CprE 381 Homework 10

[Note: The homework below focuses on the details of caching and cache design. Once you have completed this homework you should be able to motivate the need for caches using specific program examples. You should also be able to configure and hand-simulate a range of caches.]

1. Principle of Locality
    a. Write a valid MIPS assembly program that executes at least 20 instructions and demonstrates spatial locality in instruction fetching, but not data accesses. Explain this locality in the assembly comments.
    Answers vary, but see prob1a.s for a sample.
    b. Write a valid MIPS assembly program that executes at least 20 instructions and demonstrates temporal locality in data accesses, but not instruction fetching. Explain this locality in the assembly comments.
    Answers vary, but see prob1b.s for a sample.
    c. Spend some time looking at open-source programs on Github.com. Find a piece of a C or C++ program on github that appears to display a significant amount of data locality. Provide the html browsable file URL and line numbers of the example. Justify why these lines demonstrate data locality. *[Note that since this is real code, you may need to reference multiple files to demonstrate locality even in a single example.]*
    Answers vary widely.

2. Cache Configuration and Simulation

In this problem we will consider several cache designs for a processor implementing the MIPS ISA *[Note that this has implications needed to answer the below questions].* Assume that the block offset is four bits and the index is four bits.

a. What is the cache block size in bytes? words? double words?
Since the block offset is 4 bits and MIPS is byte-addressable, blocks are 2^4 = 16 bytes or 4 words or 2 double words.

b. How many sets does this cache have? *[Hint: note that both direct-mapped and fully-associative caches can be considered to have sets.]*
Since the set index is 4 bits, there are 2^4 = 16 sets.

c. Record both the amount of data and meta-data (in bits) this cache holds if it is direct-mapped, two-way set associative, and four-way set associative.
If there are 16 sets and 16 data bytes per set each way requires 16*16=256 bytes of data. Tags are 32-4-4=24 bits and each block also has an invalid and a dirty bit. Therefore there are 16*26=416 bits of meta-data or 52 bytes of meta-data per way. A direct-mapped cache has one way so it has 256B of data and 52B of meta-data. A two-way set-associative cache has two ways so it has 512B of data and 104B of meta-data. A four-way set-associative cache has two ways so it has 1024B of data and 208B of meta-data. Note that this does not include any meta-data needed for the replacement policy in the set-associative caches.

d. Simulate (i.e., complete the following table) the direct-mapped and four-way set associative cache with respect to the following series of memory accesses. Provide the reason for each miss. Assume the caches have no valid entries to begin with and use a least-recently-used (LRU) replacement policy.

Memory Access	Direct-Mapped	4-way Set Associative
0x1001FEA0	Miss (invalid)	Miss (4x invalid)
0x1001EFA4	Miss (tag mismatch)	Miss (tag mismatch; 3x invalid)
0x1001FEA8	Miss (tag mismatch)	Hit (way 0)
0x100100A0	Miss (tag mismatch)	Miss (2x tag mismatch; 2x invalid)
0x100100B0	Miss (invalid)	Miss (4x invalid)
0x100100C0	Miss (invalid)	Miss (4x invalid)
0x10011FA1	Miss (tag mismatch)	Miss (3x tag mismatch; 1x invalid)
0x1001EEA2	Miss (tag mismatch)	Miss (4x tag mismatch)
0x1001EFAF	Miss (tag mismatch)	Miss (4x tag mismatch)
0x100100A2	Miss (tag mismatch)	Hit (way 2)

Write the valid entries in the final state of each cache using the format <set#, way#, tag>. What was the hit rate of each cache?
Direct-mapped:

<Set A, Way 0, 0x100100>
<Set B, Way 0, 0x100100>
<Set C, Way 0, 0x100100>

4-way set-associative:
<Set A, Way 0, 0x1001EF>
<Set A, Way 1, 0x1001EE>
<Set A, Way 2, 0x100100>
<Set A, Way 3, 0x10011F>
<Set B, Way 0, 0x100100>
<Set C, Way 0, 0x100100>

The direct-mapped cache has a hit rate of 0, while the four-way set-associative cache has a hit rate of 2/10 = 0.2.