Computer Organization and Architecture

Chapter 4

Cache Memory

Characteristics

- Location
- Capacity
- Unit of transfer
- Access method
- Performance
- Physical type
- Physical characteristics
- Organisation

Location

- CPU
- Internal
- External

Capacity

- Word size
  - —The natural unit of organisation
- Number of words
  - —or Bytes

Unit of Transfer

- Internal
  - Usually governed by data bus width
- External
  - Usually a block which is much larger than a word
- Addressable unit
  - Smallest location which can be uniquely addressed
  - Word internally
  - Cluster on M$ disks

Access Methods (1)

- Sequential
  - —Start at the beginning and read through in order
  - —Access time depends on location of data and previous location
  - —e.g. tape
- Direct
  - —Individual blocks have unique address
  - —Access is by jumping to vicinity plus sequential search
  - —Access time depends on location and previous location
  - —e.g. disk

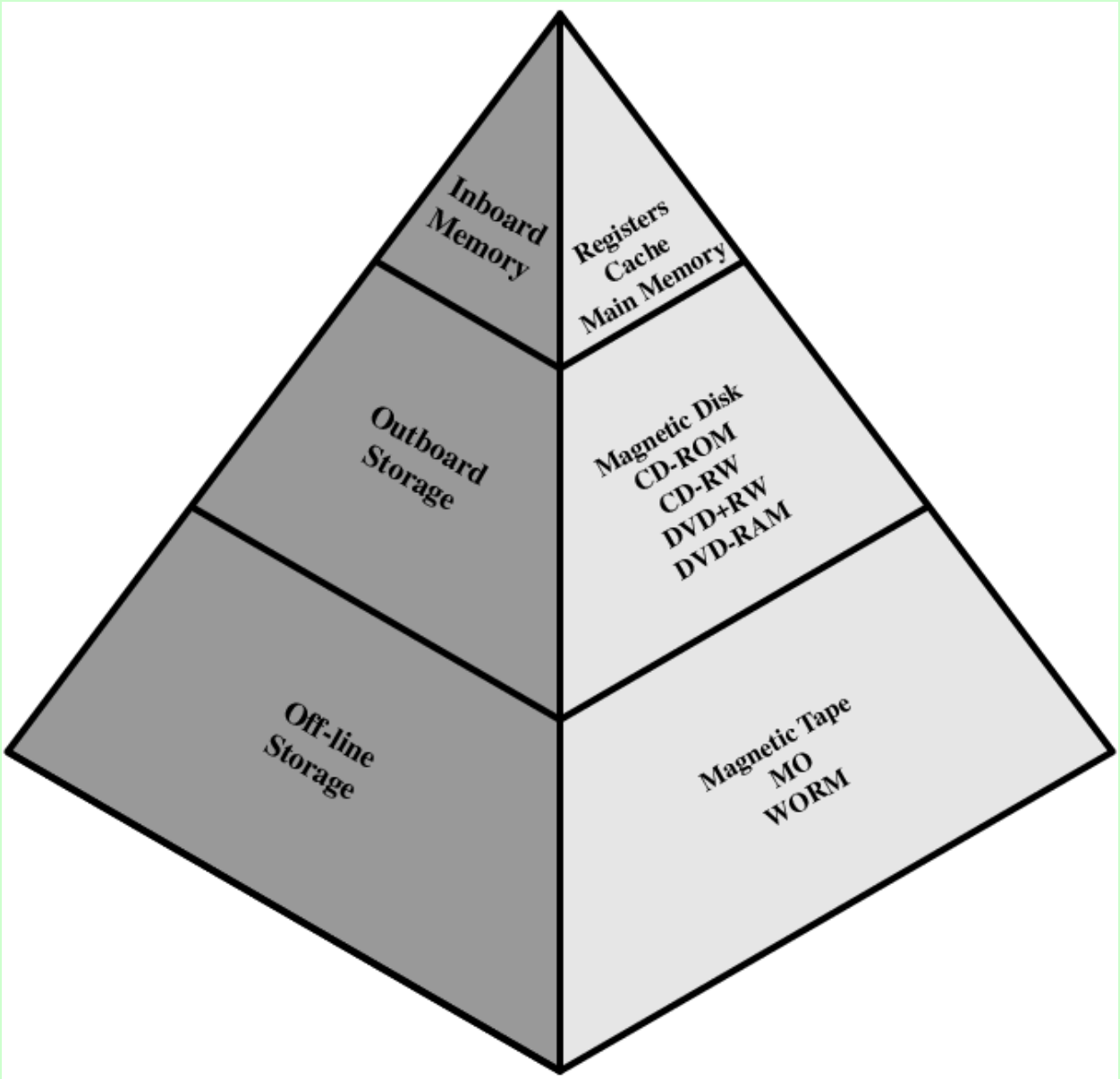
Access Methods (2)

- Random
  - Individual addresses identify locations exactly
  - Access time is independent of location or previous access
  - e.g. RAM
- Associative
  - Data is located by a comparison with contents of a portion of the store
  - Access time is independent of location or previous access
  - e.g. cache

Memory Hierarchy

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - "RAM"
- External memory
  - Backing store

Memory Hierarchy - Diagram



Performance

- Access time
  - —Time between presenting the address and getting the valid data
- Memory Cycle time
  - —Time may be required for the memory to "recover" before next access
  - —Cycle time is access + recovery
- Transfer Rate
  - —Rate at which data can be moved

Physical Types

- Semiconductor
  - —RAM
- Magnetic
  - —Disk & Tape
- Optical
  - —CD & DVD
- Others
  - —Bubble
  - —Hologram

Physical Characteristics

- Decay
- Volatility
- Erasable
- Power consumption

Organisation

- Physical arrangement of bits into words
- Not always obvious
- e.g. interleaved

The Bottom Line

- How much?
  - —Capacity
- How fast?
  - —Time is money
- How expensive?

Hierarchy List

- Registers
- L1 Cache
- L2 Cache
- Main memory
- Disk cache
- Disk
- Optical
- Tape

So you want fast?

- It is possible to build a computer which uses only static RAM (see later)
- This would be very fast
- This would need no cache
  - How can you cache cache?
- This would cost a very large amount

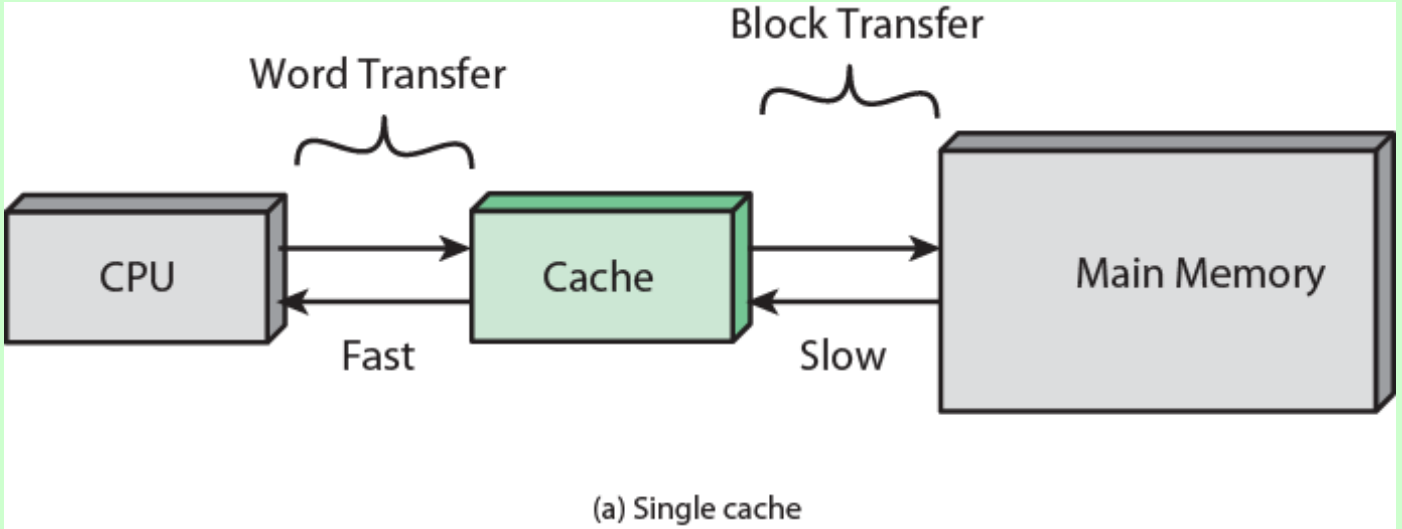
Locality of Reference

- During the course of the execution of a program, memory references tend to cluster
- e.g. loops

