As we will see in Chapter 7, memory systems are a central design issue for parallel processors. The growing importance of the memory hierarchy in determining system performance means that this important area will continue to be a focus of both designers and researchers for some years to come.

5.13 Historical Perspective and Further Reading

This history section gives an overview of memory technologies, from mercury delay lines to DRAM, the invention of the memory hierarchy, protection mechanisms, and virtual machines, and concludes with a brief history of operating systems, including CTSS, MULTICS, UNIX, BSD UNIX, MS-DOS, Windows, and Linux.

5.14 Exercises

Contributed by Jichuan Chang, Jacob Leverich, Kevin Lim, and Parthasarathy Ranganathan (all of Hewlett-Packard)

Exercise 5.1

In this exercise we consider memory hierarchies for various applications, listed in the following table.

a.	Software version control
b.	Making phone calls

5.1.1 [10] <5.1> Assuming both client and server are involved in the process, first name the client and server systems. Where can caches be placed to speed up the process?

5.1.2 [10] <5.1> Design a memory hierarchy for the system. Show the typical size and latency at various levels of the hierarchy. What is the relationship between cache size and its access latency?

5.1.3 [15] <5.1> What are the units of data transfers between hierarchies? What is the relationship between the data location, data size, and transfer latency?

5.1.4 [10] <5.1, 5.2> Communication bandwidth and server processing bandwidth are two important factors to consider when designing a memory hierarchy. How can the bandwidths be improved? What is the cost of improving them?

5.1.5 [5] <5.1, 5.8> Now consider multiple clients simultaneously accessing the server. Will such scenarios improve the spatial and temporal locality?

5.1.6 [10] <5.1, 5.8> Give an example of where the cache can provide out-of-date data. How should the cache be designed to mitigate or avoid such issues?

Exercise 5.2

In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where elements within the same row are stored contiguously.

a.
```
for (I=0; I<8; I++)
  for (J=0; J<8000; J++)
    A[I][J]=B[I][0]+A[J][I];
```

b.
```
for (J=0; J<8000; J++)
   for (I=0; I<8; I++)
    A[I][J]=B[I][0]+A[J][I];
```

5.2.1 [5] <5.1> How many 32-bit integers can be stored in a 16-byte cache line?

5.2.2 [5] <5.1> References to which variables exhibit temporal locality?

5.2.3 [5] <5.1> References to which variables exhibit spatial locality?

Locality is affected by both the reference order and data layout. The same computation can also be written below in Matlab, which differs from C by contiguously storing matrix elements within the same column.

a.
```
for I=1:8
  for J=1:8000
    A(I,J)=B(I,0)+A(J,I);
  end
end
```

b.
```
for J=1:8000
  for I=1:8
    A(I,J)=B(I,0)+A(J,I);
  end
end
```

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.	3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253
b.	21, 166, 201, 143, 61, 166, 62, 133, 111, 143, 144, 61

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 8 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 8 words of data: C1 has 1-word blocks, C2 has 2-word blocks, and C3 has 4-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.	32 KB	2 words	1 cycle
b.	32 KB	4 words	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 2-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 a 32-bit address, the following bits of the address are used to access the cache.

	Tag	Index	Offset
a.	31–10	9–5	4–0
b.	31–12	11–6	5–0

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, and their combinations can be implemented either in L1 or L2 cache.

	L1	L2
a.	Write through, non-write allocate	Write back, write allocate
b.	Write through, write allocate	Write back, 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.	250	100	0.30%	2%	64
b.	200	100	0.30%	2%	64

5.5.4 [5] <5.2, 5.5> 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?

5.5.5 [5] <5.2, 5.5> 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?

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

Exercise 5.6

Media applications that play audio or video files are part of a class of workloads called "streaming" workloads; i.e., they bring in large amounts of data but do not reuse much of it. Consider a video streaming workload that accesses a 512 KB working set sequentially with the following address stream:

0, 2, 4, 6, 8, 10, 12, 14, 16, …

5.6.1 [5] <5.5, 5.3> Assume a 64 KB direct-mapped cache with a 32-byte line. What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C model?

5.6.2 [5] <5.5, 5.1> Re-compute the miss rate when the cache line size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?

5.6.3 [10] <5.10> "Prefetching" is a technique that leverages predictable address patterns to speculatively bring in additional cache lines when a particular cache line is accessed. One example of prefetching is a stream buffer that prefetches sequentially adjacent cache lines into a separate buffer when a particular cache line is brought in. If the data is found in the prefetch buffer, it is considered as a hit and moved into the cache and the next cache line is prefetched. Assume a two-entry stream buffer and assume that the cache latency is such that a cache line can be loaded before the computation on the previous cache line is completed. What is the miss rate for the address stream above?

