CSE 213
Computer Architecture

Lecture 4: Cache Memory

Military Institute of Science and Technology

Outline

- Characteristics of Memory Systems
- The Memory Hierarchy
- Cache Memory Principles
- Cache Memory Design
- Pentium 4 Cache Organization

Key Characteristics of Computer Memory Systems

Location
- Internal (e.g., processor registers, cache, main memory)
- External (e.g., optical disks, magnetic disks, tapes)

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

Physical Type
- Semiconductor
- Magnetic
- Optical
- Magneto-optical

Physical Characteristics
- Volatile/nonvolatile
- Erasable/nonerasable

Organization
- Memory modules

Table 4.1 Key Characteristics of Computer Memory Systems

+

Characteristics of Memory Systems

- **Location**
  - Refers to whether memory is internal and external to the computer
  - Internal memory is often equated with main memory
  - Processor requires its own local memory, in the form of registers
  - Cache is another form of internal memory
  - External memory consists of peripheral storage devices that are accessible to the processor via I/O controllers
- **Capacity**
  - Memory is typically expressed in terms of bytes
- **Unit of transfer**
  - For internal memory the unit of transfer is equal to the number of electrical lines into and out of the memory module

Method of Accessing Units of Data

Sequential access

- Memory is organized into units of data called records
- Access must be made in a specific linear sequence
- Access time is variable
- Example: Tape units

Direct access

- Involves a shared read-write mechanism
- Individual blocks or records have a unique address based on physical location
- Access time is variable
- Example: Disk units

Random access

- Each addressable location in memory has a unique, physically wired-in addressing mechanism
- The time to access a given location is independent of the sequence of prior accesses and is constant
- Any location can be selected at random and directly addressed and accessed
- Example: Main memory and some cache systems

Associative

- A word is retrieved based on a portion of its contents rather than its address
- Each location has its own addressing mechanism and retrieval time is constant independent of location or prior access patterns
- Cache memories may employ associative access

Performance

The two most important characteristics of memory – Capacity & Performance

Three performance parameters are used:

Access time (latency)

- For random-access memory it is the time it takes to perform a read or write operation
- For non-random-access memory it is the time it takes to position the read-write mechanism at the desired location

Memory cycle time

- Access time plus any additional time required before second access can commence
- Additional time may be required for transients to die out on signal lines or to regenerate data if they are read destructively
- Concerned with the system bus, not the processor

Transfer rate

- The rate at which data can be transferred into or out of a memory unit
- For random-access memory it is equal to 1/(cycle time)

Physical Type

- The most common forms are:
  - Semiconductor memory
  - Magnetic surface memory
  - Optical
  - Magneto-optical

Physical characteristics

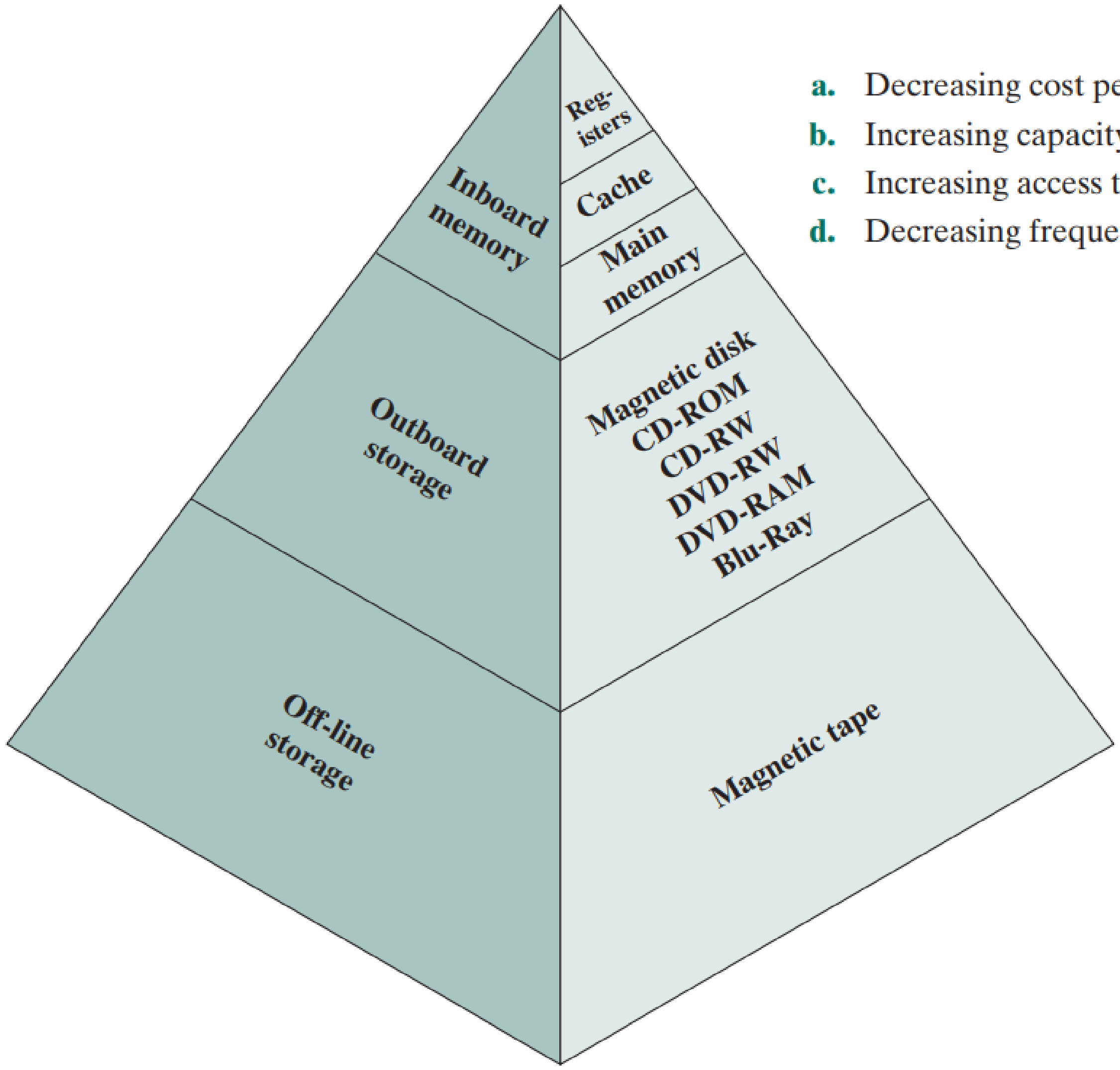
- Volatile memory
  - Information decays naturally or is lost when electrical power is switched off
- Nonvolatile memory
  - Once recorded, information remains without deterioration until deliberately changed
  - No electrical power is needed to retain information
  - Magnetic-surface memories - are nonvolatile
  - Semiconductor memory - may be either volatile or nonvolatile
- Nonerasable memory
  - Cannot be altered, except by destroying the storage unit
  - Semiconductor memory of this type is known as read-only memory (ROM)

