McGill University
ECSE 425: COMPUTER ORGANIZATION AND ARCHITECTURE
Winter 2017
Midterm Examination

1:05 – 2:25 PM, February 15th, 2017

Duration: 80 minutes

- **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 12 pages long. Please check that you have all 12 pages.**
- **There are four questions for a total of 80 points, or one point per minute. 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.**
- **Faculty standard calculators are permitted; no cell phones or laptops.**
- **Clearly state any assumptions you make.**
- **Write your answers in the space provided. Show your work to receive full credit, and clearly indicate your final answer.**

Name: **_____SOLUTION_____**

Student Number: **_____**

Name: ID:

Q1: _____/20 **Q2: _____/20** **Q3: _____/20**

Q4: _____/20

Total:

Name: ID:

Question 1: The Long and Short (Answer) of It (20 pts total)

Respond to each of the following; two to four sentences should be adequate in each case. *No credit* will be given for unjustified assertions.

a. **(4 pts)** Assume a 48-bit virtual address space, with 16 KB pages. Main memory has a 64 GB capacity. L1 is four-way set associative, virtually indexed, physically tagged, with 128B blocks. Assuming the L1 cache is as large as possible, how many bits are used for block tag, index, and offset?

16 KB pages => 14-bit page offset
64 GB main memory => 36-bit physical address
128 B blocks => 7-bit offset
L1 virtually indexed, physically tagged => up to 14 – 7 = 7-bit index
36 – 14 = 22-bit tag

22 bits for tag, 7 for index, 7 for offset

b. **(4 pts)** *Miss rate* is an imperfect measure of cache performance. Give a concrete example of when *miss rate* might decrease while *average memory access time* increases.

AMAT = HT + MR * MP

Increasing cache size decrease miss rate but increases hit time. If the change in HT is larger than MR * MP, then performance suffers.

c. **(4 pts)** Assume a cache hierarchy with two levels of caching consisting of split, 64 KB L1 caches, and a unified 256 KB L2. L1 is *direct-mapped*, and implements *write-back* and *write-allocate* policies. L2 is *inclusive*. What is expected to happen to the local L2 miss rate if the capacity of the L1 caches is doubled? Why?

The local miss rate will increase, because L2 will not have many blocks not already present in L1, a result of inclusivity.

d. (4 pts) A *structural hazard* arises in a standard five-stage pipeline when a *unified* L1 cache is used. Describe (i) *the hazard and its consequences*, and (ii) *how it is mitigated*.

The hazard occurs when a load or store enters the MEM stage: IF wants an instruction, but MEM wants data, and one must wait. This delays IF by a cycle.

This hazard can be mitigated with a split L1: one cache for instructions, and another for data.

e. (4 pts) Re-schedule the following instructions to eliminate stalls. Assume a standard five-stage pipeline with full hazard detection and forwarding.

```
LW   R2, 0(R1)
LW   R4, 0(R3)
ADD  R5, R2, R4
SW   R5, 4(R1)
LW   R6, 8(R1)
LW   R7, 8(R3)
SUB  R8, R6, R7
SW   R8, 12(R3)
```

Stalls occur between LW and consuming instructions, e.g., LW R4 -> ADD, and LW R7 -> SUB. All we need to do is move the LW instructions and all stalls can be eliminated.

```
LW   R2, 0(R1)
LW   R4, 0(R3)
LW   R6, 8(R1)
LW   R7, 8(R3)
ADD  R5, R2, R4
SW   R5, 4(R1)
SUB  R8, R6, R7
SW   R8, 12(R3)
```

Name: ID:

Question 2: Cache Me Outside (20 pts)

Consider a hierarchical cache with the following contents. All state, tags and data are in hexadecimal format. Assume that L1 and L2 are *exclusive*. Assume *little-endianness*, i.e., that the least significant byte in a word is stored in the least significant address.

