16.482 / 16.561: Computer Architecture and Design

Spring 2014

Final Exam
May 1, 2014

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.

Please note that you will not be allowed to share anything—including erasers, pens, and pencils—during the exam. If you need an eraser, pen, or pencil, ask Dr. Geiger.

The exam contains 6 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 this sheet 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 4 (Cache optimizations), you must complete either part (a) or part (b) of the question—not both.
- On Question 6 (Cache coherence protocols), you must complete either part (a) or part (b) of the question—not both.

You will have three hours to complete this exam.

Q1: Multithreading	/ 19
Q2: Caches	/ 26
Q3: Virtual memory	/ 18
Q4: Cache optimizations	/ 13
Q5: RAID	/ 10
Q6: Cache coherence protocols	/ 14
TOTAL SCORE	/ 100

1. (19 points) ***Multithreading***

a. (4 points) Assume you have three threads concurrently executing on your system. Under what circumstances would coarse-grained multithreading give the lowest total execution time for all three threads? Under what circumstances would fine-grained multithreading give the lowest total execution time?

b. (15 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
  - Note: The ALUs can handle MUL.D operations
- 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 all branches are 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, but you do not have to write the full instruction—writing the opcode (i.e. L.D, ADD.D) is sufficient.**

Thread 1:
```
L.D F0, 0(R1)
L.D F2, 8(R1)
ADD.D F4, F0, F2
SUB.D F6, F4, F2
S.D F6, 16(R1)
DSUBUI R1, R1, #16
BNEZ R1, loop
```

Thread 2:
```
L.D F2, 0(R1)
MUL.D F2, F0, F2
DADDI R1, R1, #16
ADD.D F10, F8, F2
S.D F10, -8(R1)
DSUB R20, R4, R1
BNZ R20, Loop
```

Thread 3:
```
ADD.D F4, F0, F2
SUB.D F6, F2, F6
ADD.D F8, F0, F4
MUL.D F10, F4, F6
DADDUI R1, R1, #16
S.D F10, -16(R1)
BNE R1, R2, loop
```

Note: Your second handout contains an extra copy of the latencies and threads listed above.

QUESTION 1b 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					

2. (26 points) ***Caches***

You are given a system which has an 8-byte, write-back cache with 4-byte blocks. The cache is direct-mapped. All memory accesses use 7-bit addresses.

a. (18 points) Assume the initial memory state shown below for the first 16 bytes:

Address		Address	
0	20	8	15
1	8	9	67
2	27	10	78
3	3	11	19
4	12	12	26
5	44	13	99
6	34	14	9
7	5	15	4

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. Assume the cache is initially empty.

Access	Modified register	Cache state							Modified mem. block
		V	D	Tag	Data				
`lb $t0,2($zero)`									
`sb $t0,12($zero)`									
`lb $t1,5($zero)`									
`sb $t1,4($zero)`									
`lb $t0,3($zero)`									

2 (continued)

b. (8 points) Assume the access sequence above is part of a loop, which executes five times. The remainder of the program contains no data memory accesses. Calculate the average memory access time for this program and system if the cache takes 3 cycles to access and the memory takes 250 cycles to access. Assume those are the only two levels in the memory hierarchy.

3. (18 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	1	1	1	12
1	0	0	0	--
2	1	0	1	5
3	1	0	0	8
4	0	0	0	--
5	1	1	0	2

a. (3 points) Explain the purpose of the reference bits, including conditions under which those bits are set and cleared.

b. (9 points) Assuming 16-bit addresses and 512 byte 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.

- 0x0A16

- 0x02FE

- 0x011B

3 (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 16 frames, numbered 0-15, and frame 15 is the only frame that is free before this sequence executes. The initial state of the page table is repeated below for your reference.

ACCESS SEQUENCE

- Read page 1
- Write page 3
- Write page 5
- Read page 4

INITIAL PAGE TABLE STATE:

Virtual page #	Valid bit	Reference bit	Dirty bit	Frame #
0	1	1	1	12
1	0	0	0	--
2	1	0	1	5
3	1	0	0	8
4	0	0	0	--
5	1	1	0	2

FINAL PAGE TABLE STATE:

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

4. (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. ***(Way prediction)*** Consider the following two caches for use in a system:

- A direct mapped cache with an access time of 2 ns and an average hit rate of 90%
- A four-way set associative cache with an access time of 6 ns and an average hit rate of 95%.

Assume the cache miss penalty for the system is 100 ns.

We want to use way prediction to improve the performance of the set associative cache. If a way prediction hit takes as long as a direct-mapped access, and a way-prediction miss adds an additional 6 ns, how accurate must the way predictor be to match the average access time of the direct-mapped cache?

b. ***(Multi-banked/non-blocking caches)*** Assume we have a system containing 32 blocks of memory, numbered 0-31. 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:

4, 5, 1, 2, 24, 25, 26, 27, 16, 12

Note that all of these accesses are misses; each miss takes 10 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 two 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 two banks and the blocks are interleaved sequentially across those two banks.

QUESTION 4 SOLUTION

Circle one: **a. Way prediction** **b. Multi-banked caches**

5. (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 take 200 ms, small reads take 50 ms, and small writes take 100 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.

1. read S2
2. read S10
3. write S9
4. read S4
5. read S8
6. write S7
7. write S9
8. read S2

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 5 SOLUTION

6. (14 points) ***Coherence protocols***

a. ***Solve either part (a) or part (b)—NOT BOTH***

(Snooping coherence protocols) 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	01	23
B1	S	0x108	00	88
B2	M	0x110	00	30
B3	I	0x118	00	10

P1

	State	Tag	Data	
B0	I	0x100	01	23
B1	M	0x128	AB	CD
B2	S	0x130	14	12
B3	S	0x118	14	92

P2

	State	Tag	Data	
B0	S	0x120	13	31
B1	S	0x108	00	88
B2	I	0x130	51	55
B3	I	0x138	01	38

P3

	State	Tag	Data	
B0	S	0x120	13	31
B1	S	0x108	00	88
B2	I	0x110	00	30
B3	S	0x118	14	92

Memory

Address	Data	
0x100	00	00
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 (2) does not depend on your answer to part (1). 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 6a SOLUTION

Transaction(s)	Cache blocks modified	Memory blocks modified
1. (i) P1: read 0x110 (ii) P2: read 0x110 (iii) P3: read 0x110 (iv) P0: write 0x110 ←55		
2. (i) P3: write 0x110←45 (ii) P0: write 0x110←67 (iii) P3: read 0x114 (iv) P0: read 0x114		

6 (continued)

b. ***Solve either part (a) or part (b)—NOT BOTH***

(Directory protocols) 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	1	0	1	0	0
1	0	1	0	0	1
2	0	0	0	0	0
3	0	1	1	1	0
4	0	0	0	1	1
5	1	1	0	0	0
6	0	1	0	0	1
7	1	1	1	1	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 6b SOLUTION

Transaction(s)	Messages sent	Final directory state
1. P0: read block 1 P2: read block 1 P3: write block 1 P3: read block 1		
2. P0: write block 2 P1: read block 2 P2: read block 2 P3: write block 2		