

1. Cache addressing (continued from last time)
 - a. What type of addresses does the cache use?
 - b. Cache addresses can be *physical*, using actual memory addresses
 - i. Slower, as the cache must wait for the MMU
 - ii. Memory management unit (MMU) converts virtual to physical addresses
 - iii. However, physically addressed caches don't have to deal with the aliasing problem below
 - c. Can also be *logical*, using the same virtual addresses that the process uses
 - i. Faster, as the cache doesn't need to wait for MMU
 1. Can reuse the same address that the process was using
 - ii. However, these caches encounter the *aliasing* problem
2. Cache design
 - a. Multiple-level caches
 - b. Unified or split
 - c. Cache size
 - i. Larger cache means slower cache
 - ii. There is no optimum cache size

3. Cache address layout

- a. Going to use slightly different terminology than the book
 - i. Same idea, but I find the book's variable names less than intuitive
- b. Terminology
 - i. C

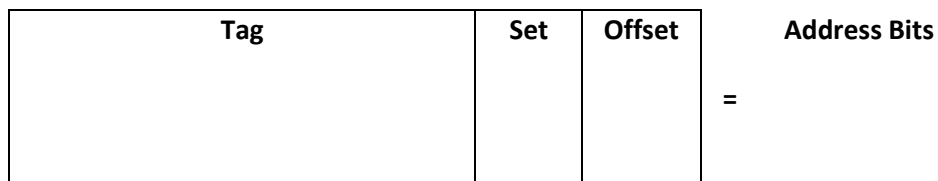
ii. LS

iii. LIC

iv. S

v. W

c. Address layout (in bits)



d. Problems of this type

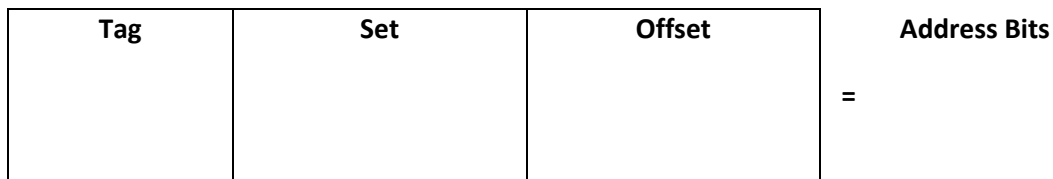
4. Direct mapped (DM) caches

- a. How does an address from main memory map to the cache?
- b. Direct mapped (DM)

- i. Easy and cheap to implement
- ii. What happens if different memory accesses map to same line of cache?

iii. Example

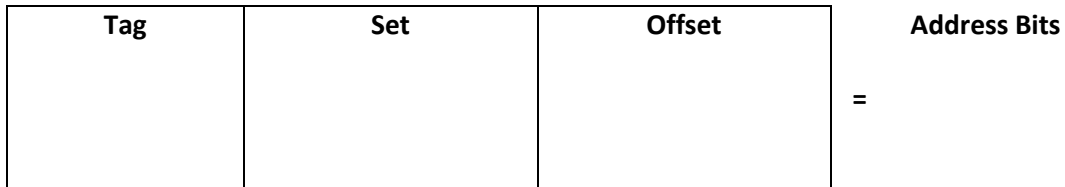
1. 8-byte DM cache with line size of 2, and 4-bit address



5. Fully associative (FA) caches

- a. Won't have conflicts like with a DM cache
- b. However, extremely expensive to implement in terms of both power and money

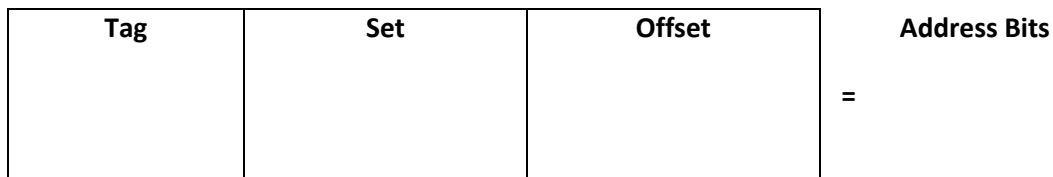
- c. Example
 - i. Same cache parameters as before, except now a FA cache
 - ii. 8-byte FA cache with line size of 2, and 4-bit address



- 6. Set associative (SA) caches
 - a. Compromise between DM and FA
 - b. N-way SA cache

- c. Advantages and disadvantages of both DM and FA

- d. Example
 - i. Same cache parameters as before, except now a 2-way SA cache
 - ii. 8-byte 2-way SA cache with line size of 2, and 4-bit address



7. Cache replacement algorithms
 - a. How do we evict lines from a given set for non-DM caches?
 - b. Least recently used (LRU)
 - c. First in, first out (FIFO)
 - d. Least frequently used (LFU)
 - e. Random
8. Cache write policies
 - a. If we change values that reside in main memory, we make changes in cache first
 - b. What happens when we need to evict a line?
 - c. Write-back
 - d. Write-through