UAF EE443, Spring 2023

Homework #5. Due: 4/20/2023 by 9:45 TOTAL: 100

5.7 This exercise examines the impact of different cache designs, specifically comparing associative caches to the direct-mapped caches from Section 5.4. For these exercises, refer to the address stream shown in Exercise 5.2.

Here's what the Exercise says: …Below is a list of 32-bit memory address references, given as word addresses:

3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253

5.7.1 [10] <§5.4> Using the sequence of references from Exercise 5.2, show the final cache contents for a three-way set associative cache with two-word blocks and a total size of 24 words. Use LRU replacement. For each reference identify the index bits, the tag bits, the block offset bits, and if it is a hit or a miss.

5.7.2 [10] <§5.4> Using the references from Exercise 5.2, show the final cache contents for a fully associative cache with one-word blocks and a total size of 8 words. Use LRU replacement. For each reference identify the index bits, the tag bits, and if it is a hit or a miss.

5.7.3 [10] <§5.4> Using the references from Exercise 5.2, what is the miss rate for a fully associative cache with two-word blocks and a total size of 8 words, using LRU replacement? What is the miss rate using MRU (most recently used) replacement? Finally what is the best possible miss rate for this cache, given any replacement policy?

Multilevel caching is an important technique to overcome the limited amount of space that a first level cache can provide while still maintaining its speed. Consider a processor with the following parameters:

Base CPI, No Memory Stalls	Processor Speed	Main Memory Access Time	First Level Cache MissRate per Instruction	Second Level Cache, Direct-Mapped Speed	Global Miss Rate with Second Level Cache, Direct-Mapped	Second Level Cache, Eight-Way Set Associative Speed	Global Miss Rate with Second Level Cache, Eight-Way Set Associative
1.5	2 GHz	100 ns	7%	12 cycles	3.5%	28 cycles	1.5%

5.7.4 [10] <§5.4> Calculate the CPI for the processor in the table using: 1) only a first level cache, 2) a second level direct-mapped cache, and 3) a second level eight-way set associative cache. How do these numbers change if main memory access time is doubled? If it is cut in half?

5.11 As described in Section 5.7, virtual memory uses a page table to track the mapping of virtual addresses to physical addresses. This exercise shows how this table must be updated as addresses are accessed. The following data constitutes a stream of virtual addresses as seen on a system. Assume 4 KiB pages, a 4-entry fully associative TLB, and true LRU replacement. If pages must be brought in from disk, increment the next largest page number.

4669, 2227, 13916, 34587, 48870, 12608, 49225

TLB

Valid	Tag	Physical Page Number
1	11	12
1	7	4
1	3	6
0	4	9

Page table

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

5.11.1 [15] <§5.7> Given the address stream shown, and the initial TLB and page table states provided above, show the final state of the system. Also list for each reference if it is a hit in the TLB, a hit in the page table, or a page fault.

Use the table given on the next page to answer:

5.11.1 - SOLUTION

Address	Virtual Page	Tag	TLB, PT H/M, PF?	TLB		
				Valid	Tag	Physical Page
4669						
2227						
13916						
34587						
48870						
12608						
49225						

Because the TLB is fully set associative, there will be no TAGs in the Virtual Page Number.
TLB, PT H/M, PF? Means: was there a hit or miss in the Translation Lookaside Buffer? Hit or Miss in the Page Table? Will there be a Page Fault generated?

5.11.3 [20] <§§5.4, 5.7> Show the final contents of the TLB if it is 2-way set associative. ~~Also show the contents of the TLB if it is direct mapped~~. Discuss the importance of having a TLB to high performance. How would virtual memory accesses be handled if there were no TLB?

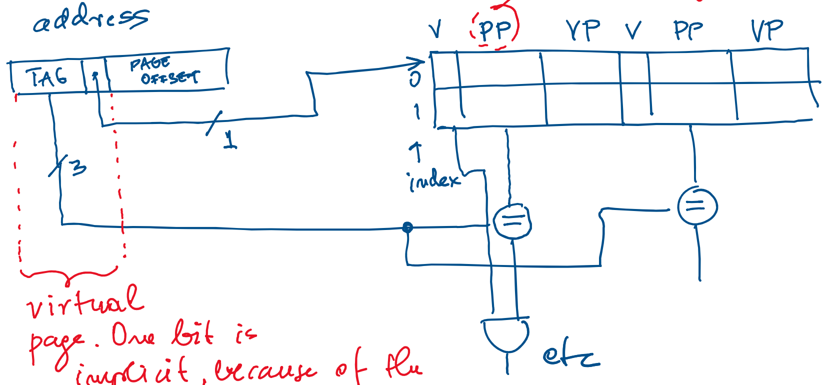
					TLB			
Address	Virtual Page	Tag	Index	TLB, PT H/M	Valid	Tag	Physical Page	Index
4669								0
								1
								0
								1
2227								0
								1
								0
								1
13916								0
								1
								0
								1
34587								0
								1
								0
								1
48870								0
								1
								0
								1
12608								0
								1
								0
								1
49225								0
								1
								0
								1

Because now TLB is 2-way set associative, so there will be a TAG. The organization of the TLB looks different, and it is shown on the next page. It is presented differently in the above table (look at the field index; I populated it now to make it less confusing).

now, TLB looks like this

Technically, can't call it a "physical page" because one bit is missing (index) ⇒ therefore, a TAG.

2-way set associative



virtual page. One bit is implicit, because of the index

There are several parameters that impact the overall size of the page table. Listed below are key page table parameters.

Virtual Address Size	Page Size	Page Table Entry Size
32 bits	8 KiB	4 bytes

5.11.4 [5] <§5.7> Given the parameters shown above, calculate the total page table size for a system running 5 applications that utilize half of the memory available.

5.11.5 [10] <§5.7> Given the parameters shown above, calculate the total page table size for a system running 5 applications that utilize half of the memory available, given a two level page table approach with 256 entries. Assume each entry of the main page table is 6 bytes. Calculate the minimum and maximum amount of memory required.

The following table shows the contents of a 4-entry TLB.

Entry-ID	Valid	VA Page	Modified	Protection	PA Page
1	1	140	1	RW	30
2	0	40	0	RX	34
3	1	200	1	RO	32
4	1	280	0	RW	31

5.12.4 [5] <§5.7> Under what scenarios would entry 2's valid bit be set to zero?

5.12.5 [5] <§5.7> What happens when an instruction writes to VA page 30?