ECS 154A Lab 3 Cache Design Specification

File name: dmcache.cpp

To begin this assignment you first need to implement a 256-byte Direct-mapped cache with a line size of 8 bytes. The cache is byte addressable. Your cache will need to support a read operation (reading a byte from the cache) and a write operation (writing a new byte of data into the cache). This cache will support a write-back write policy, which will require you to use a dirty-bit. In addition, cache must support a write-allocate write miss policy, in which a write miss causes the appropriate line to be brought into the cache from memory, and the write's value to update the correct part of the line in the cache, which will then become dirty.

Filename: sacache.cpp

After implementing the Direct-mapped cache you will alter it (in a separate file) in order to implement a 192-byte, 6-way set associative cache with line size of 4 bytes. The other specifications will remain the same, you must support read and write operations, as well as a write-back policy, write-allocate policy, and a least-recently-used (LRU) replacement policy for blocks.

Input File Specification

Both caches take as input a file name from the command line. The file specified by the file name will be an ASCII file with each line in the following 4-byte format

Bytes	Function
1-2	**16 bit address**
3	***Read/Write**
4	**8 bits of Data**

*The read will be designated by all 0's and the write will be designated by all 1's.

Upon a read operation the data segment of the 4-byte format will be ignored. However when a write occurs the data will be written into the specified byte and the dirty bit may or may not be set.

Input files will be an ASCII file with each line in the above 4-byte format and each entry given on separate lines. For ease of creation the input file will be in hex

For example:

Address	Read/Write	Data
002D	FF	FD
002E	FF	4E
002D	00	28

Which would appear in the input file as:
002D FF FD
002E FF 4E

002D 00 28

Where the first two lines in this example would write data to the designated addresses, and the third line would tell cause your cache to be read and the data bytes would be ignored at this point.

Output File Specifications

The cache produces as output a file named dm_out.txt/ sa_out.txt. Each line of the output will contain three elements separated by a space. The first element is the data requested by the read command. The second element is whether or not the read was a HIT (1) or MISS (0). The last element is the dirty-bit for that line. More specifically, it is formatted as:

"data" "HIT/MISS" "dirty bit"
Below is an example:
FD 1 0
FD is the data read from cache. 1 means it is a HIT. 0 indicates the dirty-bit.

To get you started I've provided a test input file and the resulting outputs. Please note, in the case of a read miss I will not look at the data output from your cache.

test file: test.txt
Direct-Mapped output file: dm-out.txt
Set-Associative output file: sa-out.txt

These are available here:

http://american.cs.ucdavis.edu/academic/ecs154a.sum14/html/test.txt
http://american.cs.ucdavis.edu/academic/ecs154a.sum14/html/dm-out.txt
http://american.cs.ucdavis.edu/academic/ecs154a.sum14/html/sa-out.txt

Other Specification:

1. You **do not have to use C/C++** to implement the cache simulation program. However, whatever language you are using, your program needs to be able to:
   1) Run under CSIF machine environment.
   2) Read the input file from the command line and output to the output file.
2. Please use diff command (http://unixhelp.ed.ac.uk/CGI/man-cgi?diff) to check your program's output with the 2 outputs above.
3. Please submit **all your source code** to me.

If you find any errors within this text, please e-mail to: yfhong@ucdavis.edu.