Organization: For random-access memory the organization is a key design issue. Organization refers to the physical arrangement of bits to form words

+
Memory Hierarchy

- Design constraints on a computer’s memory can be summed up by three questions:
  - How much, how fast, how expensive
- There is a trade-off among capacity, access time, and cost
  - Faster access time, greater cost per bit
  - Greater capacity, smaller cost per bit
  - Greater capacity, slower access time
- The way out of the memory dilemma is not to rely on a single memory component or technology, but to employ a memory hierarchy

Memory Hierarchy - Diagram



a. Decreasing cost per bit;
b. Increasing capacity;
c. Increasing access time;
d. Decreasing frequency of access of the memory by the processor.

Figure 4.1 The Memory Hierarchy

Locality of Reference – (d) decreasing the frequency of access

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

- **Locality of reference**, also known as the **principle of locality**, is a phenomenon describing the same value, or related storage locations, being frequently accessed. There are two basic types of reference locality – temporal and spatial locality.

- Temporal locality. If at one point in time a particular memory location is referenced, then it is likely that the **same location** will be referenced again in the near future.

- Spatial locality. If a particular memory location is referenced at a particular time, then it is likely that **nearby memory locations** will be referenced in the near future.

+
Operation of Two-Level Memory

❑ The locality property can be exploited in the formation of a two-level memory. The upper-level memory (M1) is smaller, faster, and more expensive (per bit) than the lower-level memory (M2).

❑ M1 is used as a temporary store for part of the contents of the larger M2.

❑ When a memory reference is made, an attempt is made to access the item in M1. If this succeeds, then a quick access is made. If not, then a block of memory locations is copied from M2 to M1 and the access then takes place via M1.

❑Because of locality, once a block is brought into M1, there should be a number of accesses to locations in that block, resulting in fast overall service.

+

Operation of Two-Level Memory

To express the average time to access an item, we must consider not only the speeds of the two levels of memory, but also the probability that a given reference can be found in M1. We have

