Problems

1. 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?

   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).
   (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:
      ```
      TAG CONTENTS (represented by address)
      ---- ---------------------------------
      16 200 - 203
      ```
   (c) Compute the hit rate for this example. Assume the cache is initially empty.

3. Set-associative caching.
   Repeat Problem 1 (all parts) for a four-way set-associative cache that uses the LRU replacement algorithm.

   Describe a simple technique for implementing an LRU replacement algorithm on a four-way set-associative cache. (Hint: Use two “referenced” bits rather than just one.)

Turn over for one more problem...

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?