Computer Organization and Software Systems

CONTACT SESSION 3

Mr. Vaibhav Jain

WILP – BITS Pilani

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Computer Organization and Software Systems (SS ZG516)

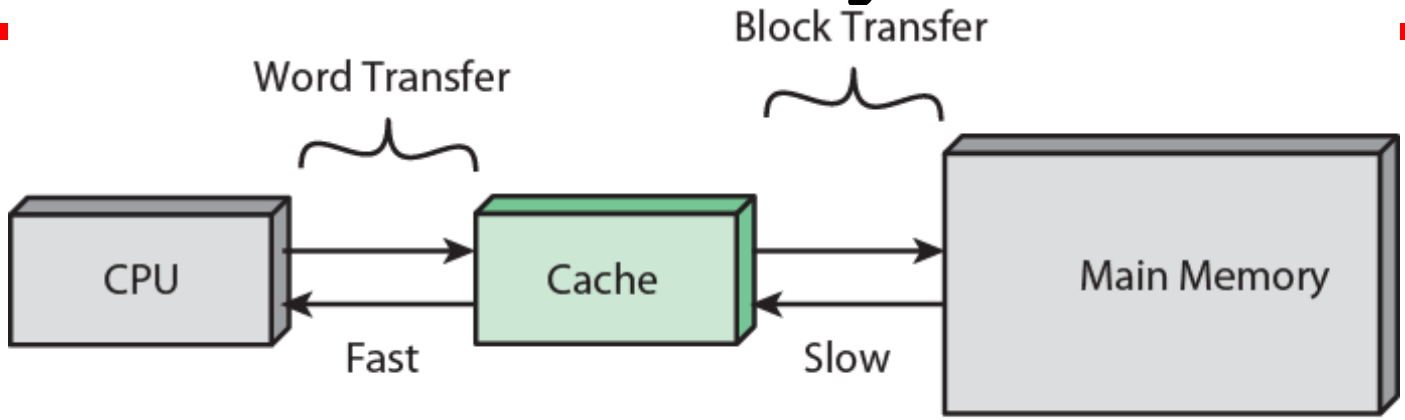
Session _3

- Cache Memory Principles
- Cache to Main Memory Mapping Functions
- Direct, Associative, and Set Associative
- Replacement Algorithm

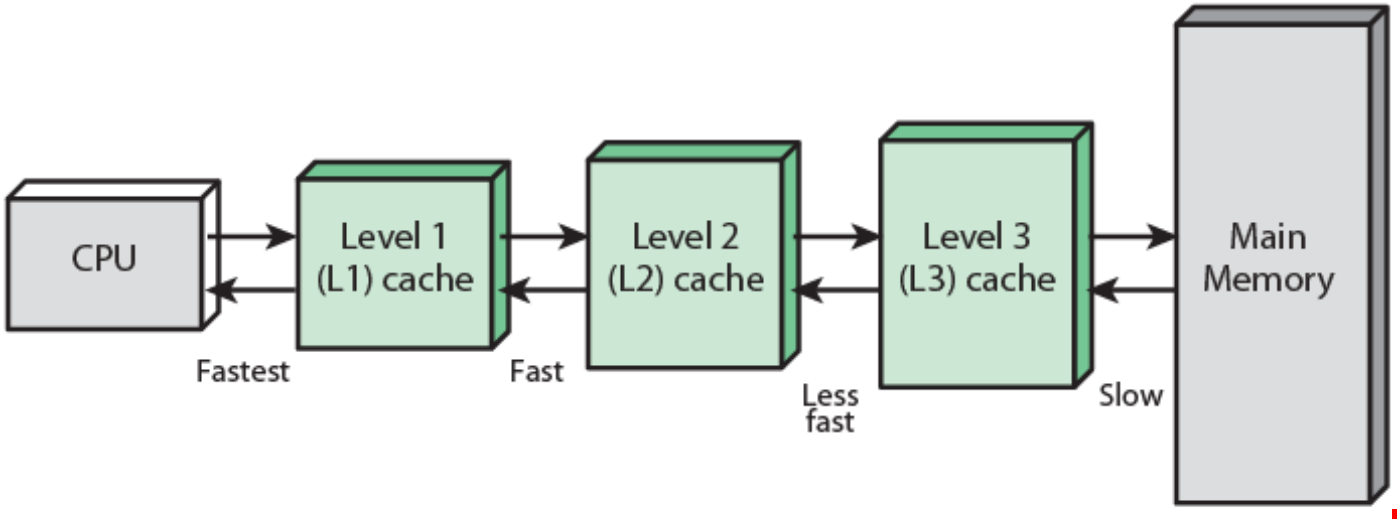
Cache

- Fastest memory in speed after registers
- At the same time provide a large memory size at the price of less expensive types of semiconductor memories.
- Sits between main memory and CPU