$$T_s = H \times T_1 + (1-H) \times (T_1 + T_2)$$

where

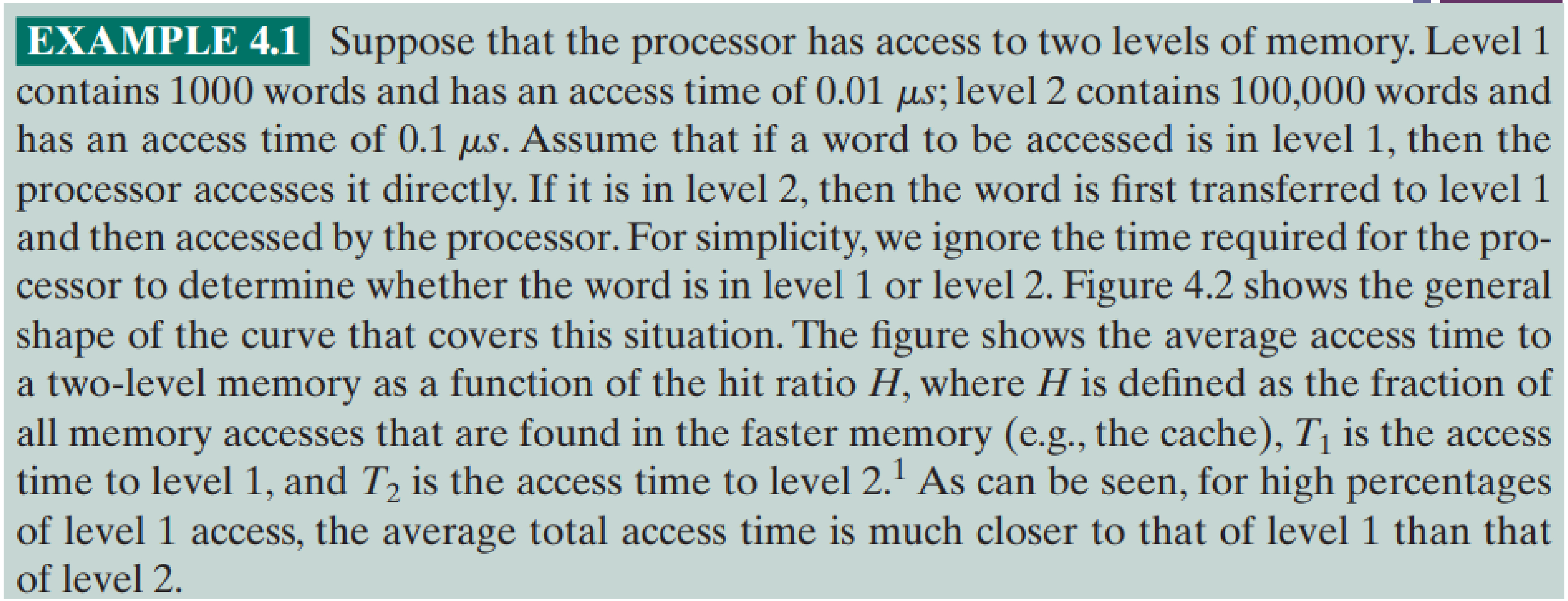
Ts = average (system) access time

T1 = access time of M1 (e.g., cache, disk cache)

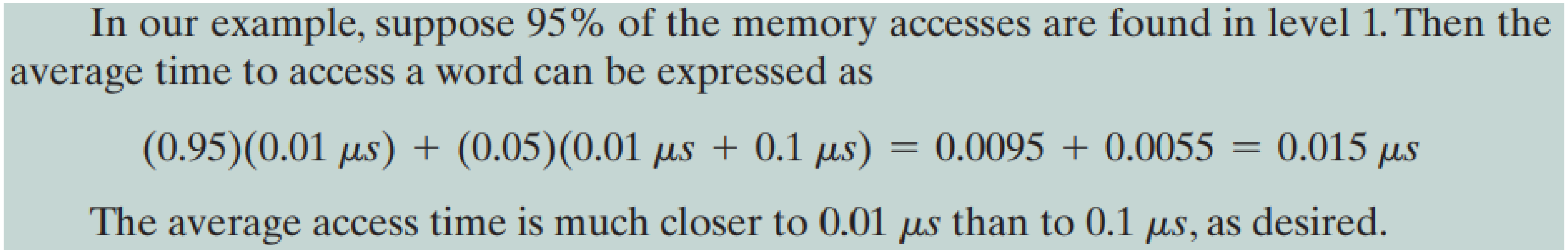
T2 = access time of M2 (e.g., main memory, disk)

H = hit ratio (fraction of time reference is found in M1)

Example-4.1



EXAMPLE 4.1 Suppose that the processor has access to two levels of memory. Level 1 contains 1000 words and has an access time of 0.01 μs; level 2 contains 100,000 words and has an access time of 0.1 μs. Assume that if a word to be accessed is in level 1, then the processor accesses it directly. If it is in level 2, then the word is first transferred to level 1 and then accessed by the processor. For simplicity, we ignore the time required for the processor to determine whether the word is in level 1 or level 2. Figure 4.2 shows the general shape of the curve that covers this situation. The figure shows the average access time to a two-level memory as a function of the hit ratio H, where H is defined as the fraction of all memory accesses that are found in the faster memory (e.g., the cache), T_1 is the access time to level 1, and T_2 is the access time to level 2.[1] As can be seen, for high percentages of level 1 access, the average total access time is much closer to that of level 1 than that of level 2.



In our example, suppose 95% of the memory accesses are found in level 1. Then the average time to access a word can be expressed as

$$(0.95)(0.01\ \mu s) + (0.05)(0.01\ \mu s + 0.1\ \mu s) = 0.0095 + 0.0055 = 0.015\ \mu s$$

The average access time is much closer to 0.01 μs than to 0.1 μs, as desired.

+

CACHE MEMORY PRINCIPLES

+
Cache

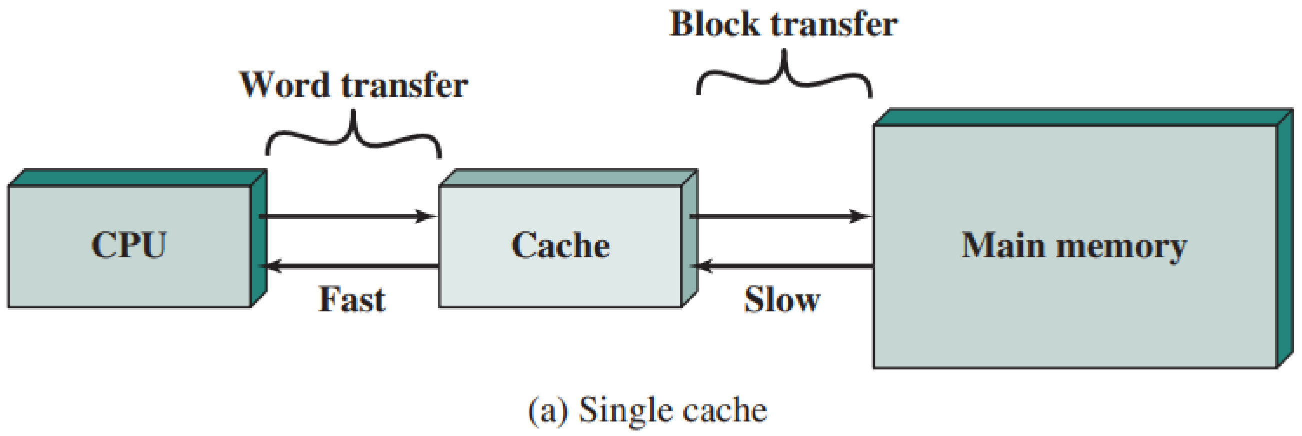
A technique, sometimes referred to as a disk cache, improves performance in two ways:

- Disk writes are clustered. Instead of many small transfers of data, we have a few large transfers of data. This improves disk performance and minimizes processor involvement.
- Some data destined for write-out may be referenced by a program before the next dump to disk. In that case, the data are retrieved rapidly from the Cache rather than slowly from the disk.

