The Cache Project Part 4:
The Memory Subsystem Controller
Due Tuesday, May 18

The fourth and last part of the memory subsystem phase of the project is to implement a simulated controller for the memory subsystem in C. This controller manages the movement of data among the CPU, the L1 cache, the L2 cache, and main memory.

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

Here are the steps you should take:

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

  ```
  ./ben_test_memory_subsystem
  ```

- **Step 2**
  Open the file memory_subsystem.c in emacs and read every line closely. You'll see that I describe the interaction among the CPU, L1 cache, L2 cache, and main memory and what the controller should do when a cache miss occurs. There are five procedures that you have to implement: memory_subsystem_initialize, memory_access(), memory_handle_l1_miss(), memory_handle_l2_miss(), and memory_handle_clock_interrupt(). You should also look closely at the related header file, memory_subsystem.h.

  In memory_subsystem.c, the comments describe exactly what you have to do to implement the above five procedures to 1) initialize the memory subsystem, 2) support memory read and write requests from a CPU, 3) handle L1 cache misses by requesting cache lines from the L2 cache, 4) handle L2 cache misses by requesting cache lines from main memory, and 5) handle a clock interrupt to cause the reference bits in the L1 cache to be cleared (for NRU purposes).

  You should also take a look at my code in test_memory_subsystem.c, so you can see how memory_subsystem_initialize(), memory_access(), and memory_handle_clock_interrupt() are called (the other two procedures, memory_handle_l1_miss(), memory_handle_l2_miss() are only called internally within memory_subsystem.c).

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

- **<u>Step 3</u>**
  To test your code in memory_subsystem.c, you can use my test_memory_subsystem.c file. To do so, compile them together by typing

  ```
  make test_memory_subsystem
  ```

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

  ```
  ./test_memory_subsystem
  ```

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

  Important: Please note that the memory subsystem code makes use of your l1 cache, l2 cache, and main memory code from the previous parts of the project, so those all have to be working correctly.

- **<u>Step 4</u>**
  Upload your memory_subsystem.c file. You're done, congratulations!