Cache and Main Memory



(a) Single cache



(b) Three-level cache organization

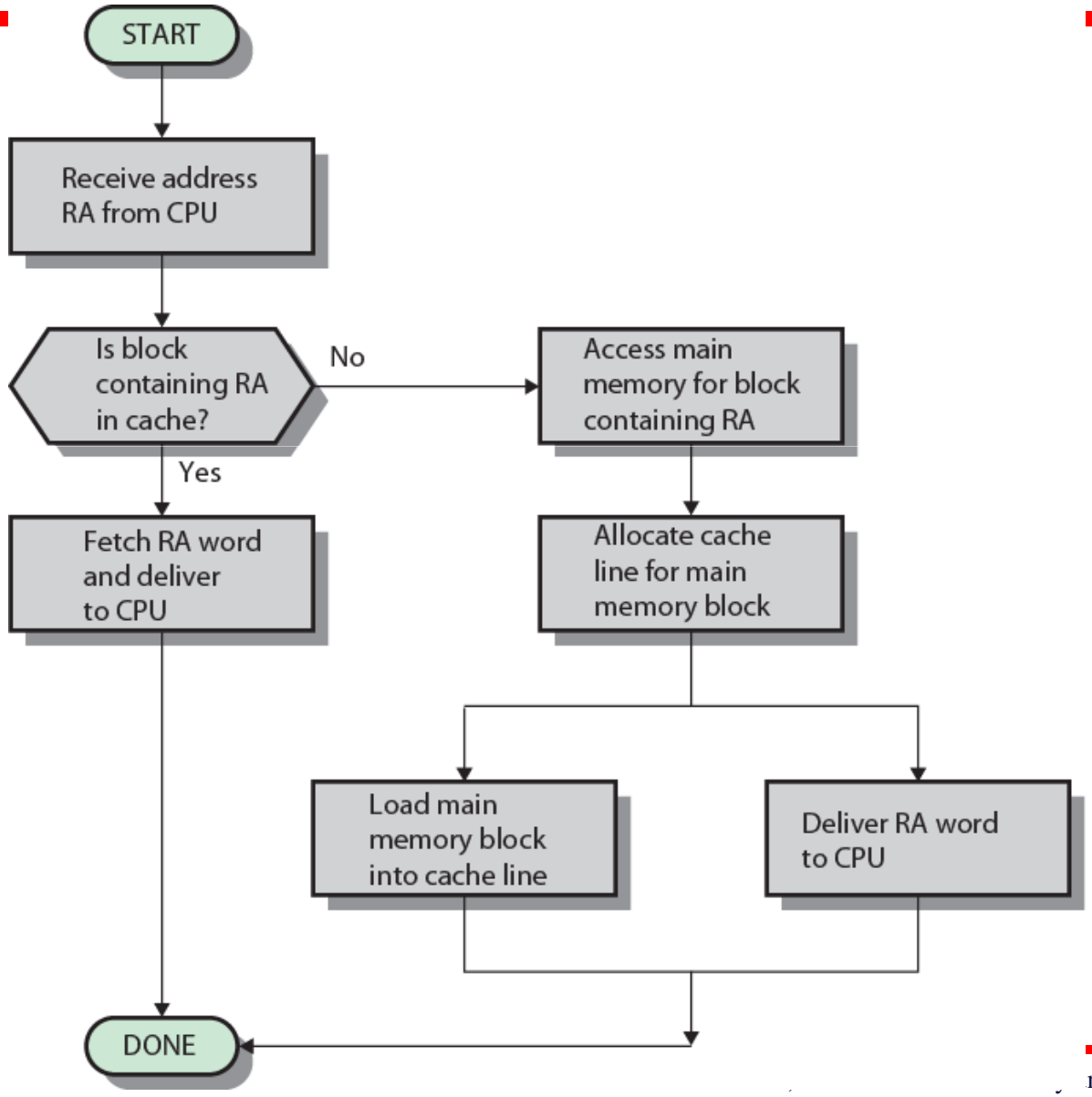
Locality of References

- Multiple locations (blocks) are mapped into cache.
- Because it is assumed that the next instruction to be executed lies near by the current instruction which is executed right now. So, cache memory transfer data to CPU without accessing the main memory more times.

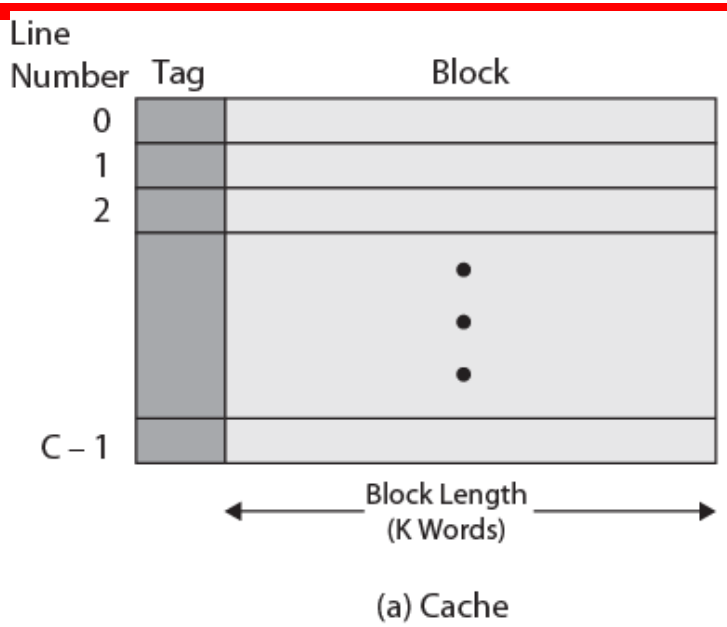
Cache operation – overview

- 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

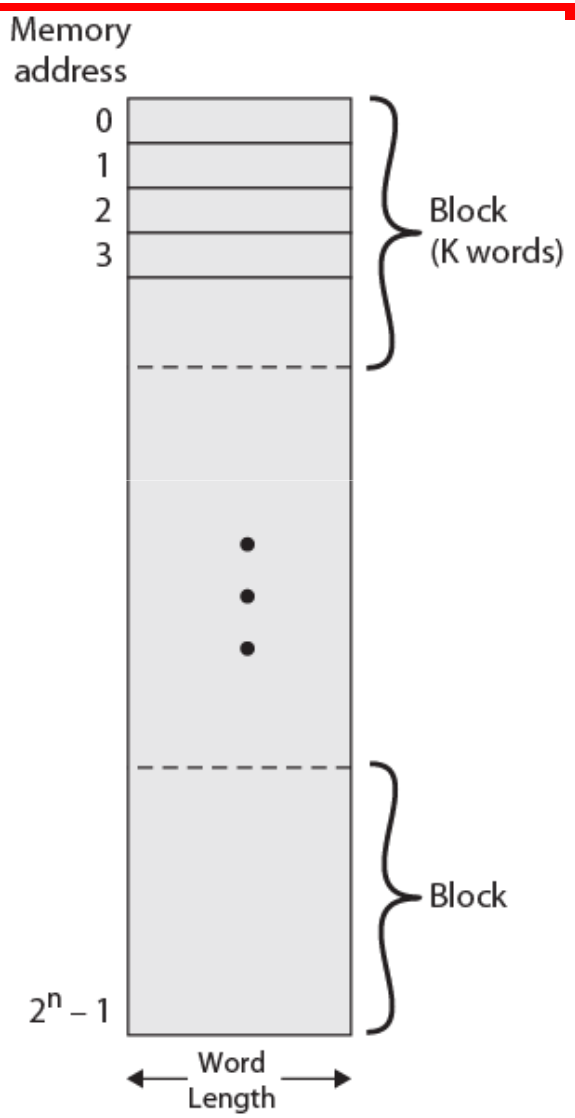
Cache Read Operation - Flowchart



Cache/Main Memory Structure



(a) Cache



(b) Main memory

Cache/Main Memory Structure

- Here, Main memory consists of up to 2^n addressable words, with each word having a unique n-bit address.
- For mapping purposes, this memory is divided into a number of fixed length blocks of K words each.
- Thus, M = 2^n/K block in main memory.
- The cache consists of m blocks, called lines. (m<<M)
- Each line contains K words, plus a tag of a few bits. Each line also includes control bits (not shown), such as a bit to indicate whether the line has been modified since being loaded into the cache.