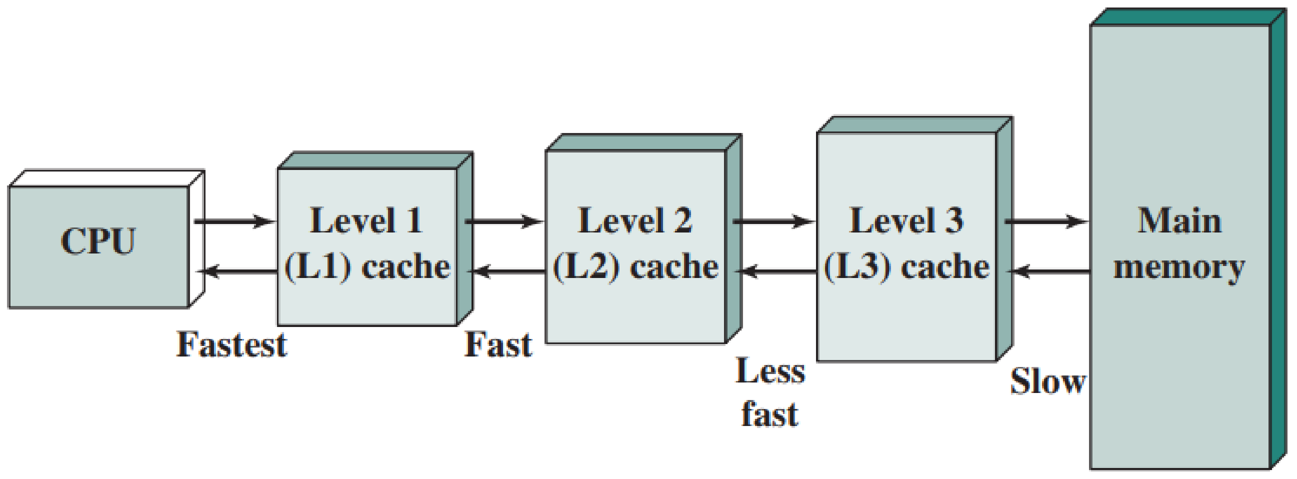
+ Cache

- 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

Figure 4.3 Cache and Main Memory

Cache/Main Memory Structure

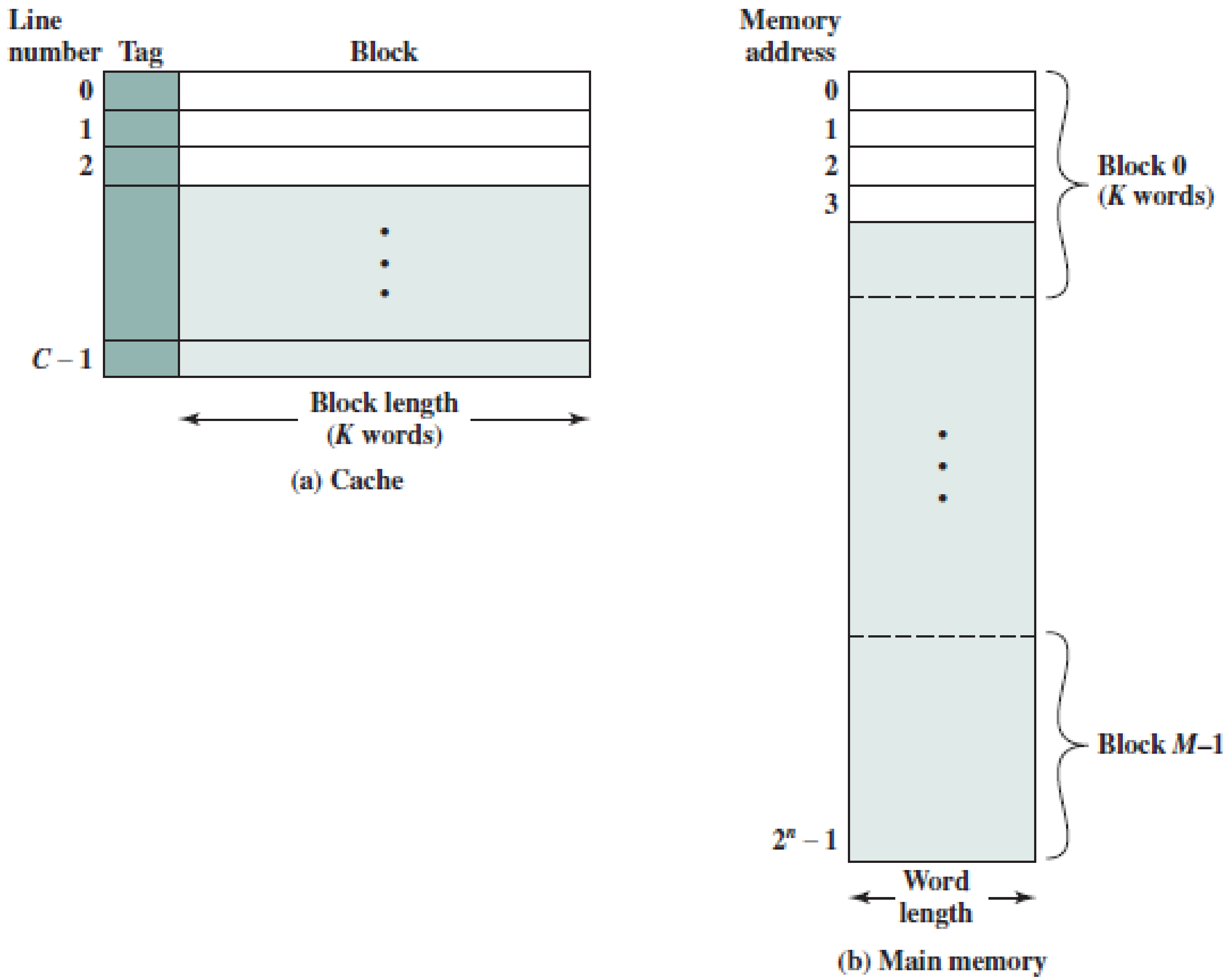


Figure 4.4 Cache/Main Memory Structure

Cache Read Operation

