


restructuring the loops that access the arrays, substantially improved locality—and, therefore, cache performance—can be obtained. The example on page 551 showed how effective even a simple change of loop structure could be.


Another direction is to try to use compiler-directed prefetching. In prefetching, a block of data is brought into the cache before it is actually referenced. The compiler tries to identify data blocks needed in the future and, using special instructions, tells the memory hierarchy to move the blocks into the cache. When the block is actually referenced, it is found in the cache, rather than causing a cache miss. The use of secondary caches has made prefetching even more attractive, since the secondary cache can be involved in a prefetch, while the primary cache continues to service processor requests.

As we will see in  Chapter 9, memory systems are also a central design issue for parallel processors. The growing importance of the memory hierarchy in determining system performance in both uniprocessor and multiprocessor systems means that this important area will continue to be a focus of both designers and researchers for some years to come.

prefetching A technique in which data blocks needed in the future are brought into the cache early by the use of special instructions that specify the address of the block.



7.9 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 and protection mechanisms, and concludes with a brief history of operating systems, including CTSS, MULTICS, UNIX, BSD UNIX, MS-DOS, Windows, and Linux.

7.10 Exercises

7.1 [5] <§7.1> SRAM is commonly used to implement small, fast, on-chip caches while DRAM is used for larger, slower main memory. In the past, a common design for supercomputers was to build machines with no caches and main memories made entirely out of SRAM (the Cray C90, for example, a very fast computer in its day). If cost were no object, would you still want to design a system this way?

7.2 [10] <§7.2> Describe the general characteristics of a program that would exhibit very little temporal and spatial locality with regard to data accesses. Provide an example program (pseudocode is fine).

7.3 [10] <§7.2> Describe the general characteristics of a program that would exhibit very high amounts of temporal locality but very little spatial locality with regard to data accesses. Provide an example program (pseudocode is fine).

7.4 [10] <§7.2> Describe the general characteristics of a program that would exhibit very little temporal locality but very high amounts of spatial locality with regard to data accesses. Provide an example program (pseudocode is fine).

7.5 [3/3] <§7.2> A new processor can use either a write-through or write-back cache selectable through software.

- a. Assume the processor will run data intensive applications with a large number of load and store operations. Explain which cache write policy should be used.
- b. Consider the same question but this time for a safety critical system in which data integrity is more important than memory performance.

7.6 [10] <§7.2>  For More Practice: Locality.

7.7 [10] <§7.2>  For More Practice: Locality.

7.8 [10] <§7.2>  For More Practice: Locality.

7.9 [10] <§7.2> Here is a series of address references given as word addresses: 2, 3, 11, 16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6, and 11. Assuming a direct-mapped cache with 16 one-word blocks that is initially empty, label each reference in the list as a hit or a miss and show the final contents of the cache.

7.10 [10] <§7.2> Using the series of references given in Exercise 7.9, show the hits and misses and final cache contents for a direct-mapped cache with four-word blocks and a *total size* of 16 words.

7.11 [15] <§7.2> Given the following pseudocode:

```
int array[10000,100000];
for each element array[i][j] {
    array[i][j] = array[i][j]*2;
}
```

write two C programs that implement this algorithm: one should access the elements of the array in row-major order, and the other should access them in column-major order. Compare the execution times of the two programs. What does this tell you about the effects of memory layout on cache performance?

7.12 [10] <§7.2> Compute the total number of bits required to implement the cache in Figure 7.9 on page 486. This number is different from the size of the

cache, which usually refers to the number of bytes of data stored in the cache. The number of bits needed to implement the cache represents the total amount of memory needed for storing all the data, tags, and valid bits.

7.13 [10] <§7.2> Find a method to eliminate the AND gate on the valid bit in Figure 7.7 on page 478. (Hint: You need to change the comparison.)

7.14 [10] <§7.2> Consider a memory hierarchy using one of the three organizations for main memory shown in Figure 7.11 on page 489. Assume that the cache block size is 16 words, that the width of organization (b) of the figure is four words, and that the number of banks in organization (c) is four. If the main memory latency for a new access is 10 memory bus clock cycles and the transfer time is 1 memory bus clock cycle, what are the miss penalties for each of these organizations?

7.15 [10] <§7.2>  For More Practice: Cache Performance.

7.16 [15] <§7.2> Cache C1 is direct-mapped with 16 one-word blocks. Cache C2 is direct-mapped with 4 four-word blocks. Assume that the miss penalty for C1 is 8 memory bus clock cycles and the miss penalty for C2 is 11 memory bus clock cycles. Assuming that the caches are initially empty, find a reference string for which C2 has a lower miss rate but spends more memory bus clock cycles on cache misses than C1. Use word addresses.

