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

Examiner: Prof. B. H. Meyer Associate Examiner: Prof. W. J. Gross →

9 AM to 12 NOON, December 9th, 2011

Duration: 3 hours

- 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 17 pages long. Please check that you have all 17 pages.
- There are six questions for a total of 120 points. Not all parts of all 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.
- 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: Solutions

Student Number: ____________

12 → 5 20 pts 10 → 5 ~~30~~ 15 5 15 ~~30~~ pts 70 min
Q1: ________ Q3: ________ Q5: ________

30 pts 20 pts 50
17 Q2: ________ ~~15~~ Q4: ________ Q6: ________

1.0

22 Total: 27

37 - 42

42
- 3
12,6 2 h

Question 1: Short Answer (20 pts; 1 pt each) 12 min

a. What is the "principle of locality"?

temporal locality: recently used items will be used again soon

spatial locality: when one item is used, nearby items will be used

b. Describe two factors influencing the cost of manufacturing a modern microprocessor.

design area

manufacturing defects

c. Ideal scalar pipelines achieve a CPI of 1; give one reason why this ideal is rarely achieved.

data
structural } hazards
control

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

when an earlier instruction writing FX finishes after a later inst writing FX

e. What makes implementing "predicted taken" more complex than implementing "predicted not-taken"?

predicted taken requires branch target resolution hardware

f. Define "precise exception."

an exception is precise if, earlier instructions can finish, later instructions do not change arch state, and the current inst can be restarted

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

WAR

h. What does a correlating branch predictor correlate?

past branch outcomes with current branch prediction

i. What is the purpose of hardware register renaming?

to eliminate WAW and WAR hazards,
thereby ~~Imp~~ reducing stalls

j. Describe one difference between static and dynamic superscalar processors.

static : in-order execution
dynamic : out-of-order execution

k. 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 — fewer compulsory misses

l. 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 - more capacity/conflict misses

m. Why aren't all caches implemented with full associativity?

cost in area
reduction in performance } parallel comparison logic

n. What is the purpose of a TLB?

locality - maintains a cache of recently used virtual → physical address translations

o. Describe one factor limiting the ILP of modern microprocessors.

limited registers for renaming limits the number of instructions that can be in flight at the same time

p. 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 FUs

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

if two processor caches have different values for the same address

r. What causes false sharing?

cache blocks larger than one word → when two processors write two words in the same block, causes coherence misses

s. Describe one disadvantage of directory coherence compared with snoopy coherence.

directory-based coherence requires additional messages compared with snoopy, and therefore possibly longer latency

t. What do memory consistency models define?

the set of possible memory access orderings that a processor ~~is~~ can observe

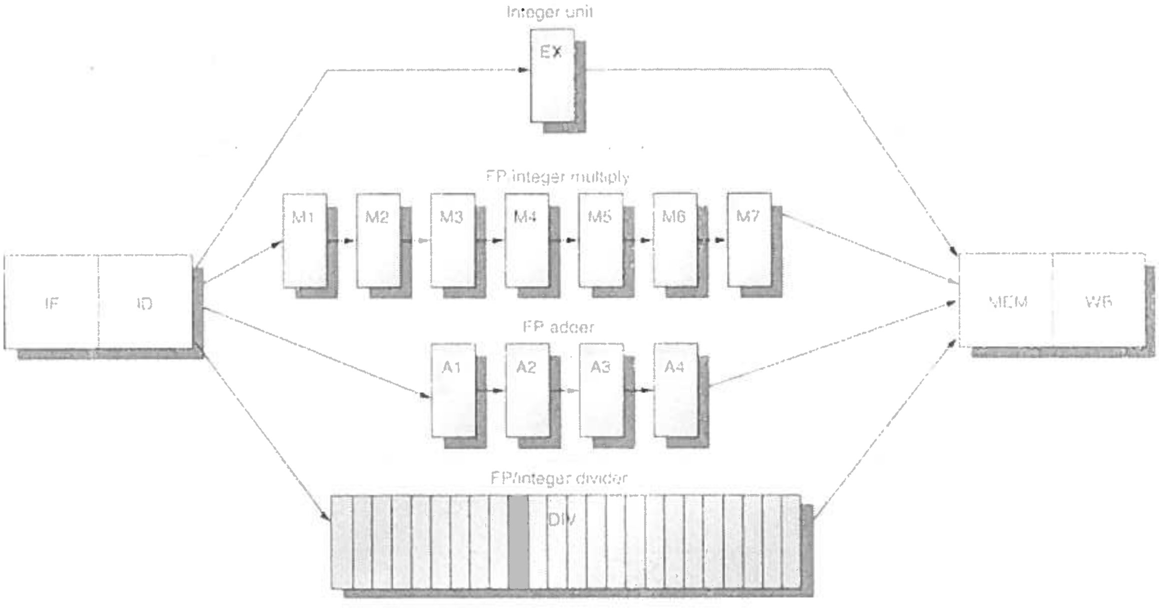
Name: ID:

