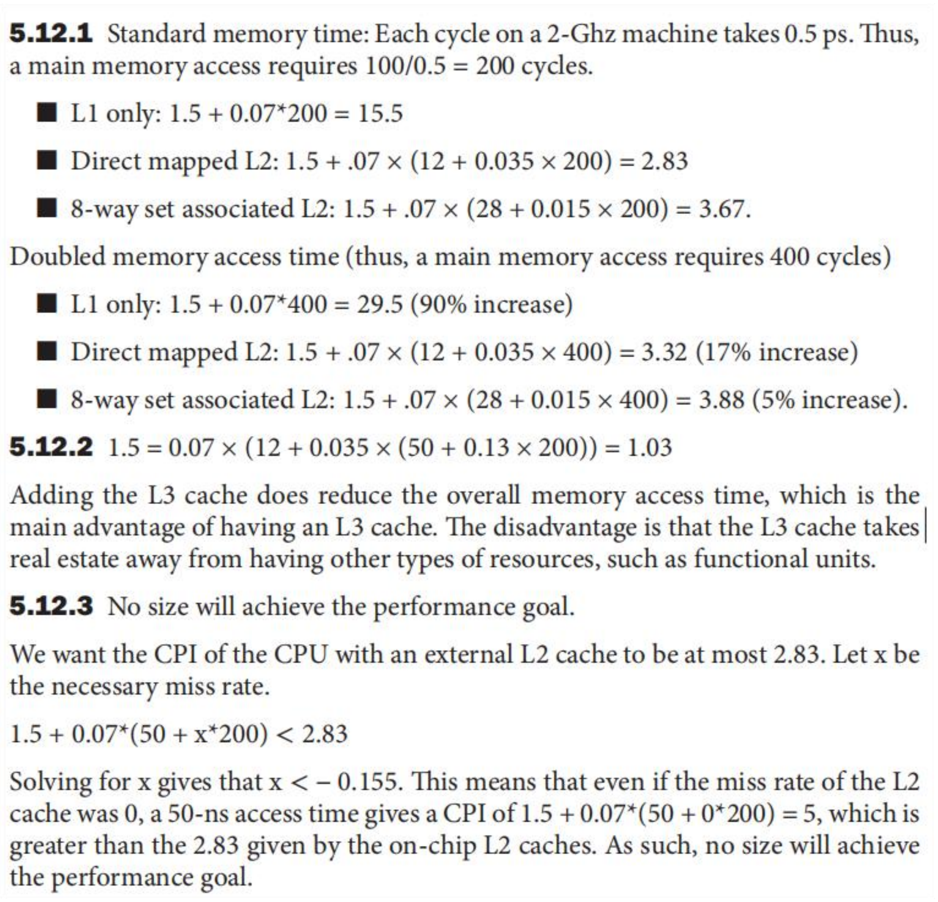
计算机组成原理第九次作业答案

5.12、5.14、5.16、5.17、5.20、5.21、5.29

5.12



5.12.1 Standard memory time: Each cycle on a 2-Ghz machine takes 0.5 ps. Thus, a main memory access requires 100/0.5 = 200 cycles.

- L1 only: 1.5 + 0.07*200 = 15.5
- Direct mapped L2: 1.5 + .07 × (12 + 0.035 × 200) = 2.83
- 8-way set associated L2: 1.5 + .07 × (28 + 0.015 × 200) = 3.67.

Doubled memory access time (thus, a main memory access requires 400 cycles)

- L1 only: 1.5 + 0.07*400 = 29.5 (90% increase)
- Direct mapped L2: 1.5 + .07 × (12 + 0.035 × 400) = 3.32 (17% increase)
- 8-way set associated L2: 1.5 + .07 × (28 + 0.015 × 400) = 3.88 (5% increase).

5.12.2 1.5 = 0.07 × (12 + 0.035 × (50 + 0.13 × 200)) = 1.03

Adding the L3 cache does reduce the overall memory access time, which is the main advantage of having an L3 cache. The disadvantage is that the L3 cache takes real estate away from having other types of resources, such as functional units.