Cache/Main Memory Structure

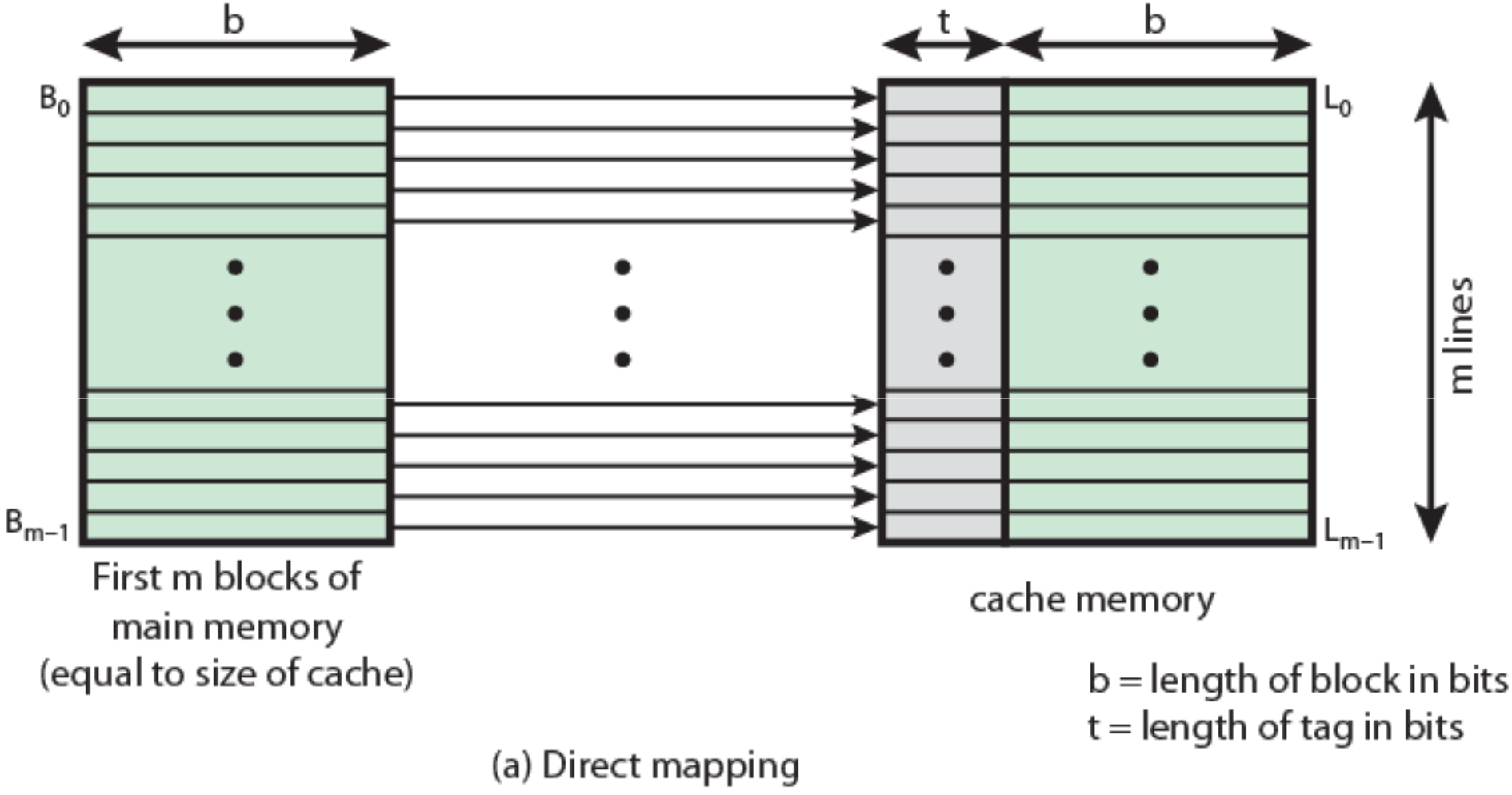
- The length of a line, not including tag and control bits, is the line size.
- At any time, some subset of the blocks of memory resides in lines in the cache.
- If a word in a block of memory is read, that block is transferred to one of the lines of the cache.
- Because there are more blocks than lines (m<<M), an individual line cannot be uniquely and permanently dedicated to a particular block.
- Thus, each line includes a tag that identifies which particular block is currently being stored.

Cache to Main Memory Mapping Functions

- Because there are fewer cache lines than main memory blocks
- An algorithm is needed for mapping main memory blocks into cache lines.
- Also it is needed for determining which main memory block currently occupies a cache line.

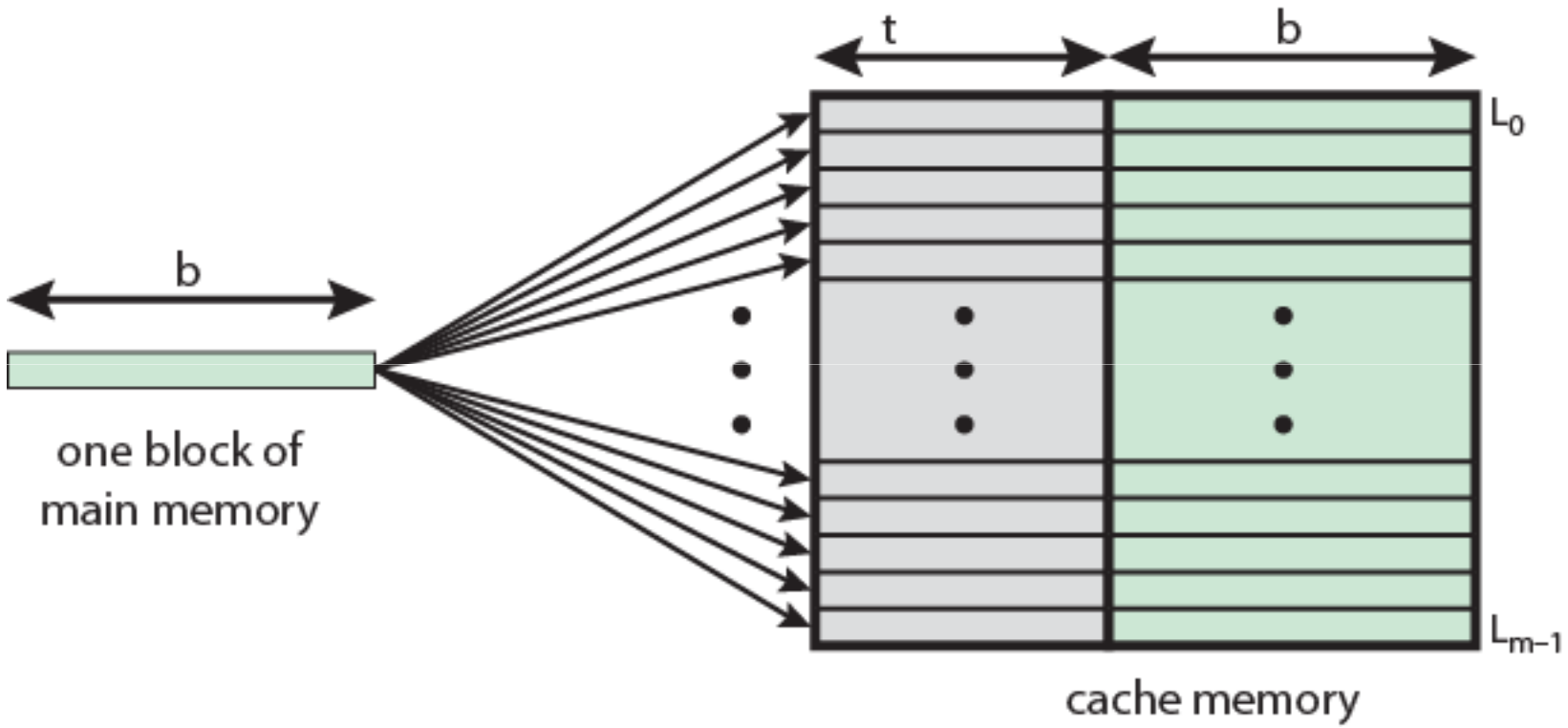
- Three techniques :
  - Direct
  - Associative
  - Set Associative

Direct Mapping from Cache to Main Memory



(a) Direct mapping

Associative Mapping from Cache to Main Memory



Mapping Function Example Specification

- Cache of 32Byte
- Cache block of 4 bytes
  - i.e. cache has lines (=8) of 4 bytes
- 128Bytes main memory
- 7 bit address
  - (2^7=128 bytes)
  - 32 blocks of 4 bytes each

Another Mapping Function Example Specification

- 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)
  - 4M blocks of 4 bytes each