L1 Cache

Set	Valid	Dirty	Tag	Data															
				f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0
0	0	0	34	83	ab	57	88	31	e7	a0	5e	8f	4c	e4	df	63	77	c6	5e
1	0	0	1f	d2	ce	aa	7c	82	23	d7	d7	1f	64	76	c7	a3	dc	19	78
2	1	0	d3	e6	65	8d	3b	e3	53	f8	d7	f8	40	3d	f2	80	42	0a	f0
3	1	1	2d	0c	b8	00	e4	d7	60	58	b0	71	4e	10	f7	b9	5f	58	c7
4	0	0	67	5f	91	85	64	de	45	ca	e8	84	b7	c0	41	6e	5c	d4	64
5	1	1	d3	be	8a	35	b0	9b	75	70	48	08	06	a0	bd	13	c6	69	a0
6	0	0	22	2a	a4	7b	bd	72	a1	f7	51	19	3b	55	fc	d8	52	30	35
7	1	0	00	30	4b	12	c0	2f	51	8d	3a	8e	57	c7	8b	be	44	93	a2
8	1	0	ab	df	cd	5a	f7	7f	4b	cd	b3	58	a5	4d	46	f5	30	91	e6
9	0	0	12	ee	f9	86	7d	24	dd	98	7c	8c	a7	de	ec	bc	a5	06	cb
a	1	1	e3	2c	5b	af	c6	92	09	cf	ca	a9	e5	f2	87	d2	91	9b	1d
b	1	0	df	e8	af	5b	2e	4a	77	e0	0c	1d	d0	ce	0b	b8	ac	a6	9c
c	0	0	ee	9d	b7	11	e9	68	7c	a3	4f	91	17	2f	b3	3d	85	92	33
d	1	0	1d	cd	8b	2f	5d	c8	de	16	90	d2	4e	e3	20	12	83	f2	62
e	0	0	a0	d5	d0	a3	4b	9e	1a	d4	55	a3	9b	1d	98	26	38	a5	e0
f	1	1	55	a5	65	2b	9f	55	20	72	e6	25	23	07	e1	bf	7f	72	1e

L2 Cache

Set	Valid	Dirty	Tag	Data															
				f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0
0	0	0	22	d0	3c	82	d0	e4	04	93	50	9c	4f	9a	4a	f1	6d	21	22
1	1	1	d2	54	9d	9b	59	ba	aa	a5	d0	66	ea	4d	7c	b6	ef	9d	9d
2	1	0	17	05	ff	99	48	d9	a8	86	0b	07	60	74	60	97	42	c0	51
3	1	0	8d	ea	b4	e8	01	cf	8c	54	96	3a	8f	5e	b1	d1	58	53	07
4	0	0	88	20	03	80	6e	6b	c1	71	3b	12	2f	3c	59	f0	6c	f9	1c
5	1	0	2d	99	ec	4f	d6	40	df	28	4d	0f	8f	64	66	ec	10	ff	cc
6	1	1	22	ec	de	a2	6c	4b	93	ad	f4	5b	ed	48	2a	cf	b4	ca	45
7	1	0	8d	97	73	42	76	bb	e0	4c	40	bd	eb	5c	b1	74	72	6a	c4
8	0	0	a9	10	08	90	d3	59	48	d4	5d	af	5c	97	b7	66	f2	fa	3c
9	0	0	0e	85	8f	34	e7	8a	f9	2e	d9	c5	1f	21	85	88	a0	03	15
a	1	1	31	5c	6a	ec	f3	e0	72	84	97	55	65	cc	a1	7b	9d	61	24
b	1	0	b8	77	93	42	c7	a1	98	44	2b	15	90	1f	d7	cf	a5	94	40
c	1	1	ab	e5	05	bf	ac	67	2d	7b	f1	76	e9	cc	16	7d	e8	1a	c3
d	1	0	4d	c9	27	66	71	42	df	83	6d	ca	a9	59	e3	a9	fe	15	18
e	0	0	ab	5c	ea	ab	14	8a	fa	11	b0	00	40	46	6c	72	89	0f	f8
f	1	1	1c	a0	0b	47	8d	b1	3e	6b	42	7d	be	76	b8	1e	0c	45	88

Indicate the changes made to any cache blocks in L1 or L2 as a consequence of servicing the following requests. In the case of read requests, indicate what data is returned.

a. (4 pts) 16-bit read access from address `0xe3aa`.

Tag, idx, offset: e3, a (10), a

1. Hits in set a in L1.
2. Returns 0x9209.

No change to caches.

b. (8 pts) Write of `0xbeef` to address `0xb8be`.

Tag, idx, offset: b8, b (11), e

1. Miss in set b in L1.
2. Hit in set b in L2.
3. Swap: L1[b] <- L2[b] while L2[b] <- L1[b]; tag for L1[b] becomes b8; tag for L2[b] becomes df.
4. Most significant bytes of L1[b] becomes 0xbeef; dirty bit is set.

c. (8 pts) 16-bit read access from address `0x2262`.