7.17 [5] <§7.2>  In More Depth: Average Memory Access Time

7.18 [5] <§7.2>  In More Depth: Average Memory Access Time

7.19 [10] <§7.2>  In More Depth: Average Memory Access Time

7.20 [10] <§7.2> Assume a memory system that supports interleaving either four reads or four writes. Given the following memory addresses in order as they appear on the memory bus: 3, 9, 17, 2, 51, 37, 13, 4, 8, 41, 67, 10, which ones will result in a bank conflict?

7.21 [3 hours] <§7.3> Use a cache simulator to simulate several different cache organizations for the first 1 million references in a trace of gcc. Both dinero (a cache simulator) and the gcc traces are available—see the Preface of this book for information on how to obtain them. Assume an instruction cache of 32 KB and a data cache of 32 KB using the same organization. You should choose at least two kinds of associativity and two block sizes. Draw a diagram like that in Figure 7.17 on page 503 that shows the data cache organization with the best hit rate.

7.22 [1 day] <§7.3> You are commissioned to design a cache for a new system. It has a 32-bit physical byte address and requires separate instruction and data caches. The SRAMs have an access time of 1.5 ns and a size of $32K \times 8$ bits, and you have a total of 16 SRAMs to use. The miss penalty for the memory system is $8 + 2 \times \text{Block}$

size in words. Using set associativity adds 0.2 ns to the cache access time. Using the first 1 million references of gcc, find the best I and D cache organizations, given the available SRAMs.

7.23 [10] <§7.2, B.5>  For More Practice: Cache Configurations

7.24 [10] <§7.2, B.5>  For More Practice: Cache Configurations

7.25 [10] <§7.3>  For More Practice: Cache Operation

7.26 [10] <§7.3>  For More Practice: Cache Operation

7.27 [10] <§7.3>  For More Practice: Cache Operation

7.28 [5] <§7.3> Associativity usually improves the miss ratio, but not always. Give a short series of address references for which a two-way set-associative cache with LRU replacement would experience more misses than a direct-mapped cache of the same size.

7.29 [15] <§7.3> Suppose a computer's address size is k bits (using byte addressing), the cache size is S bytes, the block size is B bytes, and the cache is A -way set-associative. Assume that B is a power of two, so $B = 2^b$. Figure out what the following quantities are in terms of S , B , A , b , and k : the number of sets in the cache, the number of index bits in the address, and the number of bits needed to implement the cache (see Exercise 7.12).

7.30 [10] <§7.3>  For More Practice: Cache Configurations.

7.31 [10] <§7.3>  For More Practice: Cache Configurations.

7.32 [20] <§7.3> Consider three processors with different cache configurations:

- *Cache 1*: Direct-mapped with one-word blocks
- *Cache 2*: Direct-mapped with four-word blocks
- *Cache 3*: Two-way set associative with four-word blocks

The following miss rate measurements have been made:

- *Cache 1*: Instruction miss rate is 4%; data miss rate is 6%.
- *Cache 2*: Instruction miss rate is 2%; data miss rate is 4%.
- *Cache 3*: Instruction miss rate is 2%; data miss rate is 3%.

For these processors, one-half of the instructions contain a data reference. Assume that the cache miss penalty is $6 + \text{Block size}$ in words. The CPI for this workload was measured on a processor with cache 1 and was found to be 2.0. Determine which processor spends the most cycles on cache misses.

7.33 [5] <§7.3> The cycle times for the processors in Exercise 7.32 are 420 ps for the first and second processors and 310 ps for the third processor. Determine which processor is the fastest and which is the slowest.

7.34 [15] <§7.3> Assume that the cache for the system described in Exercise 7.32 is two-way set associative and has eight-word blocks and a total size of 16 KB. Show the cache organization and access using the same format as Figure 7.17 on page 503.

7.35 [10] <§§7.2, 7.4> The following C program is run (with no optimizations) on a processor with a cache that has eight-word (32-byte) blocks and holds 256 bytes of data:

```
int i,j,c,stride,array[512];
...
for (i=0; i<10000; i++)
    for (j=0; j<512; j=j+stride)
        c = array[j]+17;
```

If we consider only the cache activity generated by references to the array and we assume that integers are words, what is the expected miss rate when the cache is direct mapped and stride = 256? How about if stride = 255? Would either of these change if the cache were two-way set associative?

