CprE 381 Homework 11

[Note: This is your final homework of the semester. These problems will help you gain some familiarity with the impact caches have on performance.]

1. Virtual Memory
    a. ZyBooks (Textbook) 5.19.16.a-e
    b. ZyBooks (Textbook) 5.19.17a-c

5.19.16 a
TLB Table Update Iterations (final state is bolded):

Address	Virtual Page	TLB H/M	TLB		
			Valid	Tag	Physical Page
4669	1	TLB miss PT miss PF	1	11	12
			1	7	4
			1	3	6
			1 (last access 0)	1	13
2227	0	TLB miss PT hit	1 (last access 1)	0	5
			1	7	4
			1	3	6
			1 (last access 0)	1	13
13916	3	TLB hit	1 (last access 1)	0	5
			1	7	4
			1 (last access 2)	3	6
			1 (last access 0)	1	13
34587	8	TLB miss PT miss PF	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 2)	3	6
			1 (last access 0)	1	13
48870	11	TLB miss PT hit	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 2)	3	6
			1 (last access 4)	11	12
12608	3	TLB hit	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 5)	3	6
			1 (last access 4)	11	12
49225	**12**	**TLB miss PT miss PF**	**1 (last access 6)**	**12**	**15**
			1 (last access 3)	**8**	**14**
			1 (last access 5)	**3**	**6**

			1 (last access 4)	**11**	**12**

Page Table Updated:

Valid	Physical Page or in Disk
1	5
1	13
0	Disk
1	6
1	9
1	11
0	Disk
1	4
1	14
0	Disk
1	3
1	12
1	15

The generation of the first entry is as follows.

The virtual address must be converted to a binary address so we can identify all relevant parts.

4669_{10} = 0000 0000 0000 0000 0001 0010 0011 1101_2

The page size is given as 4 KiB = 2^{12} bytes. So, the upper 32-12=20 bits of the virtual address will be the page number. Hence, the Virtual Page Number of 1 is shown in the second column of the table.

Note: TLB is a cache for page table. The question states that TLB is fully associative 4-entry, meaning that the TLB has 1 set, and all 4 entries (ways) are searched in parallel. You do not need to calculate set index. The virtual page number is used as Tag.

Next, we'll check if we have a TLB hit, miss, or page fault. The criteria for checking this is:

1. If the virtual page number we determined in the second column is the tag in the TLB and the corresponding valid bit is 1 then we have a TLB hit.
2. If the virtual page number is not one of the valid tags, then we have a TLB miss and we must go to the page table in physical memory and check it using the virtual page number as an index. If the returned page table entry has a valid bit of 1, the memory location is in physical memory and only the TLB needs updating according to the LRU replacement policy (LRU metadata we represent by "last access" – no initial LRU data set so we assume we replace from the top down if no info). Otherwise (page table valid bit is 0), it was not only a TLB miss, but also a page fault.

Since virtual page number of the first entry is 1, we look for the tag in the TLB. No tag entry in the initial TLB state is 1, so we have a TLB miss and proceed to the page table. Index 1 in the page table has a valid bit of 0 – so we have a page fault. A new page is brought in from disk and the page table entry is updated. Note that the question states that when a new page is brought in from disk, use the next largest page number. Thus, the new page table entry at index 1 maps to page 13 (12+1).
Next, we will add to the TLB following LRU. The entry with valid bit 0 is replaced first. The physical address is 13.
The same process follows for each subsequent entry for the given addresses. Note, once you move to the next address, make sure to use the updated values made to the TLB!

Also, since for the last address, the virtual page number is 12, we not only miss in the TLB, but also miss the page table, resulting in a page fault.

5.19.16 b

Address	Virtual Page	TLB H/M	TLB		
			Valid	Tag	Physical Page
4669	0	TLB miss PT hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 0)	0	5
2227	0	TLB hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 1)	0	5
13916	0	TLB hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 2)	0	5
34587	2	TLB miss PT miss PF	1 (last access 3)	2	13
			1	7	4
			1	3	6
			1 (last access 2)	0	5
48870	2	TLB hit	1 (last access 4)	2	13
			1	7	4
			1	3	6
			1 (last access 2)	0	5
12608	0	TLB hit	1 (last access 4)	2	13
			1	7	4
			1	3	6

			1 (last access 5)	0	5
49225	3	TLB hit	1 (last access 4)	2	13
			1	7	4
			1 (last access 6)	3	6
			1 (last access 5)	0	5

Valid	Physical Page or in Disk
1	5
0	Disk
1	13
1	6
1	9
1	11
0	Disk
1	4
0	Disk
0	Disk
1	3
1	12

The same process follows for generating this, except that we are given 16 KiB pages instead of 4. So, the upper 32-14=18 bits of the virtual address will be the page number.

A larger page size reduces the TLB miss rate, but can lead to higher fragmentation and lower utilization of the physical memory.