Question 2: MIPS FP Pipeline (20 pts) 18 min

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

```
Loop:   L.D       F2, 0(R1)       F10
        L.D       F4, 8(R1)       F12
        L.D       F6, 16(R1)      F14
        MULT.D    F2, F2, F6      F10  F10  F14
        ADD.D     F6, F4, F2      F14  F12  F10
        SUB.D     F8, F4, F2      F16  F12  F10
        S.D       F6, 0(R2)       F14
        S.D       F8, 8(R2)       F16
        DSUBUI    R1, R1, #24
        DSUBUI    R2, R2, #16
        BNEZ      R1, Loop
```

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



© 2007 Elsevier, Inc. All rights reserved

(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	F	D	E	M	W																				
LD		F	D	E	M	W																			
LD			F	D	E	M	W																		
MULTD				F	D	S	M1	M2	M3	M4	M5	M6	M7	M	W										
ADDD					F	S	D	S	S	S	S	S	S	A1	A2	A3	A4	M	W						
SUBD							F	S	S	S	S	S	S	D	A1	A2	A3	A4	M	W					
SD														F	D	S	E	M	W						
SD															F	S	D	E	M	W					
DSUBUI																S	F	D	E	M	W				
DSUBUI																		F	D	E	M	W			
BNEZ																			F	D	E	M	W		
LD																				F	F	D	E	M	W

LD → MULTD
MULTD → ADD
~~MULTD → SUB~~

ADD → SD
AD
~~SUB → SD~~
BNEZ → LD

DSUB → BNEZ

ADD, SD
SUB, SD

Name: ID:

rename registers as is necessary

(b) (8 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.

```
LD       F2, 0(R1)
LD       F4, 8(R1)
LD       F6, 16(R1)
LD       F10, 24(R1)
LD       F12, 32(R1)
LD       F14, 40(R1)
MULT.D   F2, F2, F6
MULT.D   F10, F10, F14
DSUBUI   R1, R1, #48
DSUBUI   R2, R2, #32
3 stall > ADD.D    F6, F4, F2
ADD.D    F14, F12, F10
SUB.D    F8, F4, F2
SUB.D    F16, F12, F10
S.D      F6, -32(R2)
S.D      F14, -24(R2)
S.D      F8, -16(R2)
BNEZ     R1, Loop
S.D      F16, -8(R2)
```

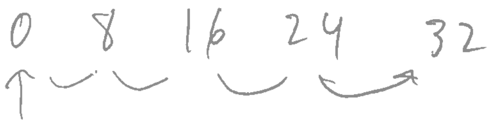
```
F  D  A1 A2 A3 A4
   F  D  A1 A2 A3 A4
      F  D  A1 A2 A3 A4
         F  D  A1 A2 A3 A4

            F  D  E  M
```

2 register renaming
2 ld offsets
2 sd offsets

2 delay slot

stall elimin. from
2 LD — grouping
4 MULT — grouping, moving
2 ADD — grouping



(c) (2 pts) What is the speedup of your re-scheduled code?

Two iterations in 19+3 cycles.

↳ $\frac{22}{2} \to 11$ cycles/iter

original : $\overset{20}{\cancel{16}}$ cycles

Speedup = $\frac{\cancel{16}\ 20}{11} = 1.81$

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

~~Three~~. Four.

