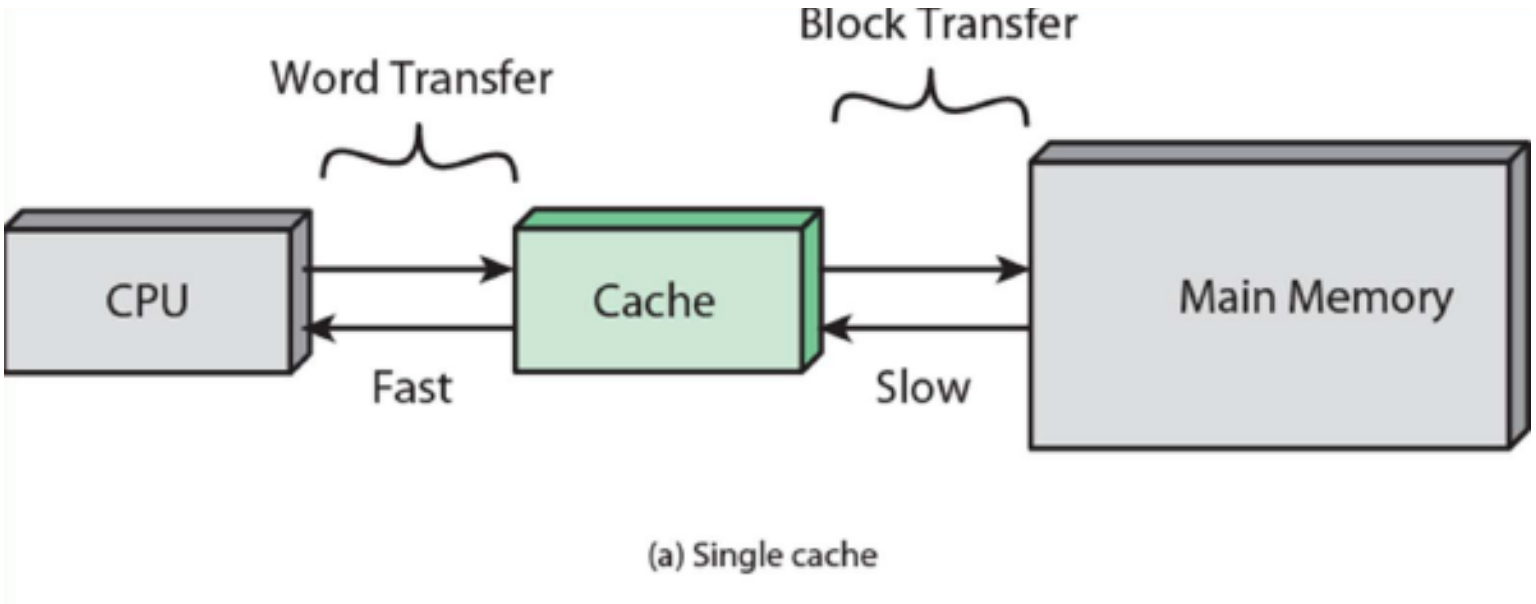
# Cache Memory

Computer Organization and Architecture
by
William Stallings

Prof.S.Meenatchi, SITE, VIT.