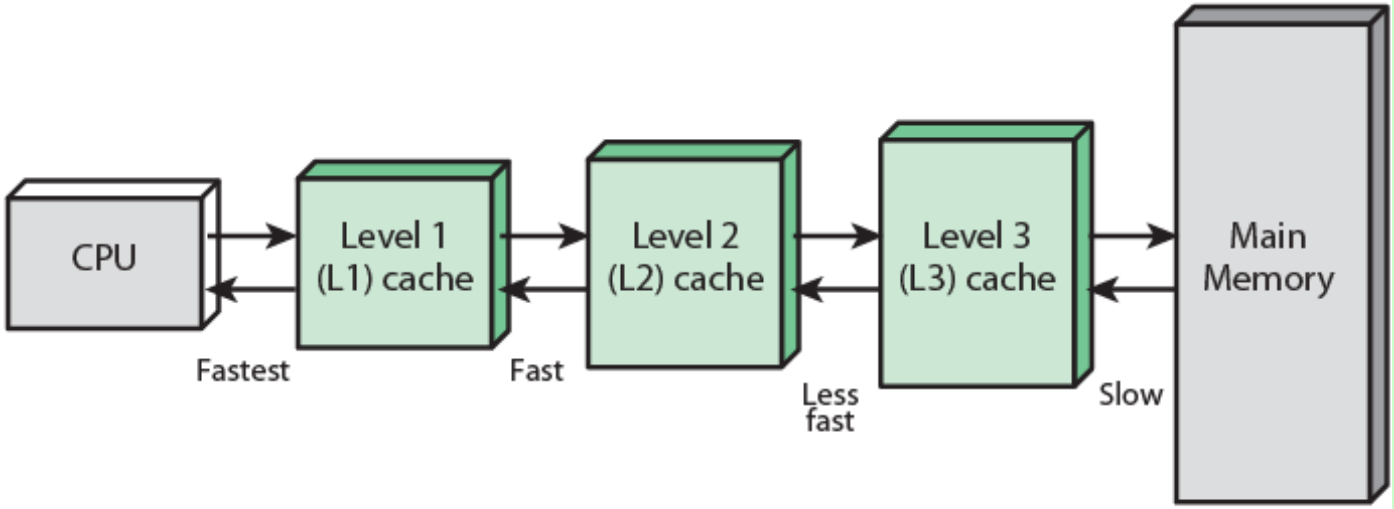
Cache

- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module

Cache and Main Memory

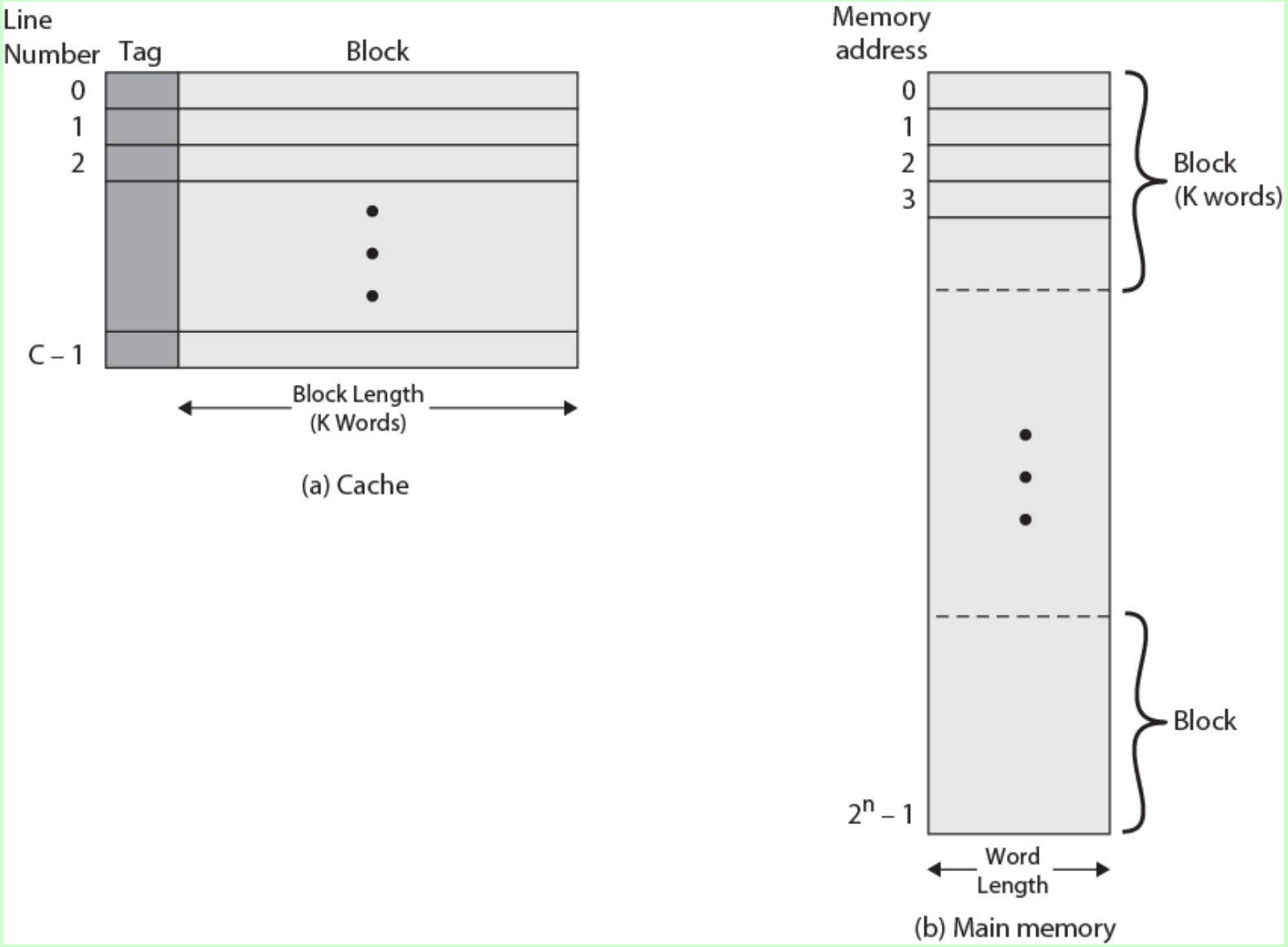


(a) Single cache



(b) Three-level cache organization

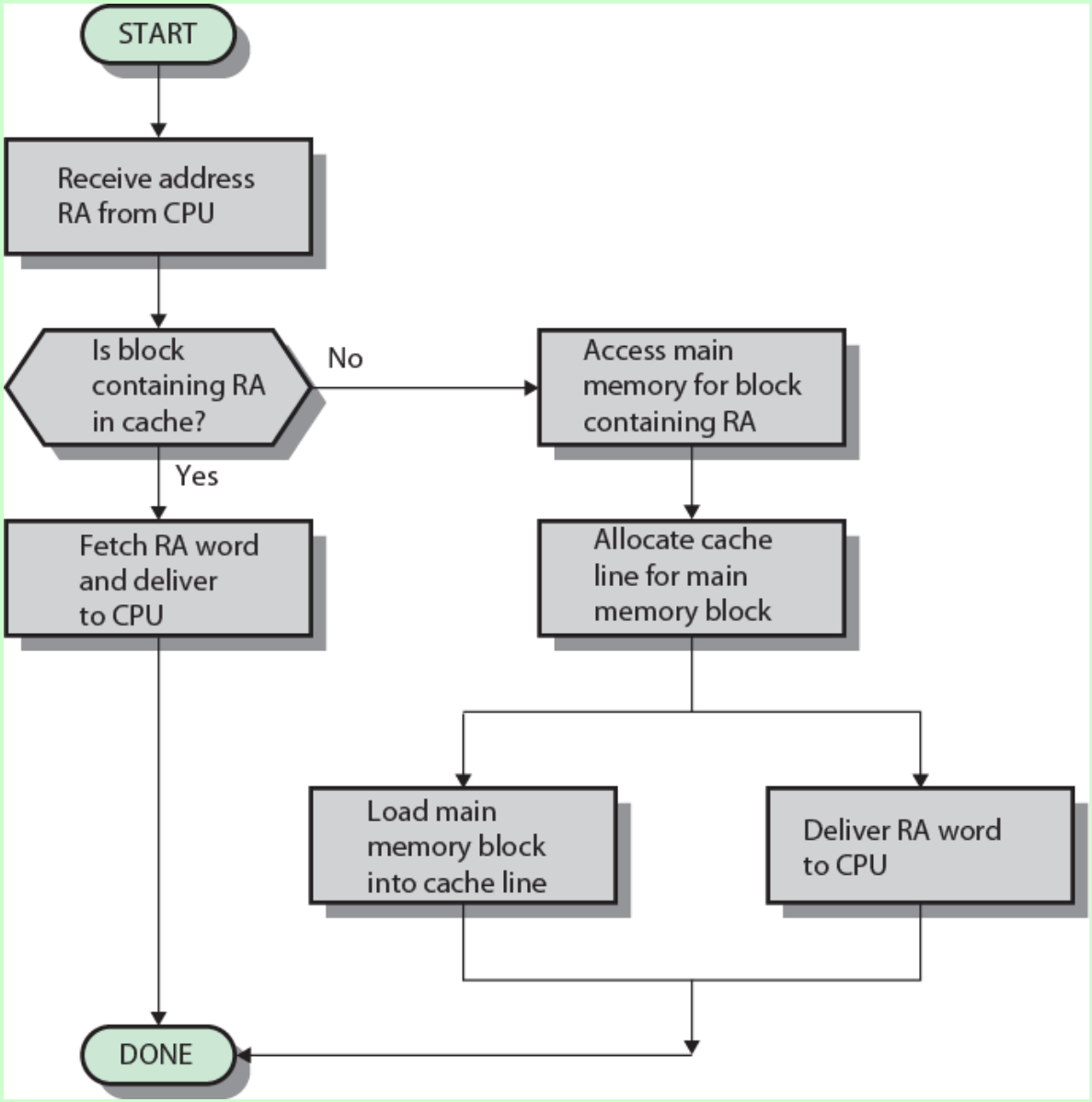
Cache/Main Memory Structure



Cache operation – overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

Cache Read Operation - Flowchart



Cache Design

- Addressing
- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

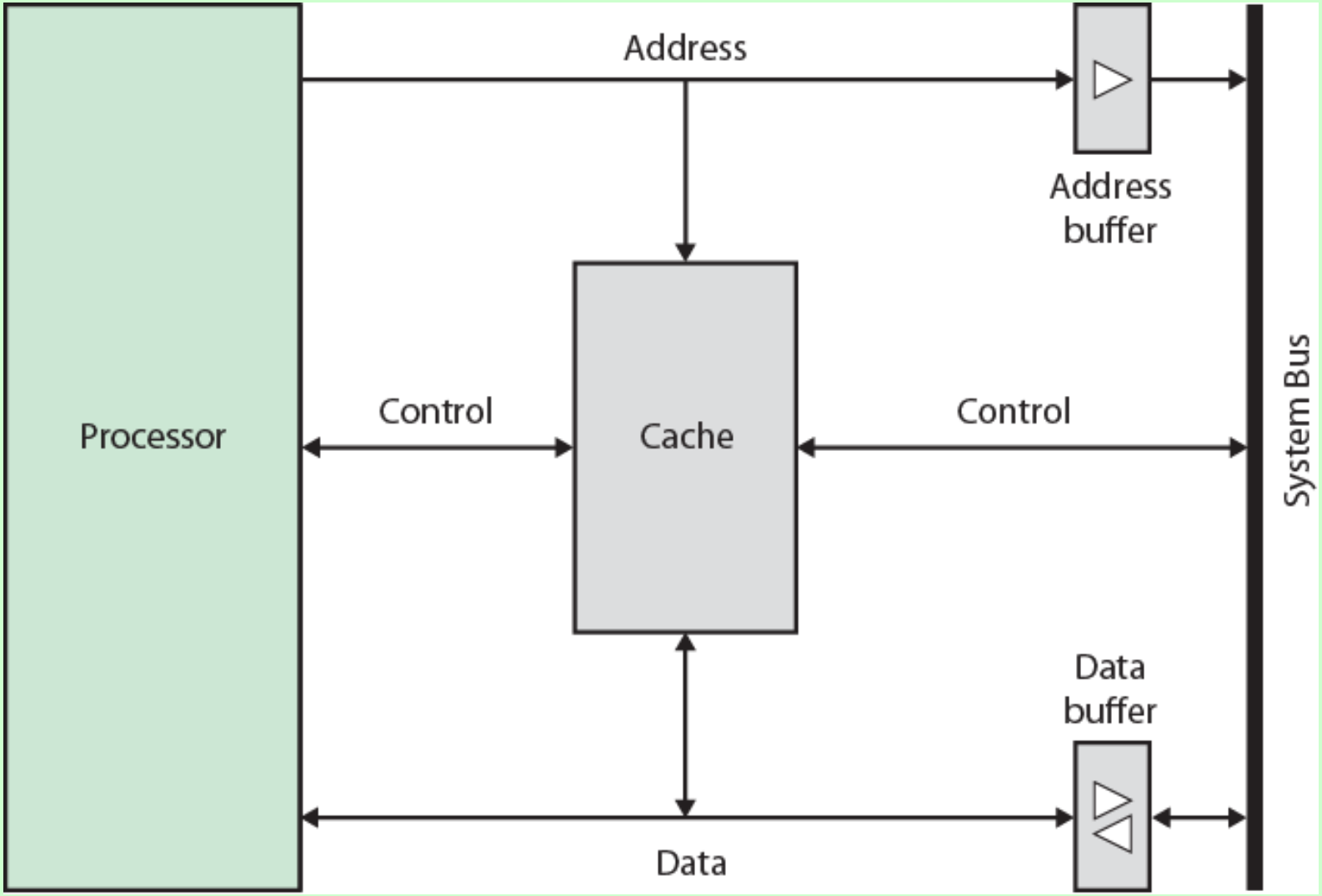
Cache Addressing

- Where does cache sit?
  - Between processor and virtual memory management unit
  - Between MMU and main memory
- Logical cache (virtual cache) stores data using virtual addresses
