16.482 / 16.561: Computer Architecture and Design

Fall 2013

Final Exam
December 16, 2013

Name: _____ **ID #:** _____

For this exam, you may use a calculator and two 8.5" x 11" double-sided page of notes. All other electronic devices (e.g., cellular phones, laptops) are prohibited. If you have a cellular phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 7 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

You will be provided with two pages (1 double-sided sheet) containing additional copies of information used in some of the problems, so that you can more easily refer to this material while answering these questions. You do not have to submit these pages when you turn in your exam.

Note that some problems do not require you to complete all parts of the question. In particular:

- On Question 5 (Cache optimizations), you must complete either part (a) or part (b) of the question—not both.
- On Question 7 (Multiprocessors), you must complete part (a). You must also complete either part (b) or part (c)—not both.

You will have three hours to complete this exam.

Q1: Dynamic scheduling and speculation	/ 16
Q2: Multithreading	/ 16
Q3: Cache operation	/ 16
Q4: Virtual memory	/ 15
Q5: Cache optimizations	/ 13
Q6: RAID	/ 10
Q7: Multiprocessors	/ 14
TOTAL SCORE	/ 100

1. (16 points) ***Dynamic scheduling and speculation***

a. (4 points) Explain why instruction results in a speculative, dynamically scheduled processor are stored in the reorder buffer rather than directly being written to the register file or memory. How does this setup impact register renaming (in other words, how is register renaming handled differently in a dynamically scheduled processor with speculation than it is without speculation)?

b. (12 points) You are given the following piece of code to run on a dynamically scheduled machine that allows speculation:

```
Loop:   L.D    F2, 0(R1)
        MUL.D  F4, F4, F2
        DADDUI R1, R1, #8
        BNE    R1, R2, Loop
        S.D    F4, 8(R1)
```

Assume the following latencies:

- L.D and S.D have 2 cycle latencies (1 cycle execution, 1 cycle in memory)
- MUL.D has a 5 cycle latency
- All other operations have 1 cycle latencies

Also, assume:

- The loop executes three times.
- The branch at the end of the loop is always predicted taken.
    - If there is a misprediction, you should accurately show how it would be handled. Unknown instructions should be shown as "?" in the Inst. column of your diagram
- Integer and floating point operations are handled in separate functional units.
- You only have one common data bus available—if two instructions need to use the common data bus, the earliest instruction in terms of program order has priority.
- You only have the ability to commit one instruction per cycle.

How many cycles will this sequence take, from the time the first instruction fetches to the time the last instruction commits? Complete in the pipeline diagram on the following page to support your answer. Use the space below and the back of this page for any additional work.

NOTE: Your extra handout contains an extra copy of the latencies and code listed above.

QUESTION 1b SOLUTION

Inst.	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
L.D F2,0(R1)	IF	IS	EX																						
MUL.D F4,F2,F0		IF	IS																						
DADDUI R1,R1,#8			IF																						

2. (16 points) ***Multithreading***

a. (4 points) List one benefit of fine-grained multithreading over coarse-grained multithreading and one benefit of coarse-grained multithreading over fine-grained multithreading.

b. (12 points) Given the 3 threads below, determine how long they take to execute using simultaneous multithreading on a processor with the following characteristics:

- 4 functional units: 2 ALUs, 1 memory port (load/store), 1 branch
  - The ALUs can handle MUL.D operations
  - The branch unit can handle jumps
- In-order execution
- The following instruction latencies:
  - L.D/S.D: 3 cycles (1 EX, 2 MEM)
  - MUL.D: 4 cycles
  - ADD.D/SUB.D: 2 cycles
  - All other operations: 1 cycle
- Thread 1 is the preferred thread, followed by Thread 2 and Thread 3.

Assume the BEQ in Thread 3 is not taken.

Your solution should use the table on the next page, which contains columns to show each cycle, the functional units being used during that cycle, and space to indicate stall cycles. Note that you only need to label a cycle as a stall if all active threads are stalled. **Clearly indicate which thread contains each instruction when completing the table.**

NOTE: The last page of the exam contains an extra copy of the latencies and threads.

Thread 1:

```
L.D   F0, 0(R1)
MUL.D F4, F0, F2
ADD.D F6, F4, F10
S.D   F6, 16(R1)
SUB.D F10, F6, F2
S.D   F10, 32(R1)
DADDUI R1, R1, #-48
BNEZ  R1, loop
```

Thread 2:

```
L.D   F2, 0(R1)
ADD.D F4, F4, F0
SUB.D F6, F6, F0
MUL.D F8, F2, F6
MUL.D F10, F2, F4
ADD.D F0, F10, F8
DSUB  R1, R1, R2
BNEZ  R1, loop
```

Thread 3:

```
LW    R1, 0(R2)
BEQ   R1, R3, t1
ADD.D F0, F2, F4
MUL.D F6, F2, F4
S.D   F0, 0(R1)
S.D   F6, 8(R1)
JR    R31
```

QUESTION 2b SOLUTION

Cycle	ALU1	ALU2	Mem1	Branch	Stalls?
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