Name: ID:

Additional Page for Q2

Name: 10 mm ID:

Question 3: (20 pts) Branch Prediction

a. (10 pts) Two machines A and B are identical except for their branch prediction schemes. Each machine uses a seven-stage pipeline:

always 1 cycle

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.

$$CPI = 1 + \text{branch stalls} = 1 + 0.25 \cdot (\text{penalty})$$

not-taken:

BNEZ IF ID1 ID2 EX1

$$1 + 0.25\left(\underset{\text{taken}}{X} \cdot \underset{\substack{2\\2}}{\underline{\underline{3}}} + \underset{2}{\underline{\underline{0}}} \cdot (1-x)\right) \quad x \quad x \quad x \quad IF$$

taken

$$1 + 0.25 \cdot \left(x \cdot \overset{1}{\underline{\underline{1}}} + (1-x) \cdot \overset{2}{\underline{\underline{3}}}\right)$$

$$\cancel{1}+0.25 \cdot X \cdot 3 = \cancel{1}+0.25\,x \cdot 1 + 0.25(1-x) \cdot 3$$

$$0.75x = 0.25x + \overset{\cdot 3}{0.25} - 0.25x \cdot 3$$

$$= 0.25x + 0.75 - 0.75x$$

$$1.5x$$

$$1.25x = 0.75$$

$$5x = 3 \qquad x = \frac{3}{5} = 0.6$$

5

60% taken

Name: ID:

b. **(10 pts)** Complete the branch prediction table below for a (2, 1) correlating predictor. Assume all predictor state is initialized to “not-taken,” or N. Assume that when indexing the predictor state and BHR, the most recent branch result is in the least significant bit on the right.

Predictor State				BHR	Prediction	Outcome	
NN	NT	TN	TT				
N	N	N	N	TN	N	T	
N	N	T	N	NT	N	T	
N	T	T	N	TT	N	T	
N	T	T	T	TT	T	N	
N	T	T	N	TN	T	T	✓
N	T	T	N	NT	T	N	
N	N	T	N	TN	T	T	✓
N	N	T	N	NT	N	T	
N	T	T	N	TT	N	T	
N	T	T	T	TT	T	N	
N	T	T	N	TN	T	T	✓
N	T	T	N	NT	T	N	

What is the accuracy of the predictor?

$3/12 = 0.25$

Name: ID: 15 min

Question 4: (20 pts) Tomasulo's Algorithm

3 min

a. (10 pts) Consider a processor which implements dynamic scheduling with speculation using Tomasulo's Algorithm. Assume that the functional units are **fully pipelined** and that all memory accesses hit the cache. There is a memory unit with 5 load buffers. The reorder buffer has 64 entries. The reorder buffer can function as a store buffer, so there are no separate store buffers. Each load or store takes 2 cycles to execute, 1 to calculate the address, and 1 to load/store the data. Assume a perfect branch predictor. Assume there are dedicated integer functional units for effective address calculation and branch condition evaluation. The other functional units are described in the following table. Assume only one result can be written to the common data bus at the same time.

Functional unit type	Cycles to execute	Number of functional units	Number of reservation stations per functional unit
Integer ALU	1+0	1	4
FP adder	1+3	1	3
FP multiplier	1+6	1	2
Load	1+1	1	5

Fill in the table below the clock cycle number that each instruction issues, begins execution, writes its result, and commits for the first iteration of the loop and first instruction of the second.

Loop Iter	Operation		Issue	Begin Exec	Finish Exec	Write Result	Commit
1	L.D	F2, 0(R1)	1	2	3	4	5
1	L.D	F4, 8(R1)	2	3	4	5	6
1	L.D	F6, 16(R1)	3	4	5	6	7
1	MULT.D	F2, F2, F6	4	7	13	14	15
1	ADD.D	F6, F4, F2	5	15	18	19	20
1	SUB.D	F8, F4, F2	6	16	19	20	21
1	S.D	F6, 0(R2)	7	20	21	22	23
1	S.D	F8, 8(R2)	8	21	22	23	24
1	DSUBUI	R1, R1, #24	9	10	10	11	25
1	DSUBUI	R2, R2, #16	10	11	11	12	26
1	BNEZ	R1, Loop	11	12	12	13	27
2	L.D	F2, 0(R1)	12	13	14	15	28

maybe make 2nd iteration → just to test write result again?

Name: 12 min ID:

b. (10 pts)

Show the contents of the Reservation Stations, the Registers, and the Re-Order Buffer when the 2nd iteration of L.D F2,0(R1) **begins execution**.

