UNIVERSITI TENAGA NASIONAL

CHAPTER 8
MEMORY

CSNB153 COMPUTER SYSTEM

Limitation in Computer Performance

- Memory speed slower than the processor's speed

Objective

- To study the **development** of an effective **memory organization** that supports the processing power of the CPU

Key Characteristics of Computer Memory Systems

- Location
  - Internal
  - External
- Capacity
  - Number of words
  - Number of bytes
- Unit of transfer
  - Word
  - Block
- Access method
  - Sequential
  - Direct
  - Random
  - Associative
- Performance
  - Access time
  - Cycle time
  - Transfer rate

Key Characteristics of Computer Memory Systems (Cont.)

- Physical type
  - Semiconductor
  - Magnetic
  - Optical
  - Magneto-optical
- Physical characteristics
  - Volatile/nonvolatile
  - Erasable/nonerasable
- Organization
  - Memory modules

Terminology

Terminology	Description
Capacity	Amount of information that can be contained in a memory unit (word / bytes)
Word	Natural unit of organization in the memory. 1 byte or word = 8 bits
Addressable unit	Fundamental data element size that can be addressed in the memory
Unit of transfer	Number of data elements transferred at a time • Bits – main memory • Word – secondary memory
Transfer rate	Rate at which data is transferred to/from the memory device

Terminology (Cont.)

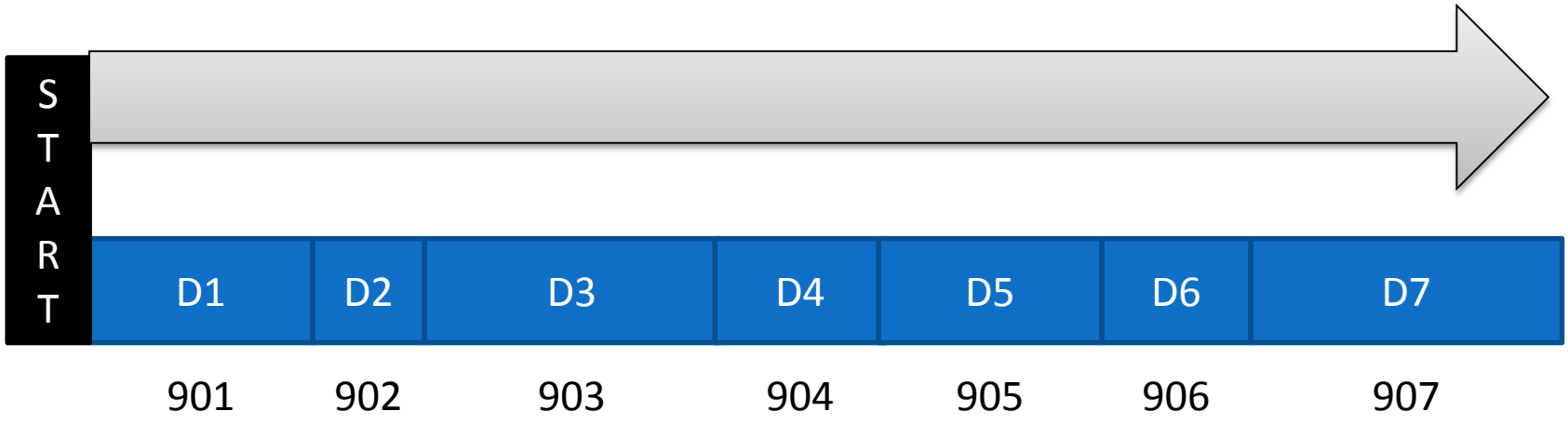
Terminology	Description
Access time	• RAM – time to address unit and transfer • Secondary storage – time to position R/W head over the location
Memory cycle time	Access time taken before starting the next access
Access technique	Method to access the memory's content

Access Method

- Sequential
- Direct
- Random
- Associative

Access Method - Sequential

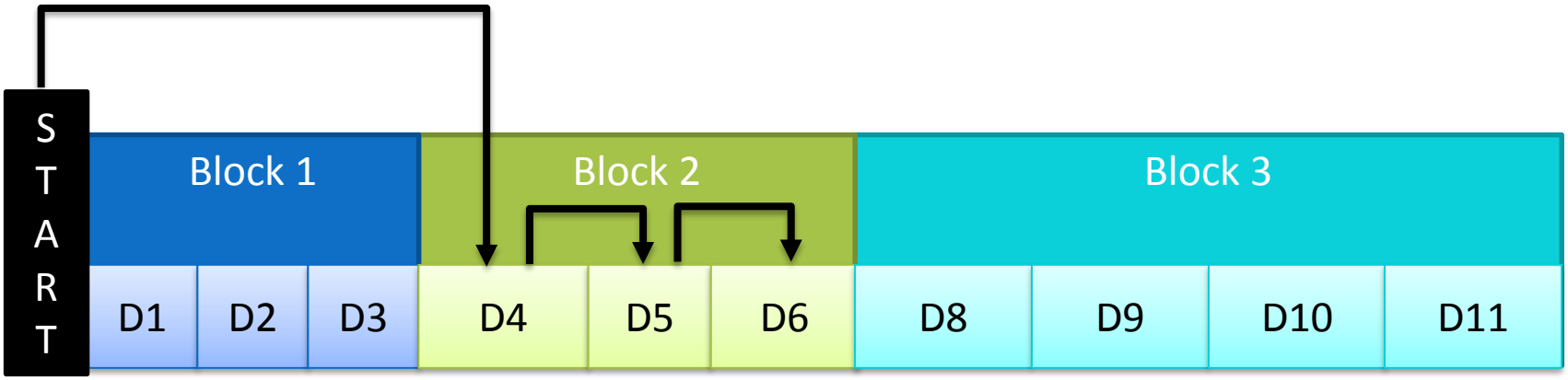
- Start at the beginning and read through in order
- Access time **depends** on location of data and previous location
- Example: tape



Access Method - Direct

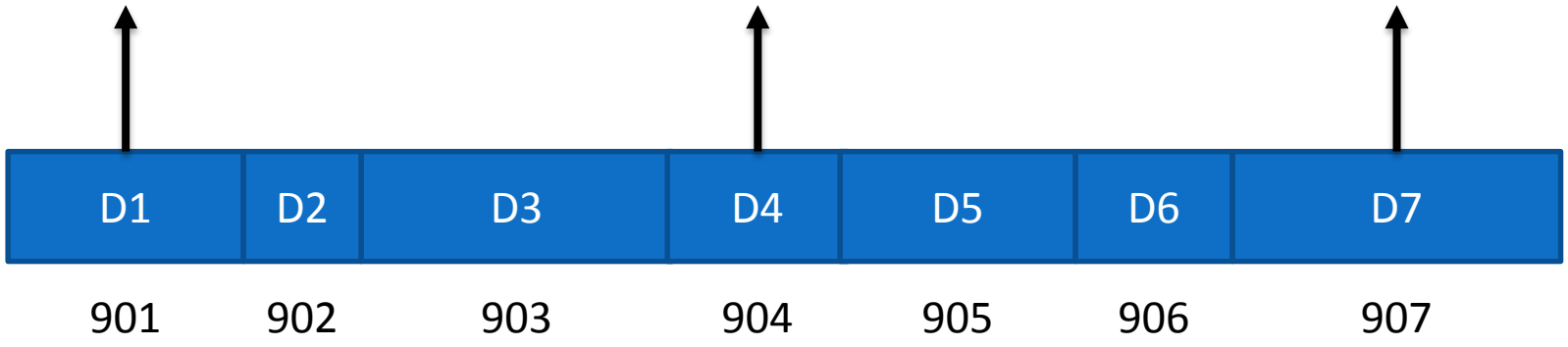
- Individual blocks have unique address
- Access is by jumping to vicinity plus sequential search
- Access time **depends** on location and previous location
- Example: Disk

Access Method – Direct (Cont.)