5.19.16 c

Two-way set associative:

Address		Tag	Index	TLB H/M	TLB

	Virtual Page				Valid	Tag	Physical Page	Index
4669	1	0	1	TLB miss PT miss PF	1	11	12	0
					1	7	4	1
					1	3	6	0
					1 (last access 0)	0	13	1
2227	0	0	0	TLB miss PT hit	1 (last access 1)	0	5	0
					1	7	4	1
					1	3	6	0
					1 (last access 0)	0	13	1
13916	3	1	1	TLB miss PT hit	1 (last access 1)	0	5	0
					1 (last access 2)	1	6	1
					1	3	6	0
					1 (last access 0)	1	13	1
34587	8	4	0	TLB miss PT miss PF	1 (last access 1)	0	5	0
					1 (last access 2)	1	6	1
					1 (last access 3)	4	14	0
					1 (last access 0)	1	13	1
48870	11	5	1	TLB miss PT hit	1 (last access 1)	0	5	0
					1 (last access 2)	1	6	1
					1 (last access 3)	4	14	0
					1 (last access 4)	5	12	1
12608	3	1	1	TLB hit	1 (last access 1)	0	5	0
					1 (last access 5)	1	6	1
					1 (last access 3)	4	14	0
					1 (last access 4)	5	12	1
49225	12	6	0	TLB miss PT miss PF	1 (last access 6)	6	15	0
					1 (last access 5)	1	6	1
					1 (last access 3)	4	14	0
					1 (last access 4)	5	12	1

Now the TLB is divided into two blocks (or sets). So, the first two rows represent the first set, and the last two rows represent the second set.

For the first entry:

4669_{10} = 0000 0000 0000 0000 **0001** 0010 0011 1101_2

Taking the bolded 4-bits, the upper three bits tell us the tag, and the last lower bit gives us the index. Similar methodology follows for calculating table values as previous problems.

5.19.16 d

Direct-mapped:
One assumption for this problem: a page table miss occurs when the valid bit is 0 for a PTE, a PF is when the page only exists on disk.

Address	Virtual Page	Tag	Index	TLB H/M	TLB			
					Valid	Tag	Physical Page	Index
4669	1	0	1	TLB miss PT miss PF	1	11	12	0
					1	0	13	1
					1	3	6	2
					0	4	9	3
2227	0	0	0	TLB miss PT hit	1	0	5	0
					1	0	13	1
					1	3	6	2
					0	4	9	3
13916	3	0	3	TLB miss PT hit	1	0	5	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
34587	8	2	0	TLB miss PT miss PF	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
48870	11	2	3	TLB miss PT hit	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	2	12	3
12608	3	0	3	TLB miss PT hit	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
49225	12	3	0	TLB miss PT miss PF	1	3	15	0
					1	0	13	1
					1	3	6	2
					1	0	6	3

Direct-mapped needs 2 bits to represent the index and 2 bits for the tag.

4669_{10} = 0000 0000 0000 0000 00**01** 0010 0011 1101_2

So, for this, the upper 18 bits give a tag of 0, and the lower 2 bits give us index of 1.

5.19.16 e

All memory references must be cross referenced against the page table and the TLB allows this to be performed without accessing off-chip memory (in the common case). If there were no TLB, memory access time would increase significantly. There would need to be two memory access: one for the page table, and one for the main memory.

5.19.17 a

To find the total page table size, we must find the size of a page table for each application. To do this, we must start by finding the number of page table entries.
Page size = 8 KiB = 2^{13} bytes.
Virtual Memory space size = 2^x bytes, where x is virtual address size.
Page Table Entries = virtual memory space size / page size.
Page Table Entries = 2^{32} / 2^{13} = 2^{19} entries.

Since we know the number of entries per table and the size of an entry is given, we can calculate the size of a page table.
Page table size = number of page table entries * page table entry size
Page table size = 2^{19} entries * 2^2 bytes per entry = 2^{21} bytes (2 MB) per table
2 MB per table * 5 tables = 10 MB total page table size

5.19.17 b

To calculate the minimum and maximum amount of memory, some preliminary calculations must be done first:

We know that we have 256 entries in the first level, so the total number of entries found above must be split into 256 equal parts for the second level tables.
2^{19} entries / 2^8 (256) first level entries = 2^{11} second level entries

We also must find the size of the first and second level tables:
Size of first level page table = 256 entries * 6 bytes per entry = 1536 bytes
Size of second level tables = 2^{11} entries * 2^2 bytes per entry = 2^{13} bytes or 8 KB per table

We can now find the minimum and maximum amount of memory required:
Minimum: only half of the first level entries are used, so there are 128 (2^7) second level tables.
Total memory size for second level tables = 2^7 tables * 2^{13} bytes per table* 5 = 5 MB.
Total memory size for the first level table = (1536 / 2) bytes per table * 5 = 3840 bytes

Maximum: All 256 (2^8) first level entries are used.
Total memory size for second level tables = 2^8 tables * 2^{13} bytes per table* 5 = 10 MB.
Total memory size for the first level table = 1536 bytes per table * 5 = 7680 bytes

5.19.17 c

To make the cache possible with the provided specifications, there must be no aliasing.
Requirements for no aliasing: set index bits + block offset bits ≤ page offset bits.
Block offset bits = 4 bits (16 bytes per block)
Set index bits = 2^{14} bytes (16 Kib) in cache / 2^4 bytes per block = 2^{10} block, so 10 set index bits are required for a direct-mapped cache.
Page offset bits = 2^{13} bytes (8 KiB) page size = 13 page offset bits.
10 + 4 is not less than or equal to 13, so the cache is not possible. If the cache had associativity, then there would be less sets and the above equation would be satisfied.

2. Extra Credit: Exam 3 Question

   Develop your own exam question (roughly 10-15 points) from control hazards, data forwarding, memory technologies, and caches. Your question shouldn't simply ask students to recall information, but should ask for an application of a concept or require understanding of a concept or need analysis of a processor/application. You must include a correct and complete solution to your question. This question should be your own work and not copied from a book or an old exam. ***Post your question and solution to the Exam3 channel for others to use.***

   Answers vary.