Tag, idx, offset: 22, 6, 2

1. Miss: set 6 is invalid in L1.
2. Hit in set 6 in L2.
3. L1[6] <- L2[6]; tag for L1[6] becomes 22; set valid and dirty.
4. L2[6] set invalid.
5. Returns 0xCFB4.

Question 3: This Memory Performance is Great Again (20 pts)

Consider a pipelined processor that runs at 1 GHz and has a base CPI of 1.5 when all cache accesses hit. The only instructions that read data from or write data to memory are loads (10% of all instructions) and stores (15% of all instructions).

The memory system for this computer is composed of a split L1 cache. Both the I-cache and D-cache are direct-mapped and hold 64 KB each. The I-cache has a 1% miss rate; the D-cache has a miss rate of 10%.

The 1 MB *inclusive*, unified L2 cache has an access time of 15 ns. Of all memory references sent to the L2 cache in this system, 60% are satisfied without going to main memory. Main memory has an access latency of 200 ns. Assume that L2 and main memory transfer times are accounted for in the access times above.

Assume that L1 is *write-back*, *write-allocate*, and 50% of evicted data is *dirty*; assume that L2 is *write-through*, *no-write-allocate*, and that 90% of writes to main memory are handled by a *write buffer* without *blocking* (stalling) the processor.

a. (2 pts) What is the average memory access time (in cycles) for read accesses to L2?

$AMAT_{L2\text{-}read} = HT_{L2} + MR_{L2} * MP_{L2\text{-}read}$
$HT_{L2} = 15$ ns, $MR_{L2} = 0.4$, $MP_{L2\text{-}read} = 200$ ns, 1 CC = 1 ns

$AMAT_{L2\text{-}read} = 15 + 0.4*200 = 95$ CC

b. (2 pts) What is the average memory access time (in cycles) for write accesses to L2?

If the block is in L2, it is updated, but also sent to main memory. If the block isn't there, it is just sent to main memory (but the corresponding block is not retrieved).

$AMAT_{L2\text{-}write} = HT_{L2} + MP_{L2\text{-}write}$
$HT_{L2} = 15$ ns, $MP_{L2\text{-}write} = 0.1*200$ ns, 1 CC = 1 ns
$AMAT_{L2\text{-}write} = 15 + 0.1*200 = 35$ CC

c. (4 pts) What is the average memory access time (in cycles) for instruction accesses?

$AMAT_{L1I} = HT_{L1I} + MR_{L1I} * (AMAT_{L2\text{-}read}) = 1 + 0.01*95 = 1.95$ CC

d. (8 pts) What is the average memory access time (in cycles) for data accesses?

Read and write misses are both handled by reading from L2, as L1 is write-allocate (data is requested, then updated in L1, but not updated in L2 until write-back). Note that reads or writes may force a write-through when the replaced data is dirty (in proportion to the dirty rate, DR). Since reads and writes to L2 have different latency, we don't multiply the entire second term by (1+DR).

$AMAT_{L1D} = HT_{L1D} + MR_{L1D} * (AMAT_{L2\text{-}read} + DR*AMAT_{L2\text{-}write})$
$= 1 + 0.1 * (95 + 0.5 * 35) = 12.25$ CC

e. (4 pts) What is the overall CPI of the system?

CPI is the ideal CPI plus whatever delays are experienced due to (a) instruction accesses, and (b) data accesses (data access rate, DAR). Note that $AMAT_{L1}$ includes L1 hit time, which is already accounted for in the ideal CPI; thus, we subtract one cycle in each case.

$CPI = CPI_{Ideal} + (AMAT_{L1I} - 1) + DAR*(AMAT_{L1D} - 1)$
$= 1.5 + (1.95 - 1) + 0.25*(12.25 - 1) = 5.2625$

Name: ID:

Question 4: Assemble the Pipeline; Who Will Pay for It? (20 pts)

Consider the following code:

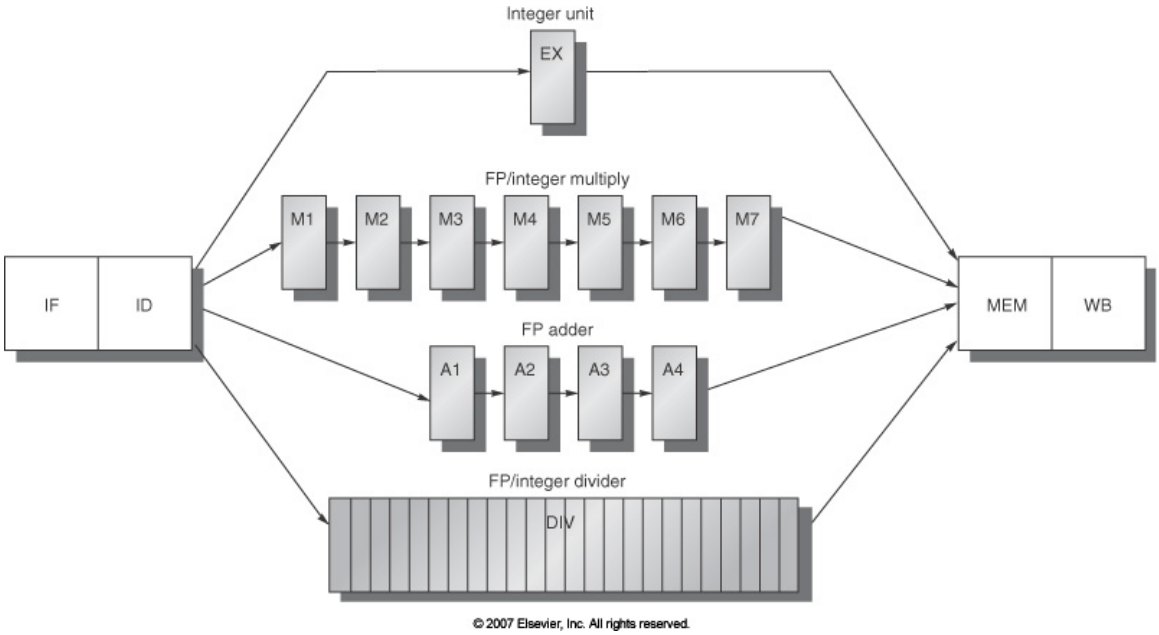
```
      LW    R6, 12(R1)
      ADDI  R7, R0, 2
EQ2:  BNEQ  R6, R7, EQ1
      ADD   R8, R2, R3
      MULT  R10, R6, R5
      J     END
EQ1:  ADDI  R7, R0, 1
      BNEQ  R6, R7, DEF
      SUB   R8, R2, R3
      LW    R9, 20(R1)
      MULT  R10, R8, R9
      J     END
DEF:  ADDI  R8, R0, 0
      ADDI  R10, R0, 0
END:  SW    R8, 28(R1)
      SW    R10, 32(R1)
```

and corresponding initial memory state:

Address	Value
0x1000	5
0x1004	1
0x1008	3
0x100C	1
0x1010	2
0x1014	4
0x1018	8
0x101C	12
0x1020	3

Name: ID:

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



Assume:

- Full hazard detection and forwarding logic;
- The register file supports one write and two reads per clock cycle;
- Branch targets and conditions and jump targets are resolved in ID;
- Branches are handled by predicting not taken;
- J is treated as a branch instruction for branch prediction purposes;
- Split L1 instruction and data caches service all requests in 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.

Complete the chart on the next page to show the execution of the code if R1 = 0x1000 initially. Indicate pipeline stages with {**F**, **D**, **X**, **Mi**, **Ai**, **M**, **W**}, and stalls with **S**.

Name: ID:

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
LW R6, 12(R1)	F	D	X	M	W																			
ADDI R7, R0, 2		F	D	X	M	W																		
BNEQ R6, R7, EQ1			F	D	X	M	W																	
ADD R8, R2, R3				F	D	X	M	W																
ADDI R7, R0, 1					F	D	X	M	W															
BNEQ R6, R7, DEF						F	D	X	M	W														
SUB R8, R2, R3							F	D	X	M	W													
LW R9, 20(R1)								F	D	X	M	W												
MULT R10, R8, R9									F	D	X	M	M	M	M	M	M	M	M	M	W			
J END										F	D	F	D	X	M	W								
ADDI R8, R0, 0											F		F	D	X	M	M	M	M	M	M			
SW R8, 28(R1)														F	D	X	M	W						
SW R10, 32(R1)															F	D	D	D	X	M	W			

Name: ID:

Additional Page