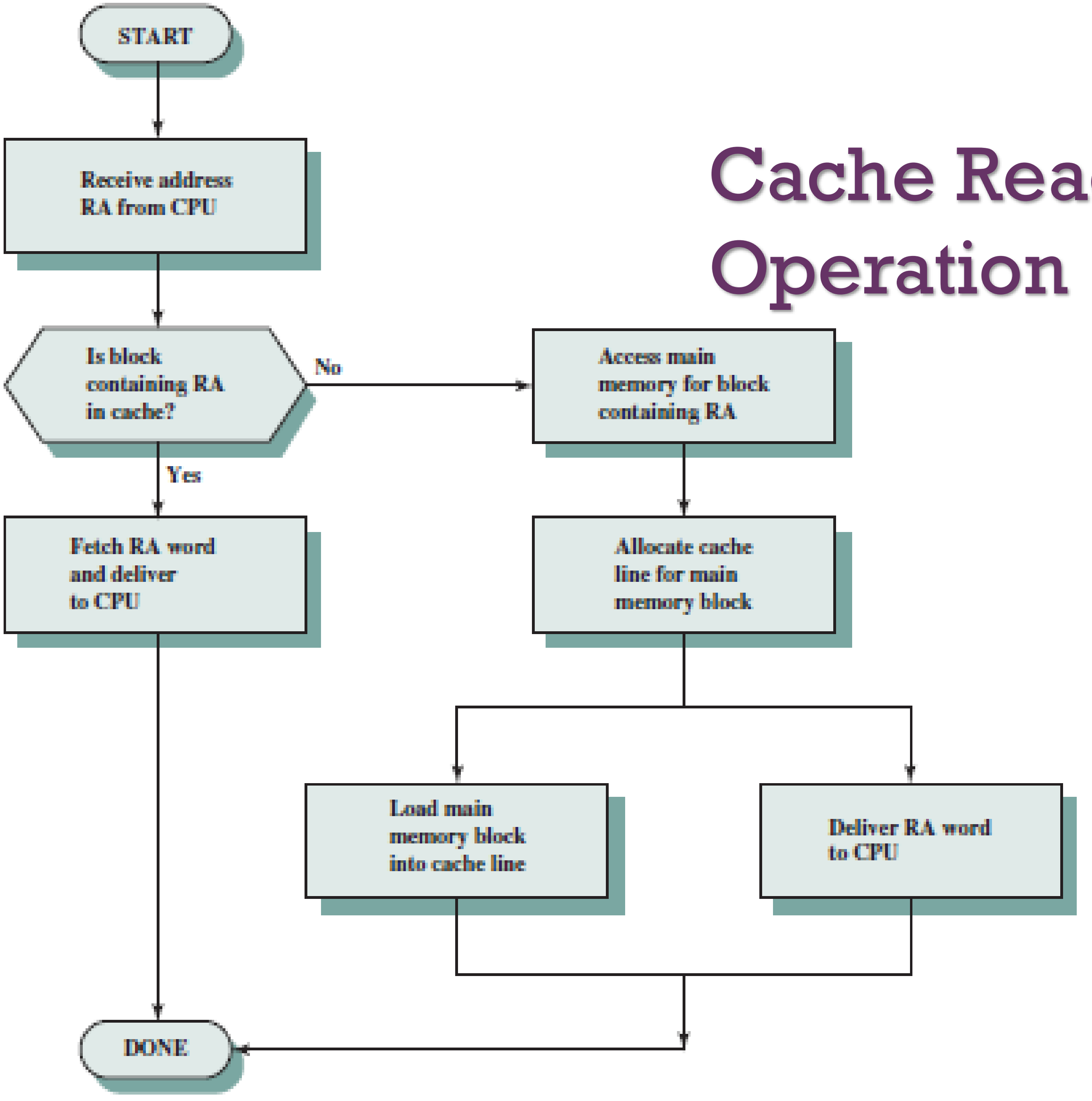

START
Receive address RA from CPU
Is block containing RA in cache?
No
Yes
Access main memory for block containing RA
Fetch RA word and deliver to CPU
Allocate cache line for main memory block
Load main memory block into cache line
Deliver RA word to CPU
DONE

