McGill University
ECSE 425
COMPUTER ORGANIZATION AND ARCHITECTURE
Fall 2011
Final Examination

9 AM to 12 NOON, December 9th, 2011

<u>Duration: 3 hours</u>

- **Write your name and student number in the space below. Do the same on the top of each page of this exam.**
- **The exam is 14 pages long. Please check that you have all 14 pages.**
- **There are five questions for a total of 100 points. Not all parts questions are worth the same number of points; read the whole exam first and spend your time wisely!**
- **This is a closed-book exam. You may use two double-sided sheets of notes; please turn these sheets in with your exam.**
- **Faculty standard calculators are permitted, but no cell phones or laptops are allowed.**
- **Clearly state any assumptions you make.**
- **Write your answers in the space provided. Show your work to receive partial credit, and clearly indicate your final answer.**

Name: ______________________________

Student Number: ______________________________

Q1: __________/20	**Q3: __________/15**	**Q5: __________/20**
Q2: __________/30	**Q4: __________/15**	

Total:

Name: ID:

Question 1: Short Answer (20 pts; 2 pts each)

a. What is the "principle of locality"?

The principle of locality states that (a) recently used data will be used again, and (b) when one datum is used, it's neighbors in memory are also likely to be used.

b. Ideal scalar pipelines achieve a CPI of 1; explain why this ideal is rarely achieved.

This ideal is rarely achieved because pipelines must stall to address hazards: structural hazards (resulting from there being too few resources to simultaneously satisfy the execution requirements of a particular set of executing instructions), data hazards (due to data dependencies), and control hazards (due to not knowing what instruction to fetch until several instructions into the execution of a branch instruction).

c. Under what circumstances can a WAW hazard occur in the standard MIPS FP pipeline?

When instructions writing the same destination register complete out of order; e.g., DIV F2, F4, F6 precedes ADD F2, F8, F10 in program order, but ADD finishes first.

d. What hazard is caused by an anti-dependence?

WAR

e. What does a correlating branch predictor correlate?

Past branch outcomes are correlated with the current branch prediction.

Name: ID:

f. When cache size is fixed, why do miss rates initially decrease as cache block sizes increase (e.g., from 16B to 64B for small L1 caches)?

Spatial locality—there are fewer compulsory misses because spatial locality says that the data in the (growing) blocks will likely be used.

g. When cache size is fixed, why do miss rates eventually increase as cache block size increases (e.g., from 64B to 256B for small L1 caches)?

Insufficient spatial locality—not all data in the blocks are used, and as a result, unused data forces out data that is still useful (i.e., there are more capacity or conflict misses).

h. Describe one advantage of simultaneous multi-threading over coarse grained multi-threading.

SMT can issue instructions from several threads at the same time, better utilizing the functional units in the core than a single thread can. SMT can also prioritize the execution of a single thread (filling in FUs only after that thread has issued as many instructions as possible), limiting the slowdown experienced by the "preferred" thread.

i. Describe a situation in which multiprocessor caches are incoherent.

If two processor caches have different values for the same memory address, the caches are incoherent.

j. What causes false sharing?

False sharing occurs when, as a result of block size, two processors compete for the right to read or modify the same block while accessing different words within that block.

