

CmpSci 201

Lab 11

In this lab you'll be implementing a simple cache simulator. You will be simulating a direct mapped cache with 8 lines of 2 words each. This means that you'll need $8 \cdot 2 \cdot 4$ bytes of memory to store the cache data in. You can allocate this wherever you want, but because the size is statically known, it may be convenient to do so in the .data segment.

1. This is a direct-mapped cache, so your first problem is to implement a hashing function that takes a 32-bit memory address as its argument and returns an index into the cache. Because the cache has 8 lines large, the hash function will simply use the low-order 6 bits of the address as the index (discard the rest of the address). Note that the three lowest bits are also discarded because they determines which byte in the line you're accessing. So your hash function should return bit5 bit4 bit3, not bit5 bit4 bit3 bit2 bit1 bit0.
2. Using the hash function above, write a function that will take an address, get the index, and load the 2 words into the appropriate cache line.
3. Now that we can get indices and load a line into the cache, we need to be able to check for elements in the cache. For this, you're going to need 8 additional words of storage. Each word is associated with a given line and contains the address of the first word in that line. These are known as the tags. Write a function that takes an address as an argument, hashes it, then compares the address against the appropriate tag. If the tag matches, then the function returns the cached value. Otherwise, it fetches the line from memory, stores it in the cache, and then returns the value. Note that this function will be returning byte values.
4. Implement a function identical to the previous one, except that it returns word values rather than bytes.