Reservation Stations								
Name	**Busy**	**Op**	**V_j**	**V_k**	**Q_j**	**Q_k**	**Dest**	**Address**
Load 1		SD			#5		–	R2+0
Load 2		SD			#6		–	R2+8
Load 3		LD					#12	#9+0
Load 4		LD					#13	#9+8
Load 5	n							
FP Add 1		ADD.D	F4			#4	#5	
FP Add 2		SUB.D	F4			#4	#6	
FP Add 3	n							
FP Mult 1		MULT.D	F2	F6			#4	
FP Mult 2	n							

Reorder Buffer					
Entry	**Busy**	**Instruction**	**State**	**Destination**	**Value**
1	n	LD	commit	F2	M[0(R1)]
2	n	LD	commit	F4	M[8(R1)]
3	n	LD	commit	F6	M[16(R1)]
4	y	MULTD	exec	F2	
5	y	ADD D	issue	F6	
6	y	SUB D	issue	F8	
7	y	SD	issue	F6	
8	y	SD	issue	F8	
9	y	DSUBUI	write	R1	R1+24
10	y	DSUBUI	write	R2	R2+16
11	y	BNEZ	write	–	
12	y	LD	exec	F2	
13	y	LD	issue	F4	
14					
15					
16					

Register Status							
	R1	**R2**	~~**F0**~~	**F2**	**F4**	**F6**	**F8**
Reorder #	9	10		12	13	7	8
Busy (yes/no)	y	y		y	y	y	y

Name: ID:

Question 5: (20 pts) Cache Hierarchy 5 min

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%

CPU Time = ~~AC~~ · CPI · CT   CPI · CT

CPU Time DM

CT = 1ns

$(1 + 1.25 \cdot 0.03 \cdot 1.25 \cdot 100 + 0.1 \cdot 0.08 \cdot 1.25 \cdot 100) \cdot 1$

~~$(1 + 1 \cdot (0.03 \cdot 100) + 0.25 \cdot (0.03 + 100) + 0.1 \cdot (0.08 \cdot (100 + 0.25 \cdot 100))) \cdot 1$~~

$= 6.6875$

CPU Time 2-way

$$\left(1 + 1 \cdot \left(0.02 \cdot \frac{100}{1.1}\right) + 0.25 \cdot \left(0.02 + \frac{100}{1.1}\right) + 0.1 \cdot \left(0.06 \cdot \left(\frac{100}{1.1} + 0.25 \cdot \frac{100}{1.1}\right)\right)\right) \cdot 1.1 =$$

$$\left(1 + 1.25 \cdot 0.02 \cdot 1.25 \cdot \frac{100}{1.1} + 0.1 \cdot 0.06 \cdot 1.25 \cdot \frac{100}{1.1}\right) \cdot 1.1 = 4.975$$

3 —	1 ns
2 —	1.25
1 ~~2~~ —	0.02
2 —	1.25
1 —	100
2 —	0.1
1 ~~2~~ —	0.06
2 —	1.25
1 —	100

1.1 ns

$$\frac{\text{CPU Time DM}}{\text{CPU Time 2-way}} = 1.344$$

Name: ID: 12 min

Question 6: (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.** 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	B0
S	108	15	08	M	128	2D	68	S	108	15	08	I	128	43	30	B1
I	110	F7	30	I	110	6F	10	M	110	76	01	M	130	64	00	B2
I	118	C2	10	S	118	3E	18	I	118	C2	10	I	118	40	28	B3

	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
P2: write 112 <-- 0A

4

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 address, and D = data contents

To show the bus messages, use the following format:

block

Bus {message type, requesting processor or directory, 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}

Name: ID:

a. **(8 pts) P0: read 130**

Bus {read miss, P0, 130, —} ✓

Bus {fetch, $\overset{P3}{\text{Dir}}$, 130, —} ✓

P3 <block 12> {S, 130, 6400}

Bus {data write back, P3, 130, 64 00}

M <130> {DS, [P0, P3], 64 00}

Bus {data value reply, $\overset{P3}{\text{Dir}}$, 130, 64 00}

P0 <block2> {S, 130, 64 00}

4

b. **(12 pts) P3: write 112 <-- 0A**

([illegible]) Bus {data write back, P3, 130, 64 00}

P3 <2> {I, 130, 64 00}

− (1) 1 M <130> {DI, [ ], 64 00}

(1) Bus {write miss, P3, 110, —}

(1) Bus {fetch / invalidate, Dir, 110, —}

(1) Bus {data write back, P2, 110, 76 01}

− (1) 2 P2 <2> {I, 110, 76 01}

− (1) 3 M <110> {DM, [P3], 76 01}

(1) Bus {data value reply, Dir, 110, 76 01}

− (1) 4 P3 <2> {M, 110, 0A 01}

4 state
5 messages
1 right address
1 right data
1 order