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

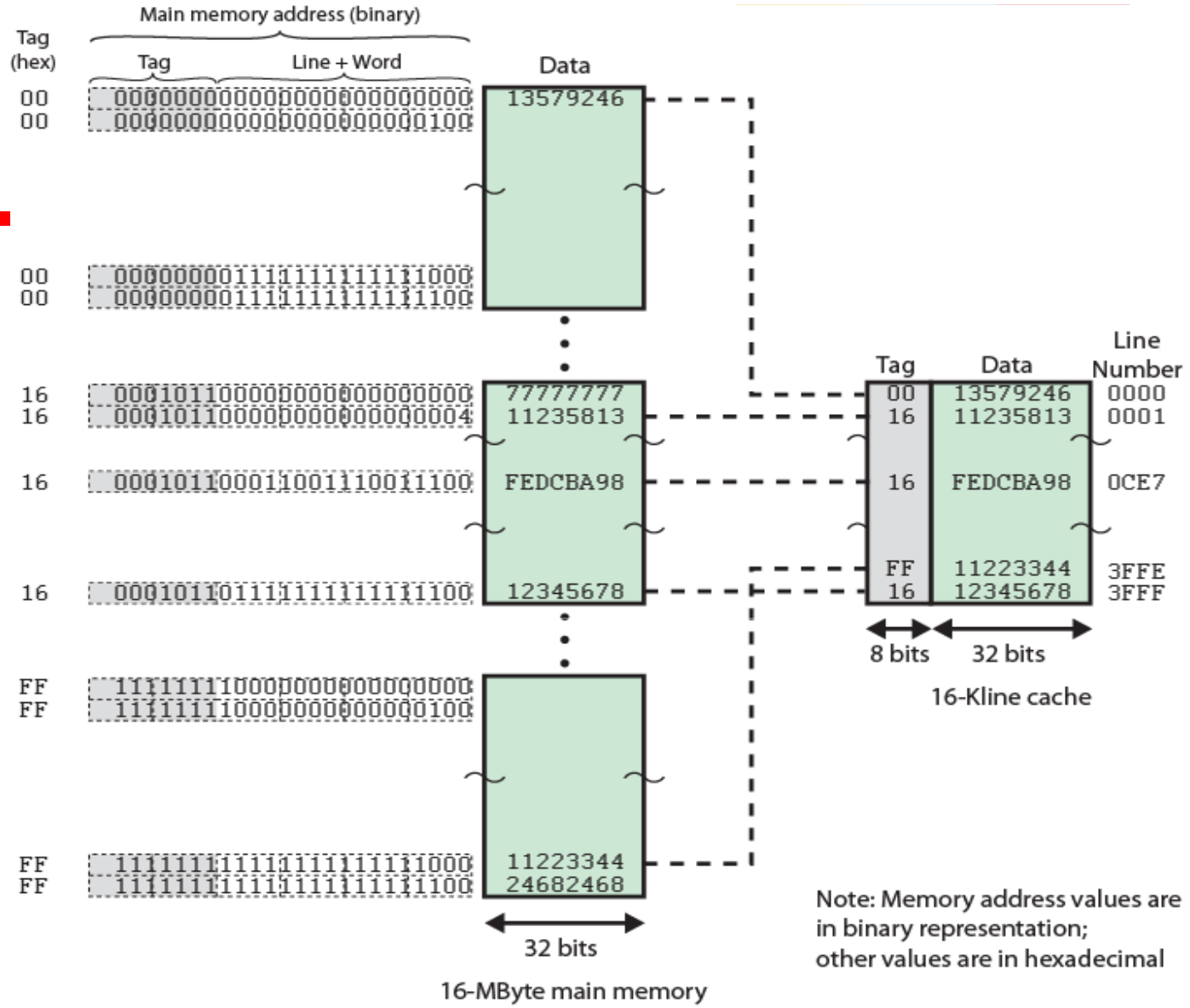
Tag s-r	Line or Slot r	Word w
8	14	2

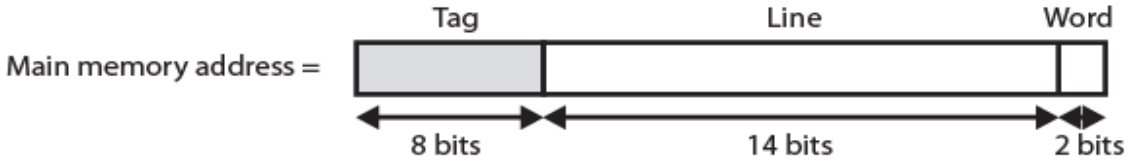
- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

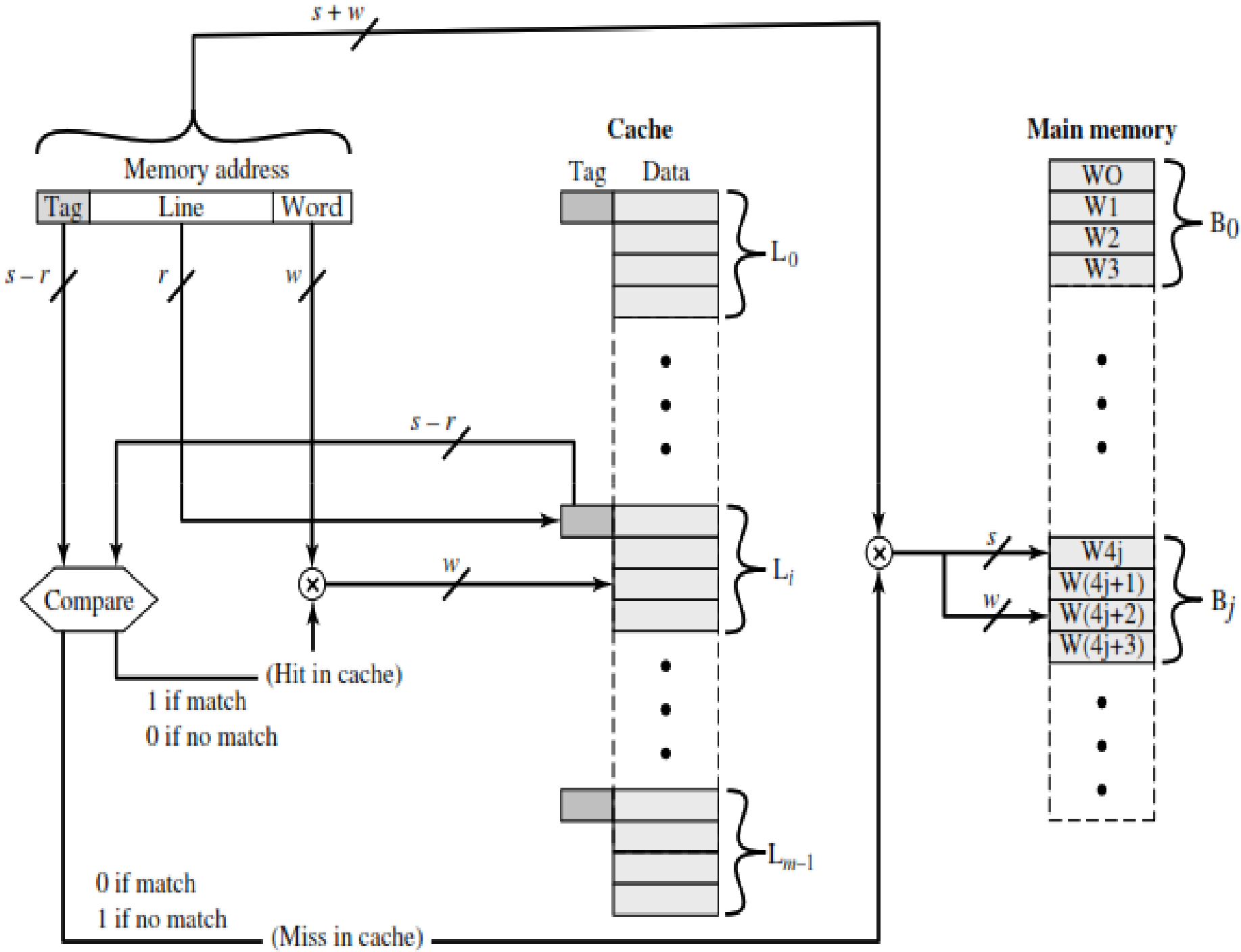
Direct Mapping
Cache Line Table

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 Example






s + w
Memory address
Tag
Line
Word
s − r
r
w
Cache
Tag
Data
L0
Li
Lm−1
Main memory
WO
W1
W2
W3
B0
W4j
W(4j+1)
W(4j+2)
W(4j+3)
Bj
s
w
s − r
w
Compare
(Hit in cache)
1 if match
0 if no match
0 if match
1 if no match
(Miss in cache)

Direct Mapping Summary

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

Exercises

- Show the address split up for the direct cache mapping function. Cache and main memory details are as follows:
  - Cache size is 128K Byte
  - Cache line size is 8 Bytes
  - Main memory size is 16M Bytes
  - Main Memory is Byte addressable
- Will main memory addresses x234560 and x374562 map to same cache line?