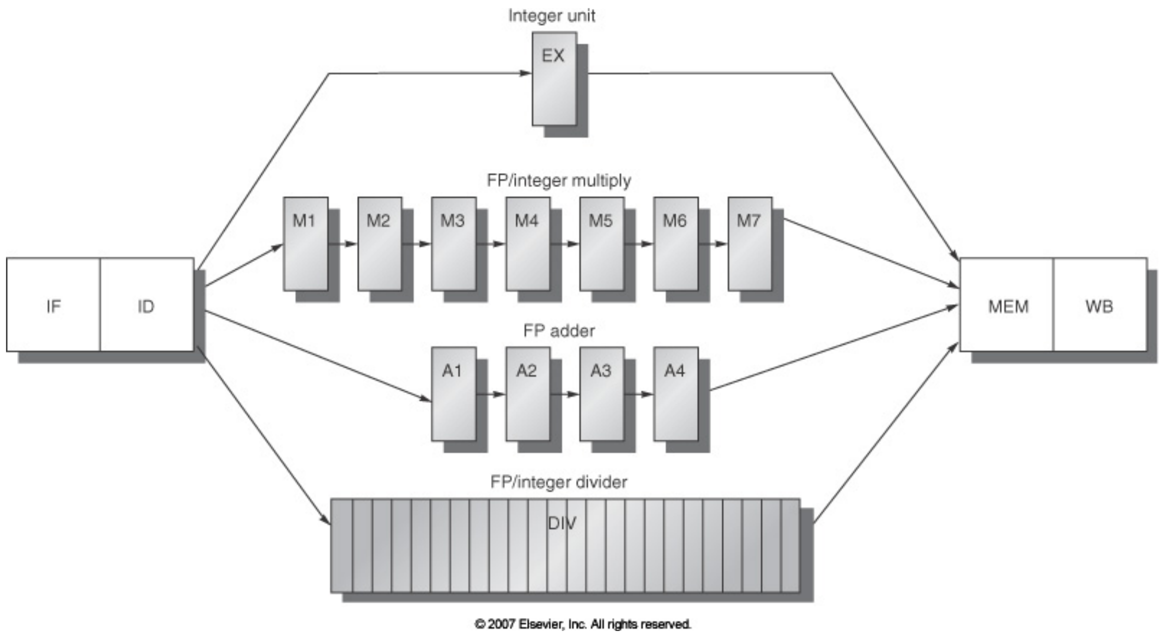
Name: ID:

Question 2: MIPS FP Pipeline (30 pts)

Consider the following loop that computes performs the Fast Fourier Transform.

```
Loop:   L.D      F2, 0(R1)
        L.D      F4, 8(R1)
        L.D      F6, 16(R1)
        MULT.D   F2, F2, F6
        ADD.D    F6, F4, F2
        SUB.D    F8, F4, F2
        S.D      F6, 0(R2)
        S.D      F8, 8(R2)
        DSUBUI   R1, R1, #24
        DSUBUI   R2, R2, #16
        BNEZ     R1, Loop
```

This code will run on the standard MIPS floating point pipeline, illustrated below.



(a) (8 pts) Assume:

- Full hazard detection and forwarding logic;
- The register file supports one write and two reads per clock cycle;
- Branches are resolved in ID; branches are handled by flushing the pipeline;
- Memory accesses take one clock cycle;
- Two or more instructions may simultaneously pass through MEM (WB) as long as only one makes use of the memory (register file).
- Structural hazards are resolved by giving priority to older instructions.

Fill in the chart on the next page to show the timing of one loop iteration.

Name: ID:

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
LD F2,0(R1)																									
LD F4,8(R1)																									
LD F6,16(R1)																									
MULT.D F2,F2,F6																									
ADD.D F6,F4,F2																									
SUB.D F8,F4,F2																									
S.D F6,0(R2)																									
S.D F8,8(R2)																									
DSUBUI R1,R1,#24																									
DSUBUI R2,R2,#16																									
BNEZ R1, Loop																									

(b) (16 pts) Unroll the loop once and re-schedule the code to minimize delay. Assume the branch delay is managed using a delayed branch slot. In the space below, write out the re-scheduled instructions, including their operands.

Name: ID:

(c) (4 pts) What is the speedup of your re-scheduled code?

(d) (2 pts) How many times must the loop be unrolled before it executes without stalling?

Name: ID:

Additional Page for Q2

Name: ID:

Question 3: (15 pts) Branch Prediction

Two machines A and B are identical except for their branch prediction schemes. Each machine uses a seven-stage pipeline:

IF ID1 ID2 EX1 EX2 MEM WB

Machine A uses a static predicted not-taken scheme. Machine B uses a static predicted taken scheme. In each case, branch targets are calculated in ID1, and branch conditions are resolved in EX1.

If 25% of instructions are conditional branches, what fraction of these branches must be taken for machine A and machine B to have equivalent performance? Assume each processor achieves the ideal CPI for every other instruction other than conditional branches.

