CMSC411 Spring 2011 Midterm 2

Name: _____

Instructions

- You have 50 minutes to take this exam.
- There are 100 points in this exam, so spend about 30 seconds per point.
- You do not need to provide a number if you can show the appropriate fraction. E.g., 1/13 is acceptable in place of .0769.
- This is a closed book exam. No notes or other aids are allowed.
- If you have a question, please raise your hand and wait for the instructor.
- Answer essay questions concisely using 1-2 sentences. Longer answers are not necessary and a penalty may be applied.
- In order to be eligible for partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.

	Problem	Score
1	Memory Hierarchy	/24
2	Cache Organization	/8
3	Dynamic Scheduling	/36
4	Speculation	/8
5	Improving CPI	/12
6	Limits to ILP	/12
	Total	/100

Instruction	Effect
LD F1, 0(Rx)	F1 ← Mem(Rx)
ADD.D F1, F2, F3	F1 ← F2 + F3
MULT.D F1, F2, F3	F1 ← F2 * F3

Power of 2	Value
$\mathbf{2^4}$	16
$\mathbf{2^5}$	32
$\mathbf{2^6}$	64
$\mathbf{2^7}$	128
$\mathbf{2^8}$	256
$\mathbf{2^9}$	512
$\mathbf{2^{10}}$	1K
$\mathbf{2^{20}}$	1M
$\mathbf{2^{32}}$	1G

1. (24 pts) Memory hierarchy

    a. (4 pts) Give a reason why increasing cache block size may improve performance.

    b. (4 pts) Give a reason why increasing cache block size may reduce performance.

    c. (4 pts) Give a reason why a write-through cache may cause less memory traffic than a write-back cache.

d. (4 pts) Name a compiler technique for improving cache performance, and explain why that particular technique can reduce cache misses.

e. (4 pts) Describe interleaved memory banks

f. (4 pts) Explain why interleaving memory banks may improve performance.

2. (8 pts) Cache organization

Suppose we have a byte addressable memory of size 1GB (2^{30} bytes).

a. (6 pts) We are given a cache of size 2MB (2^{21} bytes, not including tag bits) and a cache block size of 32 (2^5) bytes. Compute for a 2-way associative cache the length in number of bits for the tag, index and offset fields of a 30-bit memory address (show your calculations)

b. (2 pts) Considering the answer to part (a), circle the bits representing the index in the following 30-bit memory address (in binary):

1 1 0 1 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 1 1 1

3. (40 pts) Dynamic scheduling

Consider the execution of a single-issue Tomasulo-style CPU. Assume the following:

- The CPU has 1 of each: load buffer, FP adder, FP multiplier functional unit.
- An unlimited number of reservation stations for each functional unit & load buffer.
- Functional units are not pipelined.
- No forwarding between functional units; results can only come from the CDB.
- **If multiple instructions attempt to use the CDB in the same cycle, the instruction issued earliest goes first**.

For each problem, show what clock cycle each instruction is issued and when it begins execution (i.e., enters its first EX cycle). Also show when each instruction writes the CDB. If an instruction execution or completion stalls, **list the length of stall and provide a reason for the stall(s)**.

Instruction execution times (latencies) are provided as +n cycles, where an instruction executes in 1+n cycles (i.e., spends 1+n cycles in the EX stage). For example: an instruction with latency +2 executed at cycle 4 would finish in cycle 6 (4+2) and put its result on the CDB in cycle 7 (if CDB is not busy).

a. (18 pts) Assume instructions take the following time:

Instruction	Latency
Memory LD	+1
ADD.D	+0
MULT.D	+2

Instruction	Cycle #			Stalls
	Issue	Exec	Write CDB	
I1: LD F1, 0(Rx)	1	2	4	
I2: MULT.D F2, F1, F1				
I3: LD F3, 8(Rx)				
I4: ADD.D F2, F2, F3				
I5: ADD.D F2, F1, F1				
I6: MULT.D F4, F1, F2				

b. (18 pts) Assume instructions take the following time:

Instruction	Latency
Memory LD	+3
ADD.D	+0
MULT.D	+2

Instruction	Cycle #			Stalls
	Issue	Exec	Write CDB	
I1: LD F1, 0(Rx)	1	2	6	
I2: MULT.D F2, F1, F1				
I3: LD F3, 8(Rx)				
I4: ADD.D F2, F2, F3				
I5: ADD.D F2, F1, F1				
I6: MULT.D F4, F1, F2				

4. (8 pts) Speculation
   a. (4 pts) The instruction ADD.D F3, F1, F2 is supposed to put the result of F1+F2 in the register F3. How can the instruction be executed speculatively, calculating the value of F1+F2, yet be later canceled without affecting F3?

   b. (4 pts) In the previous example, how can other instructions access F3 to use the value of F1+F2 calculated by the ADD.D instruction before it is known whether the instruction will be committed?

5. (12 pts) Improving CPI
    a. (4 pts) What are VLIW architectures?

    b. (4 pts) Explain how VLIW architectures can increase performance compared to dynamic scheduling, even if the number of functional units are identical.

    c. (4 pts) Explain how predicated instructions may improve performance.

6. (12 pts) Limits to ILP
    a. (4 pts) Why are there any limits to ILP, given unlimited resources?

    b. (4 pts) What is simultaneous multithreading (SMT)?

    c. (4 pts) Why can SMT improve on single-threaded ILP, even with unlimited resources?