Due:  Friday, December 6 at 5:30 p.m.

Submission:  Hand me a paper copy or slide it under my office door (SCI 3809). Please also submit your .circ file for Problem 1 via email to [davisjan].

Problems

1. Implementing a direct-mapped cache.
   Complete your direct-mapped cache from Lab 9.
   (a) Print out the circuit diagram (please be sure parts are labeled!) and also email me your .circ file. Be sure to credit your lab partner(s).
   (b) What was the most challenging part of implementing this cache?
   (c) What did you learn from building a cache?

   For extra credit, build a two-way set-associative cache using the LRU replacement scheme. You should be able to abstract and reuse many components from your direct-mapped cache. Or, use a finite state-machine approach to implement a direct-mapped, write-allocate, write-back cache, as described in Patterson & Hennessy section 5.7.

2. Locality. [Adapted from Hennessy & Patterson, Computer Organization and Design, 4th ed.]
   The following code is written in C, where elements within the same row are stored contiguously. Assume that A and B are arrays of 32-bit integers.
   
   ```c
   for (i=0; i < 8000; i++)
     for (j=0; j < 8; j++)
   ```
   (a) How many 16-byte cache lines are required to hold the entire contents of both arrays?
   (b) References to which variables exhibit both spatial and temporal locality?
   (c) References to which variables exhibit only temporal locality?

3. Direct-mapped cache behavior. [Adapted from Hamacher, Vranesic, and Zaky, Computer Organization, 5th ed.]
   A word-addressable computer has a small data cache that holds 32 words, in eight 4-word blocks. Main memory holds 4K words. When a given program is executed, the processor reads data from the following sequence of hex addresses: 200, 204, 208, 20C, 2F4, 2F0, 200, 204, 218, 21C, 24C, 2F4. This pattern is repeated four times.
   Suppose the cache uses direct mapping.

   *Hint: It may help to convert the addresses to binary, but don't convert them to decimal.*

   (a) Show the format of a memory address for the purpose of cache mapping. (i.e., show the number of bits needed for the tag, block, and word fields).

   Turn over...
(b) Show the contents of the cache at the end of each pass through this loop.

A reasonable format for this might be to include a tag field and a “data” field in which you record the addresses of the words stored in a given block. For example, after reading the first word in the sequence, your cache table might look like this:

<table>
<thead>
<tr>
<th>TAG</th>
<th>CONTENTS (represented by address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>200 - 203</td>
</tr>
</tbody>
</table>

(c) Compute the hit rate for this example. Assume the cache is initially empty.

4. Set-associative cache behavior.

Repeat Problem 3 (all parts) for a four-way set-associative cache that uses the LRU replacement algorithm.


A computer has a cache, main memory, and a disk used for virtual memory. If a referenced word is in the cache, 20 ns are required to access it. If it is in main memory but not in the cache, 60 ns are needed to load it into the cache, and then the reference is started again. If the word is not in main memory, 12 ms are required to fetch the word from disk, followed by 60 ns to copy it to the cache, and then the reference is started again. The cache miss rate is 0.1 and the page fault rate is 0.4. What is the average time in ns required to access a referenced word on this system?


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, 4, 6, 8, 12, 16, 20, 24, ...

(a) Assume a 64 KB direct-mapped cache with a 16-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 experiences, based on the 3C model?

(b) Re-compute the miss rate when the cache line size is 32 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?

(c) “Prefetching” is a technique that leverages predictable address patterns to speculatively bring in additional memory blocks when a particular memory block is accessed. One example of prefetching is a stream buffer that prefetches sequentially adjacent memory blocks into a separate “prefetch buffer”. On each memory access, the prefetch buffer is checked along with the cache. If the data is found in the prefetch buffer, this is considered a hit. The prefetched block is moved into the cache, and the next memory block is prefetched and stored in the prefetch buffer. On each miss, two blocks are fetched from memory: one to satisfy the cache miss, and one to fill the prefetch buffer.

Assume that the memory latency is such that the next memory block can be loaded into the prefetch buffer before the computation on the previous block is completed. What is the steady-state miss rate for the address stream above?

(d) What cache performance characteristic(s) might be worsened by adding a prefetch buffer? Why?

(e) It might be simpler to place prefetched blocks directly in the cache, instead of using a separate prefetch buffer. Why not do that? Describe a different workload for which placing prefetched blocks directly in the cache would increase the miss rate, relative to using a separate prefetch buffer or not prefetching at all. How would you categorize the additional misses under the 3C model?