Figure 4.5 Cache Read Operation

+

Typical Cache Organization

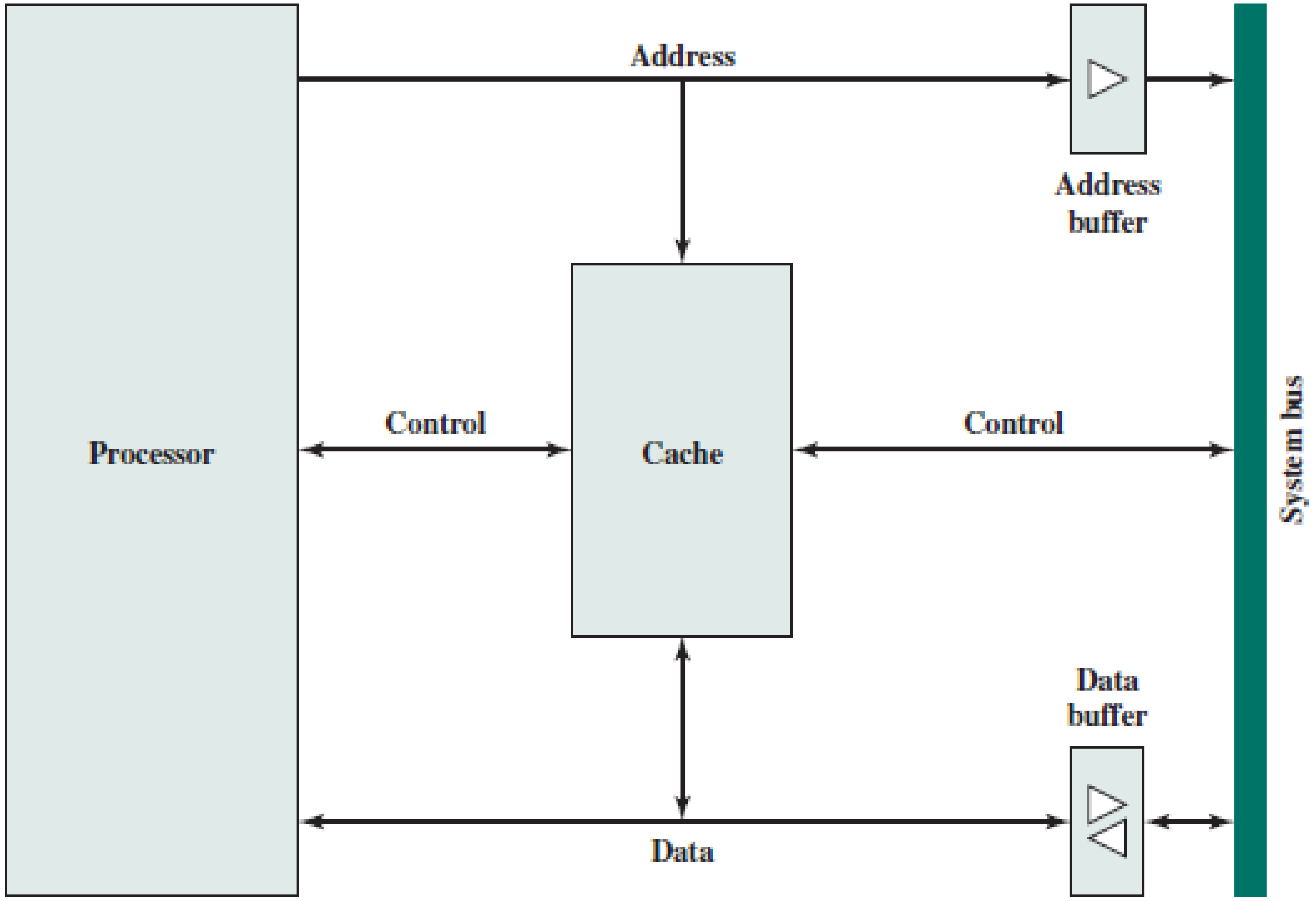


Figure 4.6 Typical Cache Organization

ELEMENTS OF CACHE DESIGN

Table 4.2 Elements of Cache Design

Cache Addresses
- Logical
- Physical

Cache Size

Mapping Function
- Direct
- Associative
- Set associative

Replacement Algorithm
- Least recently used (LRU)
- First in first out (FIFO)
- Least frequently used (LFU)
- Random