  - Processor accesses cache directly, not thorough physical cache
  - Cache access faster, before MMU address translation
  - Virtual addresses use same address space for different applications
    - Must flush cache on each context switch
- Physical cache stores data using main memory physical addresses

Size does matter

- Cost
  - More cache is expensive
- Speed
  - More cache is faster (up to a point)
  - Checking cache for data takes time

Typical Cache Organization



Mapping Function

- Cache of 64kByte
- Cache block of 4 bytes
  - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16MBytes main memory
- 24 bit address
  - (2^{24}=16M)

Direct Mapping

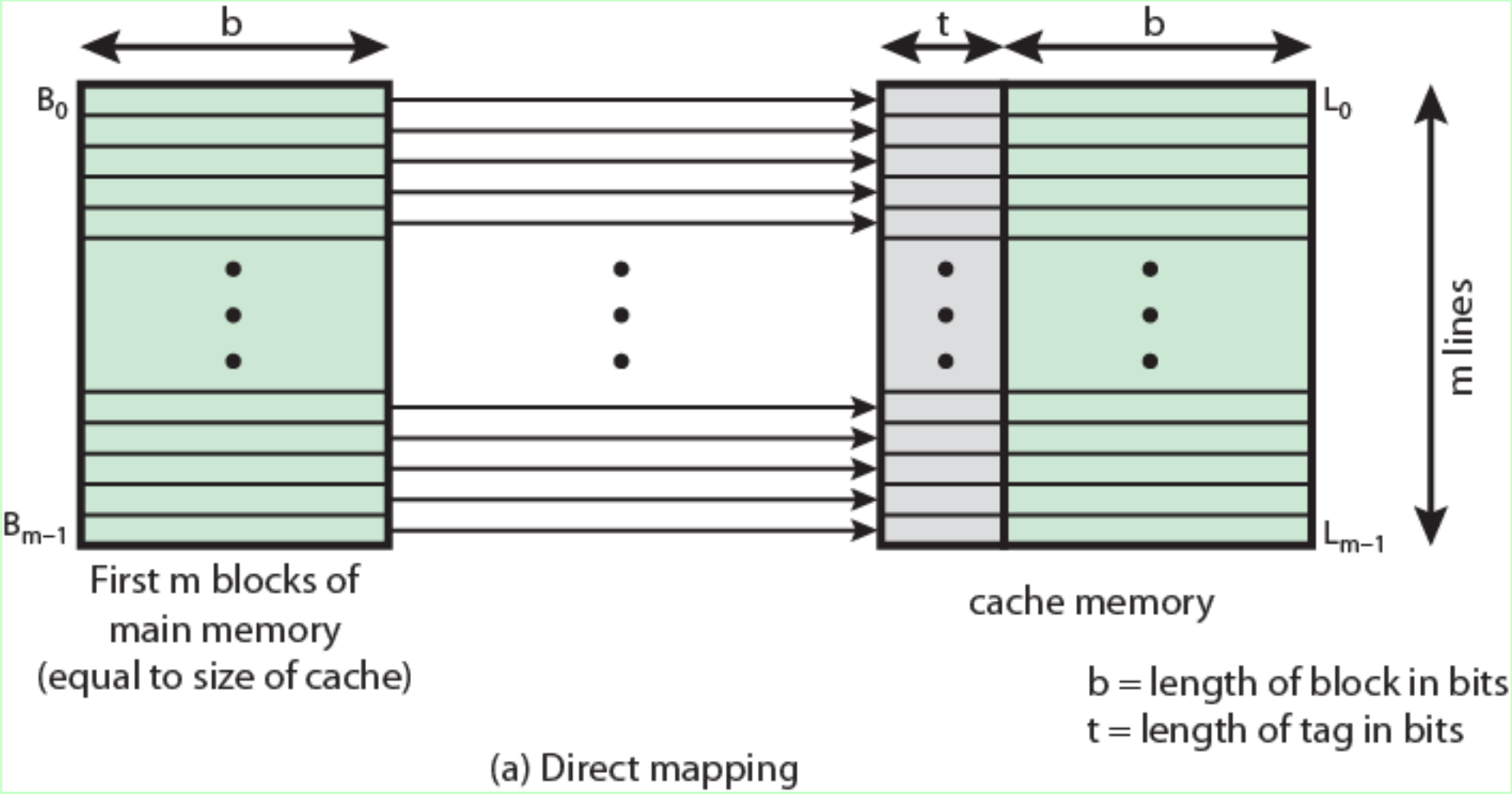
- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of s-r (most significant)

