As we will see in Chapter 6, 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 for both designers and researchers for some years to come.

**5.18 Exercises**

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. Assume each word is a 32-bit integer.

```c
for (I=0; I<8; I++)
  for (J=0; J<8000; J++)
```

5.1.1 [§5.1] How many 32-bit integers can be stored in a 16-byte cache block?

5.1.2 [§5.1] References to which variables exhibit temporal locality?

5.1.3 [§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 storing matrix elements within the same column contiguously in memory.

```matlab
for I=1:8
  for J=1:8000
    A(I,J)=B(I,0)+A(J,I);
  end
end
```
5.1.4 [10] <§5.1> How many 16-byte cache blocks are needed to store all 32-bit matrix elements being referenced?

5.1.5 [5] <§5.1> References to which variables exhibit temporal locality?

5.1.6 [5] <§5.1> References to which variables exhibit spatial locality?

5.2 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.

3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253

5.2.1 [10] <§5.3> 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.

5.2.2 [10] <§5.3> 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.

5.2.3 [20] <§§5.3, 5.4> 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. Below are listed parameters for different direct-mapped cache designs.

Cache Data Size: 32 KiB

Cache Block Size: 2 words

Cache Access Time: 1 cycle

5.2.4 [15] <§5.3> Calculate the total number of bits required for the cache listed above, 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.

5.2.5 [20] <§§5.3, 5.4> Generate a series of read requests that have a lower miss rate on a 2 KiB 2-way set associative cache than the cache listed above. Identify one possible solution that would make the cache listed have an equal or lower miss rate than the 2 KiB cache. Discuss the advantages and disadvantages of such a solution.

5.2.6 [15] <§5.3> The formula shown in Section 5.3 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
For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Index</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-10</td>
<td>9-5</td>
<td>4-0</td>
</tr>
</tbody>
</table>

**5.3.1** [5] §5.3 What is the cache block size (in words)?

**5.3.2** [5] §5.3 How many entries does the cache have?

**5.3.3** [5] §5.3 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.

<table>
<thead>
<tr>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

**5.3.4** [10] §5.3 How many blocks are replaced?

**5.3.5** [10] §5.3 What is the hit ratio?

**5.3.6** [20] §5.3 List the final state of the cache, with each valid entry represented as a record of <index, tag, data>.

**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:

<table>
<thead>
<tr>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write through, non-write allocate</td>
<td>Write back, write allocate</td>
</tr>
</tbody>
</table>

**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.
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.

<table>
<thead>
<tr>
<th>L1 Size</th>
<th>L1 Miss Rate</th>
<th>L1 Hit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 KiB</td>
<td>8.0%</td>
<td>0.66 ns</td>
</tr>
<tr>
<td>4 KiB</td>
<td>6.0%</td>
<td>0.90 ns</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>L2 Size</th>
<th>L2 Miss Rate</th>
<th>L2 Hit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MiB</td>
<td>95%</td>
<td>5.62 ns</td>
</tr>
</tbody>
</table>

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?

5.7 This exercise examines the impact of different cache designs, specifically comparing associative caches to the direct-mapped caches from Section 5.4. For these exercises, refer to the address stream shown in Exercise 5.2.

5.7.1 [10] <§5.4> Using the sequence of references from Exercise 5.2, 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.

5.7.2 [10] <§5.4> Using the references from Exercise 5.2, 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.
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 $Disk/\text{accesses per sec}$, while the cost to keep a page in DRAM is $\text{DRAM MiB/page size}$. The typical DRAM and disk costs and typical database page sizes at several time points are listed below:

<table>
<thead>
<tr>
<th>Year</th>
<th>DRAM Cost ($/\text{MiB}$)</th>
<th>Page Size (KiB)</th>
<th>Disk Cost ($/\text{disk}$)</th>
<th>Disk Access Rate (access/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>5000</td>
<td>1</td>
<td>15,000</td>
<td>15</td>
</tr>
<tr>
<td>1997</td>
<td>15</td>
<td>8</td>
<td>2000</td>
<td>64</td>
</tr>
<tr>
<td>2007</td>
<td>0.05</td>
<td>64</td>
<td>80</td>
<td>83</td>
</tr>
</tbody>
</table>

5.10.4 [10] <§§5.1, 5.7> What are the reuse time thresholds for these three technology generations?

5.10.5 [10] <§5.7> What are the reuse time thresholds if we keep using the same 4K page size? What’s the trend here?

5.10.6 [20] <§5.7> 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.

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

<table>
<thead>
<tr>
<th>Valid</th>
<th>Tag</th>
<th>Physical Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Virtual Address Size</th>
<th>Page Size</th>
<th>Page Table Entry Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bits</td>
<td>8 KiB</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

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?