

Ex5: 5.4, 5.6, 5.11, 5.12, 5.14 due June 6th, 2019.

5.4 Recall that we have two write policies and write allocation policies, and their combinations can be implemented either in L1 or L2 cache. Assume the following choices for L1 and L2 caches:

L1	L2
Write through, non-write allocate	Write back, write allocate

5.4.1 [5] <§§5.3, 5.8> Buffers are employed between different levels of memory hierarchy to reduce access latency. For this given configuration, list the possible buffers needed between L1 and L2 caches, as well as L2 cache and memory.

5.4.2 [20] <§§5.3, 5.8> Describe the procedure of handling an L1 write-miss, considering the component involved and the possibility of replacing a dirty block.

5.4.3 [20] <§§5.3, 5.8> For a multilevel exclusive cache (a block can only reside in one of the L1 and L2 caches), configuration, describe the procedure of handling an L1 write-miss, considering the component involved and the possibility of replacing a dirty block.

Consider the following program and cache behaviors.

Data Reads per 1000 Instructions	Data Writes per 1000 Instructions	Instruction Cache Miss Rate	Data Cache Miss Rate	Block Size (byte)
250	100	0.30%	2%	64

5.4.4 [5] <§§5.3, 5.8> For a write-through, write-allocate cache, what are the minimum read and write bandwidths (measured by byte per cycle) needed to achieve a CPI of 2?

5.4.5 [5] <§§5.3, 5.8> For a write-back, write-allocate cache, assuming 30% of replaced data cache blocks are dirty, what are the minimal read and write bandwidths needed for a CPI of 2?

5.4.6 [5] <§§5.3, 5.8> What are the minimal bandwidths needed to achieve the performance of CPI=1.5?

5.6 In this exercise, we will look at the different ways capacity affects overall performance. In general, cache access time is proportional to capacity. Assume that main memory accesses take 70 ns and that memory accesses are 36% of all instructions. The following table shows data for L1 caches attached to each of two processors, P1 and P2.

	L1 Size	L1 Miss Rate	L1 Hit Time
P1	2 KiB	8.0%	0.66 ns
P2	4 KiB	6.0%	0.90 ns

5.6.1 [5] <§5.4> Assuming that the L1 hit time determines the cycle times for P1 and P2, what are their respective clock rates?

5.6.2 [5] <§5.4> What is the Average Memory Access Time for P1 and P2?

5.6.3 [5] <§5.4> Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 and P2? Which processor is faster?

For the next three problems, we will consider the addition of an L2 cache to P1 to presumably make up for its limited L1 cache capacity. Use the L1 cache capacities and hit times from the previous table when solving these problems. The L2 miss rate indicated is its local miss rate.

L2 Size	L2 Miss Rate	L2 Hit Time
1 MiB	95%	5.62 ns

5.6.4 [10] <§5.4> What is the AMAT for P1 with the addition of an L2 cache? Is the AMAT better or worse with the L2 cache?

5.6.5 [5] <§5.4> Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 with the addition of an L2 cache?

5.6.6 [10] <§5.4> Which processor is faster, now that P1 has an L2 cache? If P1 is faster, what miss rate would P2 need in its L1 cache to match P1's performance? If P2 is faster, what miss rate would P1 need in its L1 cache to match P2's performance?

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 [10] <\$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.

5.11.2 [15] <\$5.7> Repeat 5.11.1, but this time use 16 KiB pages instead of 4 KiB pages. What would be some of the advantages of having a larger page size? What are some of the disadvantages?

5.11.3 [15] <\$\$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?

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.

5.11.6 [10] <\$5.7> A cache designer wants to increase the size of a 4 KiB virtually indexed, physically tagged cache. Given the page size shown above, is it possible to make a 16 KiB direct-mapped cache, assuming 2 words per block? How would the designer increase the data size of the cache?

5.12 In this exercise, we will examine space/time optimizations for page tables. The following list provides parameters of a virtual memory system.

Virtual Address (bits)	Physical DRAM Installed	Page Size	PTE Size (byte)
43	16 GiB	4 KiB	4

5.12.1 [10] <§5.7> For a single-level page table, how many page table entries (PTEs) are needed? How much physical memory is needed for storing the page table?

5.12.2 [10] <§5.7> Using a multilevel page table can reduce the physical memory consumption of page tables, by only keeping active PTEs in physical memory. How many levels of page tables will be needed in this case? And how many memory references are needed for address translation if missing in TLB?

5.12.3 [15] <§5.7> An inverted page table can be used to further optimize space and time. How many PTEs are needed to store the page table? Assuming a hash table implementation, what are the common case and worst case numbers of memory references needed for servicing a TLB miss?

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? When would a software managed TLB be faster than a hardware managed TLB?

5.12.6 [5] <§5.7> What happens when an instruction writes to VA page 200?

5.14 To support multiple virtual machines, two levels of memory virtualization are needed. Each virtual machine still controls the mapping of virtual address (VA) to physical address (PA), while the hypervisor maps the physical address (PA) of each virtual machine to the actual machine address (MA). To accelerate such mappings, a software approach called “shadow paging” duplicates each virtual machine’s page tables in the hypervisor, and intercepts VA to PA mapping changes to keep both copies consistent. To remove the complexity of shadow page tables, a hardware approach called nested page table (NPT) explicitly supports two classes of page tables ($VA \Rightarrow PA$ and $PA \Rightarrow MA$) and can walk such tables purely in hardware.

Consider the following sequence of operations: (1) Create process; (2) TLB miss; (3) page fault; (4) context switch;

5.14.1 [10] <§5.6, 5.7> What would happen for the given operation sequence for shadow page table and nested page table, respectively?

5.14.2 [10] <§5.6, 5.7> Assuming an x86-based 4-level page table in both guest and nested page table, how many memory references are needed to service a TLB miss for native vs. nested page table?

5.14.3 [15] <§5.6, 5.7> Among TLB miss rate, TLB miss latency, page fault rate, and page fault handler latency, which metrics are more important for shadow page table? Which are important for nested page table?

Assume the following parameters for a shadow paging system.

TLB Misses per 1000 Instructions	NPT TLB Miss Latency	Page Faults per 1000 Instructions	Shadowing Page Fault Overhead
0.2	200 cycles	0.001	30,000 cycles

5.14.4 [10] <§5.6> For a benchmark with native execution CPI of 1, what are the CPI numbers if using shadow page tables vs. NPT (assuming only page table virtualization overhead)?

5.14.5 [10] <§5.6> What techniques can be used to reduce page table shadowing induced overhead?

5.14.6 [10] <§5.6> What techniques can be used to reduce NPT induced overhead?