Write Policy
- Write through
- Write back

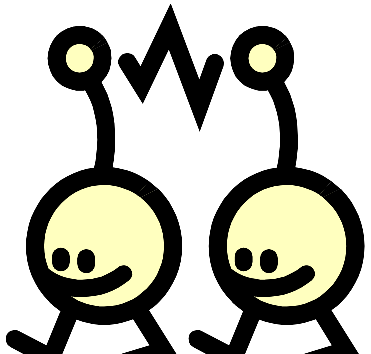
Line Size

Number of Caches
- Single or two level
- Unified or split

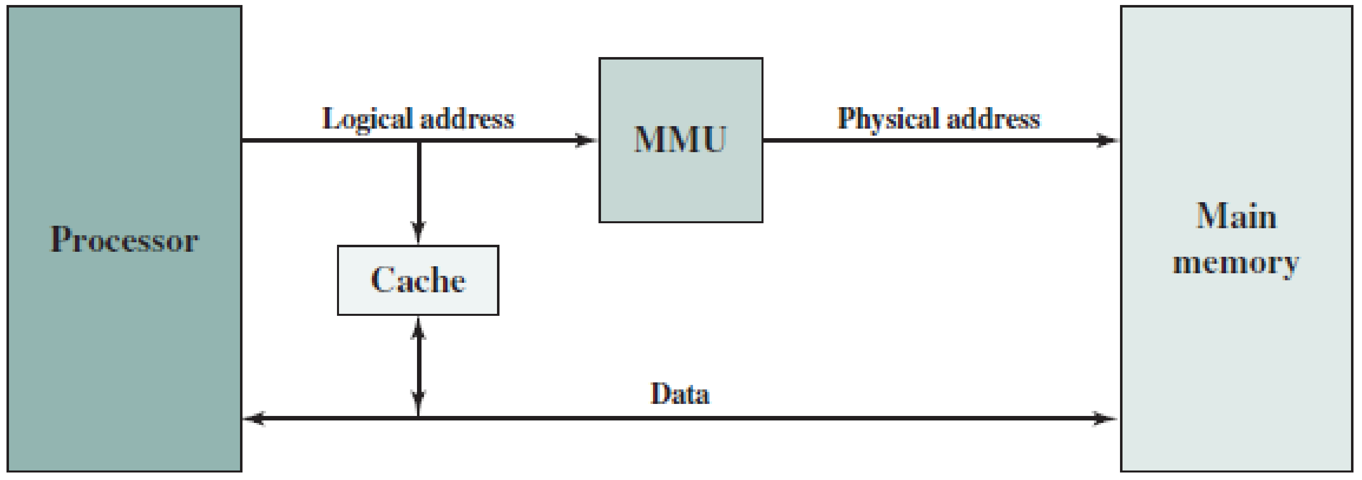
Cache Addresses

Virtual Memory

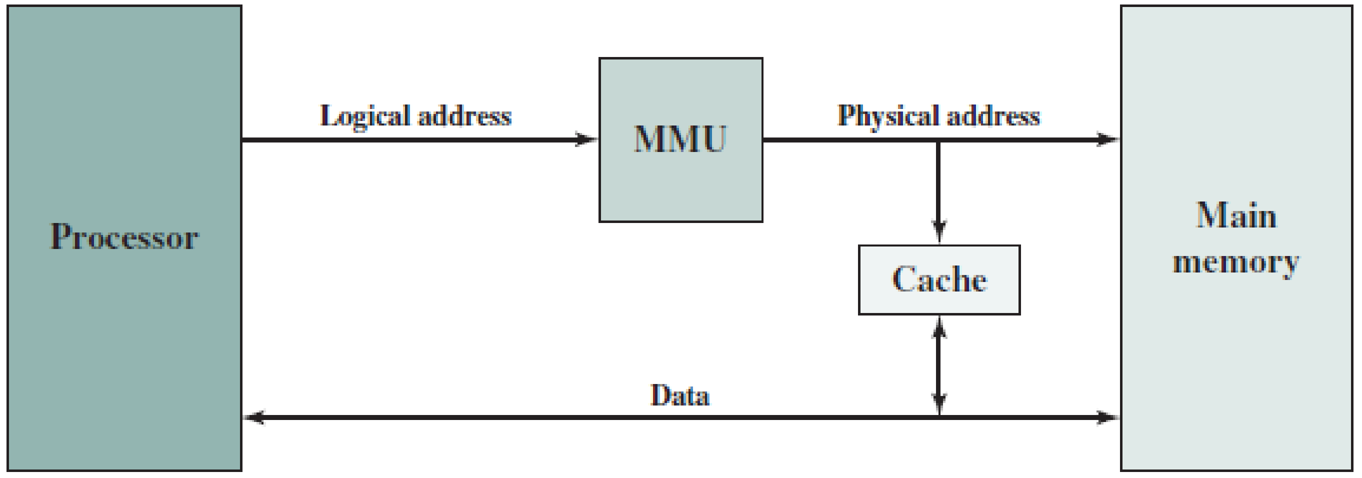
- Virtual memory
  - Facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available
  - When used, the address fields of machine instructions contain virtual addresses
  - For reads to and writes from main memory, a hardware memory management unit (MMU) translates each virtual address into a physical address in main memory



+ Logical and Physical Caches



(a) Logical cache



(b) Physical cache

Table 4.3

Cache Sizes of Some Processors

Processor	Type	Year of Introduction	L1 Cache[a]	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 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
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	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24-48 MB
Intel Core i7 EE 990	Workstaton/ server	2011	6 × 32 kB/32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ Server	2011	24 × 64 kB/ 128 kB	24 × 1.5 MB	24 MB L3 192 MB L4

[a] Two values separated by a slash refer to instruction and data caches.

[b] Both caches are instruction only; no data caches.

Mapping Function

- Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines
- Three techniques can be used:

Direct

- The simplest technique
- Maps each block of main memory into only one possible cache line

