Computer Architecture
CS-211

Spring 2017

Recitation #8

Memory

- Thus far, we have used a very simple model of memory
  - Main Memory is a linear array of bytes that can be accessed given a memory address
  - Also used registers to store values
- Reality is more complex. There is an entire memory system.
  - Different memories exist at different levels of the computer
  - Each vary in their speed, size, and cost

Memory speed gap

- Processor Speeds : 1 GHz processor speed is 1 nano sec cycle time.
- Memory Speeds (50 nano sec)
- Speed gap is huge !

Cache

- Cache this portion of the address space in faster memories
- Temporal locality
  - Recently accessed locations will likely be accessed again in near future
- Spatial locality
  - Will likely access locations close to ones recently accessed in near future

Source of locality

- Temporal locality
  - Code within a loop
  - Same instructions fetched repeatedly
- Spatial locality
  - Data arrays
  - Data allocated in chunks (contiguous bytes)

Why is locality good ?

- Address the gap between CPU speed and RAM speed
- Spatial and temporal locality implies a portion of overall address space can fit in high speed memory
- CPU can access instructions and data from this high speed memory
- Small high speed memory can make computer faster and cheaper
- This is caching

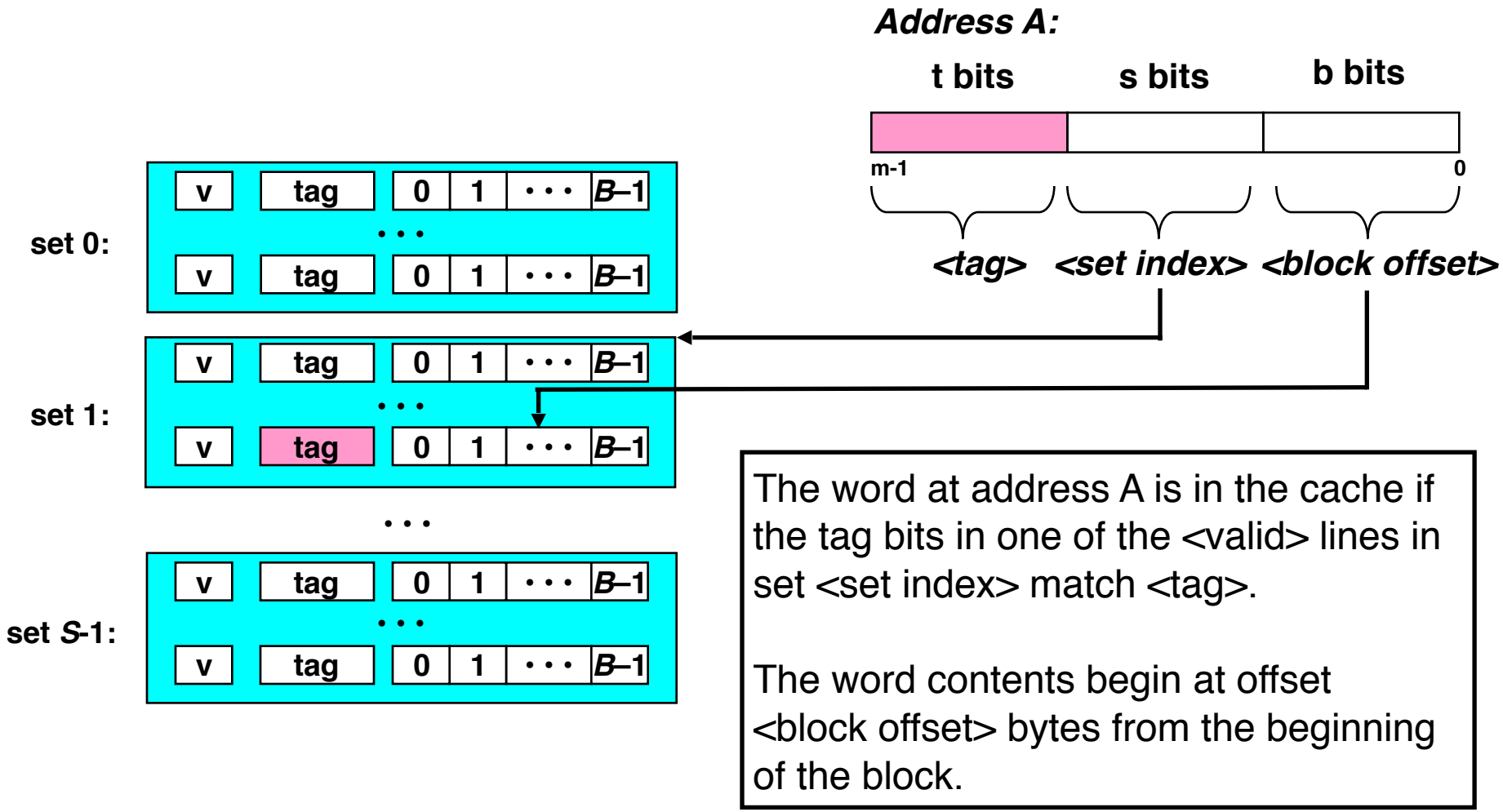
Terminology

- Cache hit Vs Cache miss
- Cache Miss :
  - Cold (compulsory) miss
    - Cold miss occurs when a memory location is accessed for the 1st time
  - Conflict miss
    - Conflict misses occur when the level k cache is large enough, but multiple data items all map to the same level k block.
      - E.g., Referencing blocks 0, 8, 0, 8, 0, 8, ... would miss every time
  - Capacity miss
    - Occurs when the set of active data blocks (working set) is larger than the cache

Cache Performance Metrics

- Miss Rate
  - Fraction of memory references not found in cache (misses/references)
  - Typical numbers:
    - 3-10% for L1
    - can be quite small (e.g., < 1%) for L2, depending on size, etc.
- Hit Time
  - Time to deliver a line in the cache to the processor (includes time to determine whether the line is in the cache)
  - Typical numbers:
    - 1 clock cycle for L1
    - 3-8 clock cycles for L2
- Miss Penalty
  - Additional time required because of a miss
    - Typically 25-100 cycles for main memory

Addressing Caches



The word at address A is in the cache if the tag bits in one of the <valid> lines in set <set index> match <tag>.

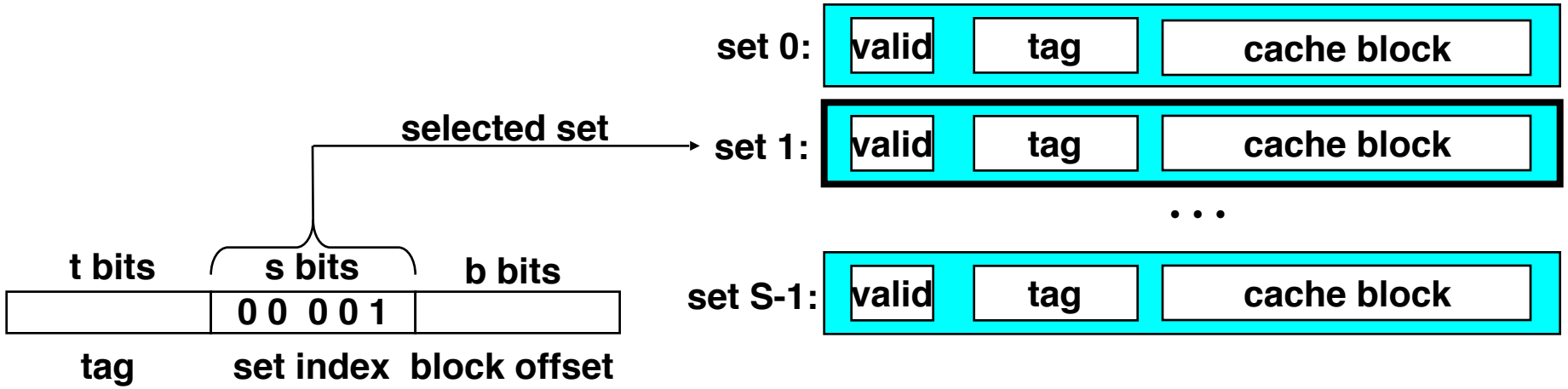
The word contents begin at offset <block offset> bytes from the beginning of the block.