Access Method - Random

- Individual addresses identify locations exactly
- Access time is **independent** of location or previous access
- Example: RAM



Access Method - Associative

- Data is located by a comparison with contents of a portion of the store
- Access time is **independent** of location or previous access
- Example: cache

Memory Hierarchy

- Design objectives
  - Adequate capacity
  - Performance
  - Cost

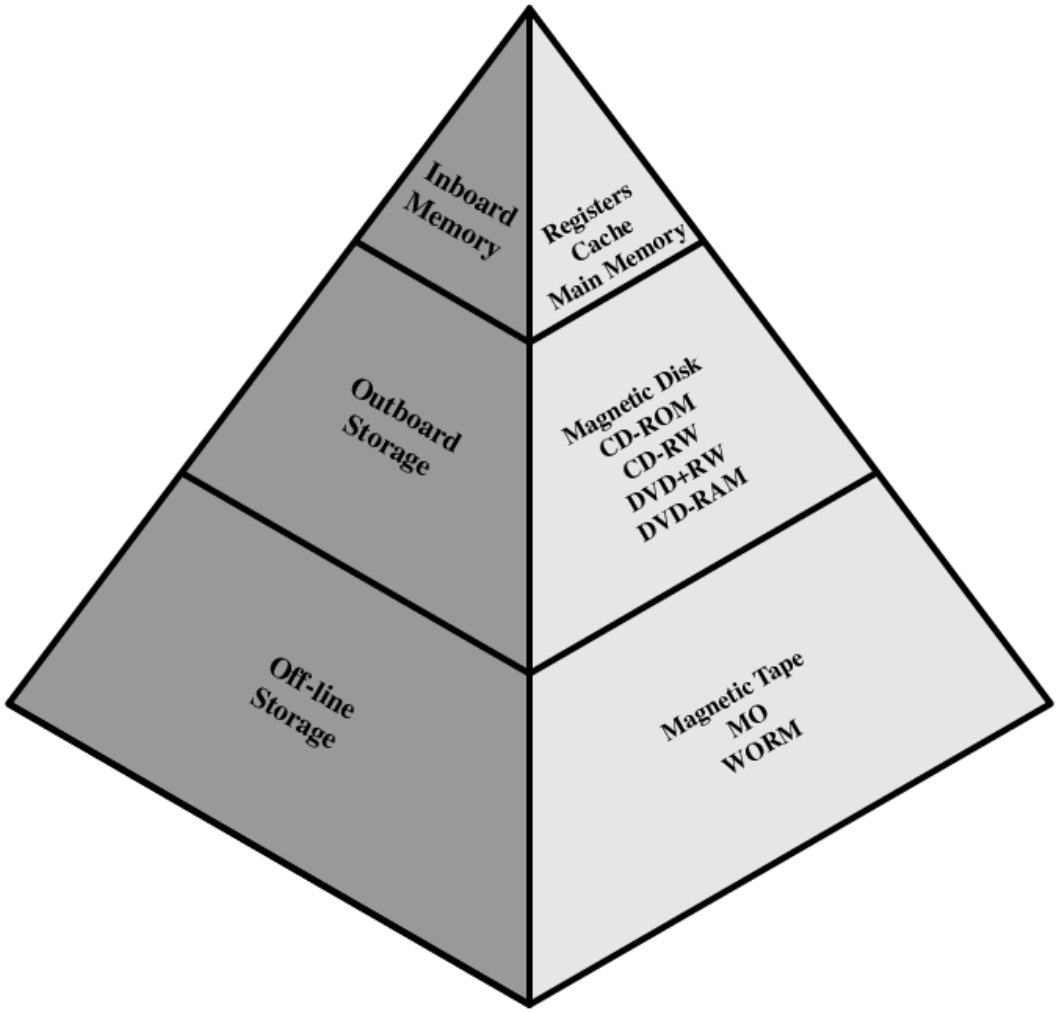
Memory Hierarchy

- How to achieve the design objectives?
  - Hierarchy of storage devices
  - Automatic space allocation methods - efficient
  - Virtual memory techniques
  - Design the memory and its related interconnection structure – compatible with processor speed

Memory Hierarchy - Basis

- Registers internal to the CPU - temporary data storage (small in number but very fast)
- External storage for data and programs (large and fast)
- External permanent storage (much larger and much slower)

Memory Hierarchy - Basis



Memory Hierarchy - Characteristic

- Consists of distinct “levels” of memory components
- Each level characterized by its size, access time, and cost per bit
- Each lower level in the hierarchy consists of modules of larger capacity, slower access time, and lower cost/bit

Common Memory Parameters

Memory Type	Technology	Size	Access Time
Cache	Semiconductor RAM	128-512 KB	10 ns
Main Memory	Semiconductor RAM	4-128 MB	50 ns
Magnetic Disk	Hard Disk	Gigabyte	10 ms,10 MB/sec
Optical Disk	CD-ROM	Gigabyte	300 ms, 600 KB/sec
Magnetic Tape	Tape	100s MB	Sec-min.,10MB/min

Hierarchy List

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

CACHE MEMORY

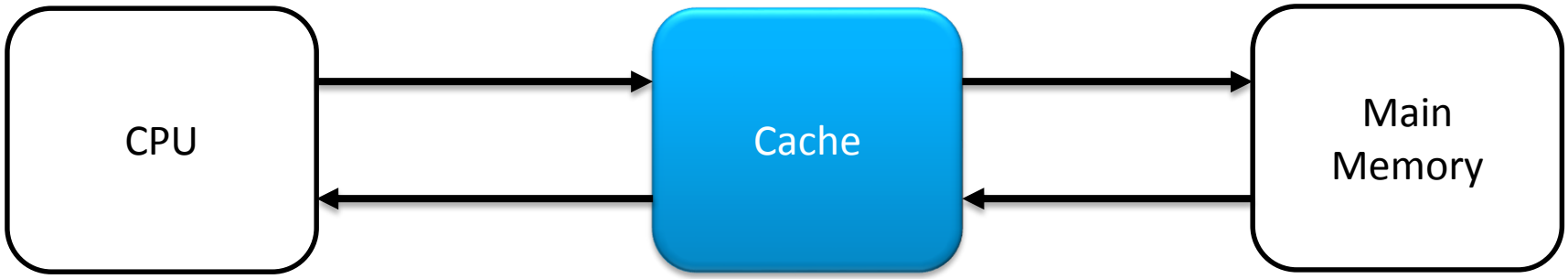
Cache Memory

- Critical component of the memory hierarchy
  - Size – smaller than main memory
  - Operates +- speed of the processor
  - Expensive than main memory
  - Stores copies of section of main memory

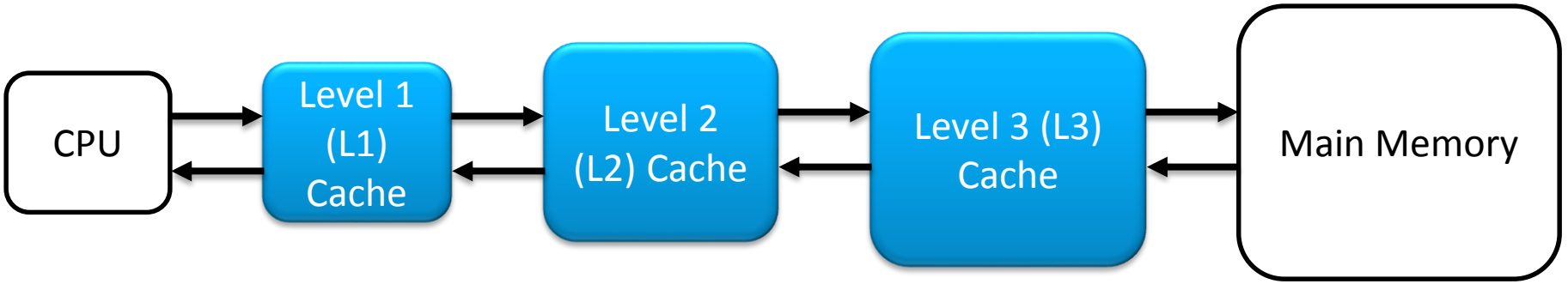
Cache

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

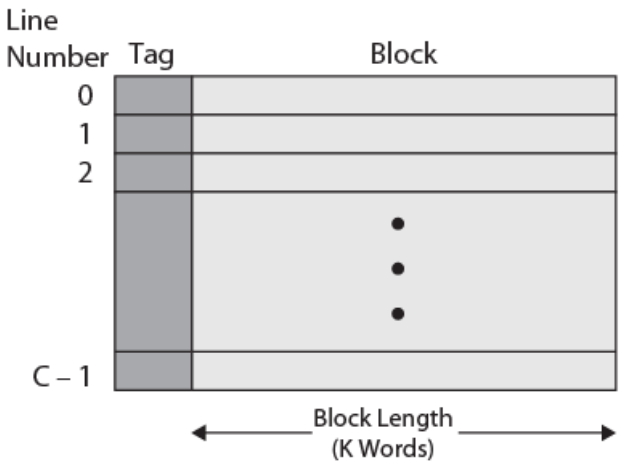
Cache and Main Memory – Single Cache



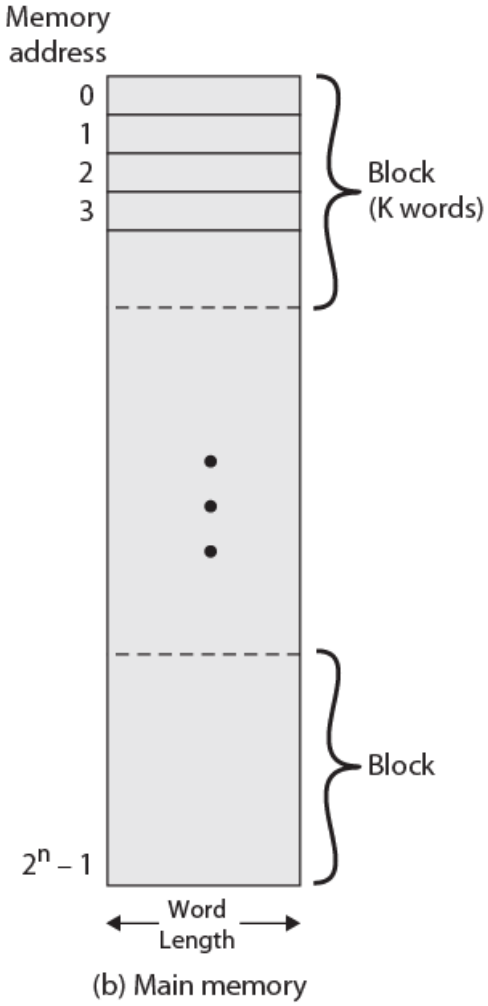
Cache and Main Memory – Three-level Cache Organization