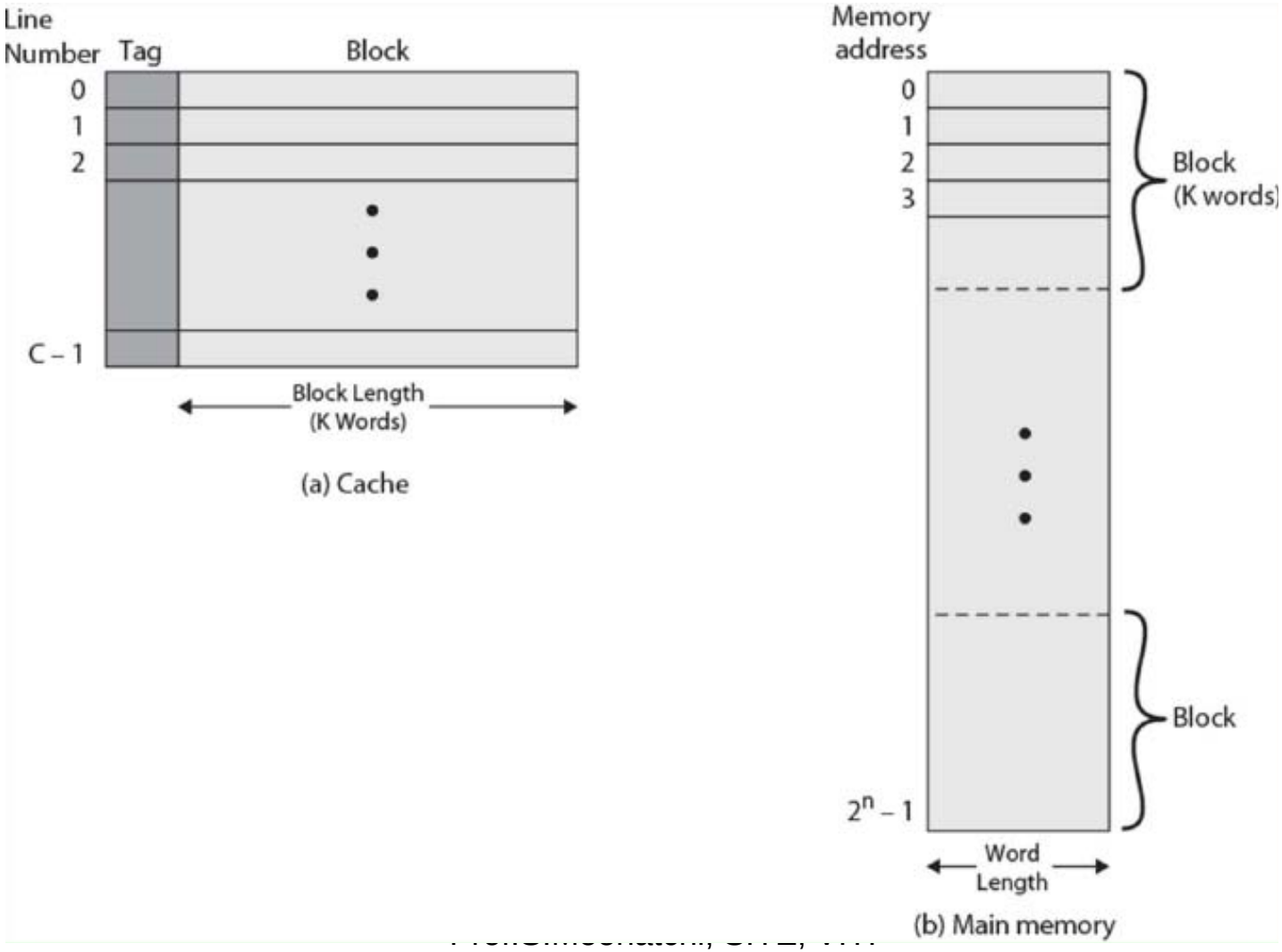
# Cache

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




(a) Single cache

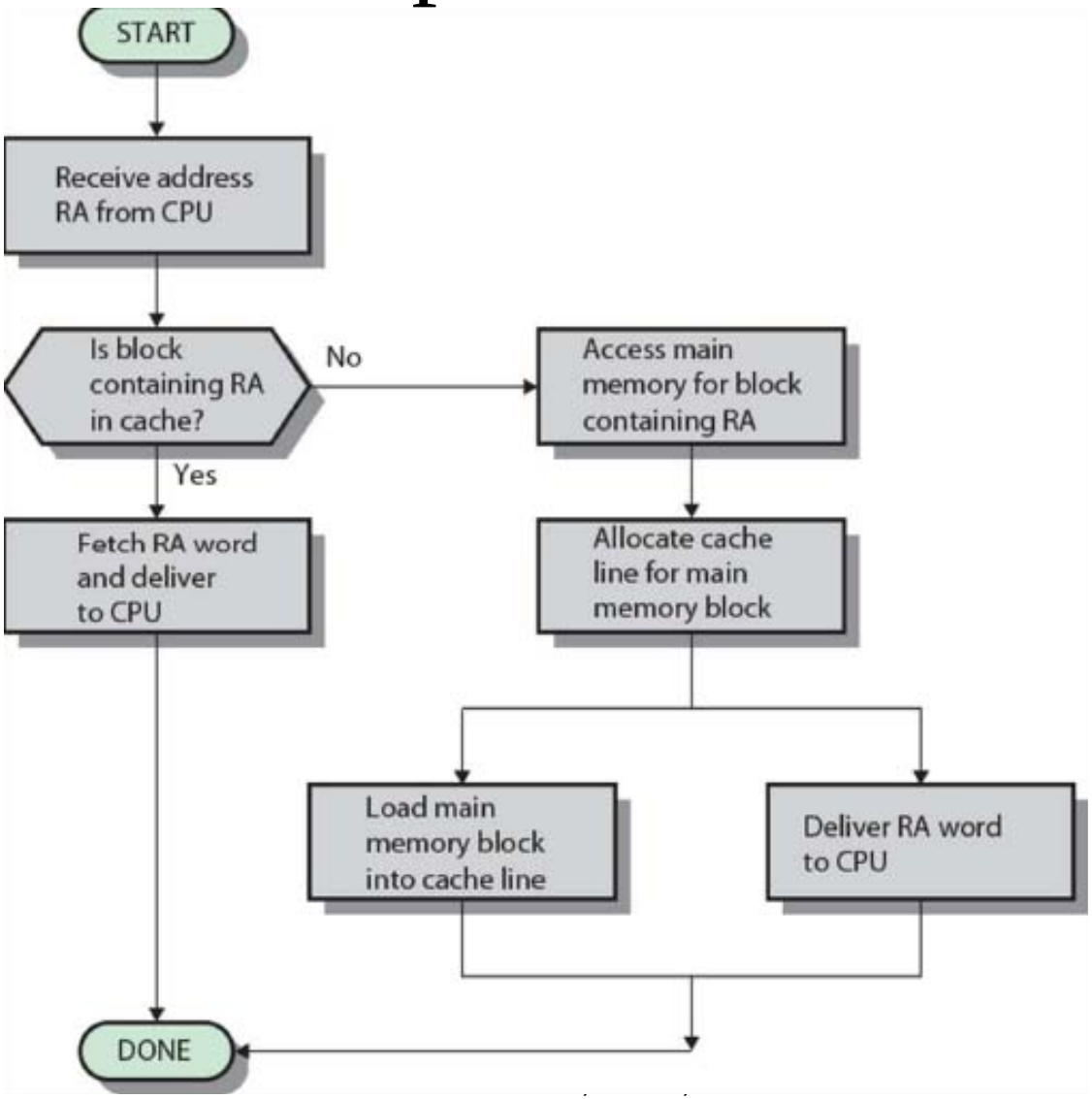
# Cache/Main Memory Structure



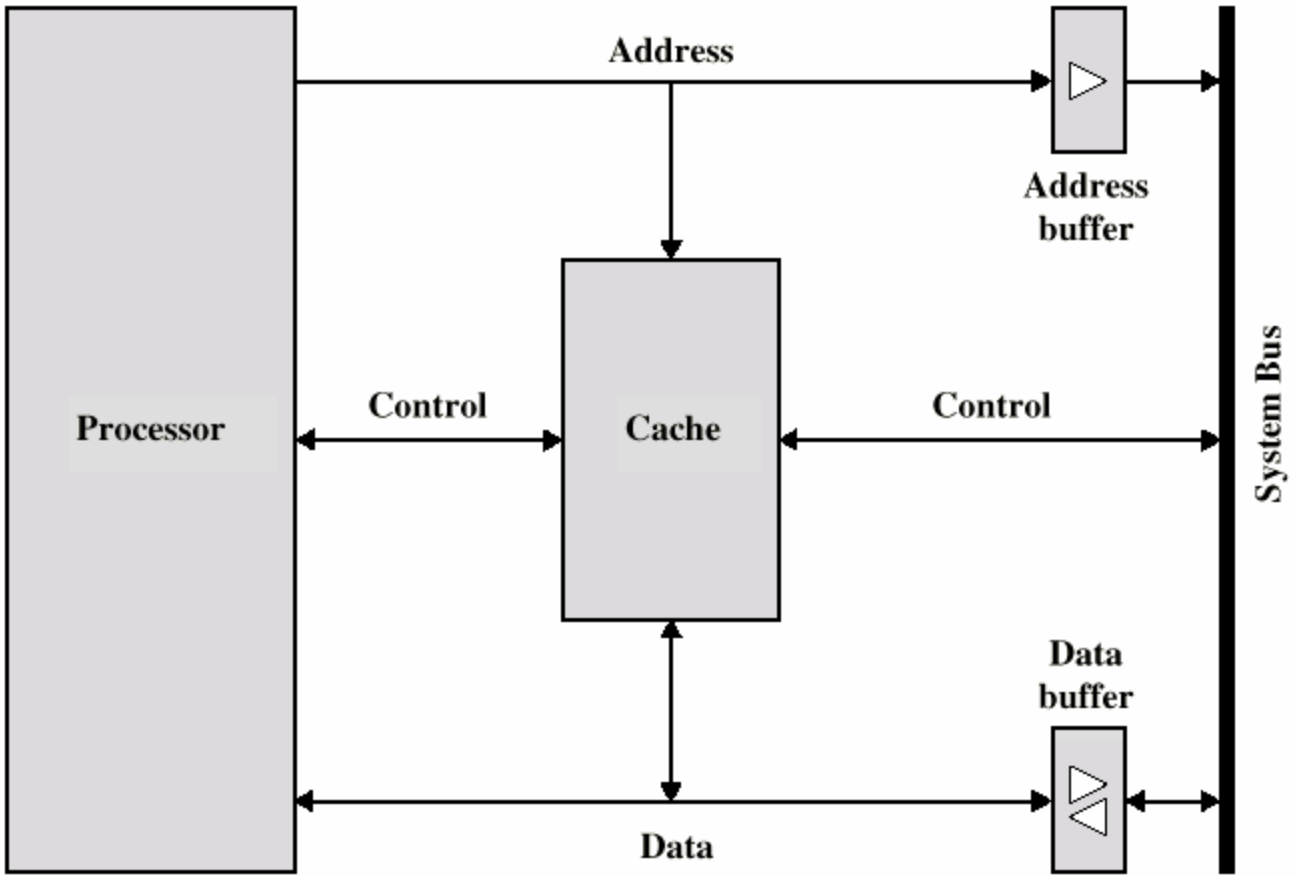
# Cache operation – overview

- 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 line

# Cache Read Operation - Flowchart



# Typical Cache Organization



# Elements of Cache Design

1. Size
2. Mapping Function
   1) associative, 2) direct, 3) set-associative
3. Replacement Algorithm
   1) LRU, 2) LFU, 3) FIFO, 4) Random
4. Write Policy
