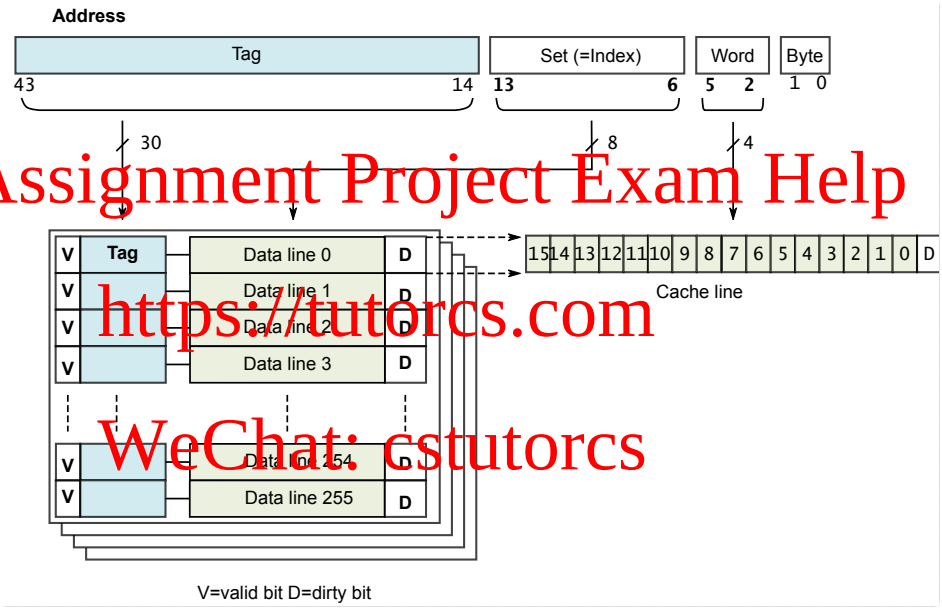
UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC 170 Introduction to Computer Architecture Winter 2021

Quiz 5 Solutions

1. The cache below was **incorrectly** described in its documentation as a 4-way set associative L1 cache with size 32 KB. (Note the size here, and in this entire problem, is the size of only *data* and *not* tag, valid, or dirty bits.) Assume that everything in the figure is accurate except for the size and/or the associativity. Note that each cell in the "Cache line" in the right side of the below diagram is a word, not a byte.



(a) (6 points) **Assuming that the associativity is correct, what is this cache's size in bytes?** Neatly showing your work will help the staff assign partial credit if appropriate.

Solution:

ARM actually does misdescribe this cache in its programmer's guide (`https://developer.arm.com/documentation/den0024/a/Caches/Cache-terminology/Cache-tags-and-Physical-Addresses`). It's actually a 32 KB 2-way set associative cache.

Note each cache line is 16 words long and each word is 4 bytes. Each way has 256 cache lines. With four ways, that's $16 \times 4 \times 256 \times 4 = 65$ KB.

(b) (6 points) **Assuming that the size is correct, what is this cache's associativity?** Neatly showing your work will help the staff assign partial credit if appropriate.

> **Solution:**
> Note each cache line is 16 words long and each word is 4 bytes. Each way has 256 cache lines. With N ways, that's $16 \times 4 \times 256 \times N = 32\,\text{KB}$, so $N = 2$ ways.

2. You have an address stream that consists of the following 16 word address references, repeated forever: 0, 0, 0, 1, 0, 2, 0, 3, 0, 4, 0, 5, 0, 6, 0, 7. ("Repeated forever" means you can ignore any compulsory misses in this problem.) Just to be clear, this means the first 32 references of this stream are 0, 0, 0, 1, 0, 2, 0, 3, 0, 4, 0, 5, 0, 6, 0, 7, 0, 0, 0, 1, 0, 2, 0, 3, 0, 4, 0, 5, 0, 6, 0, 7.

   (a) You have a cache that stores N entries of one word each. Assuming you know the address stream in advance and get to design a custom caching policy that chooses what to cache, **what is the best possible hit rate in your cache for this address stream?** You can express the hit rate as a decimal or a fraction. Explaining your answer—specifically your custom caching policy and why you chose it—will be helpful in assigning partial credit.

   (Think of a custom caching policy as writing a software program to manage the cache that decides what to store and what to evict. As examples, possible custom caching policies might be "when the cache sees an address that is not in the cache, insert it into the cache, replacing the oldest item in the cache" or "implement a standard N-element direct-mapped cache". But you can choose any policy you can describe, including conditions for specific addresses.)

   i. (4 points) Answer the above question for $N = 2$.

   > **Solution:** Cache 0 and one specific non-zero address. Never replace them. Certainly we want to cache 0. We also want to cache one other entry and keep it in cache forever; there is no better temporal locality in this problem. We can choose any other element to cache (say, 1) and thus we will hit on every reference to 0 or 1 and miss on the rest. We will hit on each of the nine 0 references and one other element in our 16-element sequence. This is a hit rate of $10/16 = 0.625$.

   ii. (4 points) Answer the above question for $N = 6$.

   > **Solution:** Cache 0 and five specific non-zero addresses. Never replace them.
   > We want to pick two (non-0) elements to never cache and then cache the rest. Then in our 16-element sequence, we will hit on all but two elements. This is a hit rate of $14/16 = 0.875$.

(b) (6 points) What is the minimum number of entries necessary for a fully-associative cache with least-recently-used (LRU) replacement to achieve a 75% hit rate on this address stream?

Solution: We need a cache that holds all 8 items to achieve at least a 75% hit rate. If we only have 7 items in the cache, the "next" (non-zero) item we need will not be in the cache because it is the least recently used. Any fully-associative cache with LRU replacement that is smaller than 8 elements will miss on 7 items out of 16. We need to store every item in a LRU cache to achieve greater than a 50% hit rate.

(c) (8 points) You are evaluating two cache designs for this address stream. Your goal is to maximize the cache hit rate.

- Emmet proposes a four-entry direct-mapped cache.
- Lucy proposes a one-entry (primary) cache with a one-entry victim cache. (Recall that an v-entry victim cache holds the last v items that have been evicted from the primary cache.)

Whose cache—Emmet's, Lucy's, or both—would give the highest hit rate on this address stream? Justify your answer.

Solution: Emmet's four-entry direct-mapped cache puts 0 and 4 in the same entry; also 1 and 5, 2 and 6, and 3 and 7. Every non-zero element will miss every time. The zero elements will hit every time except for when the 4 replaces the 0.

```
         0 0 0 1 0 2 0 3 0 4 0 5 0 6 0 7
cache:   H H H M H M H M H M M M H M H M
```

Thus the overall hit rate for the direct-mapped cache is $8/16 = 0.5$.

In Lucy's one-entry-one-victim-entry cache, the victim cache is effective in ensuring that entry 0 is always cached. Basically, every single access to entry 0 is always a hit.

```
           0 0 0 1 0 2 0 3 0 4 0 5 0 6 0 7
   cache: M H H M M M M M M M M M M M M M
  victim: H M M M H M H M H M H M H M H M
combined: H H H M H M H M H M H M H M H M
```

Thus the overall hit rate for the cache is $9/16 = 0.5625$. Lucy's cache yields the best hit rate.