Cache/Main Memory Structure



(a) Cache

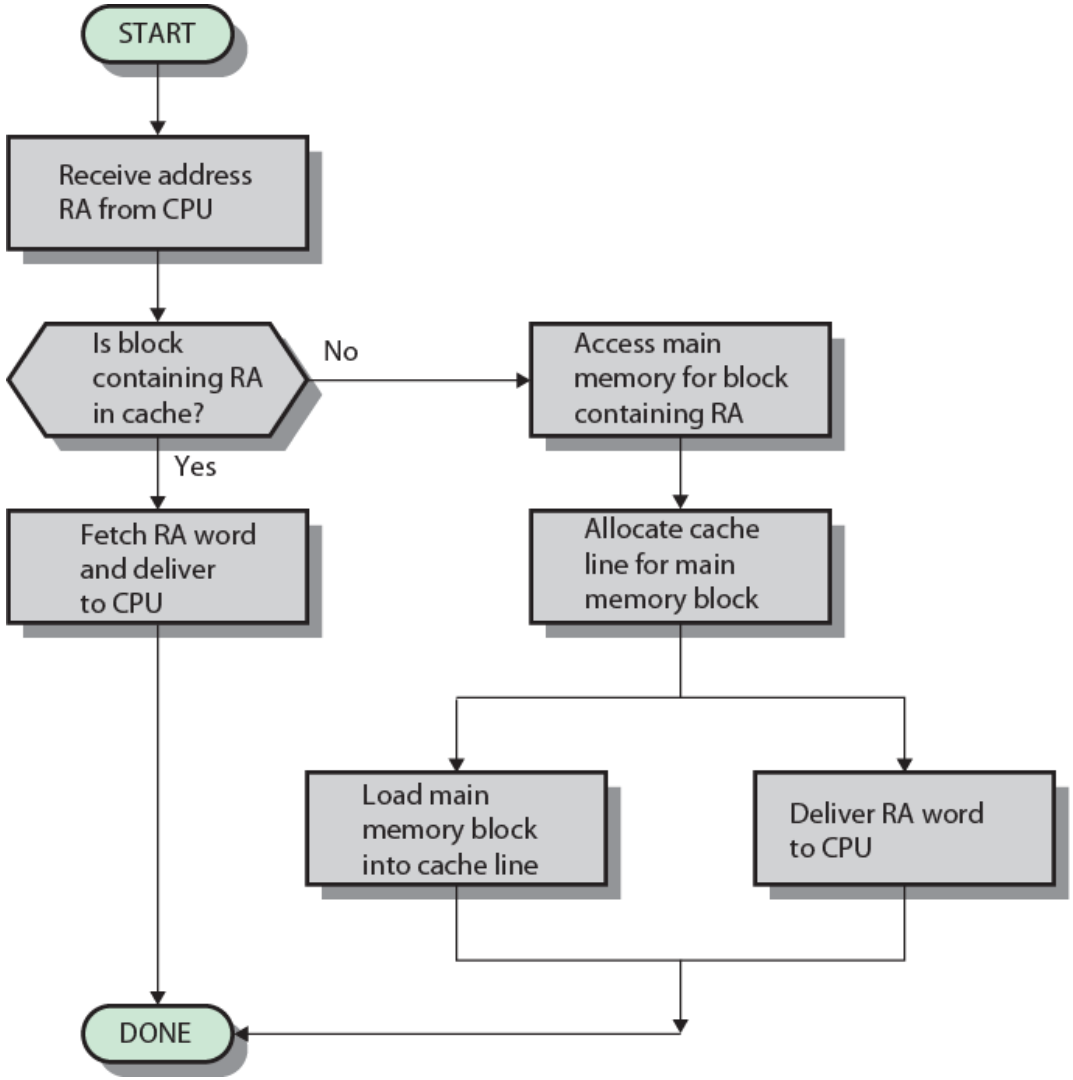


(b) Main memory

Cache Operation - Overview

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

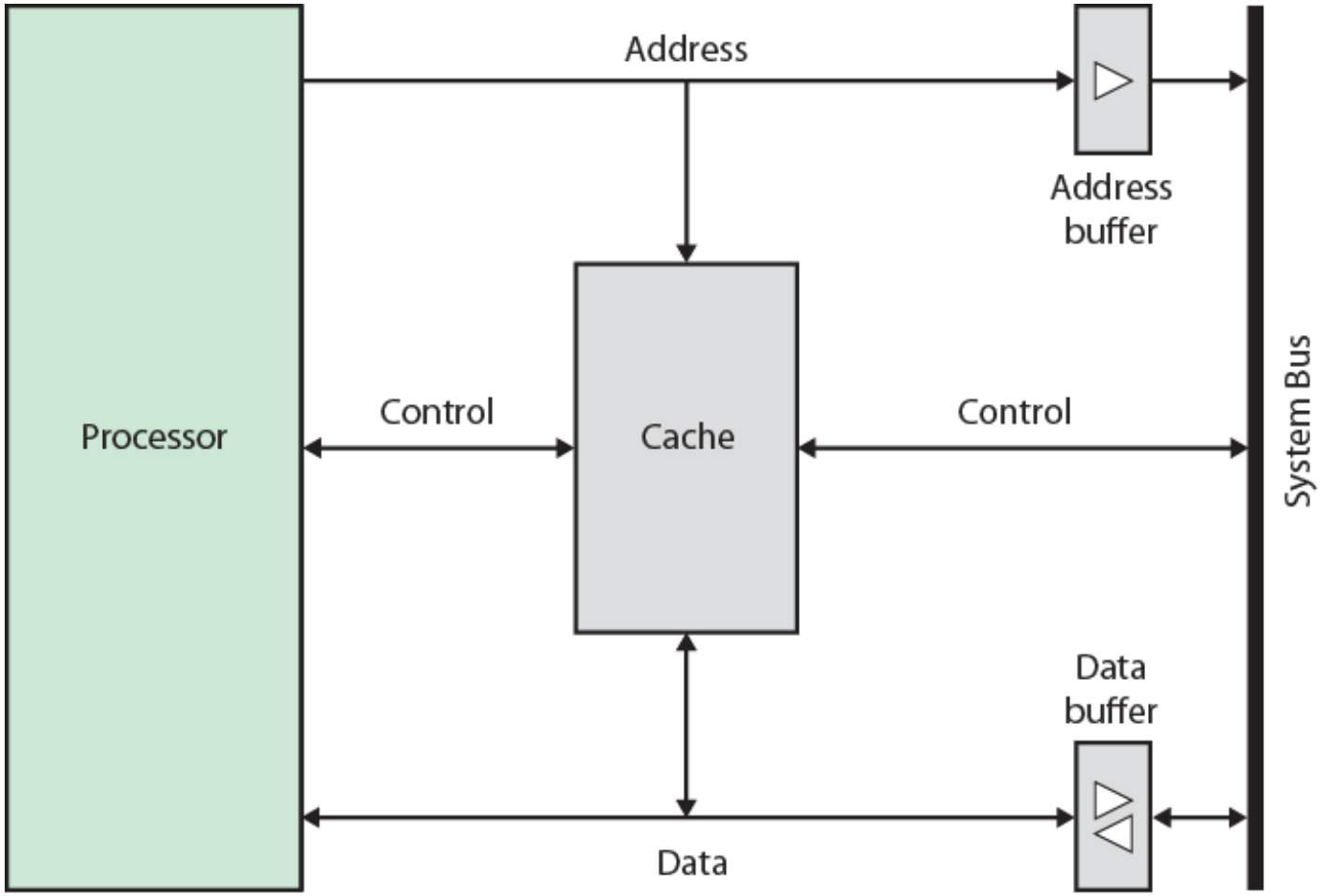
Cache Read Operation



Locality of Reference

- Memory references made by the processor, for both instructions and data, tend to cluster together
  - Instruction loops, subroutines
  - Data arrays, tables
- Keep these clusters in high speed memory to reduce the average delay in accessing data
- Over time, the clusters being referenced will change -- memory management must deal with this