Review Questions

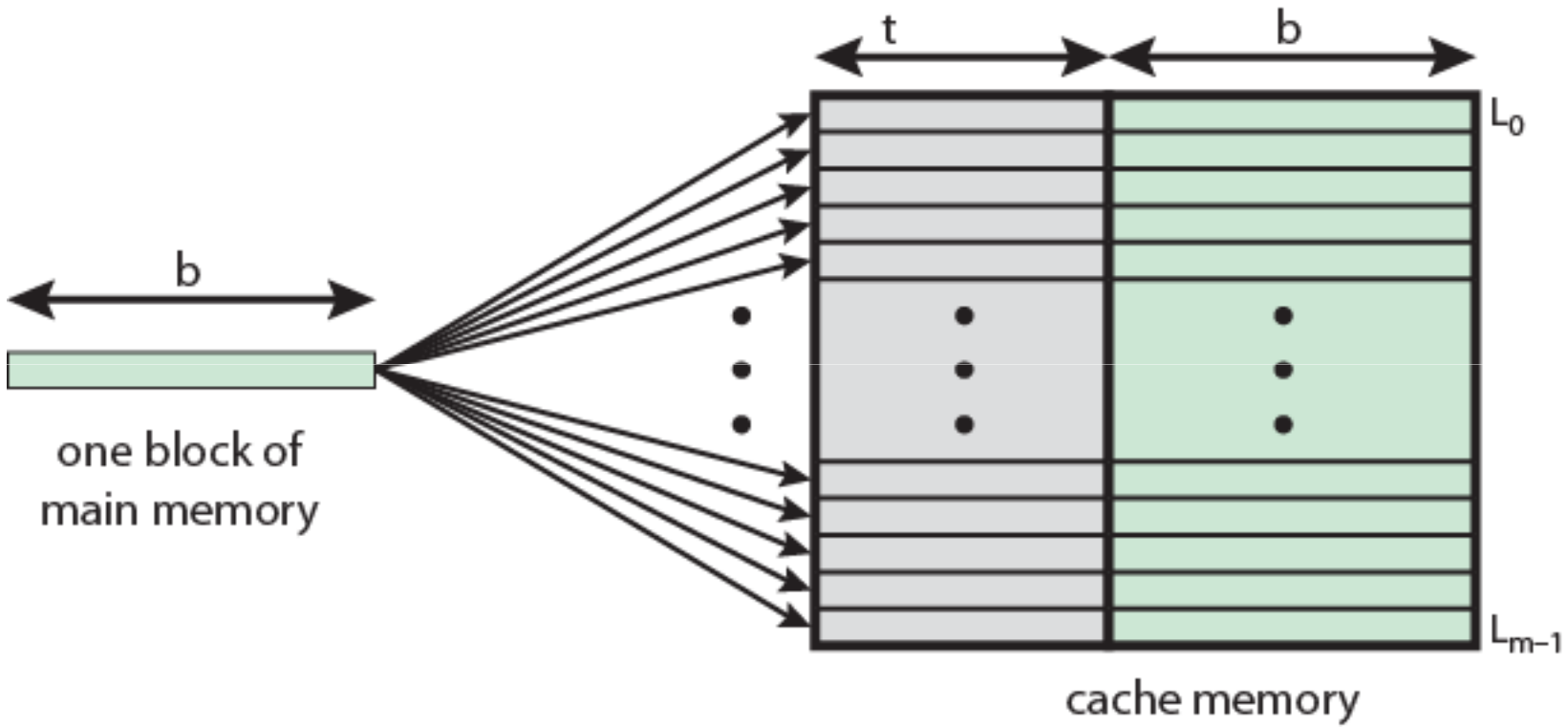
- Why tag bits are necessary to store along with the data bits in cache line?
  - *Hint: How to identify the two blocks of main memory mapped to same cache line*

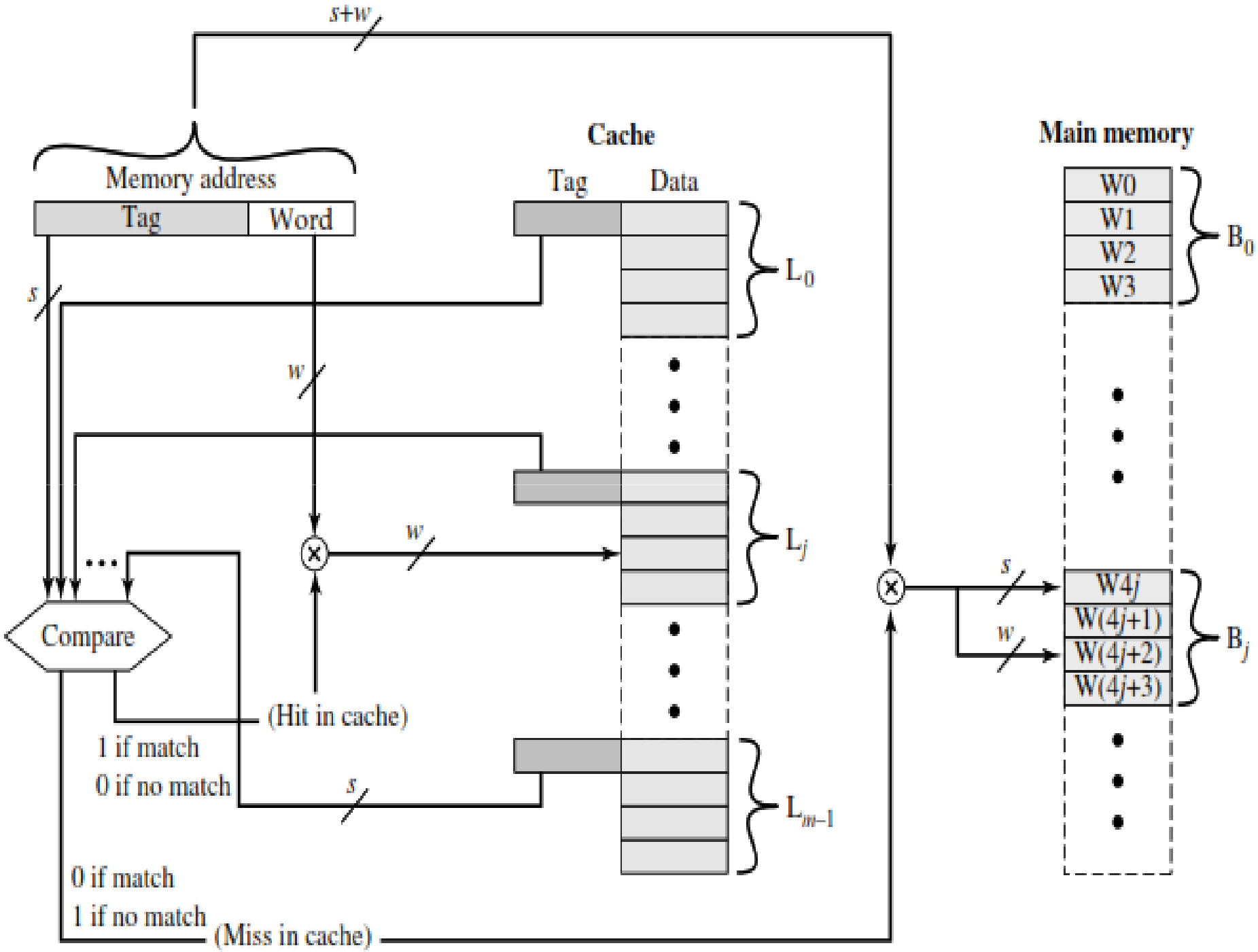
- How many tag comparisons are required to check the presence of the word requested by the processor in the direct mapped cache? Justify your answer.
  - *Hint: Block to line mapping is fixed*