Question 4: (15 pts) Cache Hierarchy

For a given benchmark running on a given processor, 25% of the instructions are loads, and 10% are stores. Using that benchmark, we want to choose between a direct-mapped cache and a 2-way set associative cache.

In each case, assume that the cache hit time is 1 cycle and the memory access penalty is 100 ns. Whenever the cache needs to replace a block, there is a 25% that the block it is replacing is dirty.

The processor clock speed with the direct-mapped cache is 1 GHz. Because of the extra time required for tag search, the clock cycle time with the 2-way set associative cache is 1.1 times greater than that with the direct-mapped cache.

Assume a base CPI of 1. Using the miss rates in the table below, calculate the CPU time for both the direct-mapped and 2-way set associative cache architectures.

Cache	**Read miss rate**	**Write miss rate**
Direct-mapped	3%	8%
2-way set associative	2%	6%

Name: ID:

Question 5: (20 pts) Cache Coherence

Consider a four-processor directory-based distributed shared-memory system. Each processor has a single direct-mapped cache that holds four blocks, each containing two words. To simplify the illustration, the **cache-address tag contains the full address (in hexadecimal)** and each word shows only two hexadecimal characters, with the least significant word on the right. The cache states are denoted M, S, and I for Modified, Shared, and Invalid. The directory states are denoted DM, DS, and DI for Directory Modified, Directory Shared, and Directory Invalid (Uncached). This simple directory protocol uses messages given in the table on the next page.

Assume the cache contents of the four processors and the content of the main memory as shown in the figure below. A CPU operation is of the form

P#: <*op*> <*address*> [<-- <*value*>]

where P# designates the CPU (e.g., P0), <op> is the CPU operation (e.g., read or write), <address> denotes the memory address, and <value> indicates the new word to be assigned on a write operation.

For the sequence of CPU operations given below, **show the changes in the contents** of the caches and memory after the each operation has completed (including coherence state, tags, and data). For each operation, **show the resulting sequence of coherence messages** placed on the bus**.** Refer to the table and the suggested message format on the following page.

P0				**P1**				**P2**				**P3**			
state	tag	data		state	tag	data		state	tag	data		state	tag	data	
I	100	26	10	I	100	26	10	S	120	02	20	S	120	02	20
S	108	15	08	M	128	2D	68	S	108	15	08	I	128	43	30
I	110	F7	30	I	110	6F	10	M	110	76	01	M	130	64	00
I	118	C2	10	S	118	3E	18	I	118	C2	10	I	118	40	28

	Memory			
address	state	Sharers	Data	
100	DI		20	00
108	DS	P0,P2	15	08
110	DM	P2	6F	10
118	DS	P1	3E	18
120	DS	P2,P3	02	20
128	DM	P1	3D	28
130	DM	P3	01	30

P0: read 130
P3: write 114 <-- 0A

Name: ID:

Message type	Source	Destination	Message contents	Function of this message
Read miss	Local cache	Home directory	P, A	Processor P has a read miss at address A; request data and make P a read sharer.
Write miss	Local cache	Home directory	P, A	Processor P has a write miss at address A; request data and make P the exclusive owner.
Invalidate	Local cache	Home directory	A	Request to send invalidates to all remote caches that are caching the block at address A.
Invalidate	Home directory	Remote cache	A	Invalidate a shared copy of data at address A.
Fetch	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; change the state of A in the remote cache to shared.
Fetch/invalidate	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; invalidate the block in the cache.
Data value reply	Home directory	Local cache	D	Return a data value from the home memory.
Data write back	Remote cache	Home directory	A, D	Write back a data value for address A.

P = requesting processor number, A = requested block address, and D = data contents

To show the bus messages, use the following format:

Bus {message type, requesting processor or directory, block address, data}
Example: Bus {read miss, P0, 100, --}

To show the contents in the cache of a processor, use the following format:

P# <block #> {state, tag, data}
Example: P3 <block 0> {S, 120, 02 20}

To show the contents in the memory, use the following format:

M <address> {state, [sharers], data}
Example: M <120> {DS, [P0, P3], 02 20}

a. (8 pts) P0: read 130

b. (12 pts) P3: write 114 <-- 0A

Name: ID:

Additional page