Typical Cache Organization

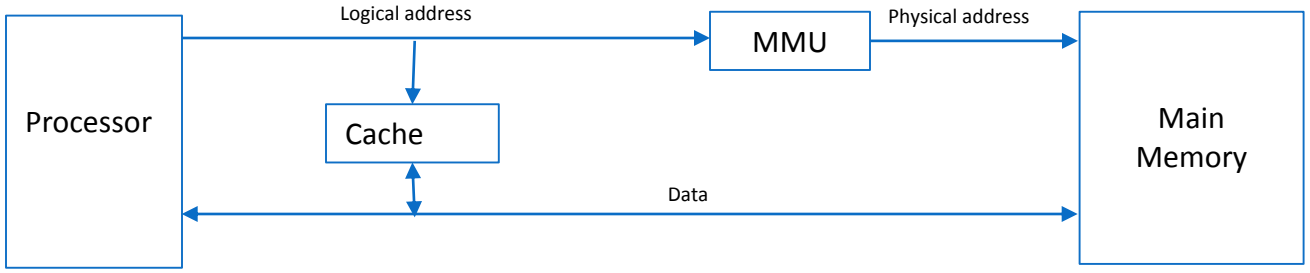


Elements of Cache Design

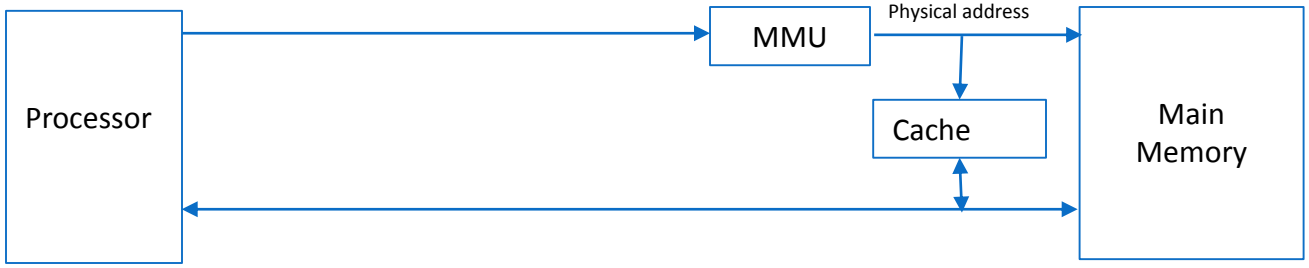
- Cache address
  - Logical
  - Physical
- Cache size
- Mapping Function
  - Direct
  - Associative
  - Set Associative
- Line Size
- Number of caches
  - Single or two level
  - Unifies or split
- Replacement algorithm
  - Least recently used (LRU)
  - First in first out (FIFO)
  - Lease frequently used (LFU)
  - Random
- Write policy
  - Write through
  - Write back
  - Write once

Cache Address

- Location
  - Between processor and virtual Memory Management Unit (MMU)
  - Between MMU and main memory
- Logical
- Physical
- MMU function – to translate each virtual address into physical address in main memory.



Logical Cache



Physical Cache

Cache Address - Logical

- 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 – changing from one thread/process to another.

Cache Address - Physical

- Stores data using main memory physical address

Cache Size

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

Cache Size on Some Processors

Processor	Type	Year of Introduction	L1 cache	L2 cache	L3 cache
IBM 360/85	Mainframe	1968	16 to 32 KB	—	—
PDP-11/70	Minicomputer	1975	1 KB	—	—
VAX 11/780	Minicomputer	1978	16 KB	—	—
IBM 3033	Mainframe	1978	64 KB	—	—
IBM 3090	Mainframe	1985	128 to 256 KB	—	—
Intel 80486	PC	1989	8 KB	—	—
Pentium	PC	1993	8 KB/8 KB	256 to 512 KB	—
PowerPC 601	PC	1993	32 KB	—	—
PowerPC 620	PC	1996	32 KB/32 KB	—	—
PowerPC G4	PC/server	1999	32 KB/32 KB	256 KB to 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 KB	8 MB	—
Pentium 4	PC/server	2000	8 KB/8 KB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 KB/32 KB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 KB	2 MB	—
Itanium	PC/server	2001	16 KB/16 KB	96 KB	4 MB
SGI Origin 2001	High-end server	2001	32 KB/32 KB	4 MB	—
Itanium 2	PC/server	2002	32 KB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 KB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 KB/64 KB	1MB	—

Mapping Function

- Determines how a block from the main memory is placed in the cache
- Mapping techniques;
  - Direct
  - Associative
  - Set associative

To be used as sample

- 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

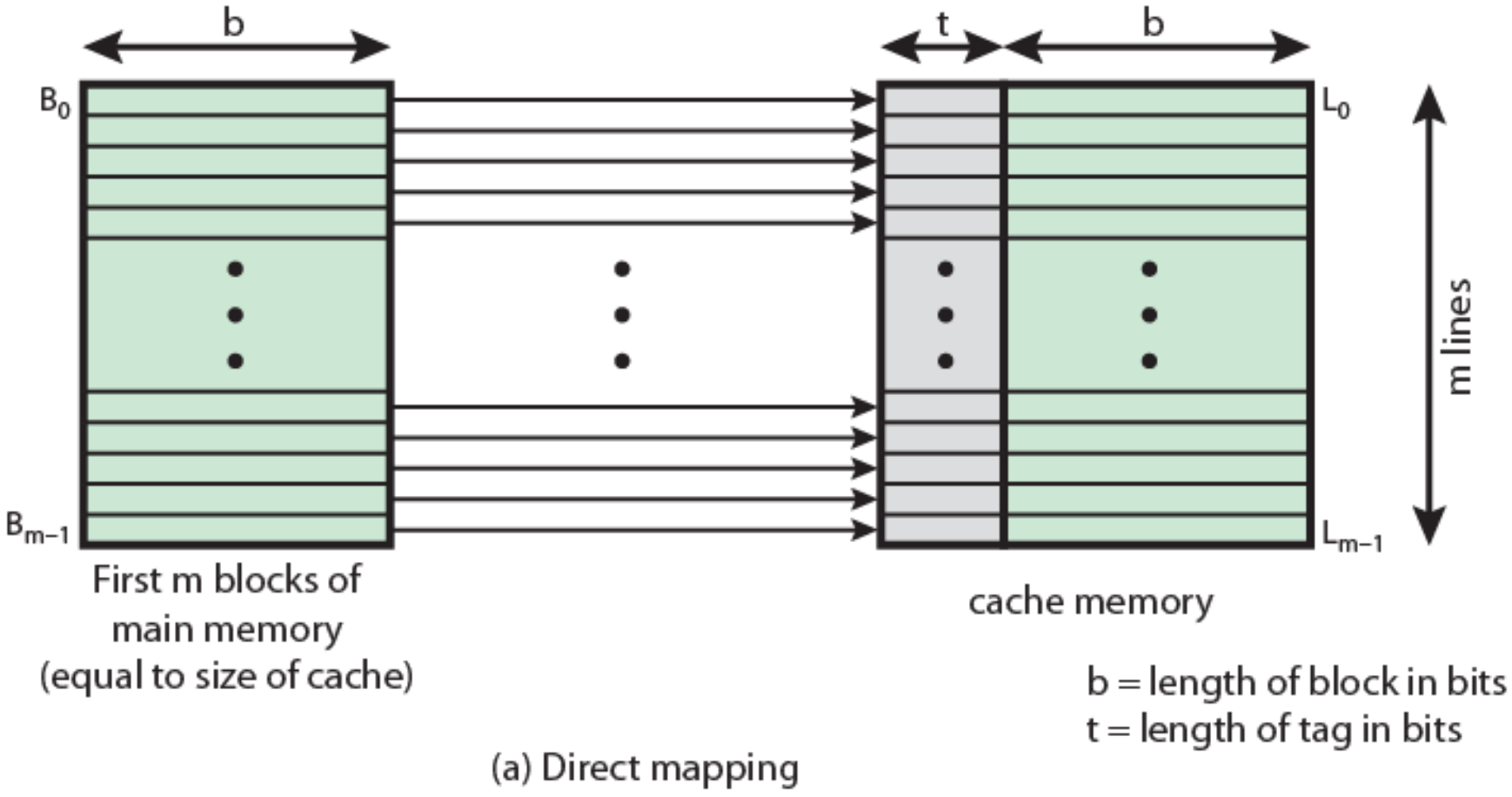
- Each block of main memory maps to only one cache line
- Address is in two parts
  - Least Significant *w* bits identify unique word
  - Most Significant *s* bits specify one memory block
    - The MSBs are split into a cache line field *r* and a tag of *s* - *r* (most significant)

