The Cache Project Part 1: Main Memory
Due Monday, April 26

The first part of the project is to simulate main memory (RAM) in C. In this simulated system, data is written to and read from the main memory as entire 8-word cache lines, not individual words (which are 32 bits).

I have written skeleton code in .c and .h files for you, along with extensive documentation in those files. Your job is to fill in the missing code where indicated. I have also included test code to help you debug your code, as well as compiled versions of the files that I wrote, so you can compare your output against mine.

You can talk to your fellow students about the project, but you may not show your code to anyone else or use anyone else's code. Be sure to read the forum for helpful information and, if you have general questions about your code, post to the forum and I will respond. Please don't post questions that have already been asked and answered, though. If you get stuck and need help with your code, please email your course assistant.

Here are the steps you should take:

- **Step 1**
  Download one of the following attached files that is appropriate for the machine you are using:
    - cache_project_macos.tgz (for macOS)
    - cache_project_cygwin.tgz (for Windows/Cygwin)
    - cache_project_linux.tgz (for Linux)

  You should create a new directory (folder) for this project and move the downloaded file into that directory. Then, in a shell, unzip the file by typing:

  ```
  tar -xzvf filename
  ```

  where filename is the name of the file you downloaded (cache_project_macos.tgz, etc.).

- **Step 2**
  You will notice a number of .c and .h files. There are also as several .o files (which are already compiled) that start with "ben". You should leave all of these files in the project directory.

  There should also be a Makefile. To create the executables based on my code, in a shell, type:

  ```
  make ben
  ```

  It will generate a number of executable files – one for each part of the project showing what the test results should be. To see what the output of the test of your main memory code should be, type

```
./ben_test_main_memory
```

If this compilation fails or you can't get the executable to run, please email your course assistant right away.

- **Step 3**
  Open the file main_memory.c your editor and read every line closely. You'll see that there are two procedures, main_memory_initialize() and main_memory_access(), that you have to implement. You should also look closely at the two related header files, memory_subsystem_constants.h and main_memory.h.

  In main_memory.c, the comments describe exactly what you have to do to implement the above two procedures to 1) allocate and initialize a large array that will simulate main memory and 2) to support reads from and writes to the simulated main memory.

  You should also take a look at my code in test_main_memory.c, so you can see how main_memory_initialize and main_memory_access are called and what the actual parameters (i.e. arguments) look like.

  Note that the type that I use throughout the code for each 32-bit word of memory is "uint32_t" (rather than "unsigned int") and the type that I used for a single byte is "uint8_t" (rather than "unsigned char"). These types, which explicitly specify the number of bits, are defined in the <stdint.h> file that is #include'd in the code and are hopefully more portable across different machines.

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

- **Step 4**
  To test your code in main_memory.c, you can use my test_main_memory.c file. To do so, compile them together by typing

  ```
  make test_main_memory
  ```

  If it compiles correctly, you can run the program by typing

  ```
  ./test_main_memory
  ```

  to see if the output is same as the output when running my compiled version (./ben_test_main_memory). Feel free to read and modify the code in test_main_memory.c to aid in your debugging.

- **Step 5**
  Once it is working, upload only your main_memory.c file to the Assignment 4 page on the course website and get started on part 2!