Associative Mapping

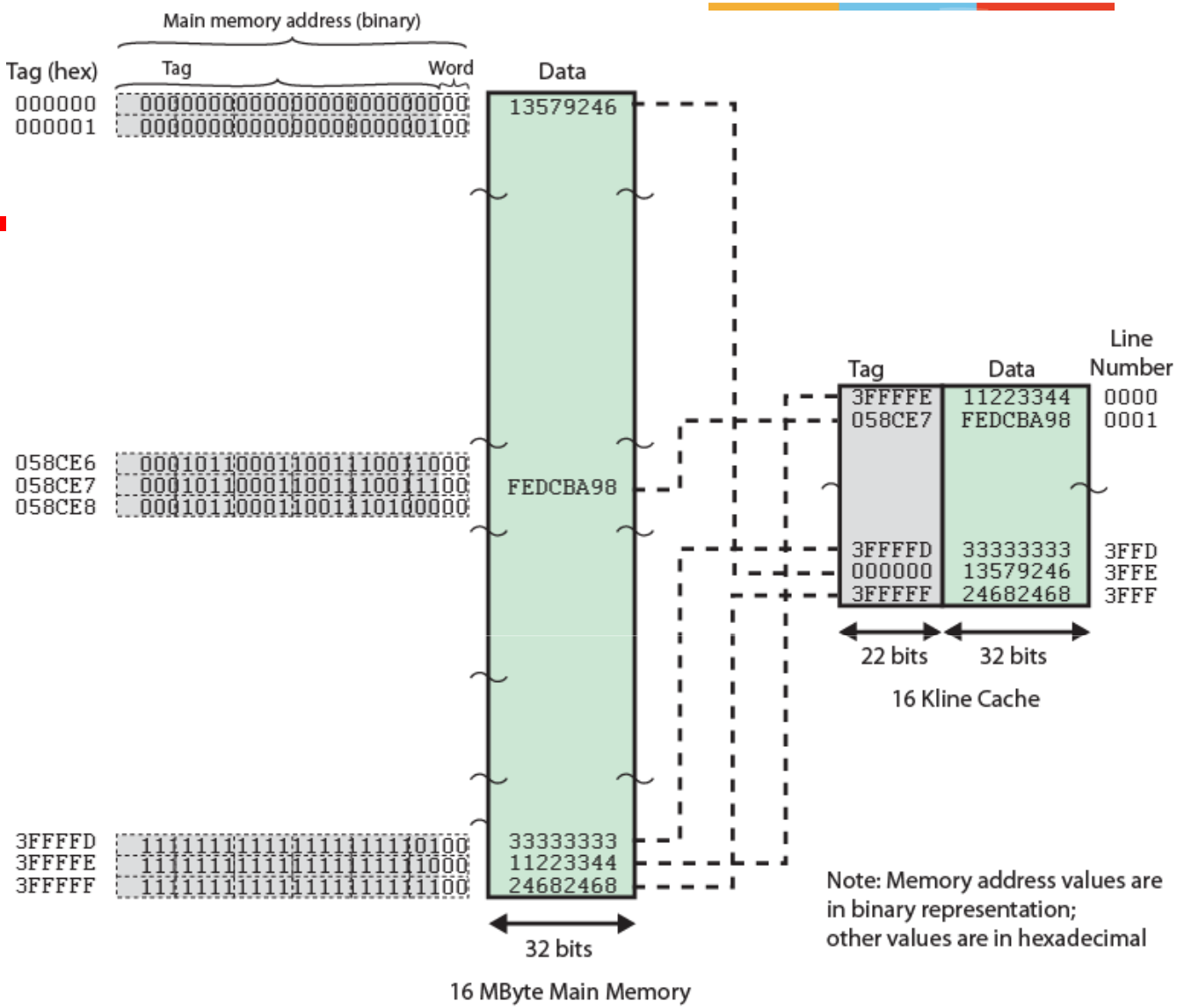
- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line's tag is examined for a match
- Cache searching gets expensive

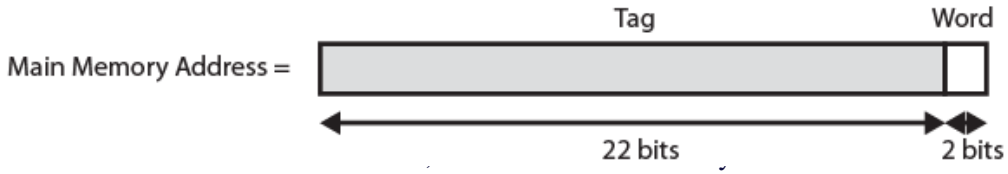
Associative Mapping from Cache to Main Memory




s+w
Memory address
Tag
Word
s
w
Cache
Tag
Data
L0
Lj
Lm–1
Main memory
W0
W1
W2
W3
B0
W4j
W(4j+1)
W(4j+2)
W(4j+3)
Bj
Compare
(Hit in cache)
1 if match
0 if no match
0 if match
1 if no match
(Miss in cache)

