The Cache Project Part 3: L1 Cache
Due Friday, December 15

The third part of the project is to simulate a 4-way set associative, write-back L1 cache in C. Unlike the main memory and L2 cache that you have already completed, data is read from and written to the L1 cache (by the CPU) one 64-bit word at a time. Following an L1 cache miss, however, an entire 8-word cache line is inserted into the L1 cache.

A detailed description of the L1 cache is found in the l1_cache.c file that you already downloaded at the start of the project. I have provided extensive comments in l1_cache.c describing what the code should do. Your job is to fill in the missing code.

Remember, do your own work on this project. Email the course assistant if you are stuck.

Here are the steps you should take:

- **Step 1**
  You should see what the output of testing my compiled L1 code is by typing

  ```
  ./ben_test_l1
  ```

- **Step 2**
  Open the file l1_cache.c in an editor and read every line closely. You'll see that I describe the organization of the L1 cache and that there are four procedures, l1_initialize(), l1_cache_access(), l1_insert_line(), and l1_clear_r_bits() that you have to implement. You should also look closely at the related header files, memory_subsystem_constants.h and l1_cache.h.

  In l1_cache.c, the comments describe exactly what you have to do to implement the above four procedures to 1) initialize the L1 cache by clearing the valid bits in each cache entry, 2) support reads from and writes to the L1 cache, one word at a time, 3) insert a new cache line into the L1 cache, possibly evicting another cache line, and 4) clearing the reference bit in each cache entry.

  You should also take a look at my code in test_l1.c, so you can see how l1_initialize(), l1_cache_access(), l1_insert_line(), and l1_clear_r_bits() are called.

  The only file you need to modify for this part of the project is l1_cache.c.

- **Step 3**
  To test your code in l1_cache.c, you can use my test_l1.c file. To do so, compile them together by typing

  ```
  make test_l1
  ```

and, if it compiles correctly, run the program by typing

```
./test_l1
```

The output when you run the program should be the same as the output when using my compiled version (./ben_test_l1). Feel free to read and modify the code in test_l1.c to aid in your debugging.

Step 4

Upload your l1_cache.c file and get started on part 4 (the last part)!