3. (16 points) ***Cache operation***

You are given a system which has a 16-byte, write-back cache with 4-byte blocks. The cache is direct-mapped. The system uses 8-bit addresses, and the cache is initially empty.

a. (12 points) Assume the initial memory state shown below for the first 32 bytes:

Address		Address		Address		Address	
0	27	8	19	16	22	24	13
1	3	9	78	17	5	25	24
2	20	10	9	18	15	26	21
3	11	11	12	19	13	27	7
4	5	12	1	20	49	28	18
5	12	13	0	21	77	29	8
6	14	14	63	22	15	30	55
7	2	15	98	23	44	31	99

For each access in the sequence listed below, fill in the cache state, indicate what register (if any) changes, and indicate if any memory blocks are written back and if so, what addresses and values are written. The cache state should carry over from one access to the next.

Access	Modified register	Cache state							Modified mem. block
		V	D	Tag	Data				
`lb $t0,6($zero)`									
`sb $t0,4($zero)`									
`lb $t1,23($zero)`									
`sb $t1,14($zero)`									

3 (continued)

b. (4 points) Assume this cache is the Level 1 cache in a hierarchy with the following characteristics:
    - Level 1 cache: 4 ns access time, 92% hit rate
    - Level 2 cache: 12 ns access time, 96% hit rate
    - Memory: 100 ns access time, 99% hit rate
    - Disk: 1000 ns access time

Assume that the processor cycle time is 2 ns. How many cycles will an average memory access take? **Show all work for full credit.**

4. (15 points) ***Virtual memory***
Answer the following questions about a process using the page table below:

Virtual page #	Valid bit	Reference bit	Dirty bit	Frame #
0	0	0	0	--
1	1	1	1	2
2	1	0	1	3
3	0	0	0	--
4	1	0	0	1
5	0	0	0	--

a. (3 points) Which pages are candidates to be evicted on a page fault? Under what conditions would a page fault cause one of these pages to be evicted?

b. (6 points) Assuming 2 KB pages, what physical addresses would the virtual addresses below map to? Note that some virtual addresses may not have a valid translation, in which case you should note that address causes a page fault.

- 0x2010

- 0x1FFE

- 0x0A1B

4 (continued)
c. (6 points) Fill in the table at the bottom of the page to show the **final** state of the page table after the following sequence of accesses. Assume main memory has 4 frames, numbered 0-3, and frame 0 is initially free. The initial state of the page table is repeated below for your reference.

ACCESS SEQUENCE

- Read page 3
- Write page 4
- Write page 5
- Read page 4

INITIAL PAGE TABLE STATE:

Virtual page #	Valid bit	Reference bit	Dirty bit	Frame #
0	0	0	0	--
1	1	1	1	2
2	1	0	1	3
3	0	0	0	--
4	1	0	0	1
5	0	0	0	--

FINAL PAGE TABLE STATE:

Virtual page #	Valid bit	Reference bit	Dirty bit	Frame #
0				
1				
2				
3				
4				
5				

5. (13 points) ***Cache optimizations***

Use the following page to answer **only 1 of the following 2 questions—part (a) or part (b)**. **Clearly indicate** which question you have chosen to answer at the top of the page.

a. ***(Multi-banked/non-blocking caches)*** Assume we have a system containing 16 blocks of memory, numbered 0-15. This system has an 8-line, direct-mapped cache that is initially empty. Assume we have a program that accesses 10 of these blocks in the following order, with one access initiated per cycle unless a stall occurs:

0, 1, 4, 14, 7, 5, 9, 8, 15, 6

Note that all of these accesses are misses; each miss takes 20 cycles to handle.

i. (3 points) Calculate the total time for these 10 accesses if the cache is not split into banks and is therefore a blocking cache (i.e., only one miss can be handled at a time).

ii. (5 points) Calculate the total time for these 10 accesses if the cache is divided evenly into four banks. Assume the blocks are not interleaved sequentially, so blocks are mapped to cache lines using normal direct mapping. In other words, B0 maps to cache line 0, B1 to cache line 1, and so on.

iii. (5 points) Calculate the total time for these 10 accesses if the cache is divided evenly into four banks and the blocks are interleaved sequentially across those four banks.

b. ***(Early restart/critical word first)*** Say we have a program running on a 32-bit processor in which a certain cache block is evicted from the cache before every access, making every access to this block a cache miss. The block contains 512 bytes. Assume every word (i.e., every 4 bytes of data) in the block is accessed exactly once.

If main memory is capable of supplying a word to the cache every 100 μs (1 μs = 10^{-6} s), calculate the total time required to access **all** words in the cache block using each of the following policies to fill the cache block on a miss:

- Sequential cache block fill (i.e., start with word 0 and fill all words in block)
- Early restart (without critical word first)
- Critical word first with early restart

QUESTION 6 SOLUTION

Circle one: a. Multi-banked caches b. Early restart/critical word first

6. (10 points) ***RAID***

You are working with a 5-disk RAID array that contains a total of 15 sectors; the exact sector configuration depends on the RAID level used. In all cases, twelve of the fifteen sectors (S0-S11) will hold data, while the remaining three sectors (P0-P2) hold parity information. Large reads and writes (reads/writes that access an entire stripe in the array) take 300 ms, small reads (reads involving only a single disk) take 150 ms, and small writes (writes involving 1 data disk + 1 parity disk) take 200 ms.

