5.2.4 [10] <5.1> How many 16-byte cache lines are needed to store all 32-bit matrix elements being referenced?

5.2.5 [5] <5.1> References to which variables exhibit temporal locality?

5.2.6 [5] <5.1> References to which variables exhibit spatial locality?

Exercise 5.3

Caches are important to providing a high performance memory hierarchy to processors. Below is a list of 32-bit memory address references, given as word addresses.

a.	1, 134, 212, 1, 135, 213, 162, 161, 2, 44, 41, 221
b.	6, 214, 175, 214, 6, 84, 65, 174, 64, 105, 85, 215

5.3.1 [10] <5.2> For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

5.3.2 [10] <5.2> For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with two-word blocks and a total size of eight blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

5.3.3 [20] <5.2, 5.3> You are asked to optimize a cache design for the given references. There are three direct-mapped cache designs possible, all with a total of eight words of data: C1 has one-word blocks, C2 has two-word blocks, and C3 has four-word blocks. In terms of miss rate, which cache design is the best? If the miss stall time is 25 cycles, and C1 has an access time of 2 cycles, C2 takes 3 cycles, and C3 takes 5 cycles, which is the best cache design?

There are many different design parameters that are important to a cache's overall performance. The table below lists parameters for different direct-mapped cache designs.

	Cache data size	Cache block size	Cache access time
a.	64 KB	1 word	1 cycle
b.	64 KB	2 word	2 cycle

5.3.4 [15] <5.2> Calculate the total number of bits required for the cache listed in the table, assuming a 32-bit address. Given that total size, find the total size

of the closest direct-mapped cache with 16-word blocks of equal size or greater. Explain why the second cache, despite its larger data size, might provide slower performance than the first cache.

5.3.5 [20] <5.2, 5.3> Generate a series of read requests that have a lower miss rate on a 2 KB two-way set associative cache than the cache listed in the table. Identify one possible solution that would make the cache listed in the table have an equal or lower miss rate than the 2 KB cache. Discuss the advantages and disadvantages of such a solution.

5.3.6 [15] <5.2> The formula shown on page 457 shows the typical method to index a direct-mapped cache, specifically (Block address) modulo (Number of blocks in the cache). Assuming a 32-bit address and 1024 blocks in the cache, consider a different indexing function, specifically (Block address[31:27] XOR Block address[26:22]). Is it possible to use this to index a direct-mapped cache? If so, explain why and discuss any changes that might need to be made to the cache. If it is not possible, explain why.

Exercise 5.4

For a direct-mapped cache design with 32-bit address, the following bits of the address are used to access the cache.

	Tag	Index	Offset
a.	31–10	9–4	3–0
b.	31–12	11–15	4–0

5.4.1 [5] <5.2> What is the cache line size (in words)?

5.4.2 [5] <5.2> How many entries does the cache have?

5.4.3 [5] <5.2> What is the ratio between total bits required for such a cache implementation over the data storage bits?

Starting from power on, the following byte-addressed cache references are recorded.

Address	0	4	16	132	232	160	1024	30	140	3100	180	2180

5.4.4 [10] <5.2> How many blocks are replaced?

5.4.5 [10] <5.2> What is the hit ratio?

5.4.6 [20] <5.2> List the final state of the cache, with each valid entry represented as a record of <index, tag, data>.

Exercise 5.5

Recall that we have two write policies and write allocation policies, their combinations can be implemented at either in L1 or L2 cache.

	L1	L2
a.	Write-back, write allocate	Write-through, non write allocate
b.	Write-back, write allocate	Write-through, write allocate

5.5.1 [5] <5.2, 5.5> 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.

5.5.2 [20] <5.2, 5.5> Describe the procedure of handling an L1 write miss, considering the component involved and the possibility of replacing a dirty block.

5.5.3 [20] <5.2, 5.5> 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)
a.	200	160	0.20%	2%	8
b.	180	120	0.20%	2%	16

5.5.4 [5] <5.2, 5.5> For a write-through, write-allocate cache, what's the minimum read and write bandwidths (measured by byte-per-cycle) needed to achieve a CPI of 2?

5.5.5 [5] <5.2, 5.5> For a write-back, write-allocate cache, assuming 30% of replaced data cache blocks are dirty, what's the minimal read and write bandwidths needed for a CPI of 2?

5.5.6 [5] <5.2, 5.5> What are the minimal bandwidths needed to achieve the performance of CPI = 1.5?

Exercise 5.8

This exercise examines the impact of different cache designs, specifically comparing associative caches to the direct-mapped caches from Section 5.2. For these exercises, refer to the table of address streams shown in Exercise 5.3.

5.8.1 [10] <5.3> Using the references from Exercise 5.3, 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.

5.8.2 [10] <5.3> Using the references from Exercise 5.3, show the final cache contents for a fully associative cache with one-word blocks and a total size of eight words. Use LRU replacement. For each reference identify the index bits, the tag bits, and if it is a hit or a miss.

5.8.3 [15] <5.3> Using the references from Exercise 5.3, what is the miss rate for a fully associative cache with two-word blocks and a total size of eight 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 miss rate 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
a.	2.0	3 GHz	125 ns	5%	15 cycles	3.0%	25 cycles	1.8%
b.	2.0	1 GHz	100 ns	4%	10 cycles	4.0%	20 cycles	1.6%

5.8.4 [10] <5.3> 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?