Different type of cache addressing

- Direct-mapped
- Set associative
- Fully associative

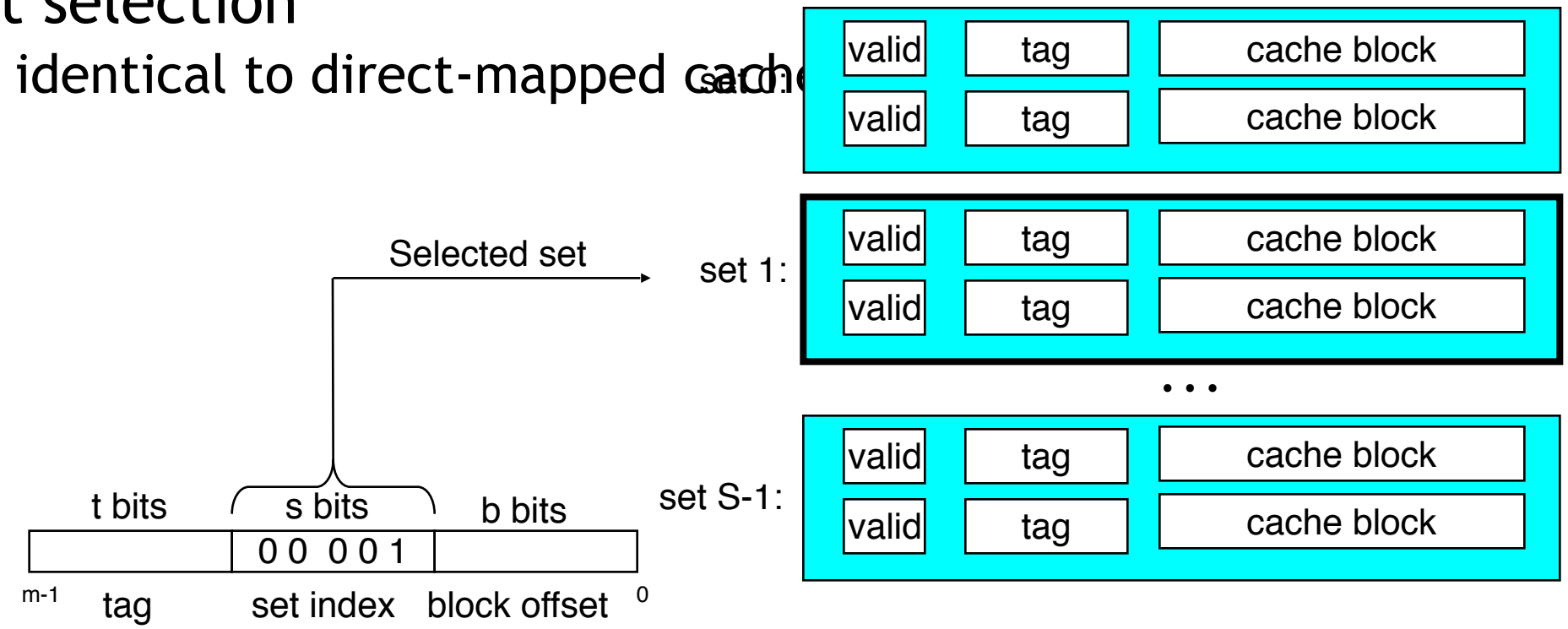
Accessing Direct-Mapped Caches

- Set selection
  - Use the set index bits to determine the set of interest.