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

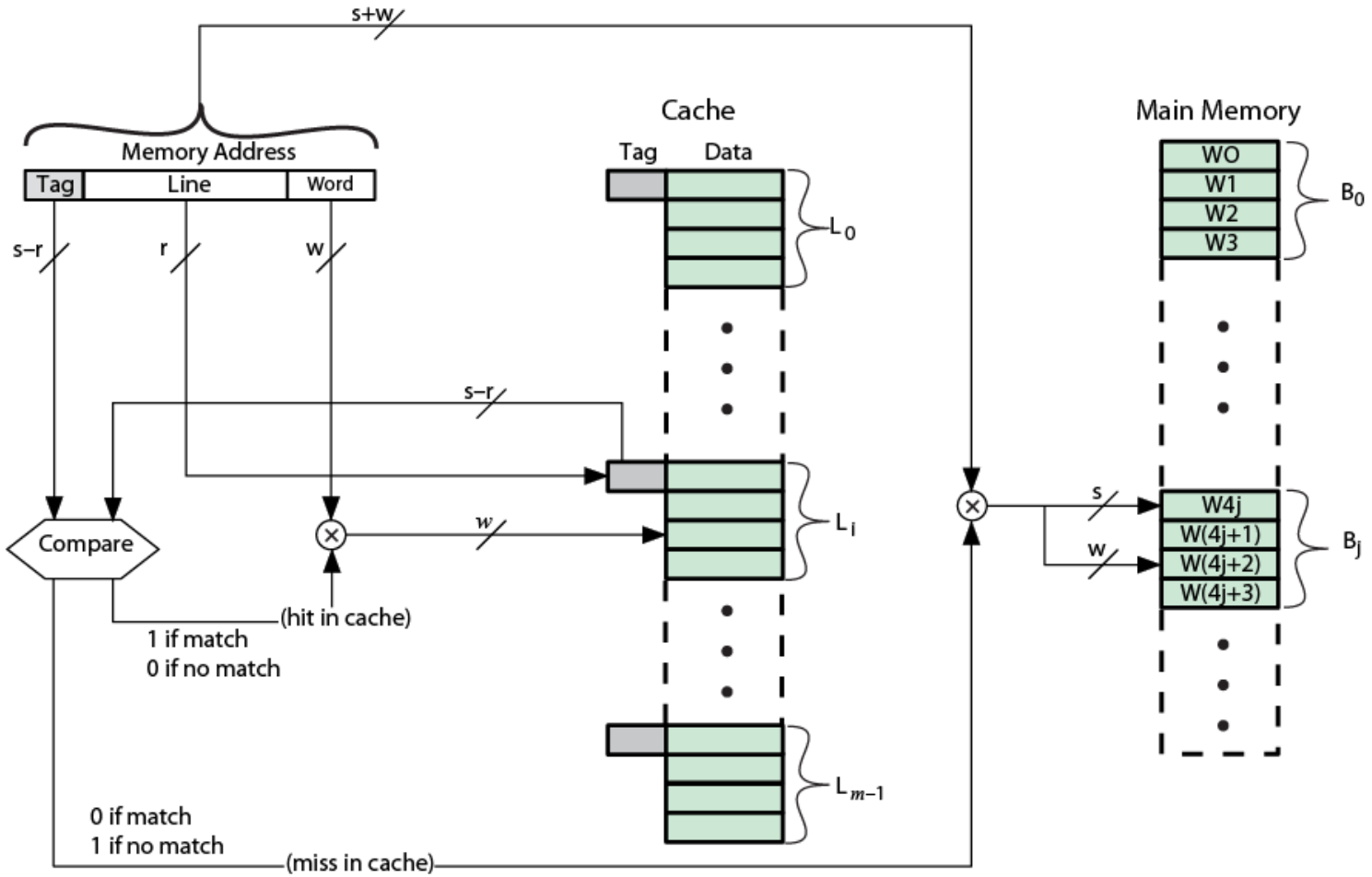


(a) Direct mapping

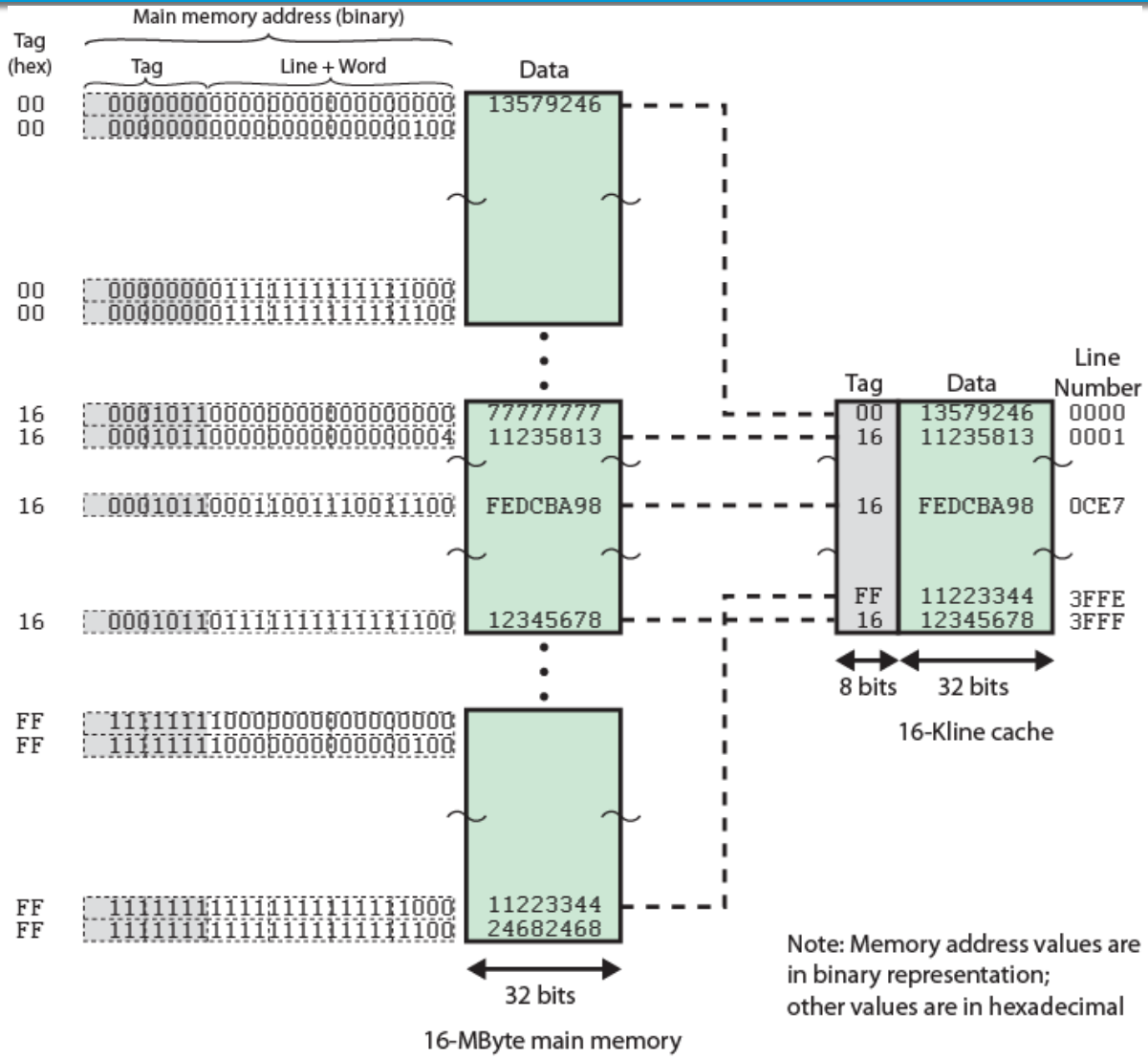
Direct Mapping Cache Line Table

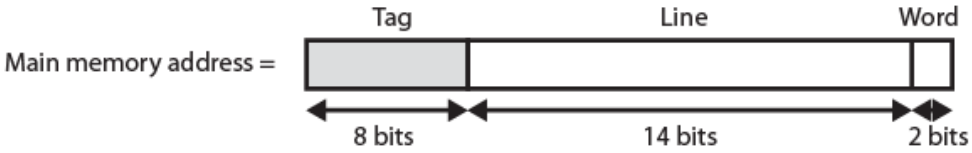
Cache line	Main Memory blocks assigned
0	0, m, 2m, 3m…2^s-m
1	1,m+1, 2m+1…2^s-m+1
…	
m-1	m-1, 2m-1,3m-1…2^s-1

