________
        nand  1  2  3        _____________________
        nand  6  5  4        _____________________
        nand  4  3  7        _____________________
        nand  3  4  1        _____________________
        nand  2  4  2        _____________________
        halt                 _____________________
data    .fill 0
```

(a) *(8 points)* In the blanks shown after the instructions, list all of the pipeline registers (IFID, IDEX, EXMEM, MEMWB, WBEND) that contain values that must be forwarded into the EX stage when that particular instruction is in the EX stage. If it makes a difference to you, you may assume that registers contain zero values initially.

(b) *(3 points)* Do any instructions in this code have to stall? If so, which one(s)?

6. Pipeline Performance

For this problem — specify all answers to **two digits beyond the decimal point.**

Consider the normal 5-stage LC2K3 pipeline discussed in class. Branches are resolved in the MEM stage and "predict not taken" is used. One half of all load instructions will incur a one cycle stall due to data hazards that can't be resolved via forwarding. Assume the instruction and data caches have perfect (100%) hit rates. Analysis has shown the following dynamic instruction breakdown for program X: 15% taken branches, 10% not taken branches, 25% loads, 15% stores, 35% ALU.

(a) *(5 points)* What is the CPI of this machine when running program X?

To avoid control hazards, the designers of the LC2K3 have decided to replace the `beq` instruction with a delayed branch. This means that both the compiler and underlying architecture have been modified such that the three instructions after the delayed branch will automatically execute regardless if the branch is taken or not taken. Program X has been recompiled to support this change and the compiler was able to fill two of the three delayed branch slots with useful instructions. Now, only 40% of all loads incur a one cycle stall due to data hazards. The clock rate is unchanged.

(b) *(7 points)* How many times faster is the performance of program X on the machine using delayed branches compared to running the program on the machine using normal branches? (Note: If version A takes 2 ns and version B takes 5 ns, then version A is 2.50 times faster than version B.)

7. Superscalar Pipelines

The CS370 computer is a superscalar machine that implements the LC2K3 instructions. It contains two pipelines each with the following five stages:

Stage 1 (IF) Instruction Fetch: Behaves in a similar fashion as described in class. Two instructions are fetched each cycle. A "predict not taken" scheme is used for branch prediction. The instruction cache has a perfect (100%) hit rate.

Stage 2 (RR) Register Read: Only two things are done in this stage: registers are read from the register file and the 16-bit offset is converted into a 32-bit signed offset. No hazard detection is done during this stage. **Instructions may only enter the RR stage in groups of two.**

Stage 3 (AG) Address Generation: This stage will compute the effective address for loads and stores and the target address for branches. It is responsible for all hazard detection and generating stall signals when necessary. **Instructions may only enter the AG stage in groups of two.**

Stage 4 (EM) Execute and Memory: This stage will serve as the execute stage for ALU and branch instructions and the memory stage (using the address computed in the previous cycle) for loads and stores. Due to an innovative memory design, both pipelines are allowed to do memory operations even if their access is to the same address. Branches are resolved in this stage. Instructions cannot stall in this stage. The data cache has a perfect (100%) hit rate.

Stage 5 (WB) Writeback: This stage is just like writeback as described in class. The register file is written with the computed result in the first half of the clock cycle. Register reads (in the RR stage) occur in the second half of the clock cycle.

When possible, data hazards are resolved by forwarding. Full forwarding exists to both the AG and EM stages.

(a) *(8 points)* On the next page, fill in the pipeline diagram for the given code sequence assuming that the branch is not taken (reg3 does not equal reg4 at that point in the program). Some of the diagram has been started for you. You may represent stalls in any way you like.

(b) *(2 points)* How many cycles does it take to execute this code sequence?

Cycles	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
add 1 2 3	IF	RR	AG	EM	WB															
add 4 5 6	IF	RR	AG																	
lw 3 5 -1																				
add 5 2 4																				
sw 1 4 -1																				
nand 4 1 7																				
add 7 3 4																				
sw 4 7 -2																				
lw 5 3 14																				
beq 3 4 3																				
add 1 2 3																				
nand 3 4 5																				

8. Pipelining sans hazard-detection

This question deals with a pipelined implementation of the LC architecture that does not have logic to detect data hazards. Therefore, it is up to the user to write assembly code that does not have any data hazards (in reality, this would be the compiler's job).

Remember that besides forwarding, there are 2 other ways to deal with data hazards:

(i) introducing noops

(ii) re-ordering instructions

Consider the following LC code:

```
        lw  0 7 n1
        lw  0 6 n2
        lw  0 1 n3
        lw  0 2 n4
loop    sw  0 7 tmp
        add 1 7 1
        add 2 6 2
        add 7 7 7
        beq 1 2 end
        beq 0 0 loop
end     halt
n1      .fill A
n2      .fill B
n3      .fill C
n4      .fill D
tmp     .fill 0
```

A, B, C, D are any 32 bit signed integers.

Assume that the register file can be written to during the first half of the clock cycle and read from during the second half. Also assume that a "predict not taken and squash if wrong" policy is used for branches.

(a) *(4 points)* Rewrite the above code, inserting the minimum number of noops so that there are no data hazards. You may not re-order the given instructions; you may only insert noops between them.

(b) *(6 points)* (continuation of question from previous page)

Is it possible to re-order the instructions so that there are no data hazards? If yes, provide such an ordering. If not, re-order the instructions and introduce noops so that the number of noops executed is minimized. For your convenience, the code sequence from the previous page is repeated below.

```
        lw  0 7 n1
        lw  0 6 n2
        lw  0 1 n3
        lw  0 2 n4
loop    sw  0 7 tmp
        add 1 7 1
        add 2 6 2
        add 7 7 7
        beq 1 2 end
        beq 0 0 loop
end     halt
n1      .fill A
n2      .fill B
n3      .fill C
n4      .fill D
tmp     .fill 0
```

9. A comparator is a combinational logic circuit that compares two binary numbers and determines whether they are equal. The number of bits in each input number is called the width of the comparator. For example, an 8 bit wide comparator takes two 8 bit binary numbers and inputs and produces a single bit of output (that indicates equal or not equal).

   (a) *(3 points)* To actually build the pipelined LC2K3 with forwarding as simulated in project 3, how many comparators are necessary specifically for control of the forwarding?

   (b) *(3 points)* How wide are the comparators?

   (c) *(3 points)* How many additional comparators are necessary for control of the one cycle stall that occurs under certain conditions?