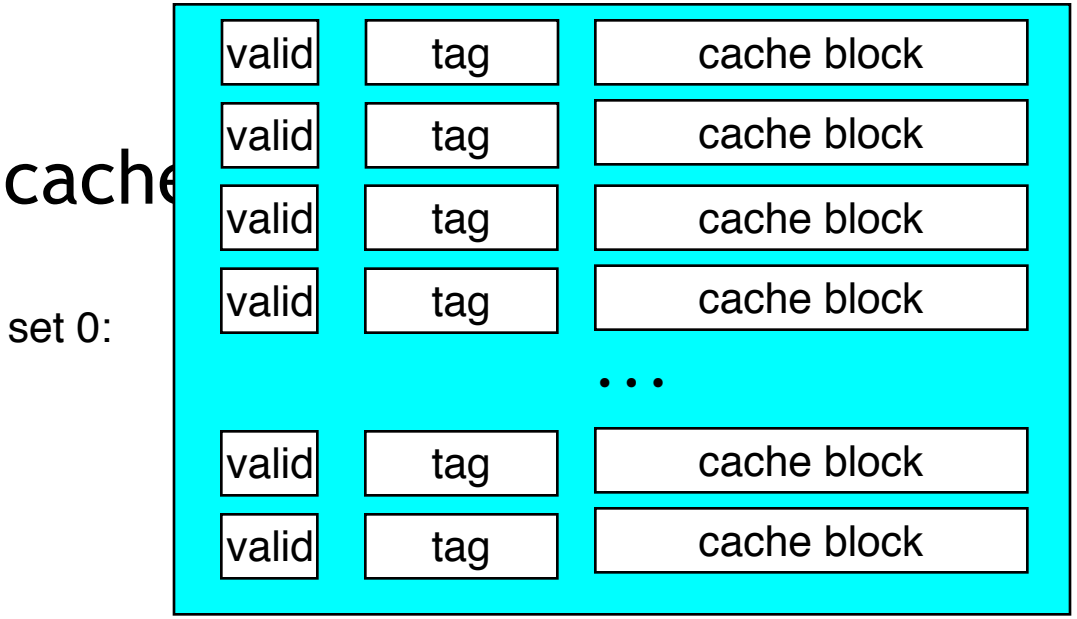
Accessing Set Associative Caches

- Set selection
  - identical to direct-mapped cache



Fully Associative Caches

- Set selection is trivial
- Accessing a line is the same as a set associative cache



Replacement

- What if has a cache miss but all entries in the set are valid?
- Direct-mapped cache: no decision
  - Discard current content and bring needed content in
- Set Associate cache: which line to evict?
  - Need a replacement algorithm
- If we knew the future, can you think of an optimal replacement algorithm?
- Since we are oracles, we need to approximate
  - FIFO: First-In-First-Out
  - LRU: Least Recently Used
  - Random: Select victim from set randomly

Writes and Cache

- Reading information from a cache is straight forward
- What about writing?
  - What if you're writing data that is already cached (write-hit)?
  - What if the data is not in the cache (write-miss)?
- Dealing with a write-hit
  - Write-through - immediately write data back to memory
  - Write-back - defer the write to memory for as long as possible
- Dealing with a write-miss
  - write-allocate - load the block into cache and update (write miss similar to read miss)
  - no-write-allocate - writes directly to memory
- Write-through caches are typically no-write-allocate
- Write-back caches are typically write-allocate

Question 0

- Define spatial and temporal locality ?
- Name 2 advantage and disadvantage for DRAM and SRAM ?

Question 1

- Which one is faster 1 or 2 ? Why ?
- for i in 0 …n
  - for j in 0 ….n

    1) A[i][j]++ ;
    2) A[j][i]++ ;

Question 2

- Calculate number of block bits, number of index bit and number of tag bits for following cache. (addresses are 64 bits)
  - 32 KB L1 cache, 2 Way associative, cache line size 64 bytes

  - Block bits (offset) = 6 bit ?
  - Index bit = 8 bit ?
  - Tag bit = 50 bit ?