Direct Mapping Cache Organization



Direct Mapping Example





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

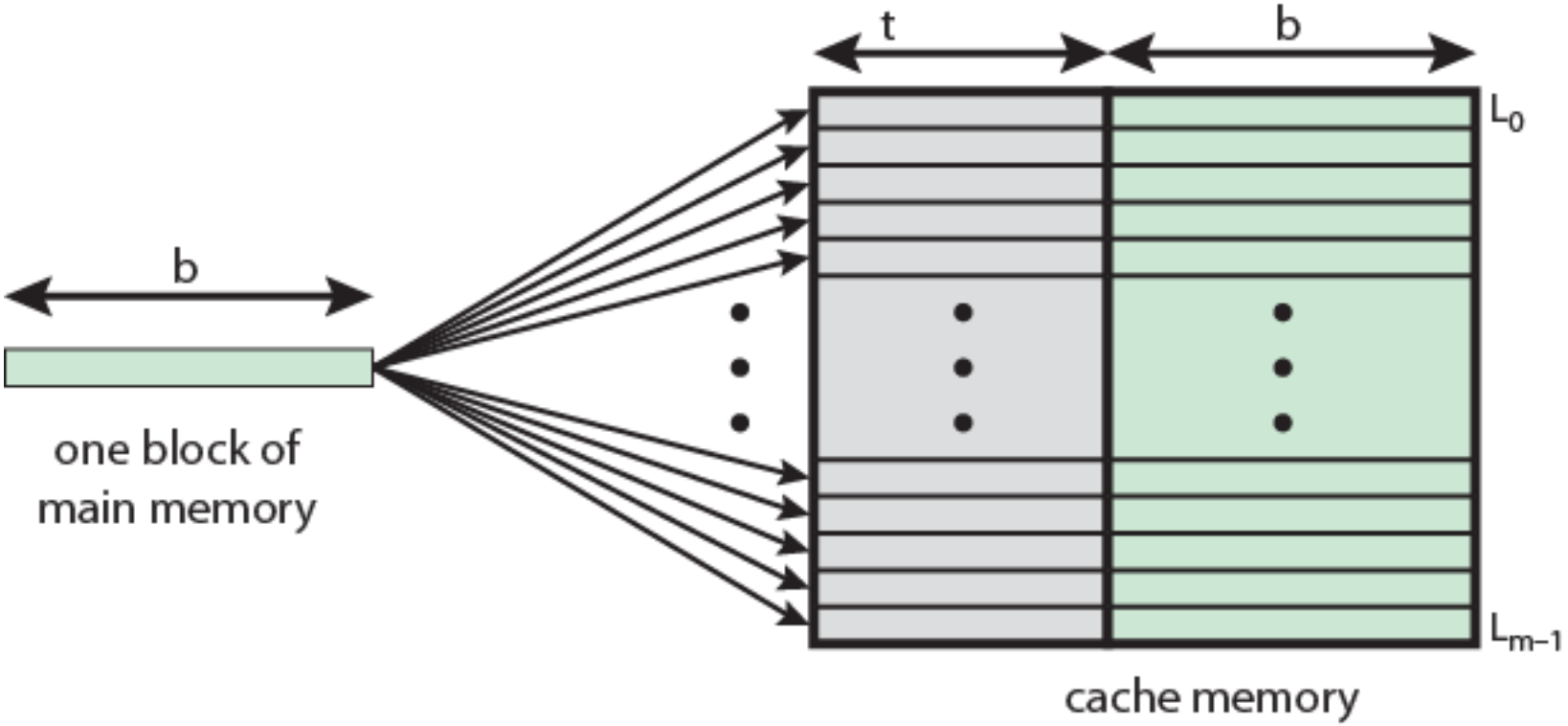
Direct Mapping Pros & Cons

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

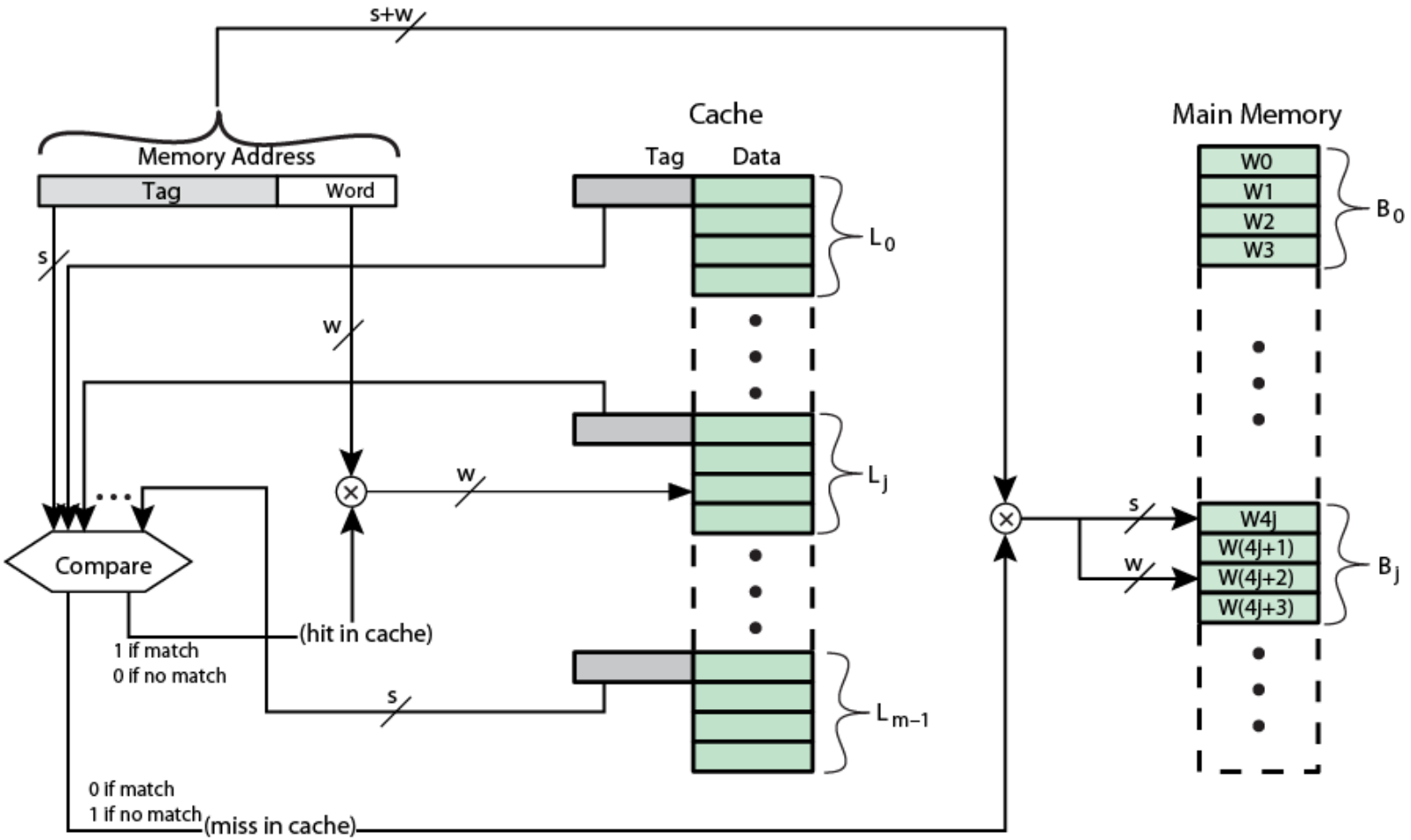
Associative Mapping

- 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

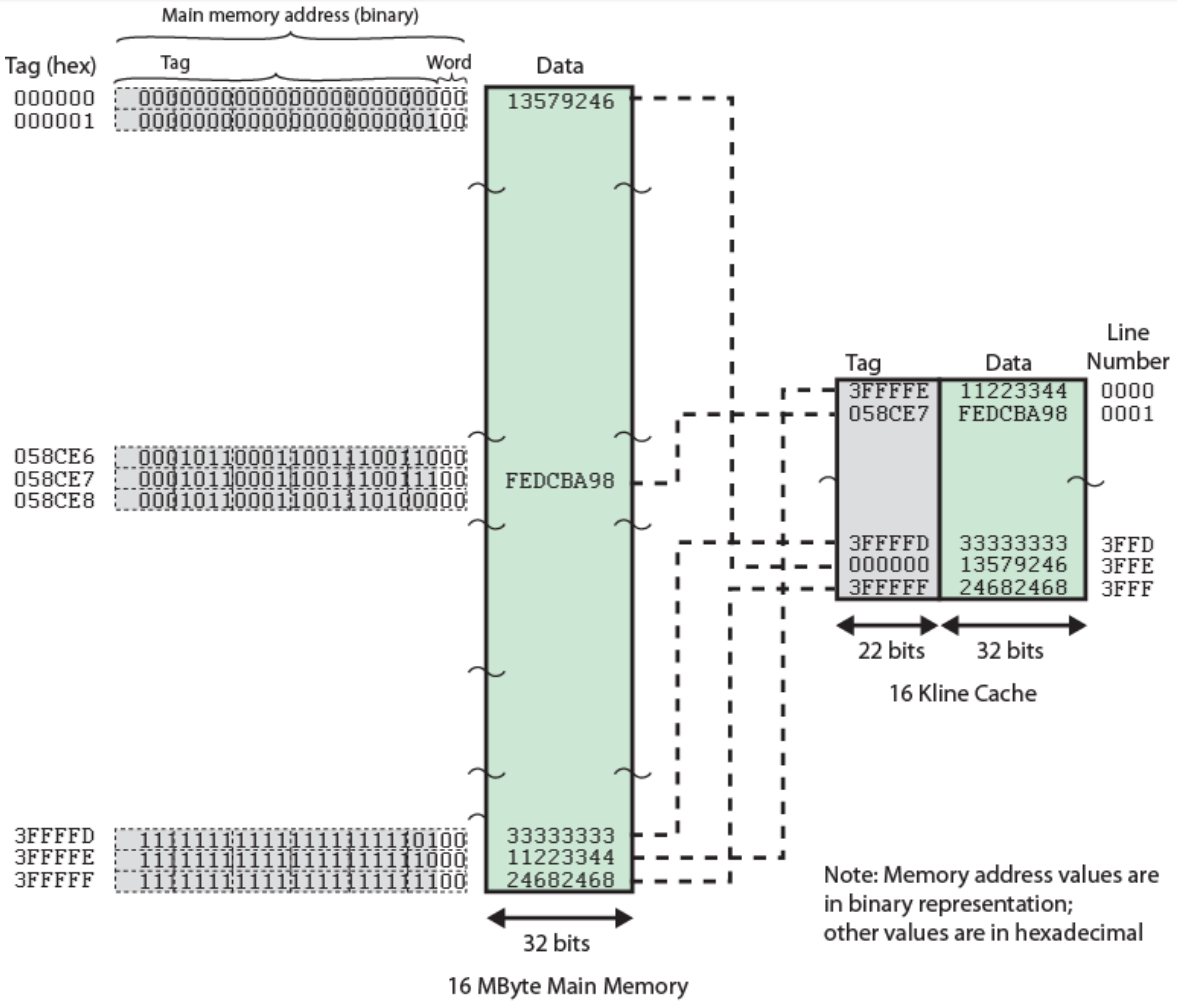
Associative Mapping from Cache to Main Memory



Fully Associative Cache Organization



Associative Mapping Example



Associative Mapping Address Structure