7.36 [10] <§§7.3, B.5>  For More Practice: Cache Configurations

7.37 [5] <§§7.2–7.4>  For More Practice: Memory Hierarchy Interactions

7.38 [4 hours] <§§7.2–7.4> We want to use a cache simulator to simulate several different TLB and virtual memory organizations. Use the first 1 million references of gcc for this evaluation. We want to know the TLB miss rate for each of the following TLBs and page sizes:

1. 64-entry TLB with full associativity and 4 KB pages
2. 32-entry TLB with full associativity and 8 KB pages
3. 64-entry TLB with eight-way associativity and 4 KB pages
4. 128-entry TLB with four-way associativity and 4 KB pages

7.39 [15] <§7.4> Consider a virtual memory system with the following properties:

- 40-bit virtual byte address
- 16 KB pages
- 36-bit physical byte address

What is the total size of the page table for each process on this processor, assuming that the valid, protection, dirty, and use bits take a total of 4 bits and that all the virtual pages are in use? (Assume that disk addresses are not stored in the page table.)

7.40 [15] <§7.4> Assume that the virtual memory system of Exercise 7.39 is implemented with a two-way set-associative TLB with a total of 256 TLB entries. Show the virtual-to-physical mapping with a figure like Figure 7.24 on page 525. Make sure to label the width of all fields and signals.

7.41 [10] <§7.4> A processor has a 16-entry TLB and uses 4 KB pages. What are the performance consequences of this memory system if a program accesses at least 2 MB of memory at a time? Can anything be done to improve performance?

7.42 [10] <§7.4> Buffer overflows are a common exploit used to gain control of a system. If a buffer is allocated on the stack, a hacker could overflow the buffer and insert a sequence of malicious instructions compromising the system. Can you think of a hardware mechanism that could be used to prevent this?

7.43 [15] <§7.4>  For More Practice: Hierarchical Page Tables

7.44 [15] <§7.4>  For More Practice: Hierarchical Page Tables

7.45 [5] <§7.5> If all misses are classified into one of three categories—compulsory, capacity, or conflict (as discussed on page 543)—which misses are likely to be reduced when a program is rewritten so as to require less memory? How about if the clock rate of the processor that the program is running on is increased? How about if the associativity of the existing cache is increased?

7.46 [5] <§7.5> The following C program could be used to help construct a cache simulator. Many of the data types have not been defined, but the code accurately describes the actions that take place during a read access to a direct-mapped cache.

```
word ReadDirectMappedCache(address a)
    static Entry cache[CACHE_SIZE_IN_WORDS];
    Entry e = cache[a.index];
    if (e.valid == FALSE || e.tag != a.tag) {
        e.valid = true;
        e.tag = a.tag;
        e.data = load_from_memory(a);
    }
    return e.data;
```

Your task is to modify this code to produce an accurate description of the actions that take place during a read access to a direct-mapped cache with multiple-word blocks.

7.47 [8] <§7.5> This exercise is similar to Exercise 7.46, except this time write the code for read accesses to an n -way set-associative cache with one-word blocks. Note that your code will likely suggest that the comparisons are sequential in nature when in fact they would be performed in parallel by actual hardware.

7.48 [8] <§7.5> Extend your solution to Exercise 7.46 by including the specification of a new procedure for handling write accesses, assuming a write-through policy. Be sure to consider whether or not your solution for handling read accesses needs to be modified.

7.49 [8] <§7.5> Extend your solution to Exercise 7.46 by including the specification of a new procedure for handling write accesses, assuming a write-back policy. Be sure to consider whether or not your solution for handling read accesses needs to be modified.

7.50 [8] <§7.5> This exercise is similar to Exercise 7.48, but this time extend your solution to Exercise 7.47. Assume that the cache uses random replacement.

7.51 [8] <§7.5> This exercise is similar to Exercise 7.49, but this time extend your solution to Exercise 7.47. Assume that the cache uses random replacement.

7.52 [5] <§§7.7–7.8> Why might a compiler perform the following optimization?

```
/* Before */  
  
for (j = 0; j < 20; j++)  
  for (i = 0; i < 200; i++)  
    x[i][j] = x[i][j] + 1;  
  
/* After */  
  
for (i = 0; i < 200; i++)  
  for (j = 0; j < 20; j++)  
    x[i][j] = x[i][j] + 1;
```

§7.1, page 472: 1.

§7.2, page 491: 1 and 4: A lower miss penalty can lead to smaller blocks, yet higher memory bandwidth usually leads to larger blocks, since the miss penalty is only slightly larger.

§7.3, page 510: 1.

§7.4, page 538: 1-a, 2-c, 3-c, 4-d.

§7.5, page 545: 2.

Answers To Check Yourself