CS 251, Spring 2022, Assignment 6

Due Friday, July 22nd, 9:00 pm
Lates accepted within 48 hours with a penalty of 10% per day or portion of a day late.

You are required to read, follow, complete, sign, and submit the following statement of Academic Integrity.

Statement of Academic Integrity for CS 251 Spring 2022, **Assignment 6**
I declare the following statements to be true:

- I have not used any unauthorized aids.
- I recognize that while I can discuss the questions in this assignment on Piazza and other forums with the instructors and with other students in the class, the write up that I am submitting is my own.
- I am aware that misconduct related to course work can result in significant penalties, including failing the course and suspension (this is covered in Policy 71:)

  `https://uwaterloo.ca/secretariat/policies-procedures-guidelines/policy-71`

Student Name: Robert Knowles

UW ID#: 20878339

Signature: R K

Date: 18/07/2022

1. (2 points) AMAT

   The Average memory access time (AMAT) is the average (expected) time it takes for a memory access considering both hits and misses. Let t_h denote the hit time, m denote the miss rate, and t_m denote the miss penalty. AMAT can be calculated using the formula:

$$AMAT = t_h + m \cdot t_m \ .$$

   The miss rate is the percentage of memory accesses that are not in cache. The miss penalty is the additional time it takes for memory access to the next higher level in the hierarchy.

   Suppose you only have a level 1 cache, and that level 1 cache hits in 1 clock cycle with a miss rate of 4%. The cost to access main memory (the miss penalty) is 57 clock cycles. Using the formula above, state the AMAT below. Show your work. You will only recieve one point by just providing the answer.

   AMAT= 1 + (0.04)*57
   1 + 2.28
   3.28

2. (10 points) Cache

Suppose we have a direct mapped cache with **block size four** that is initially empty.

(a) (4 points) Suppose we have the sequence of load words and store words given in the table. Fill in the hit/miss column of the table.

Line #	Instruction	Data memory address (hex)	Tag	Index	Block	Byte	**H**it/ **M**iss
			Data memory address (binary)				
1	LDUR	0x020	000	0001	00	000	
2	STUR	0x028	000	0001	01	000	
3	LDUR	0x438	010	0001	11	000	
4	LDUR	0x430	010	0001	10	000	
5	LDUR	0x428	010	0001	01	000	
6	LDUR	0x420	010	0001	00	000	
7	STUR	0x278	001	0011	11	000	
8	LDUR	0x220	001	0001	00	000	
9	LDUR	0x228	001	0001	01	000	
10	LDUR	0x230	001	0001	10	000	

(b) (4 points) Given below are empty tables of cache. Fill in each table with the contents of the cache after the instructions at line 3, 7 and 10 have been executed. You must state what data is loaded from Memory. For example, if memory address '0x020' is in the cache, write 'M[020]' under index 0001 and block 00. Note that '0x020' is a hexidecimal number, not a decimal number.

Be sure to fill in the Tags and Dirty bits. If a row is empty (i.e., hasn't been accessed and not valid), you may leave the row blank.

Cache after executing Lines 1, 2, and 3:

			Block			
Index	**Tag**	**D**	**00**	**01**	**10**	**11**
0000						
0001						
0010						
0011						

Cache after executing Lines 1 through 7:

			Block			
Index	**Tag**	**D**	**00**	**01**	**10**	**11**
0000						
0001						
0010						
0011						

Cache after executing Lines 1 through 10:

			Block			
Index	**Tag**	**D**	**00**	**01**	**10**	**11**
0000						
0001						
0010						
0011						

(c) (2 points)

Assume the cache uses a write-back scheme, where data written to cache is only written to memory when the cache block is replaced. At this point, the entire cache block is written back to memory.

Circle the line number of the instructions above that *cause* a cache block to be written to memory. You need to circle the **line numbers** and not the number of instructions that cause a write-back.

#1 #2 #3 #4 #5 #6 #7 #8 #9 #10

3. (12 points) Page Tables

Suppose you have a 64-bit virtual and physical address space with 4KB pages. For all addresses, the most significant 8 hex digits will always be 0, so just indicate the least significant 8 hex digits. Below, is a six-row portion of the page table. All the blank Valid, Dirty, and Reference bits are 0.

The free pages start at physical address 0x00204 0000 and go up. The OS allocates free pages in ascending order, i.e. page 0x00204, 0x00205, etc.

Below the page table is a sequence of virtual memory accesses. Show the page table after this sequence of memory accesses. The address is the memory address to which the LDUR instruction is reading from or the STUR instruction is writing to; we are not concerned with the memory addresses at which these instructions are located at in this question. For each virtual address, translate it to a physical address and indicate if a page fault occurred with that access.

Leave unused entries in the page table blank. However, for non-blank entries in the page table, *all* columns must be completed; i.e., you must fill in the 0's rather than leave an entry blank if anything else is non-zero on the row. If you want to change an existing value in the page table, cross out the old value and add the new one beside it.

Page Table

Index	V	D	R	Page
00100				
00101	1	0	1	00200
00102				
00103	1	1	0	00201
00104				
00105	0	1	1	00202

Instruction	**Virtual Address**	**Physical Address**	**Page fault?**
`LDUR`	`0x00101 340`		
`STUR`	`0x00103 408`		
`STUR`	`0x00102 888`		
`LDUR`	`0x00100 010`		
`LDUR`	`0x00105 560`		
`STUR`	`0x00101 108`		

4. (10 points) Page Tables and the TLB

Suppose you have a 64-bit virtual and physical address space with 4KB pages, an 8-word TLB, and 16 pages of physical memory. For simplicity, assume that the page table is in memory on page 15 and addressed through a Page Table Register (not shown below) and an entry for the page table is never put in the TLB. Also assume the following:

- An entry is added the TLB in the row with the lowest number that is not valid.
- When taking a page from the Free Page list, take the available page with the *lowest* page number.

For all addresses, the most significant 8 hex digits will always be 0, so just indicate the least significant 8 hex digits. Additionally, for this question, you **do not need to clear the reference bits**. Suppose the Free pages, the Page Table and TLB are as follows

Free pages: 1 3 5 7 9

Page Table

Index	V	D	R	Page (hex)
00000	1	0	1	00002
00001	1	0	0	0000A
00002	1	0	1	00008
00003	0	0	0	00007
00004	0	0	0	00003
00005	1	0	1	00004
00006	1	1	1	00006
00007	0	0	1	00005
00008	1	1	0	00000
00009	0	0	0	00001

TLB

Row	V	D	R	Tag	Page
0	1	0	1	00005	00004
1	0	0	0	00003	00007
2	1	0	1	00002	00008
3	0	0	0	00004	00003
4	1	0	1	00000	00002
5	0	0	1	00007	00005
6	0	0	0	00001	0000A
7	1	1	1	00006	00006

Suppose we have the following *sequence* of virtual memory accesses (all reads), starting with the Free pages, the Page Table and TLB above. For each memory access, write the corresponding physical address, and note if the physical page came from the Page Table (PT), the TLB, or caused a Page Fault (PF). In all cases, note the TLB row that is accessed (read from or updated). In your copy, make any updates to the page table and TLB; although this won't be graded, it will affect your answers to later addresses in the sequence.

Question continued on next page.

Memory Accesses: (addresses in hexadecimal)

Virtual Address	Physical Address	TLB/PageTable/ Page Fault	TLB Row
0x00000 5D8:			
0x00008 540:			
0x00007 A08:			
0x00004 710:			
0x00008 110:			
0x00001 360:			