Direct Mapping Address Structure

Tag s-r	Line or Slot r	Word w
8	14	2

- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

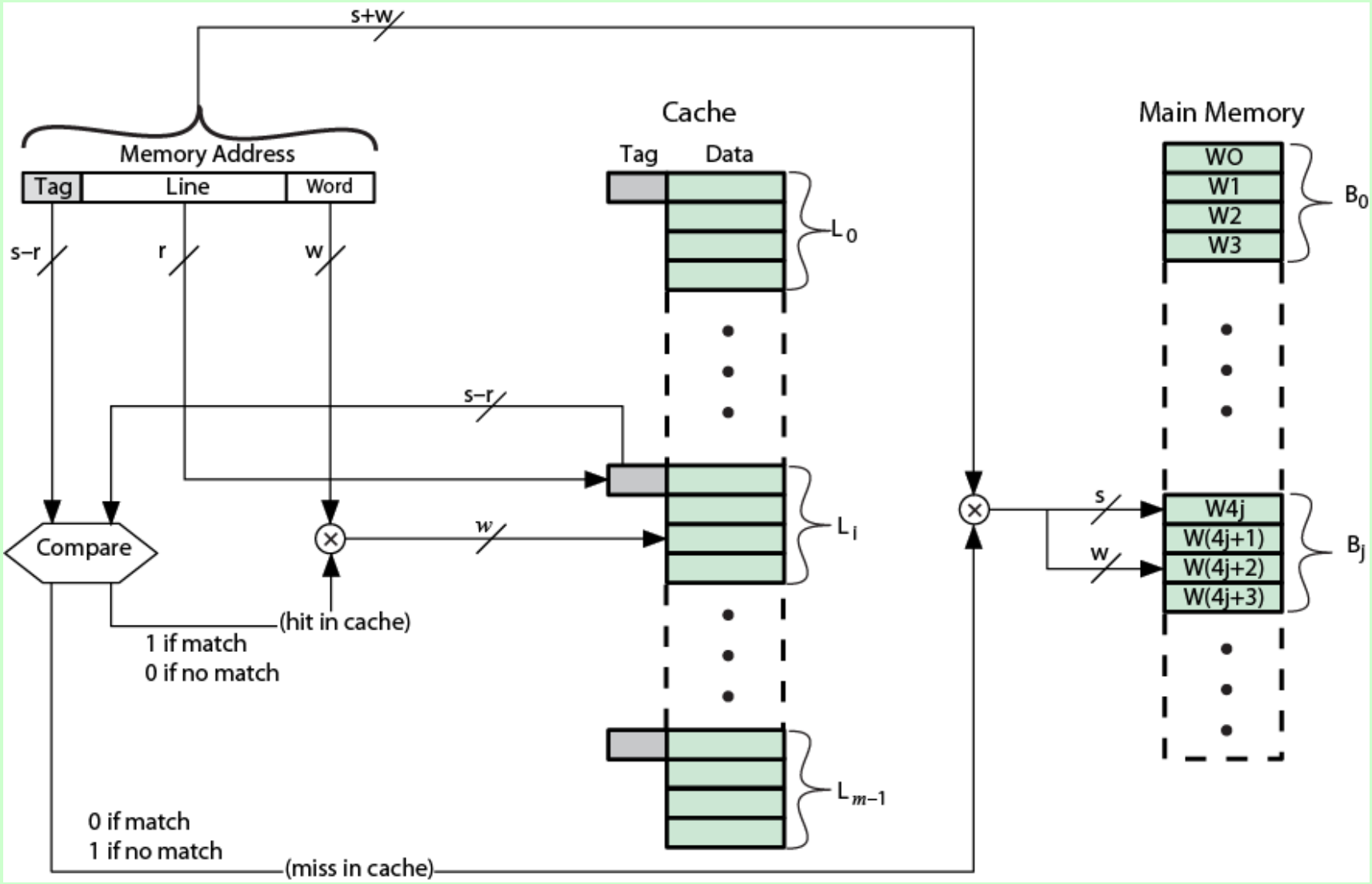
Direct Mapping from Cache to Main Memory



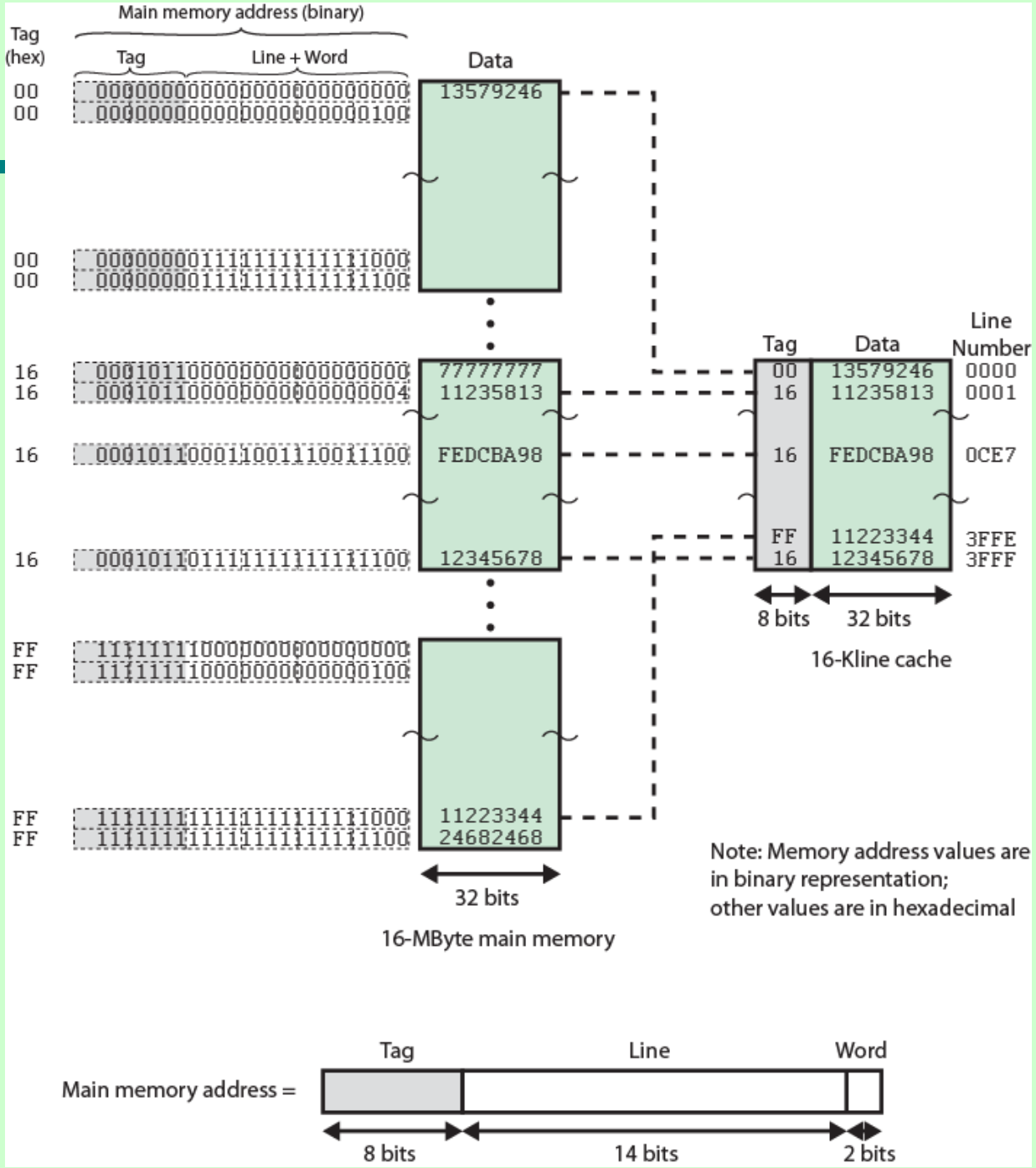
Direct Mapping
Cache Line Table

Cache line	Main Memory blocks held
0	0, m, 2m, 3m…2s-m
1	1,m+1, 2m+1…2s-m+1
…	
m-1	m-1, 2m-1,3m-1…2s-1