Given the following sequence of sector reads and writes, determine the time required if the array is configured with RAID 3, RAID 4, and RAID 5. Assume the following:

- Requests are queued in such a manner that two consecutive operations may proceed simultaneously if they do not share any disk within the array.
- If two disks, D_x and D_y, are in use, and the access to D_x finishes before the access to D_y, a new operation may start immediately assuming it does not involve D_y.
- Multiple accesses to the same stripe may overlap if they don't use the same disk.

1. read S0
2. write S5
3. write S8
4. read S9
5. read S3
6. write S7
7. read S11
8. write S1

In each case, show the organization of the array to support your answer. The next page contains additional space to solve this problem.

ADDITIONAL SPACE FOR QUESTION 6 SOLUTION

7. (14 points) ***Multiprocessors***
Note: To complete this problem, you must solve part (a) + either part (b) or (c).

a. (4 points) Say you have a dual processor system running two separate programs that share data but use different variable names. The following pieces of code are executed in the order shown, with the side effects listed:
   - P0: `x = 12;` (P0 writes `x`, invalidates copy of that block in P1's cache)
   - P1: `a = b;` (P1 experiences cache miss while reading `b`)

Assume that the cache miss for P1 is a result of the invalidation from P0. Under what conditions is the cache miss for P1 a true sharing miss? Under what conditions is that miss a false sharing miss?

b. (10 points) ***Solve either part (b) or part (c)—not both.***
Say we have a four-processor system that uses a write-invalidate, directory coherence protocol. The system contains a total of 8 memory blocks, as shown in the initial directory state below:

Block #	P0	P1	P2	P3	Dirty
0	0	0	0	0	0
1	1	0	1	1	0
2	1	0	0	0	1
3	0	1	0	1	0
4	0	0	1	1	0
5	0	1	0	0	1
6	0	1	0	0	0
7	0	0	0	0	0

For all sequences of transactions shown on the next page, list all messages sent as well as the final directory state for the block(s) in question. You should assume that each sequence of accesses is independent—your answer to part 2 does not depend on part 1—but accesses within a sequence are dependent on one another—your answer for part 1, access (ii) **does** depend on what happens in part 1, access (i).

Note: Your second handout contains an extra copy of the directory state above.

QUESTION 5b SOLUTION

Transaction(s)	Messages sent	Final directory state
1. (i) P1: read block 2 (ii) P2: read block 2 (iii) P3: read block 2		
2. (i) P2: write block 5 (ii) P2: write block 6 (iii) P2: write block 7		
3. (i) P3: write block 2 (ii) P0: write block 2 (iii) P3: read block 2		

c. (10 points) ***Solve either part (b) or part (c)—not both.***

You are given a four-processor system that uses a write-invalidate, snooping coherence protocol. Each direct-mapped, write-back cache has four lines, each of which holds eight bytes; in the diagram below, only the least-significant byte of each word is shown. The cache states are I (invalid), S (shared), and M (modified/exclusive).

The caches and memory have the following initial state; please note that all addresses and tags are shown in hexadecimal:

P0

	State	Tag	Data	
B0	I	0x100	02	80
B1	S	0x108	00	88
B2	M	0x110	30	09
B3	I	0x118	00	10

P1

	State	Tag	Data	
B0	I	0x100	02	80
B1	M	0x128	AB	CD
B2	I	0x110	20	08
B3	S	0x118	14	92

P2

	State	Tag	Data	
B0	S	0x120	13	31
B1	S	0x108	00	88
B2	I	0x130	14	12
B3	I	0x138	01	38

P3

	State	Tag	Data	
B0	M	0x100	13	31
B1	S	0x108	00	88
B2	S	0x130	14	12
B3	S	0x118	14	92

Memory

Address	Data	
0x100	02	80
0x108	00	88
0x110	20	08
0x118	14	92
0x120	13	31
0x128	FF	FE
0x130	14	12
0x138	AB	BA

For each of the transactions listed on the next page, use the table to list all cache blocks modified and their final state, as well as all memory blocks modified and their final state. Assume each set of transactions starts with the same initial state—in other words, your answer to part (b) does not depend on your answer to part (a). However, you should track the state transitions of each block throughout the problem.

NOTE: Your second handout contains an extra copy of the tables above.

QUESTION 5c SOLUTION

Transaction(s)	Cache blocks modified	Memory blocks modified
1. (i) P0: write 0x114 ← 99 (ii) P2: read 0x110 (iii) P3: write 0x110 ← 99		
2. (i) P2: write 0x138 ← 12 (ii) P2: read 0x118 (iii) P0: read 0x138		
3. (i) P3: write 0x10C ← FF (ii) P3: write 0x134 ← 41 (iii) P3: write 0x104 ← AB		