Cache block size (B) can affect both miss rate and miss latency. Assuming a 1-CPI machine with an average of 1.35 references (both instruction and data) per instruction, help find the optimal block size given the following miss rates for various block sizes.

	8	16	32	64	128
a.	4%	3%	2%	1.5%	1%
b.	8%	7%	6%	5%	4%

5.6.4 [10] <5.2> What is the optimal block size for a miss latency of 20×B cycles?

5.6.5 [10] <5.2> What is the optimal block size for a miss latency of 24+B cycles?

5.6.6 [10] <5.2> For constant miss latency, what is the optimal block size?

Exercise 5.7

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
a.	P1	2 KB	8.0%	0.66 ns
	P2	4 KB	6.0%	0.90 ns
b.	P1	16 KB	3.4%	1.08 ns
	P2	32 KB	2.9%	2.02 ns

5.7.1 [5] <5.3> Assuming that the L1 hit time determines the cycle times for P1 and P2, what are their respective clock rates?

5.7.2 [5] <5.3> What is the AMAT for P1 and P2?

5.7.3 [5] <5.3> 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
a.	1 MB	95%	5.62 ns
b.	8 MB	68%	23.52 ns

5.7.4 [10] <5.3> What is the AMAT for P1 with the addition of an L2 cache? Is the AMAT better or worse with the L2 cache?

5.7.5 [5] <5.3> 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?

5.7.6 [10] <5.3> 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?

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 8 words. Use LRU replacement. For each reference identify the index bits, the tag bits, and if it is a hit or a miss.

5.8.3 [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 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 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.	1.5	2 GHz	100 ns	7%	12 cycles	3.5%	28 cycles	1.5%
b.	1.0	2 GHz	150 ns	3%	15 cycles	5.0%	20 cycles	2.0%

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?

5.8.5 [10] <5.3> It is possible to have an even greater cache hierarchy than two levels. Given the processor above with a second level, direct-mapped cache, a designer wants to add a third level cache that takes 50 cycles to access and will reduce the global miss rate to 1.3%. Would this provide better performance? In general, what are the advantages and disadvantages of adding a third level cache?

5.8.6 [20] <5.3> In older processors such as the Intel Pentium or Alpha 21264, the second level of cache was external (located on a different chip) from the main processor and the first level cache. While this allowed for large second level caches, the latency to access the cache was much higher, and the bandwidth was typically lower because the second level cache ran at a lower frequency. Assume a 512 KB off-chip second level cache has a global miss rate of 4%. If each additional 512 KB of cache lowered global miss rates by 0.7%, and the cache had a total access time of 50 cycles, how big would the cache have to be to match the performance of the second level direct-mapped cache listed in the table? Of the eight-way set associative cache?

Exercise 5.9

For a high-performance system such as a B-tree index for a database, the page size is determined mainly by the data size and disk performance. Assume that on average a B-tree index page is 70% full with fix-sized entries. The utility of a page is its B-tree depth, calculated as $\log_2(\text{entries})$. The following table shows that for 16-byte entries, and a 10-year-old disk with a 10 ms latency and 10 MB/s transfer rate, the optimal page size is 16K.

Page Size (KB)	Page Utility or B-Tree Depth (Number of Disk Accesses Saved)	Index Page Access Cost (ms)	Utility/Cost
2	6.49 (or $\log_2(2048/16\times0.7)$)	10.2	0.64
4	7.49	10.4	0.72
8	8.49	10.8	0.79
16	9.49	11.6	0.82
32	10.49	13.2	0.79
64	11.49	16.4	0.70
128	12.49	22.8	0.55
256	13.49	35.6	0.38

5.9.1 [10] <5.4> What is the best page size if entries now become 128 bytes?

5.9.2 [10] <5.4> Based on 5.9.1, what is the best page size if pages are half full?

5.9.3 [20] <5.4> Based on 5.9.2, what is the best page size if using a modern disk with a 3 ms latency and 100 MB/s transfer rate? Explain why future servers are likely to have larger pages.

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 to 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 4-entry fully associative TLB, and true LRU replacement. If pages must be brought in from disk, increment the next largest page number.

a.	4669, 2227, 13916, 34587, 48870, 12608, 49225
b.	12948, 49419, 46814, 13975, 40004, 12707, 52236

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 initial TLB and page table states shown 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.

5.10.2 [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 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 several key page table parameters.

	Virtual Address Size	Page Size	Page Table Entry Size
a.	32 bits	8 KB	4 bytes
b.	64 bits	8 KB	6 bytes

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

5.10.5 [10] <5.4> Given the parameters in the table 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.

5.10.6 [10] <5.4> A cache designer wants to increase the size of a 4 KB virtually indexed, physically tagged cache. Given the page size listed in the table above, is it possible to make a 16 KB direct-mapped cache, assuming 2 words per block? How would the designer increase the data size of the cache?