   1) Write-through, 2) Write-back, 3) Write -Once
5. Block Size
6. Number of Caches
   1) Single or 2-level, 2) Unified or Split

# Size

- Cost

  —More cache is expensive
- Speed

  —More cache is faster (up to a point)

  —Checking cache for data takes time

# Mapping Function

- Specification of correspondence between main memory blocks and cache blocks
- The transformation of data from main memory to cache memory

    1) Associative mapping
    2) Direct mapping
    3) Set-associative mapping

# Memory Capacity

- Number of bytes that can be stored

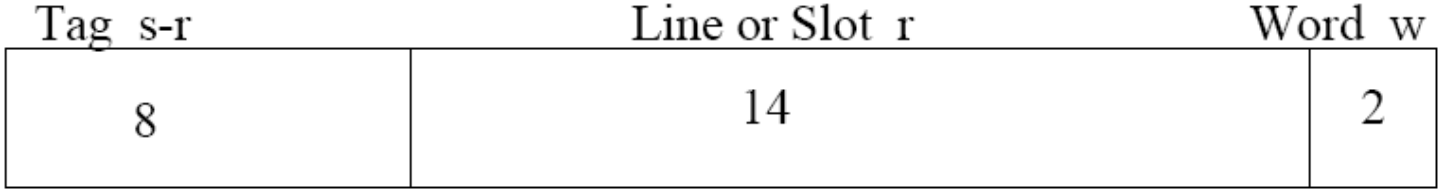
| Term | Normal Usage | Usage as Power of 2 |
|---|---|---|
| K ( Kilo) | $10^3$ | $2^{10} = 1,024$ |
| M (Mega) | $10^6$ | $2^{20} = 1,048,576$ |
| G (Giga) | $10^9$ | $2^{30} = 1,073,741,824$ |
| T (Tera) | $10^{12}$ | $2^{40} = 1,099,511,627,776$ |

# Direct Mapping

- Each block of main memory maps to only one cache line
- Address consists of two parts
  - Least Significant **w bits -** identify unique word
  - Most Significant **s bits** - specify one memory block
- Most Significant **s bits** are split into
  - a cache line field **r bits** and
  - a tag of **s-r bits** (most significant)

