Conor Curry CS1541 HW2

1. 2^{12} bytes = 4KiB
2. 2^{12} bytes / 2 bytes/word = 2^{11} words
3. 2^{11} words ⇒ 2^{11} blocks
4. 1 bit (since 2 bytes in word)
5. 11 bits (since 2^{11} blocks)
6. 12 bits (24 - (1 + 11))
7. 14 bits (12 + 1 + 1) → 7×2^{12} bits = 7×2^{9} bytes of metadata in cache
8. 7×2^{9} bytes
9. $(7 \times 2^{9}) + 2^{12}$ bytes = 7.5 KiB ← bigger than our 6KiB budget.
10. 2^{9} blocks ($2^{11}/4$)
~~10. 3 bits (since $8 = 2^3$ bytes in 4 words)~~
10. 1 bit (since 2 bytes per word)
11. 2 bits (since 2^2 bytes per word)
12. 9 bits (since 2^9 blocks in cache)
13. 12 bits (24 - (9 + 2 + 1))
14. 14 bits (same as previously)

⇒ 7×2^{10} bits = 7×2^{7} bytes of metadata in cache

15. $7 \times 2^{7} + 2^{12}$ bytes = 4.875 KiB
16. Because a cache block is 4 × 16 bit words, a 16 bit bus means that 4 values will be loaded from memory on a cache read miss.
17. 6 - 4.875 = 1.125 KiB = 1152 bytes left in our budget for write-through buffer.

Buffer entry size = 20 bits (metadata & data) + 24 bits (address) = 44 bits (5.5 bytes)

(1158 × 8)/44 ≈ 209 buffer entries.

18: The additional complexity of write-back would involve checking the dirty bit when writing to a word in the cache. If the bit was set, then the whole block (4 words) must be written to memory (over a restricted 16-bit bus). However write-back will put an updated value in the buffer on each write of one word only. Note that the write-back could be improved by widening the dirty metadata to 2-bits, and therefore selecting dirty words out of larger cache blocks.

19: You would likely use write-through, so that the device can get the most up to date data in the mmaped region. (eg. a framebuffer, you want the screen to always reflect current data.

→

20: With some small supporting features in the hardware, like the addition of a bit of metadata to each cache block specifying that the address should use either write-through or write-back, it should be possible to specify particular memory regions that use different cache writing schemes.

21: There are two separate scenarios to worry about on cache reads, which depend on the nature of the device using the mmaped region: if the device writes to the mmaped region, or if the CPU is the only source of change to the region.

If the device is writing, then caching might be detrimental. It would mean that the CPU may never invalidate a cache block and need to go to memory, meaning it is isolated from the changes occuring in the memory region. In this case, one might not want to cache that address range at all. This might be an input device, like a keyboard. For this, we'd treat all reads in the region as cache misses, and load from memory.
However, if only the CPU is writing, then a cache miss can be loaded into cache, and then treated as a normal cache block. Subsequent reads, while the block remains valid, can be treated as cache hits. This would be used in cases of output-only devices, like a screen using an mmaped frame buffer.