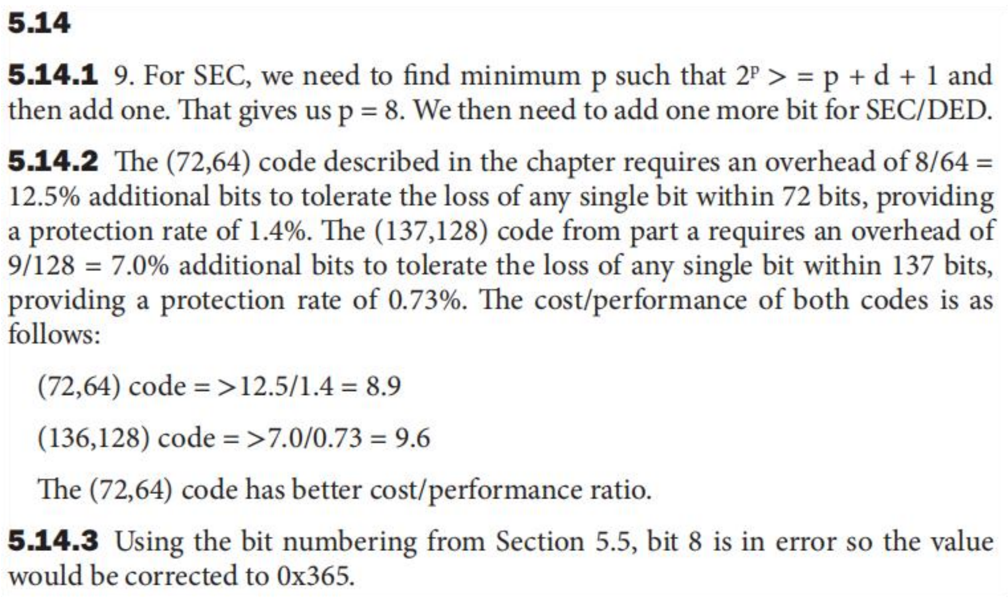
5.12.3 No size will achieve the performance goal.

We want the CPI of the CPU with an external L2 cache to be at most 2.83. Let x be the necessary miss rate.

1.5 + 0.07*(50 + x*200) < 2.83

Solving for x gives that x < − 0.155. This means that even if the miss rate of the L2 cache was 0, a 50-ns access time gives a CPI of 1.5 + 0.07*(50 + 0*200) = 5, which is greater than the 2.83 given by the on-chip L2 caches. As such, no size will achieve the performance goal.

5.14



5.14

5.14.1 9. For SEC, we need to find minimum p such that $2^p > = p + d + 1$ and then add one. That gives us p = 8. We then need to add one more bit for SEC/DED.

5.14.2 The (72,64) code described in the chapter requires an overhead of 8/64 = 12.5% additional bits to tolerate the loss of any single bit within 72 bits, providing a protection rate of 1.4%. The (137,128) code from part a requires an overhead of 9/128 = 7.0% additional bits to tolerate the loss of any single bit within 137 bits, providing a protection rate of 0.73%. The cost/performance of both codes is as follows:

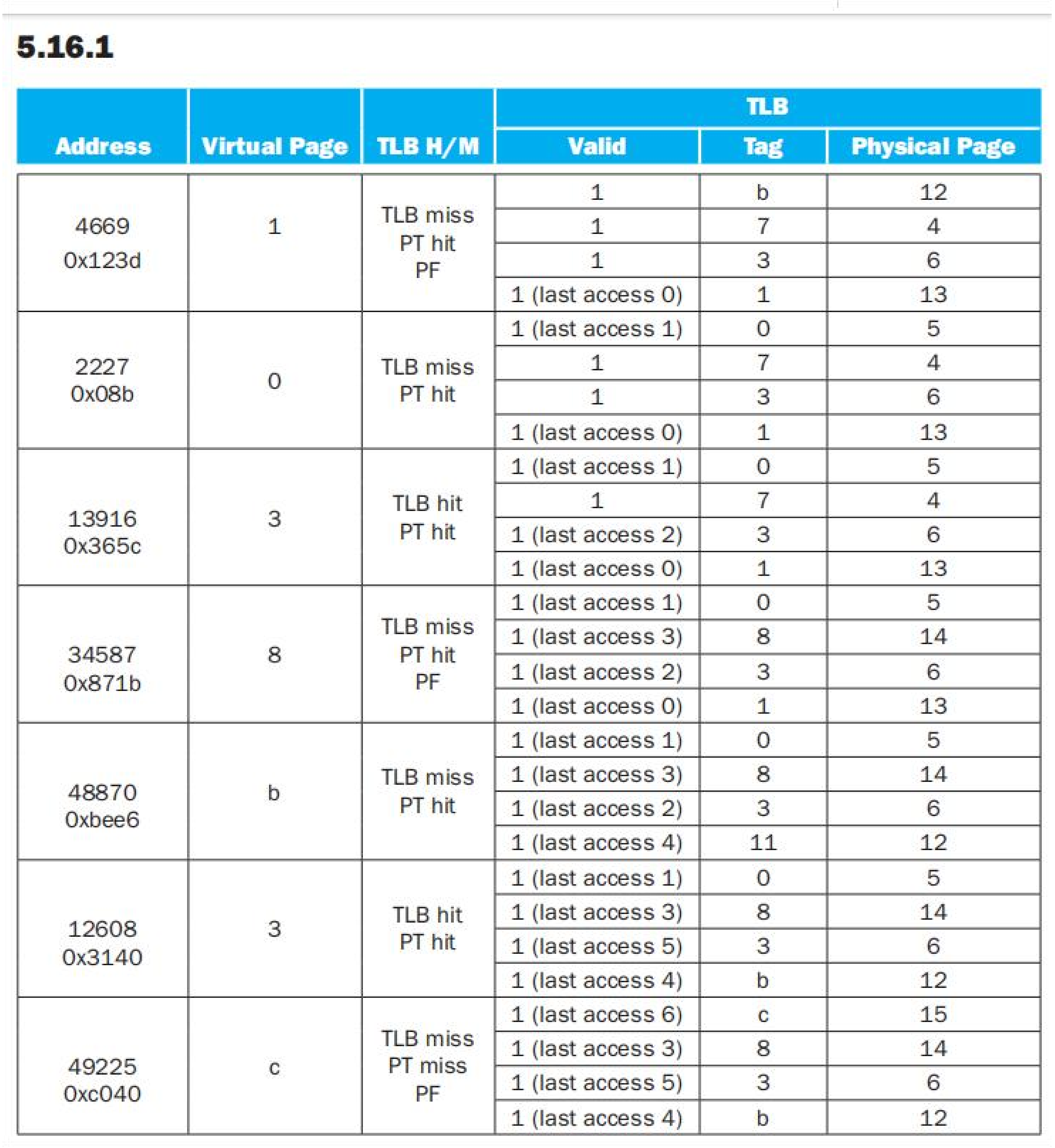
(72,64) code = >12.5/1.4 = 8.9

(136,128) code = >7.0/0.73 = 9.6

The (72,64) code has better cost/performance ratio.

5.14.3 Using the bit numbering from Section 5.5, bit 8 is in error so the value would be corrected to 0x365.

5.16



5.16.1

			TLB		
Address	**Virtual Page**	**TLB H/M**	**Valid**	**Tag**	**Physical Page**
4669 0x123d	1	TLB miss PT hit PF	1	b	12
			1	7	4
			1	3	6
			1 (last access 0)	1	13
2227 0x08b	0	TLB miss PT hit	1 (last access 1)	0	5
			1	7	4
			1	3	6
			1 (last access 0)	1	13
13916 0x365c	3	TLB hit PT hit	1 (last access 1)	0	5
			1	7	4
			1 (last access 2)	3	6
			1 (last access 0)	1	13
34587 0x871b	8	TLB miss PT hit 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 0xbee6	b	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 0x3140	3	TLB hit PT hit	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 5)	3	6
			1 (last access 4)	b	12
49225 0xc040	c	TLB miss PT miss PF	1 (last access 6)	c	15
			1 (last access 3)	8	14
			1 (last access 5)	3	6
			1 (last access 4)	b	12

5.16.2

Address	Virtual Page	TLB H/M	TLB		
			Valid	Tag	Physical Page
4669 0x123d	1	TLB miss PT hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 0)	0	5
2227 0x08b3	0	TLB miss PT hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 1)	0	5
13916 0x365c	3	TLB hit PT hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 2)	0	5
34587 0x871b	8	TLB miss PT hit PF	1 (last access 3)	2	13
			1	7	4
			1	3	6
			2	0	5
48870 0xbee6	11	TLB miss PT hit	1 (last access 4)	2	13
			1	7	4
			1	3	6
			1 (last access 2)	0	5
12608 0x3140	3	TLB hit PT hit	1 (last access 4)	2	13
			1	7	4
			1	3	6
			5	0	5
49225 0xc040	12	TLB miss PT hit	1 (last access 4)	2	13
			1	7	4
			1 (last access 6)	3	6
			1 (last access 5)	0	5

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

5.16.3 Two-way set associative

Address	Virtual Page	Tag	Index	TLB H/M	TLB			
					Valid	Tag	Physical Page	Index
4669 0x123d	1	0	1	TLB miss PT hit PF	1	b	12	0
					1	7	4	1
					1	3	6	0
					1 (last access 0)	0	13	1
2227 0x08b3	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 0x365c	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 0x871b	8	4	0	TLB miss PT hit 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 0xbee6	b	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 0x3140	3	1	1	TLB hit PT 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 0xc049	c	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

5.16.4 Direct mapped

					TLB			
Address	Virtual Page	Tag	Index	TLB H/M	Valid	Tag	Physical Page	Index
4669 0x123d	1	0	1	TLB miss PT hit PF	1	b	12	0
					1	0	13	1
					1	3	6	2
					0	4	9	3
2227 0x08b3	0	0	0	TLB miss PT hit	1	0	5	0
					1	0	13	1
					1	3	6	2
					0	4	9	3
13916 0x365c	3	0	3	TLB miss PT hit	1	0	5	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
34587 0x871b	8	2	0	TLB miss PT hit PF	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
48870 0xbee6	b	2	3	TLB miss PT hit	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	2	6	3
12608 0x3140	3	0	3	TLB hit PT hit	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
49225 0xc049	c	3	0	TLB miss PT miss PF	1	3	15	0
					1	0	13	1
					1	3	6	2
					1	0	6	3

5.16.5 Without a TLB, almost every memory access would require two accesses to RAM: An access to the page table, followed by an access to the requested data.