# Direct Mapping - Address Structure

- 24 bit address

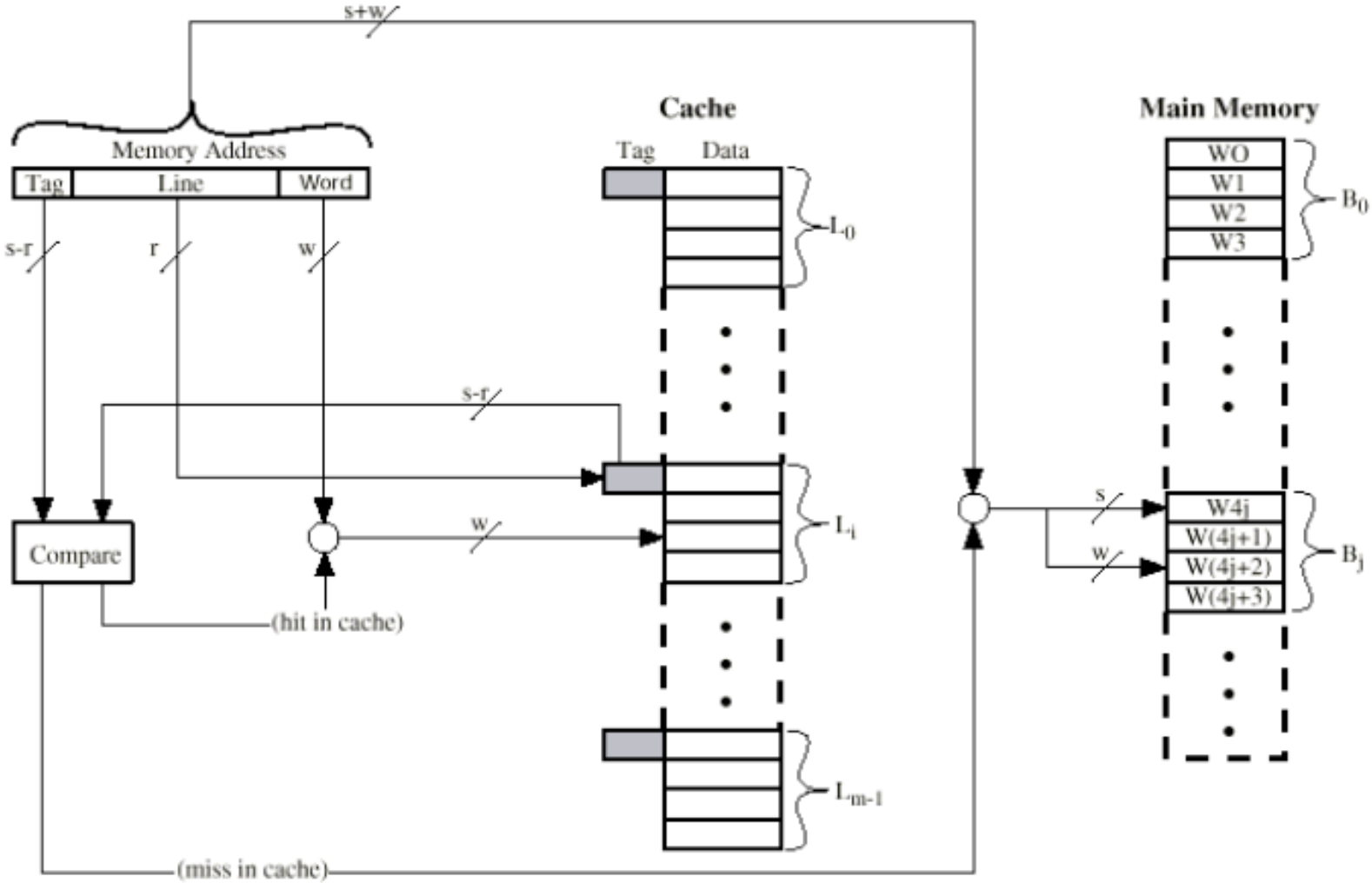



- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bits tag (=22-14)
  - 14 bits line
- No two blocks in the same line have the same Tag field
- Content of cache is checked by finding line and checking Tag

# 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 Cache Organization



# 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
- Index=Line + word
- Index field is used to access the word from cache

# 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.

# Example: Direct Mapping Function

- Cache memory = 64kByte
- Block size = 4 bytes
- Main memory = 16MBytes

- Block = 4bytes
  = $2^2$ words

- Cache = (64k/4) lines
  = 16k lines of 4 bytes each
  = $2^4 * 2^{10}$
  = $2^{14}$ lines of 4 bytes

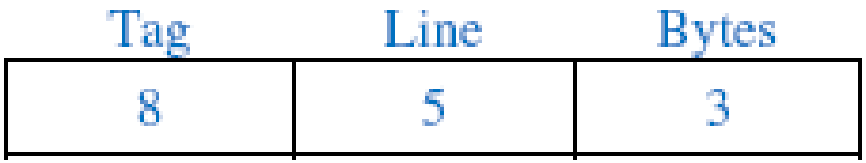
- Main = (16M/4) blocks
  = 4M blocks of 4 byte each
  = $2^2 * 2^{10} * 2^{10}$
  = $2^{22}$ blocks of 4 byte each

Main memory =16M
$= 2^4 * 2^{10} * 2^{10}$
$= 2^{24}$ words
Each word is directly addressable by 24-bit address

Addressable units: $2^{24}$
Address length: 24 bits
Block size: $2^2$ words
Blocks in main memory: $2^{22}$
Lines in cache: $2^{14}$
s = 22
w = 2
r = 14
Size of tag (s-r): 8

# Example

- 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. How is a 16-bit memory address divided into tag, line number, and byte number?
- **Answer:**
  - Block size = 8 bytes = $2^3$ bytes
    - 3 bits are used for byte offset.
  - Cache lines = 32 = $2^5$
    - 5 bits are used as index bits.
  - Main memory = $2^{16}$ bytes
    - Each word is directly addressable by 16-bit address
  - Remaining 16-(5+3) = 8 bits are used as tag bits.