Direct Mapping Cache Organization



Direct Mapping Example



Direct Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2s+w words or bytes
- Block size = line size = 2w words or bytes
- Number of blocks in main memory = 2s+ w/2w = 2s
- Number of lines in cache = m = 2r
- Size of tag = (s – r) bits

Direct Mapping pros & cons

- Simple
- Inexpensive
- Fixed location for given block
  - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

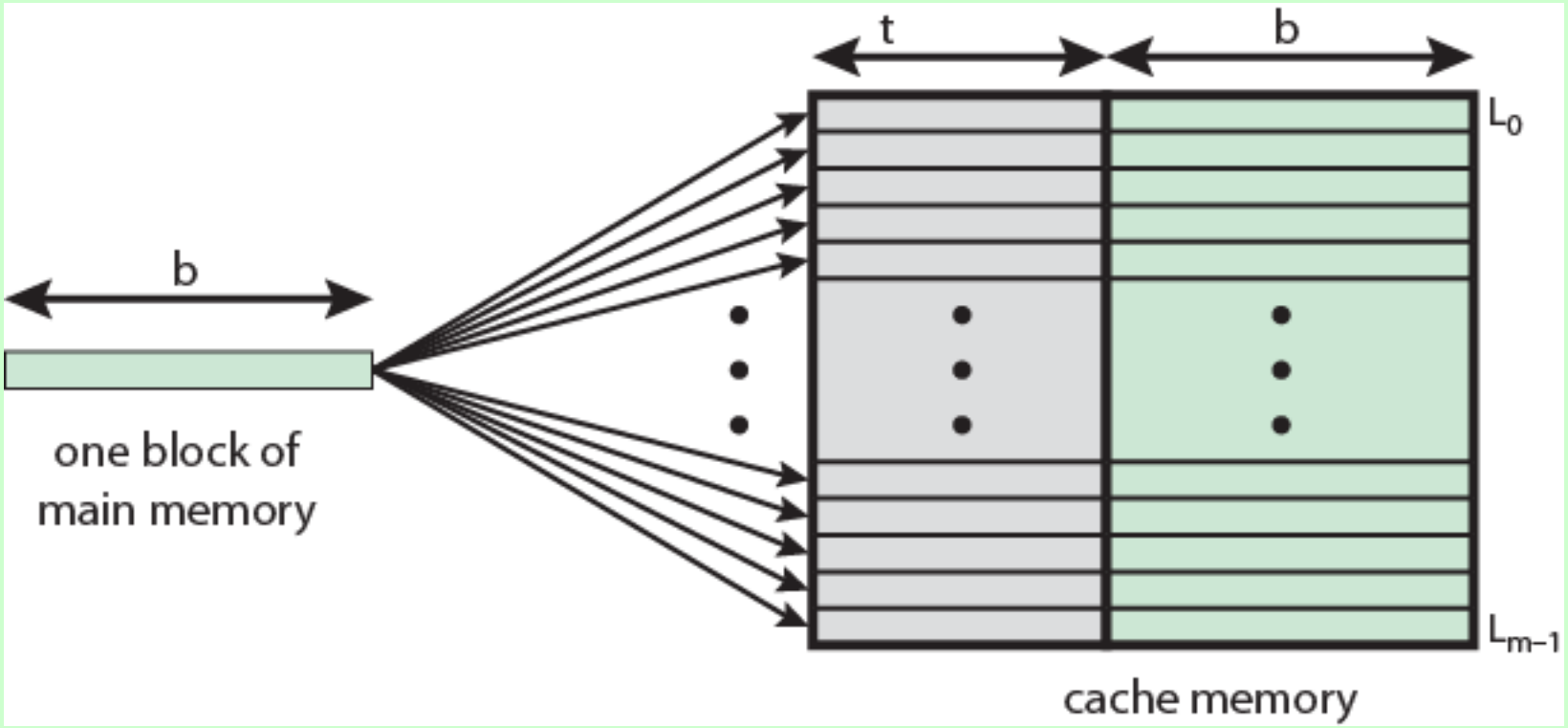
Victim Cache

- Lower miss penalty
- Remember what was discarded
  - Already fetched
  - Use again with little penalty
- Fully associative
- 4 to 16 cache lines
- Between direct mapped L1 cache and next memory level

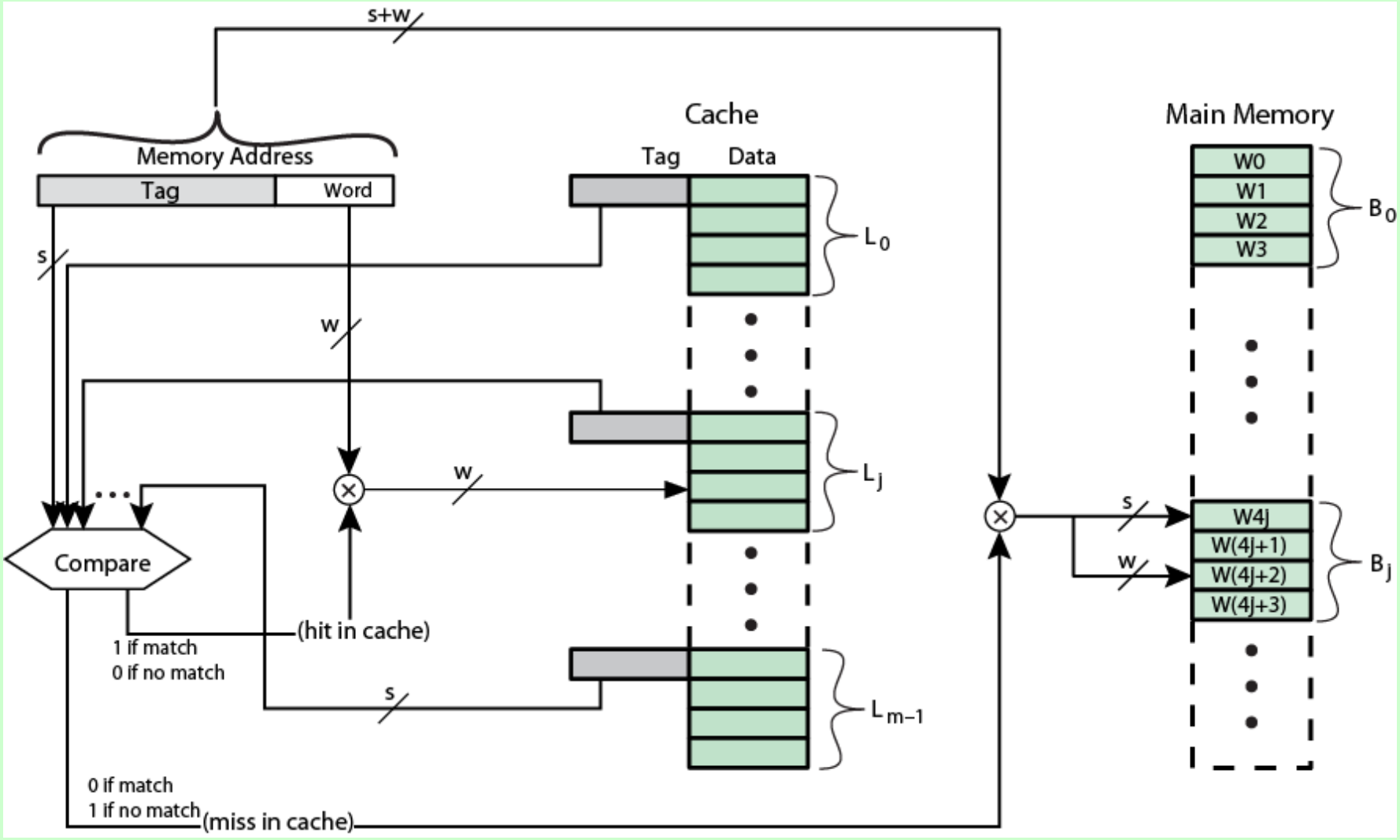
Associative Mapping

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line’s tag is examined for a match
- Cache searching gets expensive