5.17

5.17

5.17.1 The tag size is 32–log2(8192) = 32–13 = 19 bits. All five page tables would require 5 × (2^19 × 4) bytes = 10 MB.

5.17.2 In the two-level approach, the 2^19 page table entries are divided into 256 segments that are allocated on demand. Each of the second-level tables contains 2^(19 – 8) = 2048 entries, requiring 2048 × 4 = 8 KB each and covering 2048 × 8 KB = 16 MB (2^24) of the virtual address space.

If we assume that "half the memory" means 2^31 bytes, then the minimum amount of memory required for the second-level tables would be 5 × (2^31/2^24)*8 KB = 5 MB. The first-level tables would require an additional 5 × 128 × 6 bytes = 3840 bytes.

The maximum amount would be if all 1st-level segments were activated, requiring the use of all 256 segments in each application. This would require 5 × 256 × 8 KB = 10 MB for the second-level tables and 7680 bytes for the first-level tables.

5.17.3 The page index is 13 bits (address bits 12 down to 0).

A 16 KB direct-mapped cache with two 64-bit words per block would have 16-byte blocks and thus 16 KB/16 bytes = 1024 blocks. Thus, it would have 10 index bits and 4 off set bits and the index would extend outside of the page index.

The designer could increase the cache's associativity. This would reduce the number of index bits so that the cache's index fits completely inside the page index.

5.20

5.20.1 There are no hits

5.20.2 Direct mapped

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0
M	M	M	M	M	M	M	M	H	H	M	M	M	M	H	H	M

5.20.3 Answers will vary.

5.20.4 MRU is an optimal policy.

5.20.5 The best block to evict is the one that will cause the fewest misses in the future. Unfortunately, a cache controller cannot know the future! Our best alternative is to make a good prediction.

5.20.6 If you knew that an address had limited temporal locality and would conflict with another block in the cache, choosing not to cache it could improve the miss rate. On the other hand, you could worsen the miss rate by choosing poorly which addresses to cache.

5.21

5.21

5.21.1 CPI = 1.5 + 120/10000 × (15 + 175) = 3.78

If VMM overhead doubles = >CPI = 1.5 + 120/10000 × (15 + 350) = 5.88

If VMM overhead halves = >CPI = 1.5 + 120/10000 × (15 + 87.5) = 2.73

The CPI of a machine running on native hardware is 1.5 + 120/10000*15 = 1.68. To keep the performance degradation to 10%, we need

1.5 + 120/10000*(15 + x) < 1.1*1.68

Solving for x shows that a trap to the VMM can take at most 14 cycles.

5.21.2 Non-virtualized CPI = 1.5 + 120/10000 × 15 + 30/10000 × 1100 = 4.98

Virtualized CPI = 1.5 + 120/10000 × (15 + 175) + 30/10000 × (1100 + 175) = 7.60

Non-virtualized CPI with half I/O = 1.5 + 120/10000 × 15 + 15/10000 × 1100 = 3.33

Virtualized CPI with half I/O = 1.5 + 120/10000 × (15 + 175) + 15/10000 × (1100 + 175) = 5.69.

5.29

5.29.1 Shadow page table: (1) VM creates page table, hypervisor updates shadow table; (2) nothing; (3) hypervisor intercepts page fault, creates new mapping, and invalidates the old mapping in TLB; (4) VM notifies the hypervisor to invalidate the process's TLB entries. Nested page table: (1) VM creates new page table, hypervisor adds new mappings in PA to MA table. (2) Hardware walks both page tables to translate VA to MA; (3) VM and hypervisor update their page tables, hypervisor invalidates stale TLB entries; (4) same as shadow page table.

5.29.2 Native: 4; NPT: 24 (instructors can change the levels of page table)

Native: L; NPT: L × (L + 2).

5.29.3 Shadow page table: page fault rate.

NPT: TLB miss rate.

5.29.4 Shadow page table: 1.03

NPT: 1.04.

5.29.5 Combining multiple page table updates.

5.29.6 NPT caching (similar to TLB caching).