# Assignment

1. A 16MB main memory has 32KB cache with 8 bytes per line.

   i) How many lines are there in the cache?

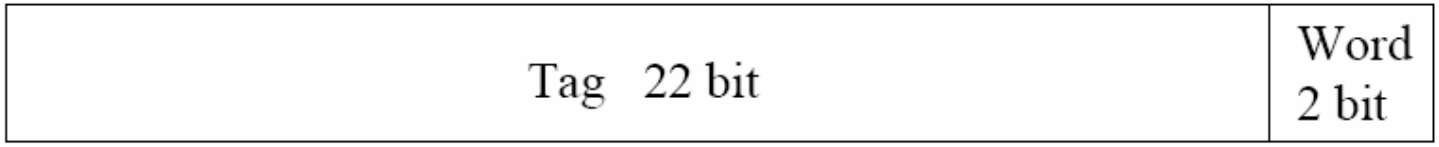
   ii) Show how the main memory and cache memory is organized when the cache is direct- mapped.

   iii) Show how the main memory address is partitioned.

2. A digital computer has a memory unit of 128K x 16 and a cache memory of 1K words. The cache uses direct mapping with a block size of four words.
   - How many bits are there in the tag, index, block and word fields of the address format?
   - How many blocks can the cache accommodate?

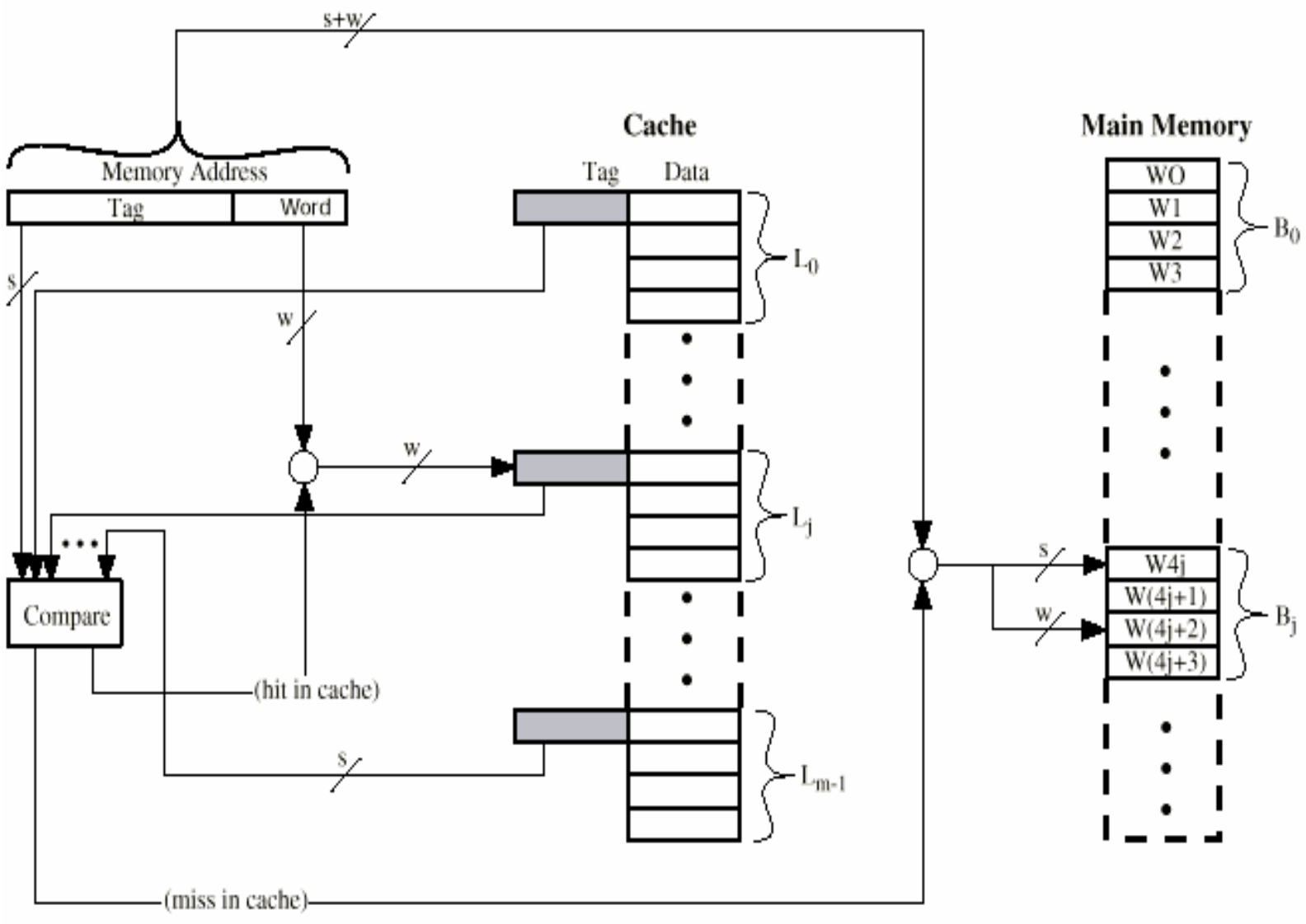
# Associative Mapping

- 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.
- **Address Structure**




  - Compare tag field with tag entry in cache to check for hit