Associative Mapping Example





Associative Mapping Address Structure

Tag 22 bit	Word 2 bit

- 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

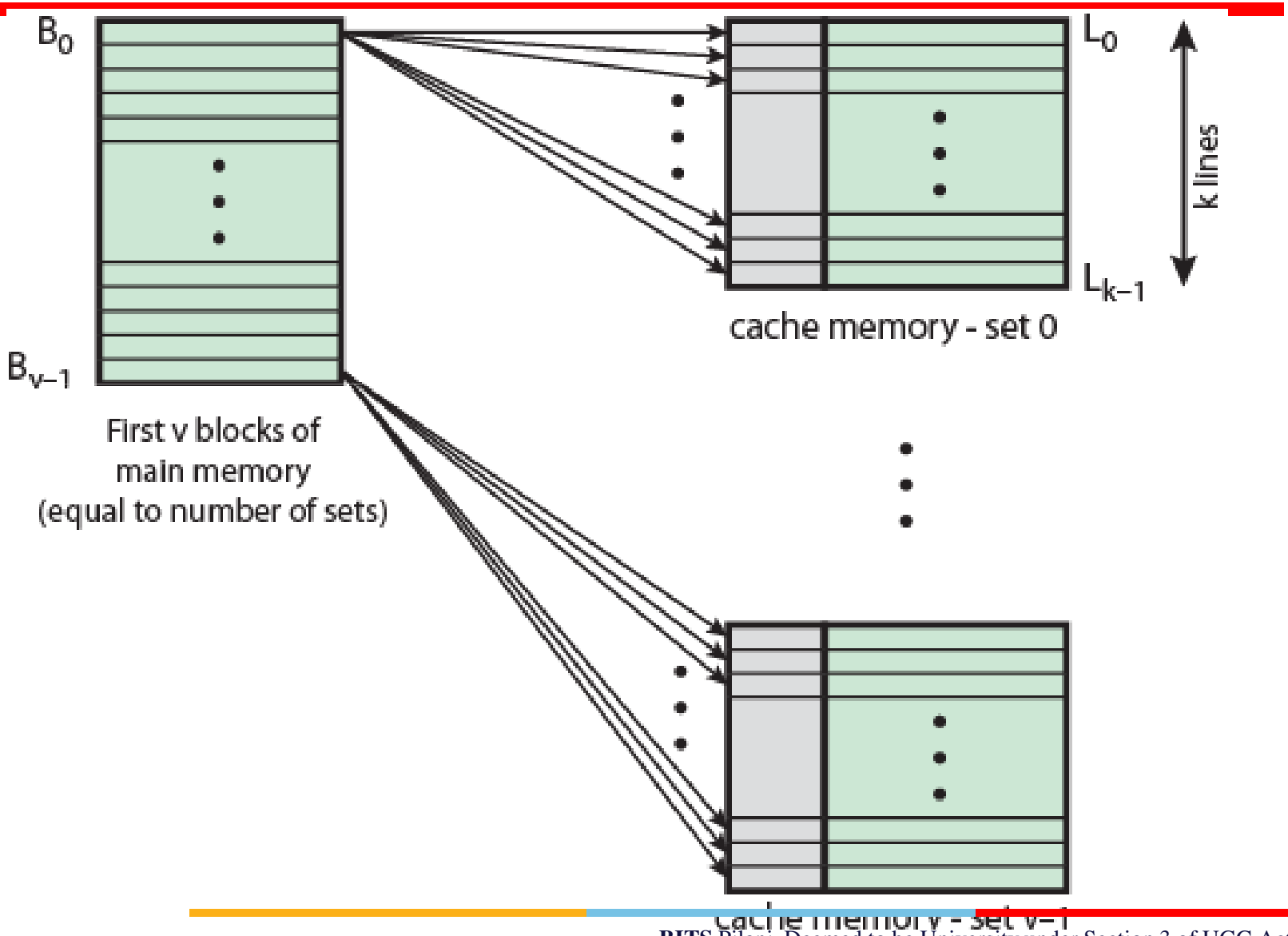
Associative Mapping Summary

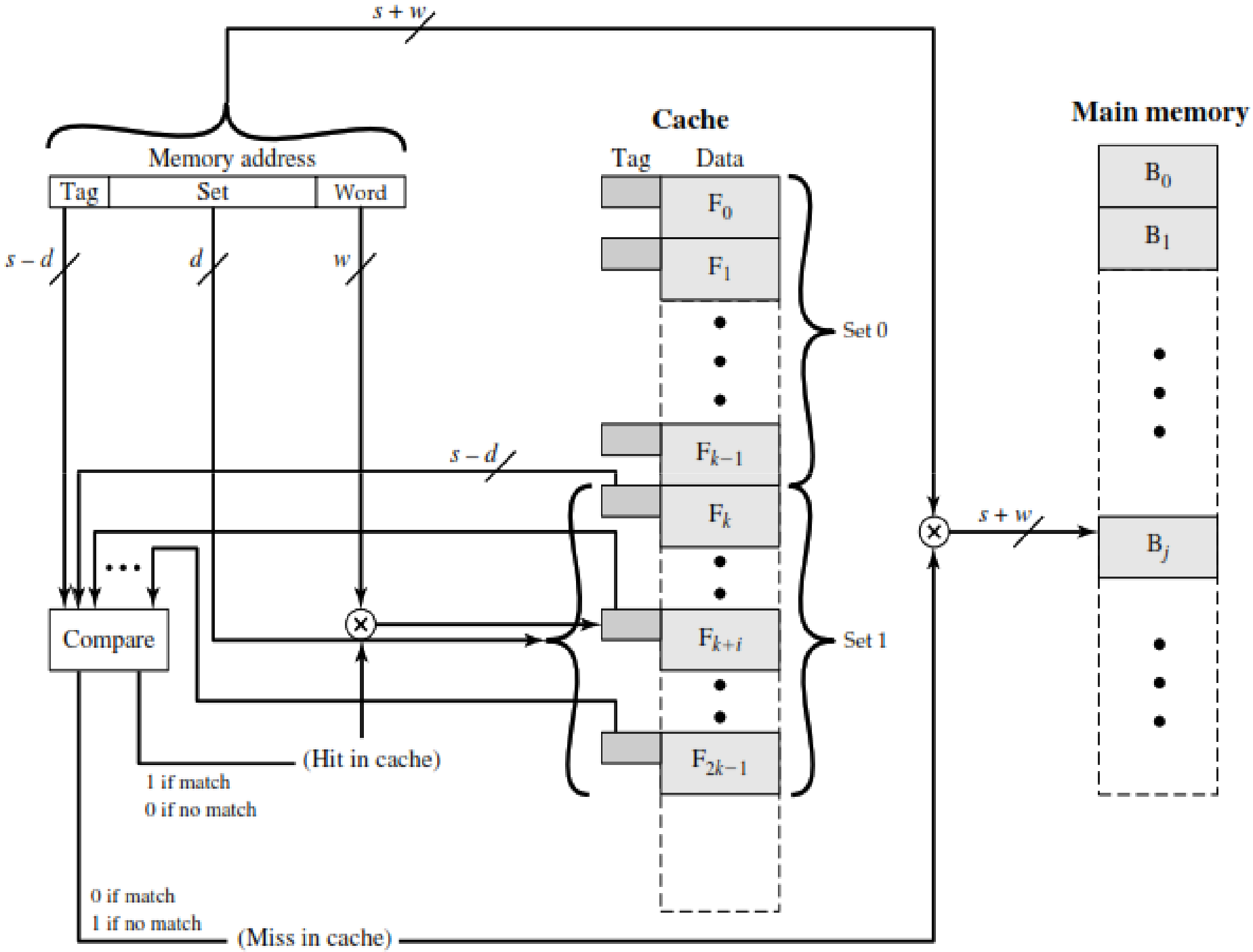
- Address length = (s + w) bits
- Number of addressable units = 2^(s+w) words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = 2^(s+ w)/2^w = 2^s
- Number of lines in cache = undetermined
- Size of tag = s bits

