CSE 378: Machine Organization and Assembly Language

Assignment #6: Cache Simulation

(Assigned November 29, 2000, Due December 6, 2000 electronically)

1 Cache Simulator

One way to test the effectiveness of a caching scheme is to simulate it using artificial memory traces. For this assignment you will program a cache simulator in C/C++. The simulator will take a set of memory locations as input and produce statistics such as the number of cache hits and write backs. It is important that you read sections 7.1 to 7.3 before attempting this assignment. The cache you are simulating will use a write-back, write allocate policy and a least recently used replacement policy for set associative caches.

2 The Program

The input to your cache simulator will be a data file formatted as follows. The first line of the file will be single integer that is the set associativity of the cache. A one indicates a direct mapped cache; two indicates a two-way set associative cache, and so on. The second line is the number of blocks in the cache. If this number is equal to the set associativity, the cache is fully associative. The third line of the file is the number of bytes per block in the cache. You may assume that the associativity will never be greater than the number of blocks. Also, the bytes per block and number of blocks will always be powers of two. The remainder of the file is made up addresses and letters indicating whether a read or a write should be performed. The format is a letter "r" or "w" followed by a 32-bit byte address. A single zero will mark the end of the file. For example, a cache that has 128 blocks of 4 bytes each using a two-way set associative scheme would look like:

```
2
128
4
r <address one>
w <address two>
…
0
```

Your program is responsible for breaking up the addresses into the byte offset, block offset, index/set and tag parts. If the letter in front of the address is an "r" the data at that address is being read from memory, so it should be brought into the cache if not already there. If it is already there you should record a cache hit. If the letter is a "w" the data is being written back to memory, so you should put that data into the cache and set the dirty bit high to indicate that the data needs to be written back.

After processing all of the memory addresses you will output the following.

- Total number of memory addresses processed.
- Number of hits in the cache.
- Number of blocks written back.

3 Cache Specifics

A single line in the cache will need a number of bits to keep track of the various properties. You will need a valid bit for each line that should be initialized to false (to indicate that there is currently no valid information in that part of the cache). You will also need a least recently used field for each set to keep track of which block in the set is oldest. You will also need a dirty bit for each block in the cache to keep track of those blocks that need to be written back. Besides this you will only need a tag for each set, as there isn't any actual data to store (this is a simulation).

4 Miscellaneous

Your program should take one command line argument that is the name of a file to process. The name of the program should be **dsim**. You are free to develop the code on either Windows or UNIX, but your code must execute correctly on UNIX, particularly the tropical island servers that you have been using to code assembly. A couple of test files will be available before the end of the week for you to test your code on.