Tag	Word *w*
22	2

- 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	FFFFFC	24682468	3FFF

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

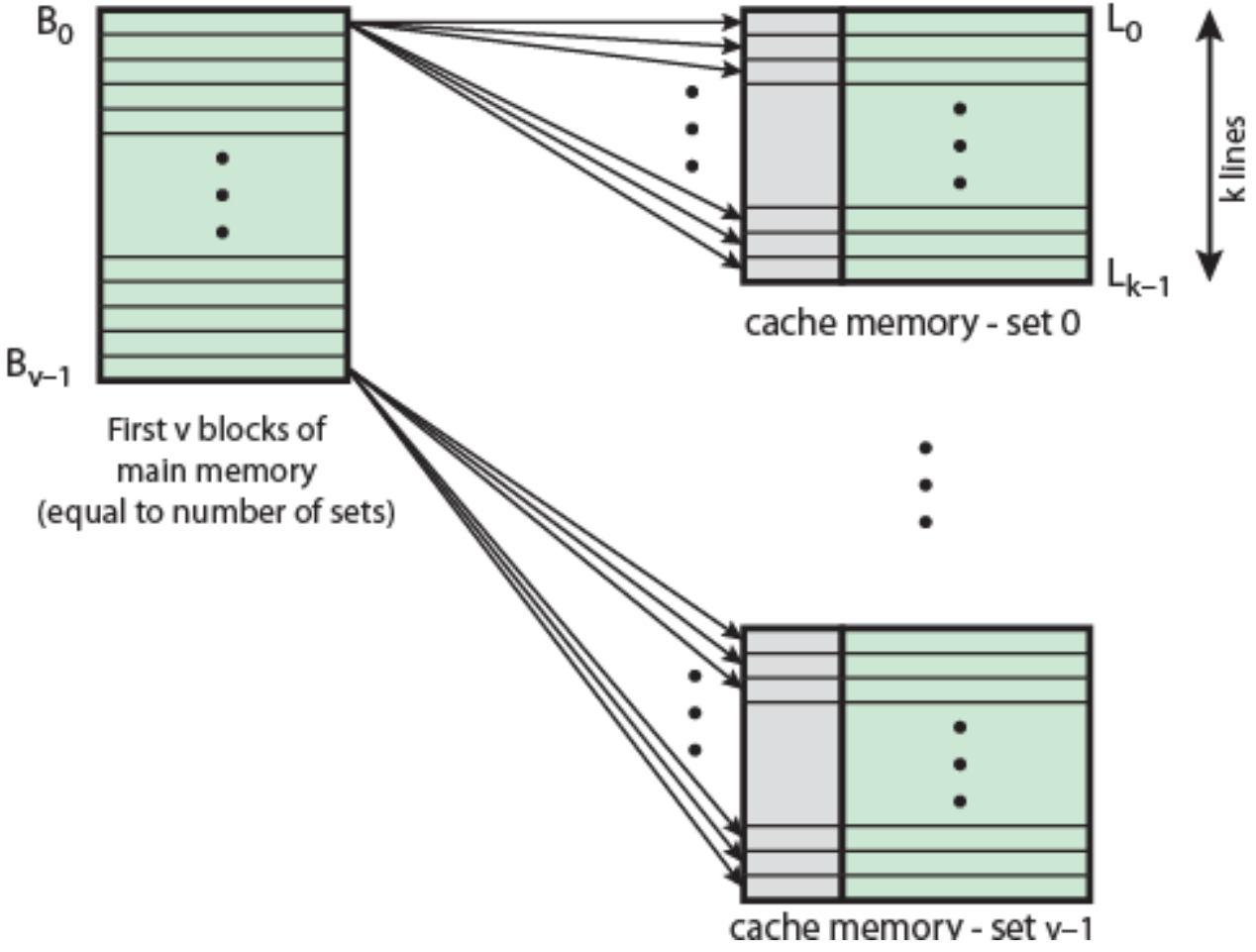
Set Associative Mapping

- 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 ways associative mapping
  - A given block can be in one of 2 lines in only one set

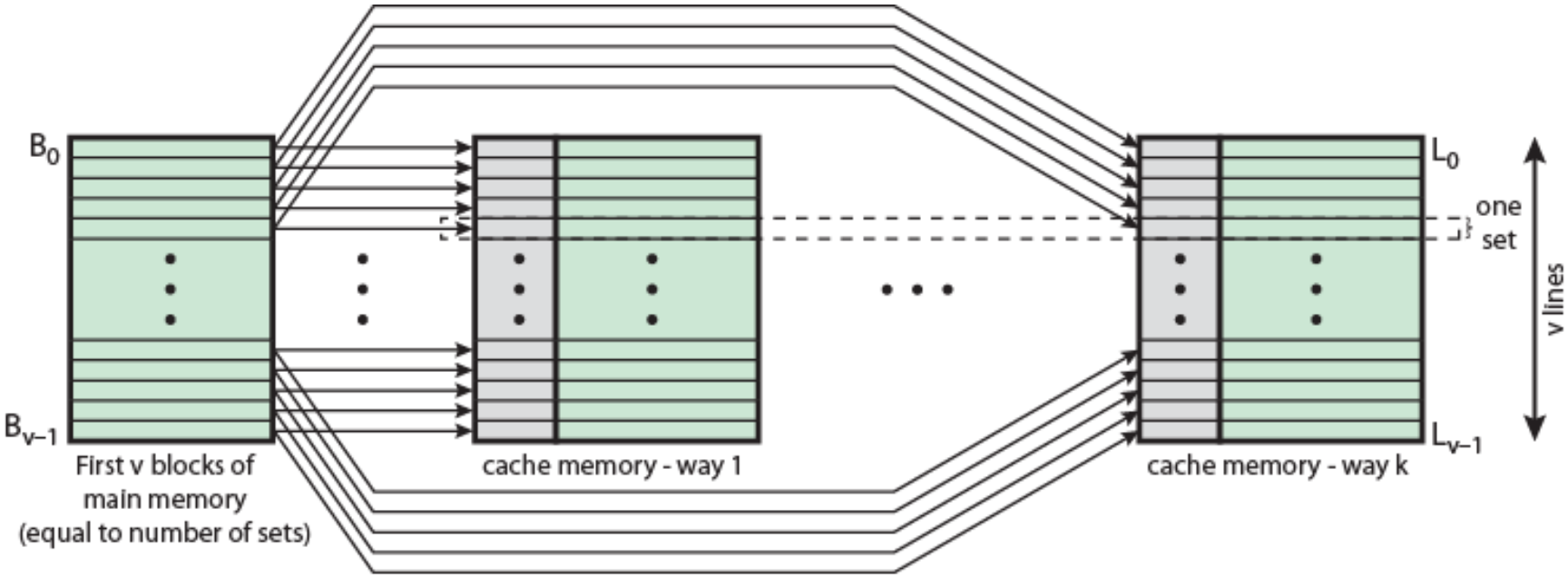
Set Associative Mapping Example

- 13 bit set number
- Block number in main memory is modulo 213
- 000000, 00A000, 00B000, 00C000 … map to same set