Exercise 5.11

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

	Virtual Address (bits)	Physical DRAM Installed	Page Size	PTE Size (byte)
a.	43	16 GB	4 KB	4
b.	38	8 GB	16 KB	4

5.11.1 [10] <5.4> 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?

5.11.2 [10] <5.4> 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?

5.11.3 [15] <5.4> 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

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

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 200?

Exercise 5.12

In this exercise, we will examine how replacement policies impact miss rate. Assume a 2-way set associative cache with 4 blocks. You may find it helpful to draw a table like those found on page 482 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, 8, 10, 12, 14, 16, 0
b.	1, 3, 5, 1, 3, 1, 3, 5, 3

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 supports 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 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?

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

5.13.4 [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)?

5.13.5 [10] <5.6> What techniques can be used to reduce page table shadowing induced overhead?

5.13.6 [10] <5.6> What techniques can be used to reduce NPT induced overhead?

Exercise 5.14

One of the biggest impediments to widespread use of virtual machines is the performance overhead incurred by running a virtual machine. The table below lists various performance parameters and application behavior.

	Base CPI	Priviliged O/S Accesses per 10,000 Instructions	Performance Impact to Trap to the Guest O/S	Performance Impact to Trap to VMM	I/O Accesses per 10,000 Instructions	I/O Access Time (Includes Time to Trap to Guest O/S)
a.	1.5	120	15 cycles	175 cycles	30	1100 cycles
b.	1.75	90	20 cycles	140 cycles	25	1200 cycles

5.14.1 [10] <5.6> Calculate the CPI for the system listed above assuming that there are no accesses to I/O. What is the CPI if the VMM performance impact doubles? If it is cut in half? If a virtual machine software company wishes to obtain a 10% performance degradation, what is the longest possible penalty to trap to the VMM?

5.14.2 [10] <5.6> I/O accesses often have a large impact on overall system performance. Calculate the CPI of a machine using the performance characteristics above, assuming a non-virtualized system. Calculate the CPI again, this time using a virtualized system. How do these CPIs change if the system has half the I/O accesses? Explain why I/O bound applications have a smaller impact from virtualization.

5.14.3 [30] <5.4, 5.6> Compare and contrast the ideas of virtual memory and virtual machines. How do the goals of each compare? What are the pros and cons of each? List a few cases where virtual memory is desired, and a few cases where virtual machines are desired.

5.14.4 [20] <5.6> Section 5.6 discusses virtualization under the assumption that the virtualized system is running the same ISA as the underlying hardware. However, one possible use of virtualization is to emulate non-native ISAs. An example of this is QEMU, which emulates a variety of ISAs such as MIPS, SPARC, and PowerPC. What are some of the difficulties involved in this kind of virtualization? Is it possible for an emulated system to run faster than on its native ISA?

Exercise 5.15

In this exercise, we will explore the control unit for a cache controller for a processor with a write buffer. Use the finite state machine found in Figure 5.34 as a starting point for designing your own finite state machines. Assume that the cache controller is for the simple direct-mapped cache described on page 529, but you will add a write buffer with a capacity of one block.

Recall that the purpose of a write buffer is to serve as temporary storage so that the processor doesn't have to wait for two memory accesses on a dirty miss. Rather than writing back the dirty block before reading the new block, it buffers the dirty block and immediately begins reading the new block. The dirty block can then be written to main memory while the processor is working.

5.15.1 [10] <5.5, 5.7> What should happen if the processor issues a request that *hits* in the cache while a block is being written back to main memory from the write buffer?

5.15.2 [10] <5.5, 5.7> What should happen if the processor issues a request that *misses* in the cache while a block is being written back to main memory from the write buffer?

5.15.3 [30] <5.5, 5.7> Design a finite state machine to enable the use of a write buffer.

Exercise 5.16

Cache coherence concerns the views of multiple processors on a given cache block. The following table shows two processors and their read/write operations on two different words of a cache block X (initially X[0] = X[1] = 0).

	P1	P2
a.	`X[0] ++; X[1] = 3;`	`X[0] = 5; X[1] +=2;`
b.	`X[0] =10; X[1] = 3;`	`X[0] = 5; X[1] +=2;`

5.16.1 [15] <5.8> List the possible values of the given cache block for a correct cache coherence protocol implementation. List at least one more possible value of the block if the protocol doesn't ensure cache coherency.

5.16.2 [15] <5.8> For a snooping protocol, list a valid operation sequence on each processor/cache to finish the above read/write operations.

5.16.3 [10] <5.8> What are the best-case and worst-case numbers of cache misses needed to execute the listed read/write instructions?

Memory consistency concerns the views of multiple data items. The following table shows two processors and their read/write operations on different cache blocks (A and B initially 0).