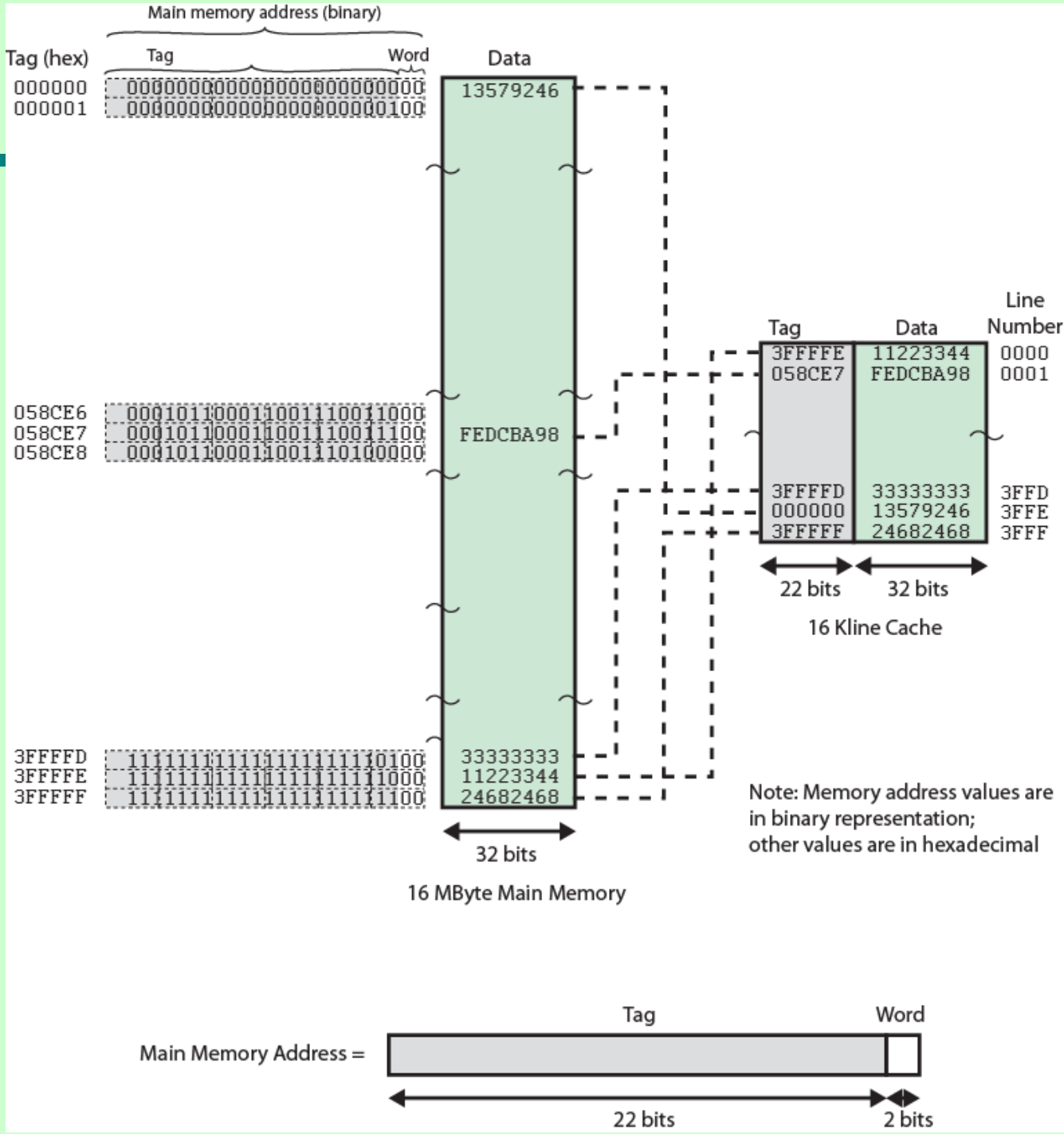
Associative Mapping from Cache to Main Memory



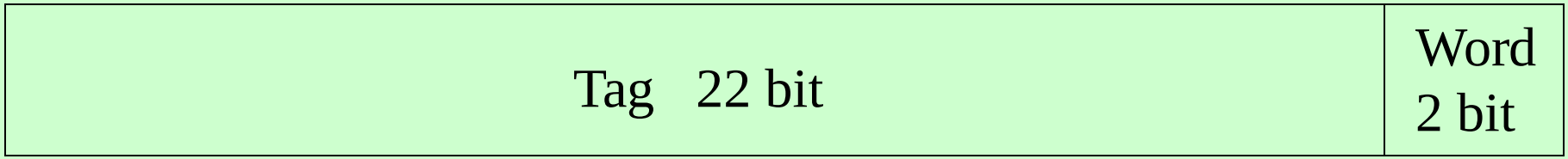
Fully Associative Cache Organization



Associative Mapping Example



Associative Mapping Address Structure



- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit
- Least significant 2 bits of address identify which 16 bit word is required from 32 bit data block
- e.g.
  - Address Tag Data Cache line
  - FFFFFC FFFFFC24682468 3FFF

Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2s+w words or bytes
- Block size = line size = 2w words or bytes
- Number of blocks in main memory = 2s+ w/2w = 2s
- Number of lines in cache = undetermined
- Size of tag = s bits

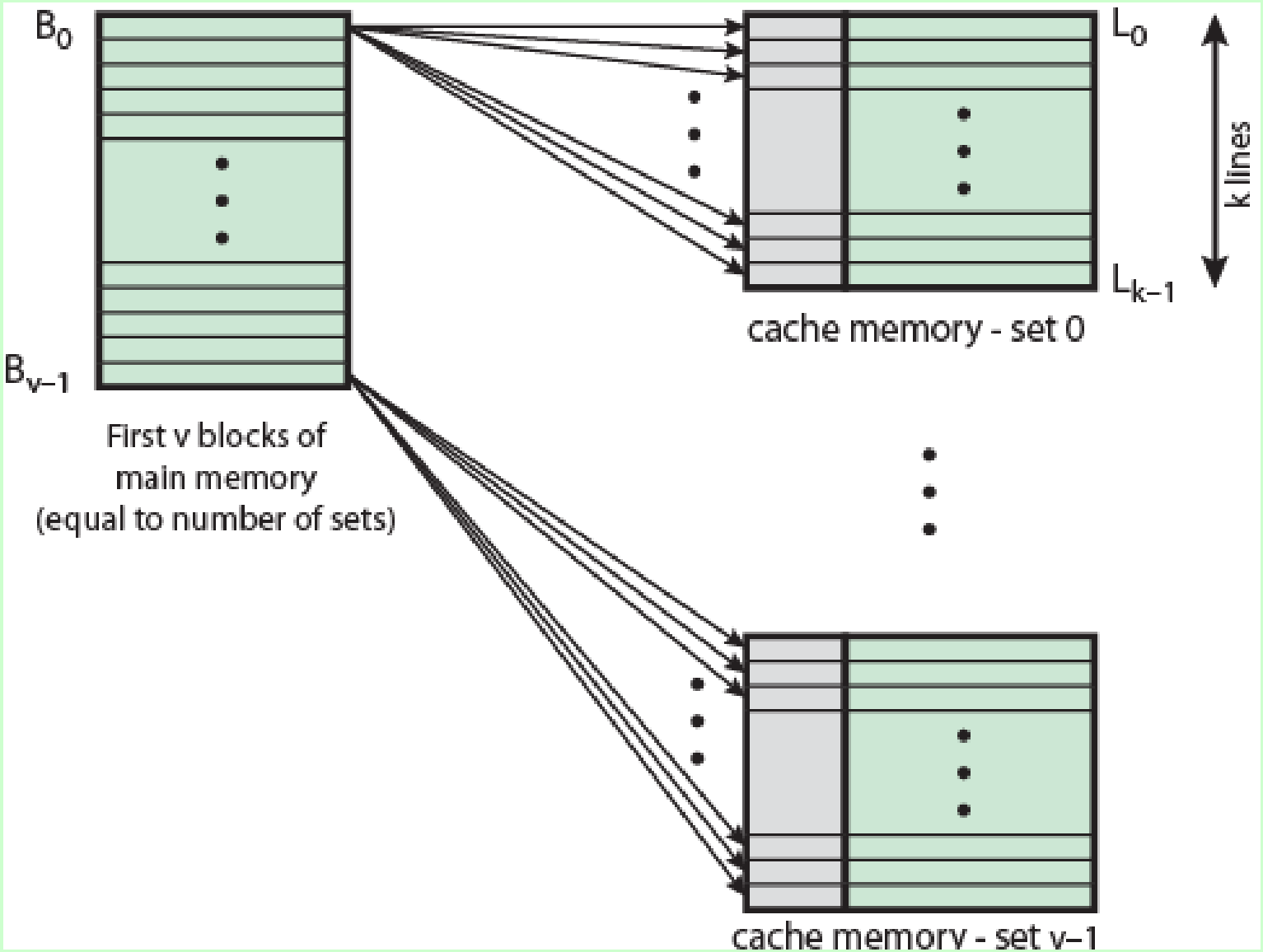
Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