  - Least significant 2 bits of address identify the word

# Associative Cache Organization



# 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

# Example: Associative Mapping

- Cache memory = 64kByte
- Block size = 4 bytes
- Main memory = 16MBytes

- Block = 4bytes
  = $2^2$ words

- Cache = (64k/4) lines
  = 16k lines of 4 bytes each
  = $2^4 * 2^{10}$
  = $2^{14}$ lines of 4 bytes

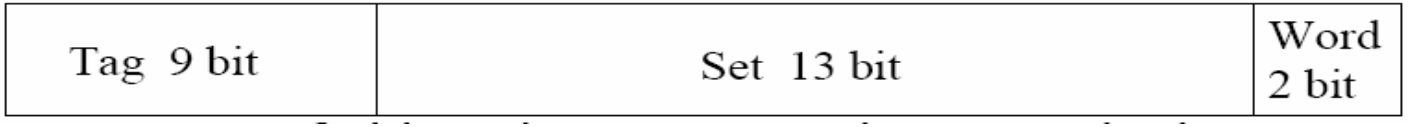
- Main = (16M/4) blocks
  = 4M blocks of 4 byte each
  = $2^2 * 2^{10} * 2^{10}$
  = $2^{22}$ blocks of 4 byte each

Main memory =16M
$= 2^4 * 2^{10} * 2^{10}$
$= 2^{24}$ words
Each word is directly addressable by 24-bit address

Addressable units: $2^{24}$
Address length: 24 bits
Block size: $2^2$ words
Blocks in main memory: $2^{22}$
Lines in cache: $2^{14}$
s = 22
w = 2
Size of tag: 22