Set Associative Mapping

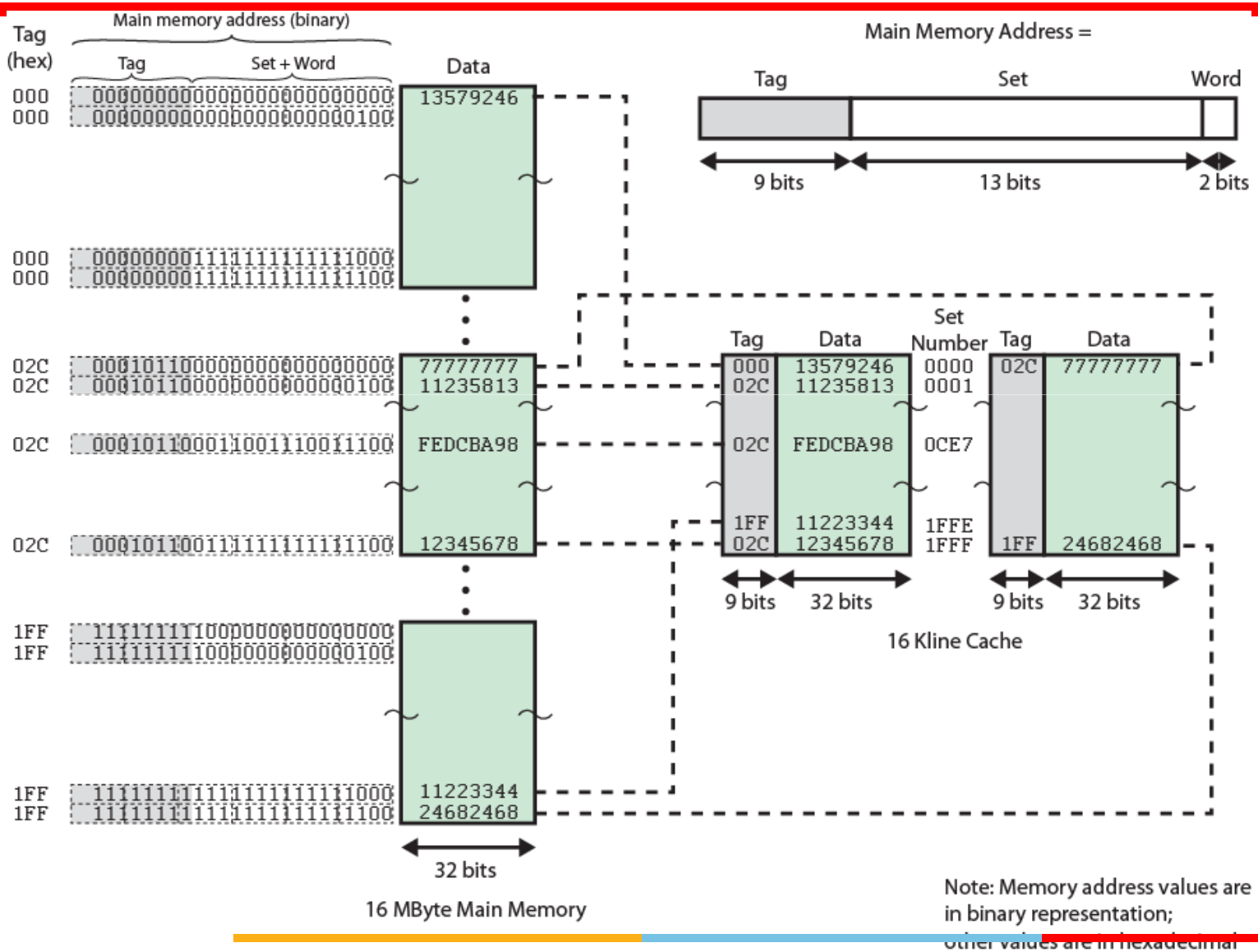
- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

Mapping From Main Memory to Cache: v Associative




s + w
Memory address
Tag
Set
Word
s – d
d
w
Cache
Tag
Data
F0
F1
Fk–1
Fk
Fk+i
F2k–1
Set 0
Set 1
s – d
Compare
(Hit in cache)
1 if match
0 if no match
0 if match
1 if no match
(Miss in cache)
s + w
Main memory
B0
B1
Bj

Two Way Set Associative Mapping Example



Set Associative Mapping Summary

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

QUESTIONS-

- A set-associative cache consists of 64 lines, or slots, divided into four-line sets. Main memory contains 4K blocks of 128 words each. Show the format of main memory addresses.

- A two-way set-associative cache has lines of 16 bytes and a total size of 8 kbytes. The 64-Mbyte main memory is byte addressable. Show the format of main memory addresses.

QUESTIONS-

- Consider a machine with a byte addressable main memory of 2^16 bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.
  - a. How is a 16-bit memory address divided into tag, line number, and byte number?
  - b. Into what line would bytes with each of the following addresses be stored?
    - 0001 0001 0001 1011
    - 1100 0011 0011 0100
    - 1101 0000 0001 1101
    - 1010 1010 1010 1010

QUESTIONS-

- Consider a machine with a byte addressable main memory of 2^16 bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.
  - c. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?
  - d. How many total bytes of memory can be stored in the cache?

Replacement Algorithms (1)
Direct mapping

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

Replacement Algorithms (2) Associative & Set Associative

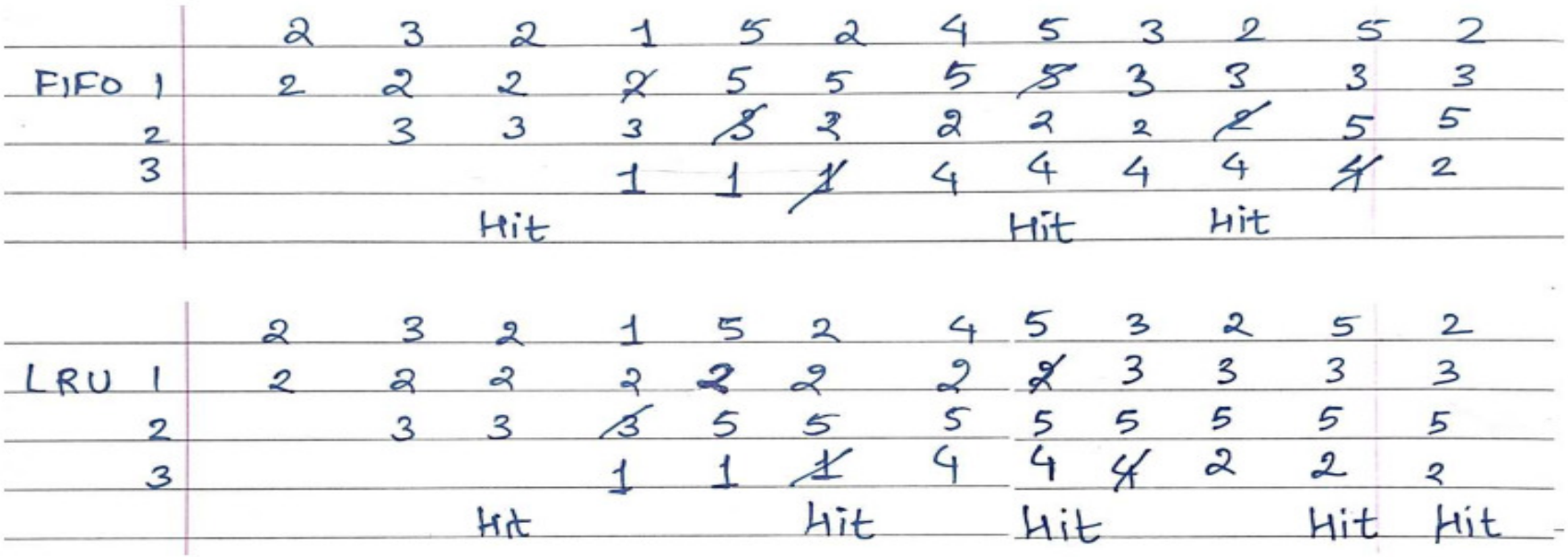
- Least Recently used (LRU)
- e.g. in 2 way set associative
  - Which of the 2 block is lru?
- First in first out (FIFO)
  - replace block that has been in cache longest
- Least frequently used
  - replace block which has had fewest hits
- Random

Replacement Algorithms (2)
Associative & Set Associative

Consider a fully associative cache memory having 3 lines. The main memory references generated by the processor are 2,3,2,1,5,2,4,5,3,2,5,2.

a) Show the final cache entries by assuming FIFO and LRU replacement policies.

b) Also count the number of HITS in each case.



FIFO: 3 Hits , LRU: 5 Hits