Keeping "frequently used" (or "hot") pages in DRAM can save disk accesses, but how do we determine the exact meaning of "frequently used" for a given system? Data engineers use the cost ratio between DRAM and disk access to quantify the reuse time threshold for hot pages. The cost of a disk access is $Disk/accesses_per_sec, while the cost of keep a page in DRAM is $DRAM_MB/page_size. The typical DRAM and disk costs, and typical database page sizes at several time points are listed below:

Year	DRAM cost ($/MB)	Page size (KB)	Disk cost ($/disk)	Disk access rate (access/sec)
1987	5000	1	15000	15
1997	15	8	2000	64
2007	0.05	64	80	83

5.9.4 [10] <5.1, 5.4> What are the reuse time thresholds for these three technology generations?

5.9.5 [10] <5.4> What are the reuse time thresholds if we keep using the same 4K page size? What's the trend here?

5.9.6 [20] <5.4> What other factors can be changed to keep using the same page size (thus avoiding software rewrite)? Discuss their likeliness with current technology and cost trends.

Exercise 5.10

As described in Section 5.4, 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 table is a stream of virtual addresses as seen on a system. Assume 4 KB pages, a four-entry fully associative TLB, and true LRU replacement. If pages must be brought in from disk, increment the next largest page number.

a.	4095, 31272, 15789, 15000, 7193, 4096, 8912
b.	9452, 30964, 19136, 46502, 38110, 16653, 48480

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

5.10.1 [10] <5.4> Given the address stream in the table, and the shown initial state of the TLB and page table, 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.

5.10.2 [15] <5.4> Repeat Exercise 5.10.1, but this time use 16 KB pages instead of 4 KB pages. What would be some of the advantages of having a larger page size? What are some of the disadvantages?

5.10.3 [15] <5.3, 5.4> Show the final contents of the TLB if it is two-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 several key page table parameters.

	Virtual address size	Page size	Page table entry size
a.	32 bits	4 KB	4 bytes
b.	64 bits	16 KB	8 bytes

5.10.4 [5] <5.4> Given the parameters in the table above, calculate the total page table size for a system running five applications that utilize half of the memory available.

5.11.4 [5] <5.4> Under what scenarios would entry 2's valid bit be set to 0?

5.11.5 [5] <5.4> What happens when an instruction writes to VA page 30? When would a software-managed TLB be faster than a hardware-managed TLB?

5.11.6 [5] <5.4> What happens when an instruction writes to VA page xxx?

Exercise 5.12

In this exercise, we will examine how replacement policies impact miss rate. Assume a two-way set-associative cache with four blocks. You may find it helpful to draw a table like those found on page 483 to solve the problems in this exercise, as demonstrated below on the address sequence "0, 1, 2, 3, 4".

Address of memory block accessed	Hit or miss	Evicted block	Contents of cache blocks after reference			
			Set 0	Set 0	Set 1	Set 1
0	Miss		Mem[0]			
1	Miss		Mem[0]		Mem[1]	
2	Miss		Mem[0]	Mem[2]	Mem[1]	
3	Miss		Mem[0]	Mem[2]	Mem[1]	Mem[3]
4	Miss	0	Mem[4]	Mem[2]	Mem[1]	Mem[3]
...						

The following table shows address sequences.

	Address sequence
a.	0, 2, 4, 0, 2, 4, 0, 2, 4
b.	0, 2, 4, 2, 0, 2, 4, 0, 2

5.12.1 [5] <5.3, 5.5> Assuming an LRU replacement policy, how many hits does this address sequence exhibit?

5.12.2 [5] <5.3, 5.5> Assuming an MRU (most recently used) replacement policy, how many hits does this address sequence exhibit?

5.12.3 [5] <5.3, 5.5> Simulate a random replacement policy by flipping a coin. For example, "heads" means to evict the first block in a set and "tails" means to evict the second block in a set. How many hits does this address sequence exhibit?

5.12.4 [10] <5.3, 5.5> Which address should be evicted at each replacement to maximize the number of hits? How many hits does this address sequence exhibit if you follow this "optimal" policy?

5.12.5 [10] <5.3, 5.5> Describe why it is difficult to implement a cache replacement policy that is optimal for all address sequences.

5.12.6 [10] <5.3, 5.5> Assume you could make a decision upon each memory reference whether or not you want the requested address to be cached. What impact could this have on miss rate?

Exercise 5.13

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 (or extended page table) explicitly support two classes of page tables (VA⇨PA and PA⇨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;

5.13.1 [10] <5.4, 5.6> What would happen for the given operation sequence, for shadow page table, and nested page table respectively?

5.13.2 [10] <5.4, 5.6> Assuming an x86-based four-level page table in both guest and nested page table, how many memory references are needed to service a TLB miss for native versus nested page table?

5.13.3 [15] <5.4, 5.6> 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?

The following table shows parameters for a shadow paging system.

TLB misses per 1000 instruction	NPT TLB miss latency	Page faults per 1000 instruction	Shadowing page fault overhead
0.2	200 cycles	0.001	30000 cycles