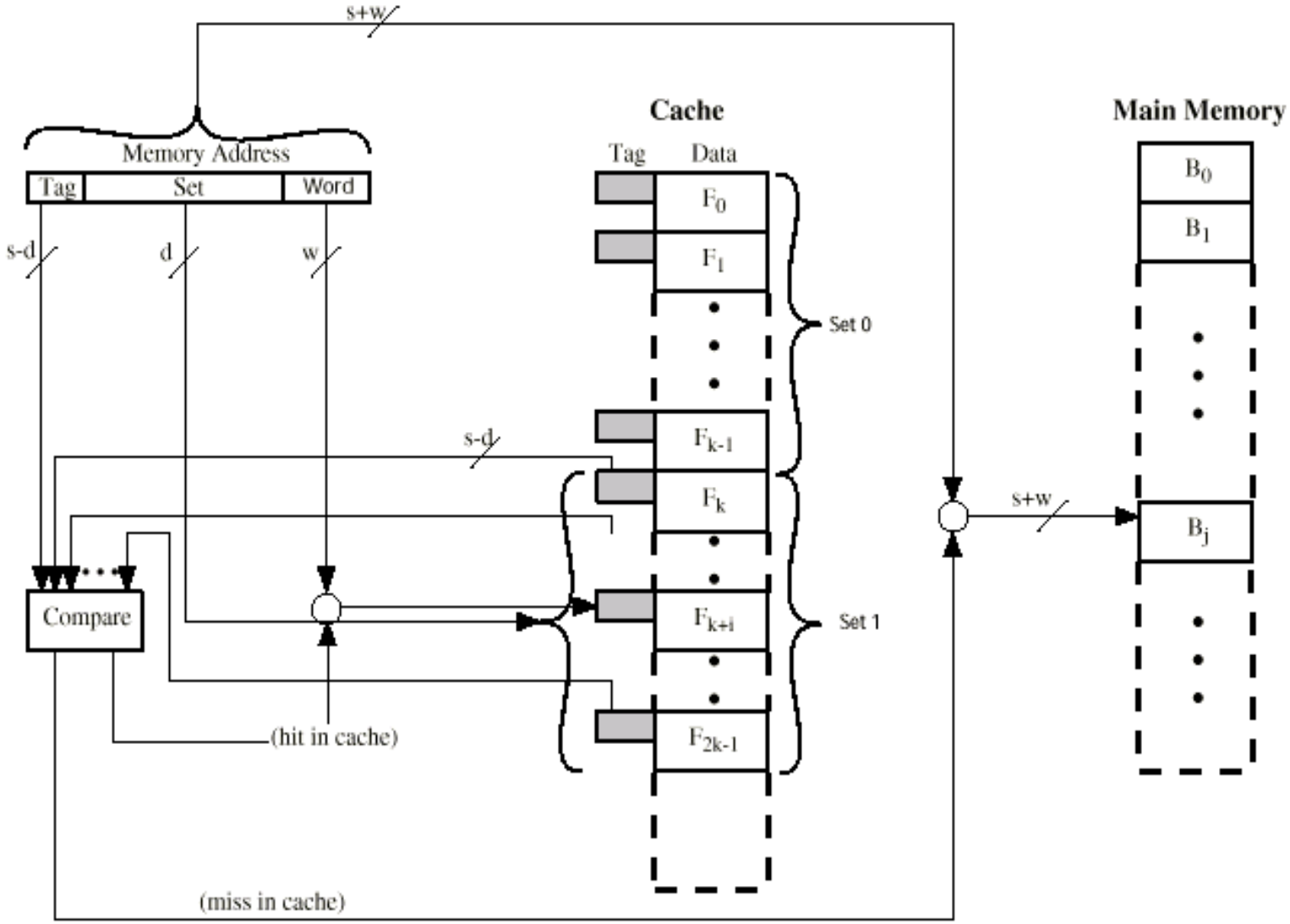
# Set Associative Mapping

- 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
- **Address Structure**




  - Use set field to determine cache set to look in
  - Compare tag field to see if we have a hit

# Two Way Set Associative Cache Organization



# 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 = $2^{d}$
- Number of lines in cache = kv = k * $2^{d}$
- Size of tag = (s – d) bits

# Example: Two way set associative Mapping

- Cache memory = 64kByte
- Block size = 4 bytes
- Main memory = 16MBytes
- Each Set = 2 lines

- Block = 4bytes
  = $2^2$ words

- Set size = 2 * 4 = 8 = $2^3$ words

- Cache = (64k/4) lines
  = 16k lines of 4 bytes each
  = $2^4 * 2^{10}$
  = $2^{14}$ lines of 4 bytes

- Cache = 64kByte
  = (64k/8) sets
  = 8k sets of 2 lines each
  = $2^3 * 2^{10}$
  = $2^{13}$ sets of 2 lines each

Main memory = (16M/4) blocks
= 4M blocks of 4 byte each
= $2^2 * 2^{10} * 2^{10}$
= $2^{22}$ blocks of 4 byte each

Main memory =16M
=24*210*210
=224 words
Each word is directly addressable by 24-bit address
Addressable units: $2^{24}$
Address length: 24 bits
Block size: $2^2$ words
Blocks in main memory: $2^{22}$
Lines in set: 2
Number of sets: $2^{13}$
Lines in cache: $2^{14}$
s = 22
w = 2
d = 13
Size of tag (s-d): 9

# Example: Set Associative Mapping

- A set associative cache consists of 64 lines divided into four-line sets. Main memory contains 4K blocks of 128 words each. Show the format of memory addresses.
- Each block contains 128 words.
  - Block = 128 words.

    $= 2^7$ words

  Therefore, 7 bits are needed to identify the word within the block.
- Cache = 64 lines / 4
  - Cache is divided into 16 sets of 4 lines each ($2^4$ sets).
  - Therefore, 4 bits are needed to identify the set number.
- Main memory = 4K blocks of 128 words each. ($2^2 * 2^{10} = 2^{12}$ blocks)
  - Therefore, 12 bits are needed to specify the block (set + tag = 12 bits).
  - Tag length is 8 bits.
- Main memory = 4K blocks of 128 words each. ($2^2 * 2^{10} * 2^7 = 2^{19}$ words)
  - Each word is directly addressable by 19-bit address.

| TAG | SET | WORD |
|---|---|---|
| 8 | 4 | 7 |

# Replacement Algorithms

- **Direct mapping**
  - No choice
  - Each block only maps to one line
  - Replace that line
- **Associative & Set Associative**
  - **Least Recently used (LRU)**
    —replace block that has been in cache longest
  - **First in first out (FIFO)**
    —replace block that has been entered first in cache
  - **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

# Write Policy

- **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

# 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…

# 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

# Line Size

- Retrieve not only desired word but a number of adjacent words as well
- Increased block size will increase hit ratio at first
- 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