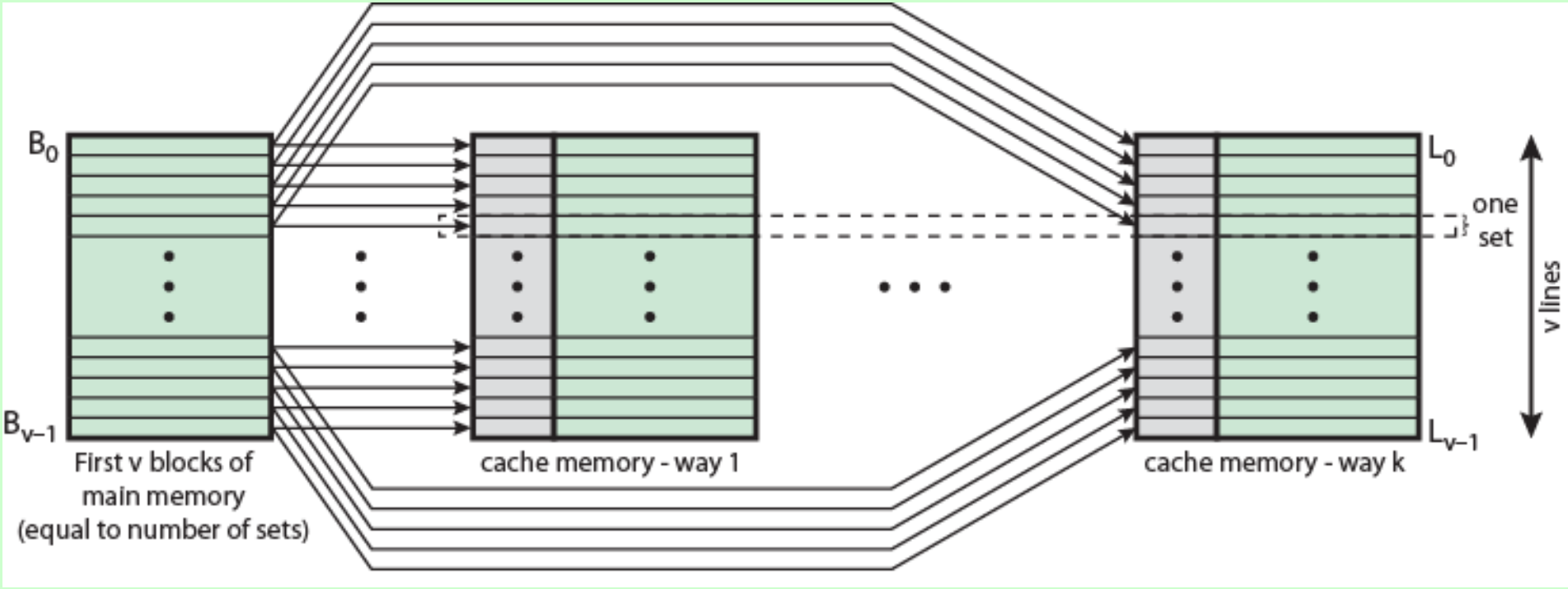
Set Associative Mapping Example

- 13 bit set number
- Block number in main memory is modulo 2^{13}
- 000000, 00A000, 00B000, 00C000 … map to same set

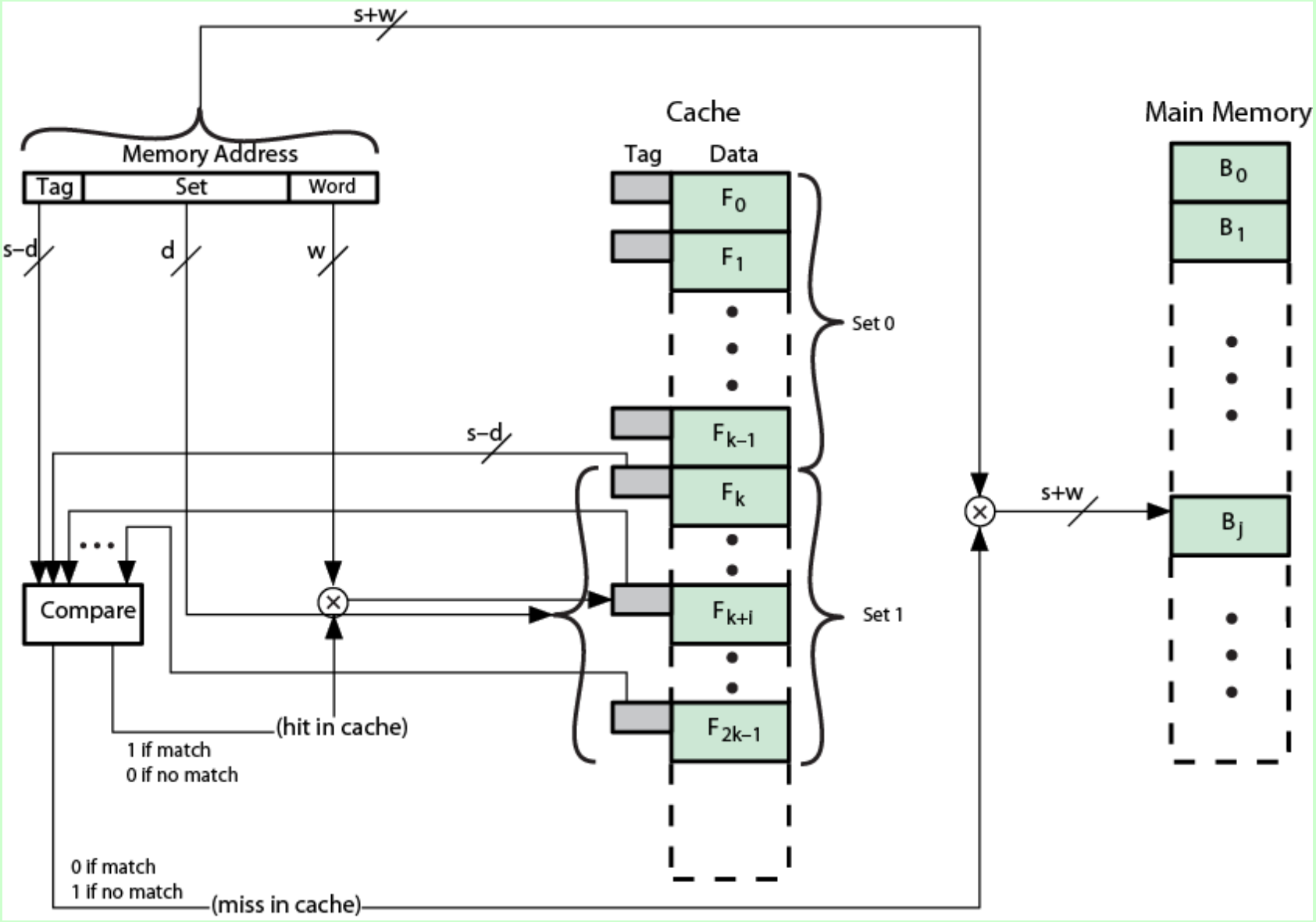
Mapping From Main Memory to Cache: v Associative