	P1	P2
a.	`A = 1; B = 2; A+=2; B++;`	`C = B; D = A;`
b.	`A = 1; B = 2; A=5; B++;`	`C = B; D = A;`

5.16.4 [15] <5.8> List the possible values of C and D for an implementation that ensures both consistency assumptions on page 538.

5.16.5 [15] <5.8> List at least one more possible pair of values for C and D if such assumptions are not maintained.

5.16.6 [15] <5.2, 5.8> For various combinations of write policies and write allocation policies, which combinations make the protocol implementation simpler?

Exercise 5.17

Both Barcelona and Nehalem are chip multiprocessors (CMPs), having multiple cores and their caches on a single chip. CMP on-chip L2 cache design has interesting trade-offs. The following table shows the miss rates and hit latencies for two benchmarks with private vs. shared L2 cache designs. Assume L1 cache misses once every 32 instructions.

	Private	Shared
Benchmark A misses-per-instruction	0.30%	0.12%
Benchmark B misses-per-instruction	0.06%	0.03%

The next table shows hit latencies.

	Private Cache	Shared Cache	Memory
a.	5	20	180
b.	10	50	120

5.17.1 [15] <5.10> Which cache design is better for each of these benchmarks? Use data to support your conclusion.

5.17.2 [15] <5.10> Shared cache latency increases with the CMP size. Choose the best design if the shared cache latency doubles. Off-chip bandwidth becomes the bottleneck as the number of CMP cores increases. Choose the best design if off-chip memory latency doubles.

5.17.3 [10] <5.10> Discuss the pros and cons of shared vs. private L2 caches for both single-threaded, multi-threaded, and multiprogrammed workloads, and reconsider them if having on-chip L3 caches.

5.17.4 [15] <5.10> Assume both benchmarks have a base CPI of 1 (ideal L2 cache). If having non-blocking cache improves the average number of concurrent L2 misses from 1 to 2, how much performance improvement does this provide over a shared L2 cache? How much improvement can be achieved over private L2?

5.17.5 [10] <5.10> Assume new generations of processors double the number of cores every 18 months. To maintain the same level of per-core performance, how much more off-chip memory bandwidth is needed for a 2012 processor?

5.17.6 [15] <5.10> Consider the entire memory hierarchy. What kinds of optimizations can improve the number of concurrent misses?

Exercise 5.18

In this exercise we show the definition of a web server log and examine code optimizations to improve log processing speed. The data structure for the log is defined as follows:

```
struct entry {
  int  srcIP;    // remote IP address
  char URL[128]; // request URL (e.g., “GET index.html”)
  long long refTime;  // reference time
  int  status;   // connection status
  char browser[64]; // client browser name
} log [NUM_ENTRIES];
```

Some processing functions on a log are:

a.	`topK_sourceIP (int hour);`
b.	`browser_histogram (int srcIP); // browsers of a given IP`

5.18.1 [5] <5.11> Which fields in a log entry will be accessed for the given log processing function? Assuming 64-byte cache blocks and no prefetching, how many cache misses per entry does the given function incur on average?

5.18.2 [10] <5.11> How can you reorganize the data structure to improve cache utilization and access locality? Show your structure definition code.

5.18.3 [10] <5.11> Give an example of another log processing function that would prefer a different data structure layout. If both functions are important, how would you rewrite the program to improve the overall performance? Supplement the discussion with code snippet and data.

For the problems below, use data from "Cache Performance for SPEC CPU2000 Benchmarks" (http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/) for the pairs of benchmarks shown in the following table.

a.	Mesa / gcc
b.	mcf / swim

5.18.4 [10] <5.11> For 64 KB data caches with varying set associativities, what are the miss rates broken down by miss types (cold, capacity, and conflict misses) for each benchmark?

5.18.5 [10] <5.11> Select the set associativity to be used by a 64 KB L1 data cache shared by both benchmarks. If the L1 cache has to be directly mapped, select the set associativity for the 1 MB L2 cache.

5.18.6 [20] <5.11> Give an example in the miss rate table where higher set associativity actually increases miss rate. Construct a cache configuration and reference stream to demonstrate this.

Answers to Check Yourself

§5.1, page 457: 1 and 4. (3 is false because the cost of the memory hierarchy varies per computer, but in 2008 the highest cost is usually the DRAM.)
§5.2, page 475: 1 and 4: A lower miss penalty can enable smaller blocks, since you don't have that much latency to amortize, yet higher memory bandwidth usually leads to larger blocks, since the miss penalty is only slightly larger.
§5.3, page 491: 1.
§5.4, page 517: 1-a, 2-c, 3-b, 4-d.
§5.5, page 525: 2. (Both large block sizes and prefetching may reduce compulsory misses, so 1 is false.)