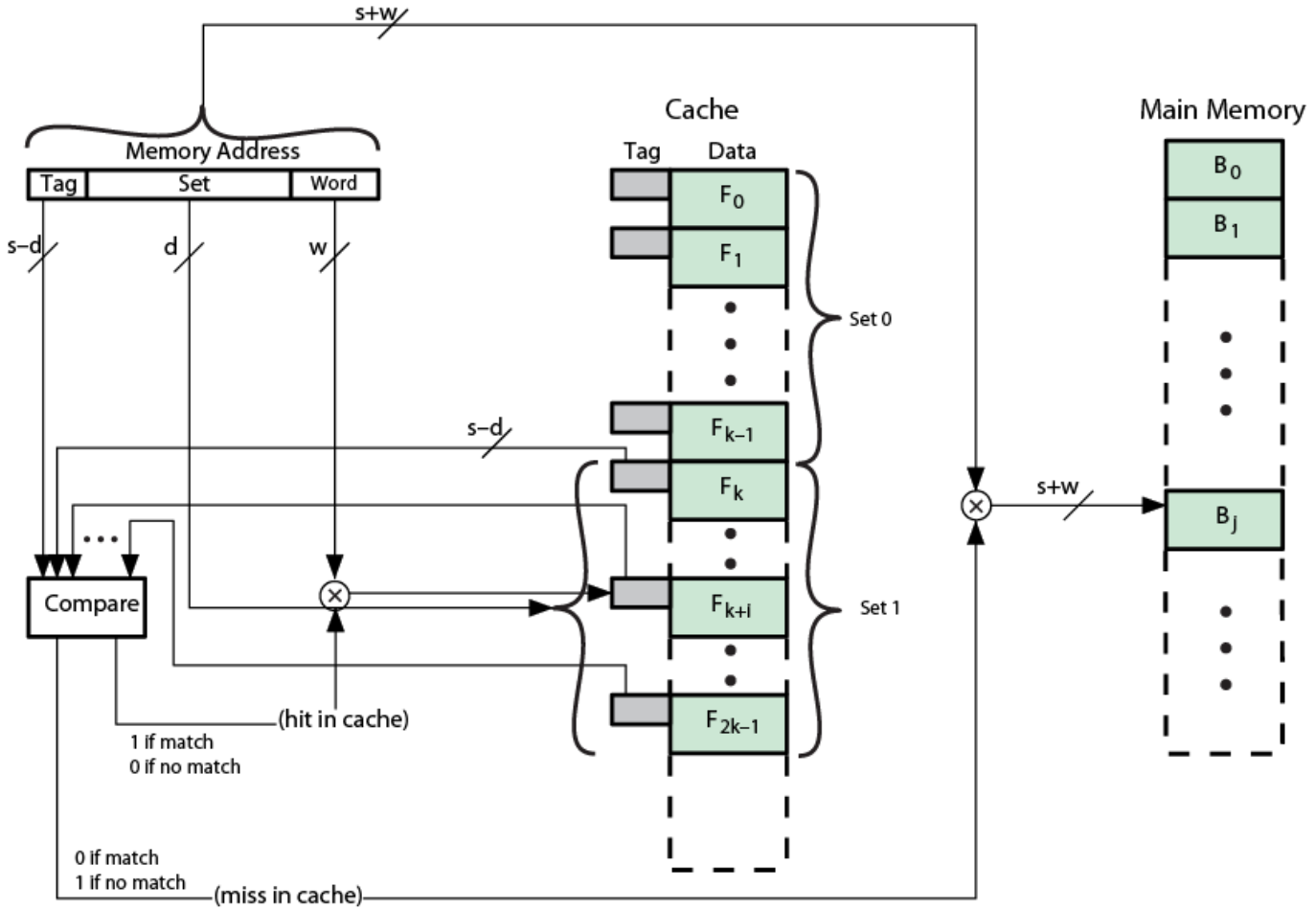
Mapping from Main Memory to Cache: *v Associative*



Mapping from Main Memory to Cache: *k-way Associative*



k-way Set Associative Cache Organization



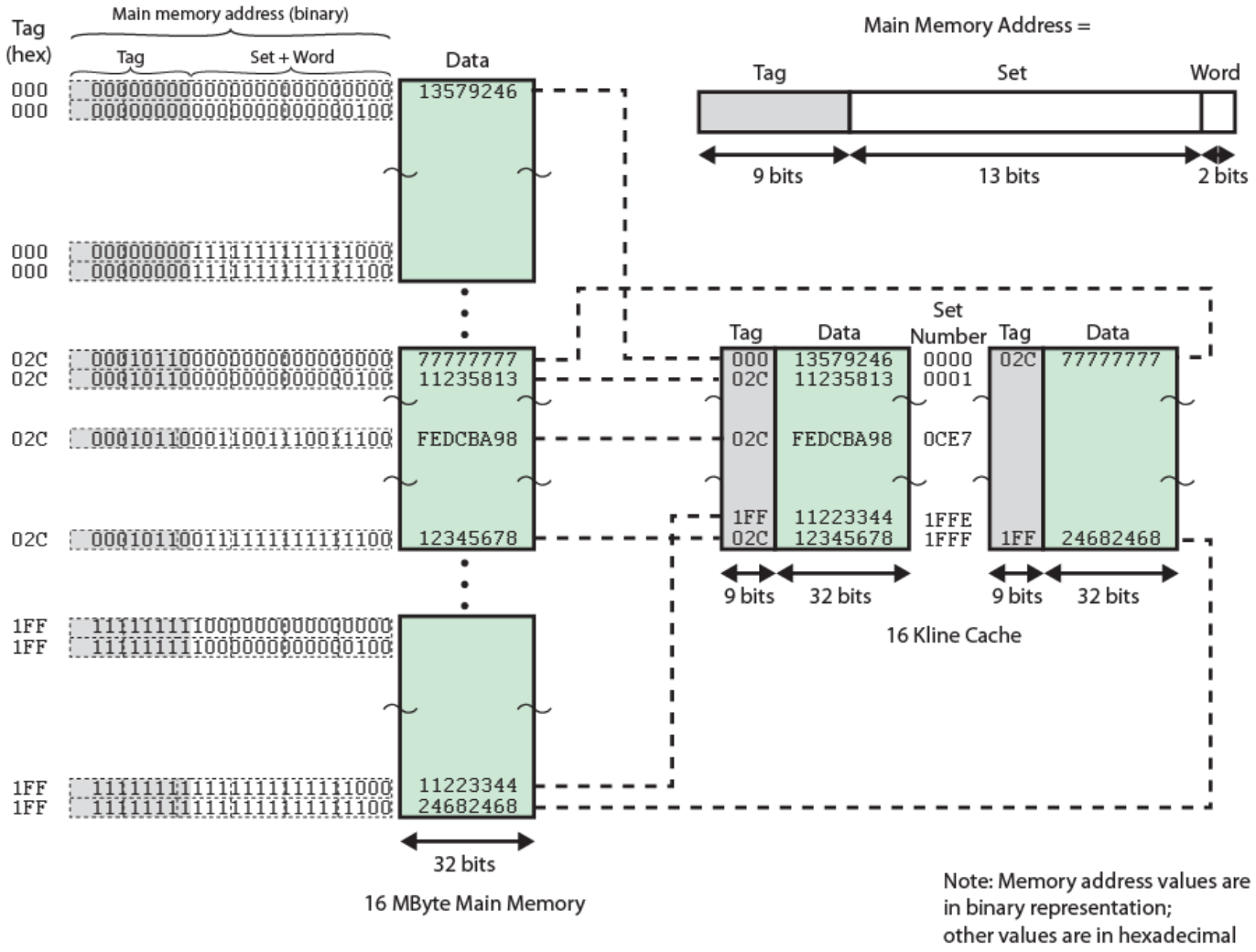
Set Associative Mapping Address Structure

Tag	Set	Word *w*
9	13	2

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

Address	Tag	Data	Set number
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 = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = 2^{d}
- 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

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 Algorithm

Replacement Algorithm: Direct Mapping

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

Replacement Algorithm: 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
- Techniques;
  - Write through
  - Write back

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 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
- % of memory references are writes = 15%

Additional Reference

- William Stallings, Computer Organization and Architecture: Designing for Performance, 8th. Edition, Prentice-Hall Inc., 2010