Mapping From Main Memory to Cache: k-way Associative



K-Way Set Associative Cache Organization

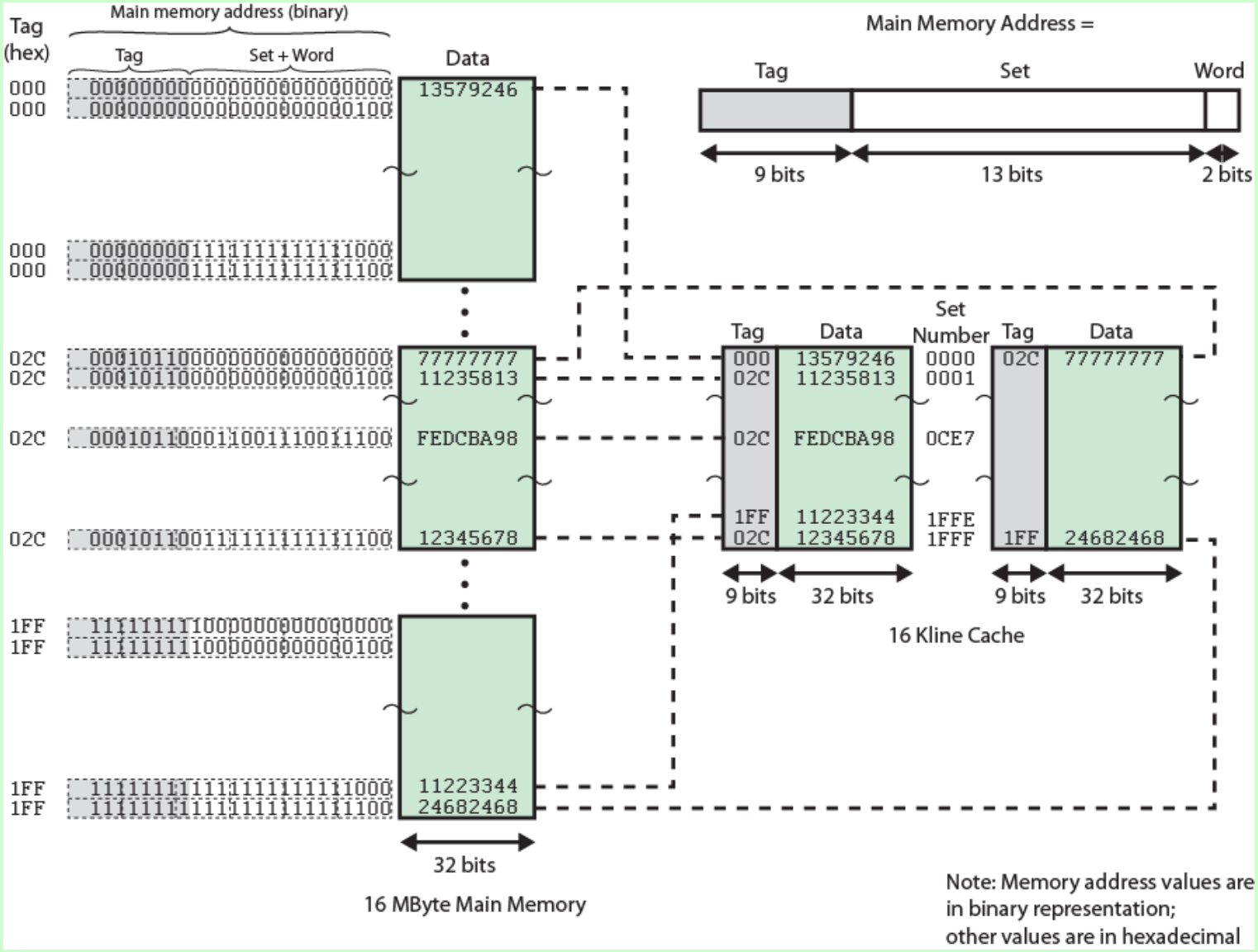


Set Associative Mapping Address Structure

Tag 9 bit	Set 13 bit	Word 2 bit

- Use set field to determine cache set to look in
- Compare tag field to see if we have a hit
- e.g
  - Address number Tag Data Set
  - 1FF 7FFC 1FF 12345678 1FFF
  - 001 7FFC 001 11223344 1FFF

Two Way Set Associative Mapping Example



Set Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2s+w words or bytes
- Block size = line size = 2w words or bytes
- Number of blocks in main memory = 2d
- Number of lines in set = k
- Number of sets = v = 2d
- Number of lines in cache = kv = k * 2d
- Size of tag = (s – d) bits

Direct and Set Associative Cache Performance Differences

- Significant up to at least 64kB for 2-way
- Difference between 2-way and 4-way at 4kB much less than 4kB to 8kB
- Cache complexity increases with associativity
- Not justified against increasing cache to 8kB or 16kB
- Above 32kB gives no improvement
- (simulation results)

Replacement Algorithms (1)
Direct mapping

- No choice
- Each block only maps to one line
- Replace that line

Replacement Algorithms (2) Associative & Set Associative

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
- e.g. in 2 way set associative
  - —Which of the 2 block is lru?
- First in first out (FIFO)
  - —replace block that has been in cache longest
- Least frequently used
  - —replace block which has had fewest hits
- Random

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write through

- All writes go to main memory as well as cache
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Lots of traffic
- Slows down writes

- Remember bogus write through caches!

Write back

- Updates initially made in cache only
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync
- I/O must access main memory through cache
- N.B. 15% of memory references are writes

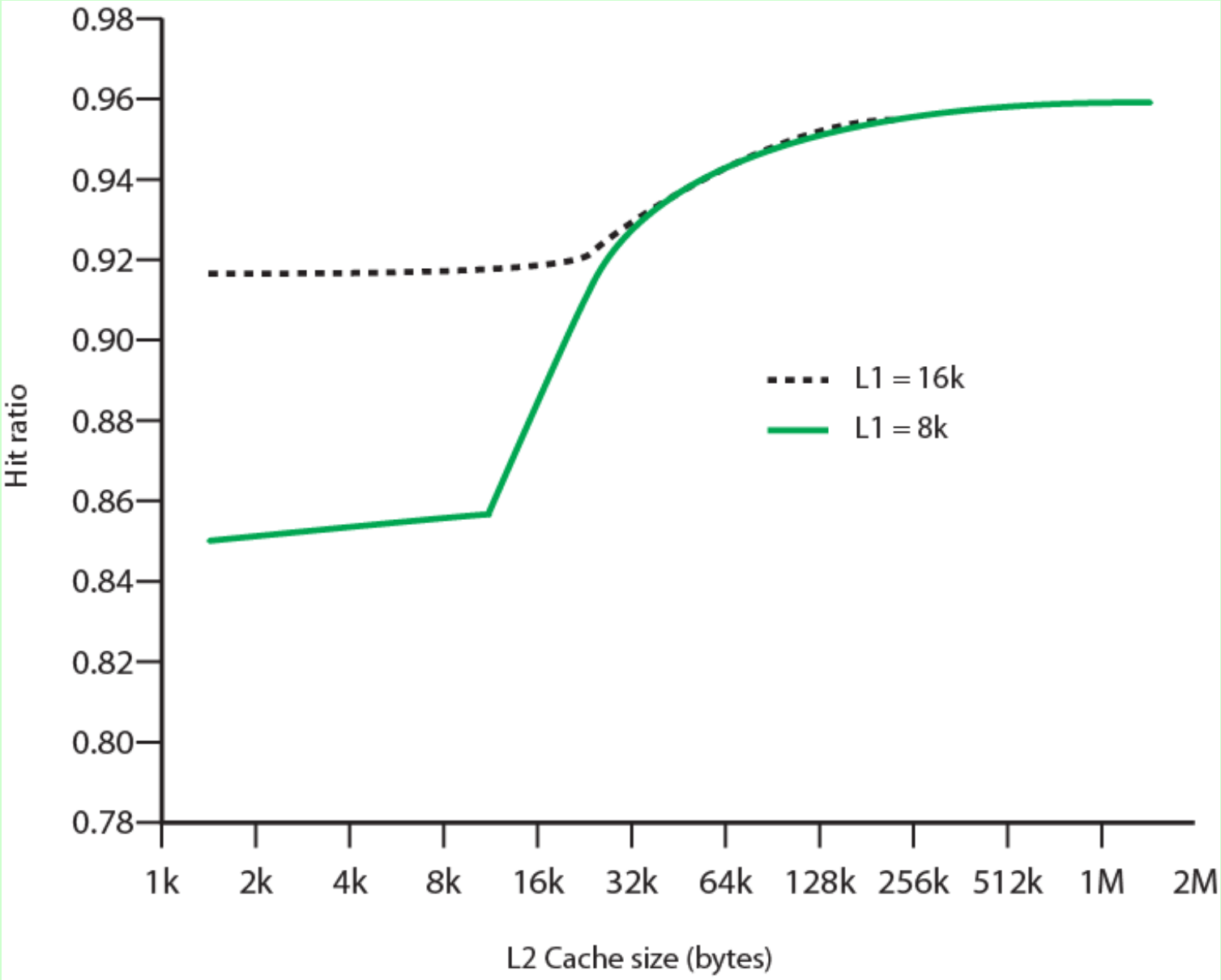
Line Size

- Retrieve not only desired word but a number of adjacent words as well
- Increased block size will increase hit ratio at first
  - the principle of locality
- Hit ratio will decreases as block becomes even bigger
  - Probability of using newly fetched information becomes less than probability of reusing replaced
- Larger blocks
  - Reduce number of blocks that fit in cache
  - Data overwritten shortly after being fetched
  - Each additional word is less local so less likely to be needed
- No definitive optimum value has been found
- 8 to 64 bytes seems reasonable
- For HPC systems, 64- and 128-byte most common

Multilevel Caches

- High logic density enables caches on chip
  - —Faster than bus access
  - —Frees bus for other transfers
- Common to use both on and off chip cache
  - —L1 on chip, L2 off chip in static RAM
  - —L2 access much faster than DRAM or ROM
  - —L2 often uses separate data path
  - —L2 may now be on chip
  - —Resulting in L3 cache
    - – Bus access or now on chip…

Hit Ratio (L1 & L2)
For 8 kbytes and 16 kbyte L1



Unified v Split Caches

- One cache for data and instructions or two, one for data and one for instructions
- Advantages of unified cache
  - Higher hit rate
    - Balances load of instruction and data fetch
    - Only one cache to design & implement
- Advantages of split cache
  - Eliminates cache contention between instruction fetch/decode unit and execution unit
    - Important in pipelining

Internet Sources

- Manufacturer sites
  - Intel
  - ARM
- Search on cache