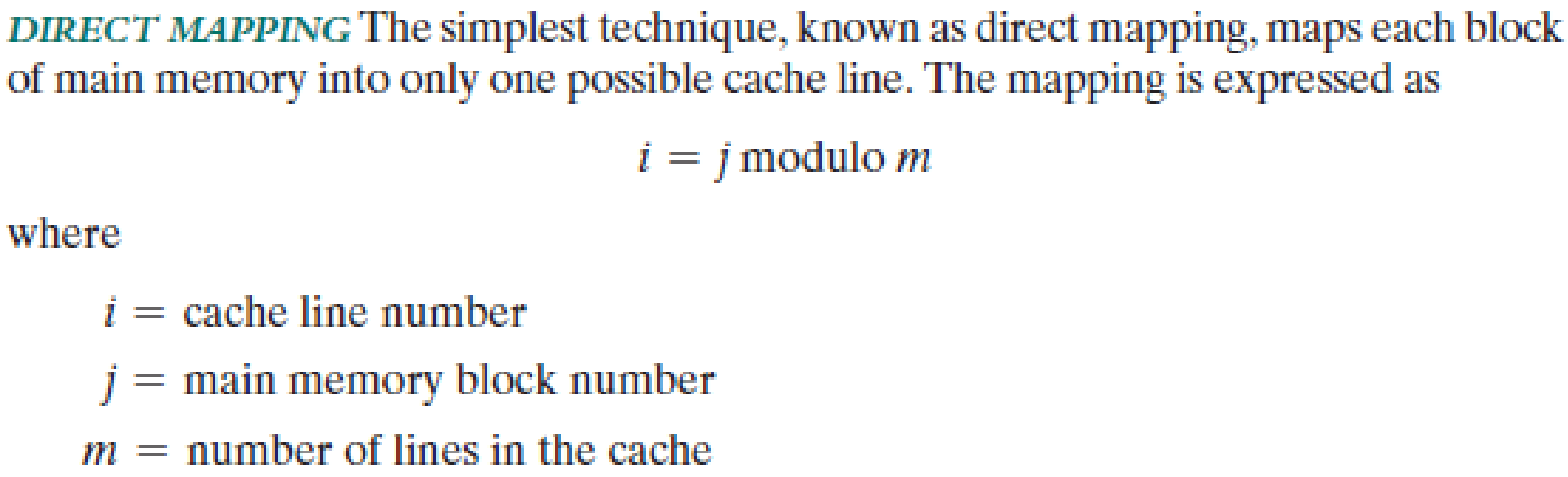
Associative

- Permits each main memory block to be loaded into any line of the cache
- The cache control logic interprets a memory address simply as a Tag and a Word field
- To determine whether a block is in the cache, the cache control logic must simultaneously examine every line's Tag for a match

Set Associative

- A compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages

Direct Mapping



DIRECT MAPPING The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. The mapping is expressed as

$$i = j \text{ modulo } m$$

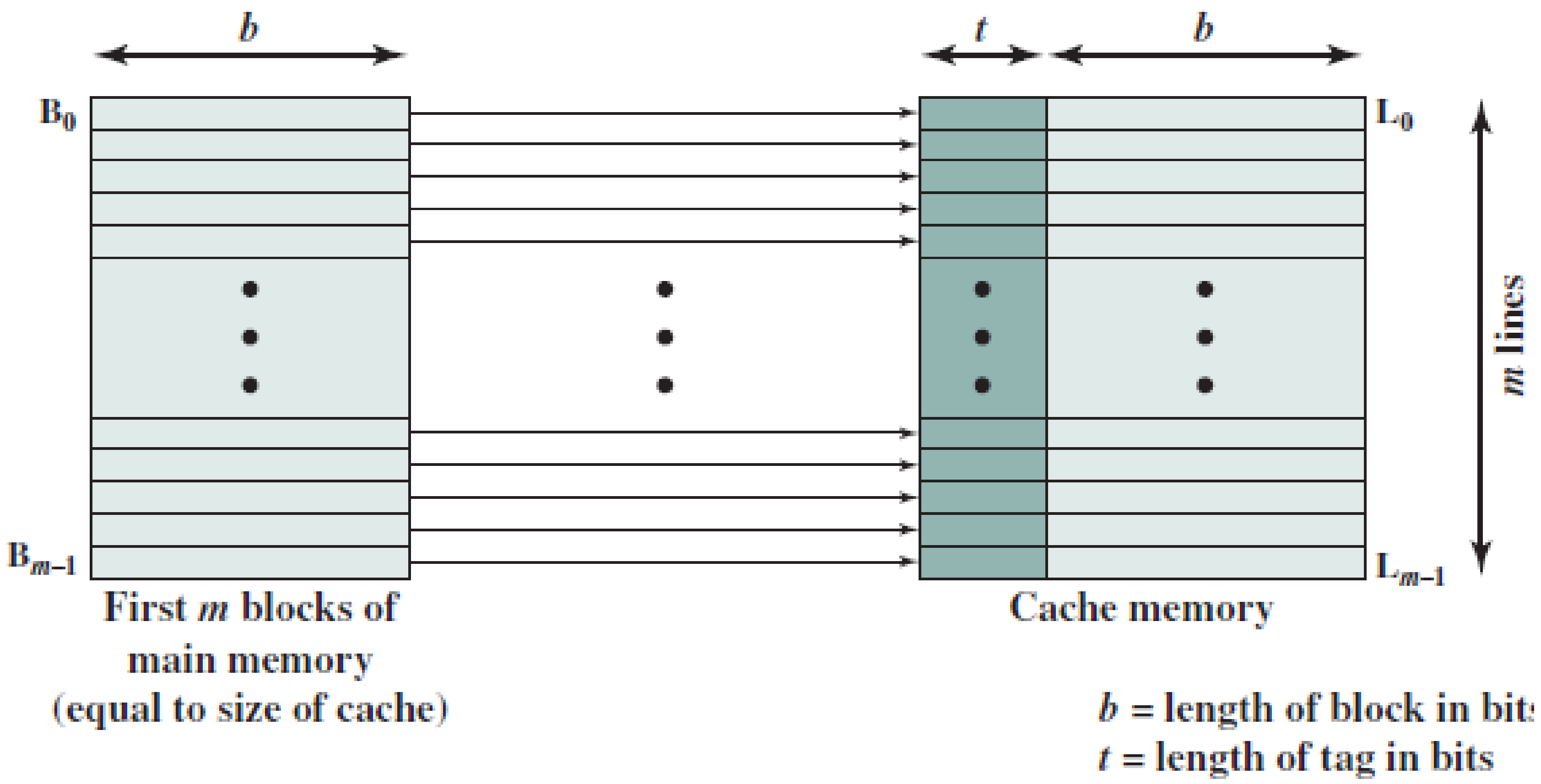
where

i = cache line number

j = main memory block number

m = number of lines in the cache

Direct Mapping



b
t
b
B_0
B_{m-1}
L_0
L_{m-1}
m lines
First m blocks of main memory (equal to size of cache)
Cache memory
b = length of block in bit
t = length of tag in bits

(a) Direct mapping

Direct Mapping Cache Organization

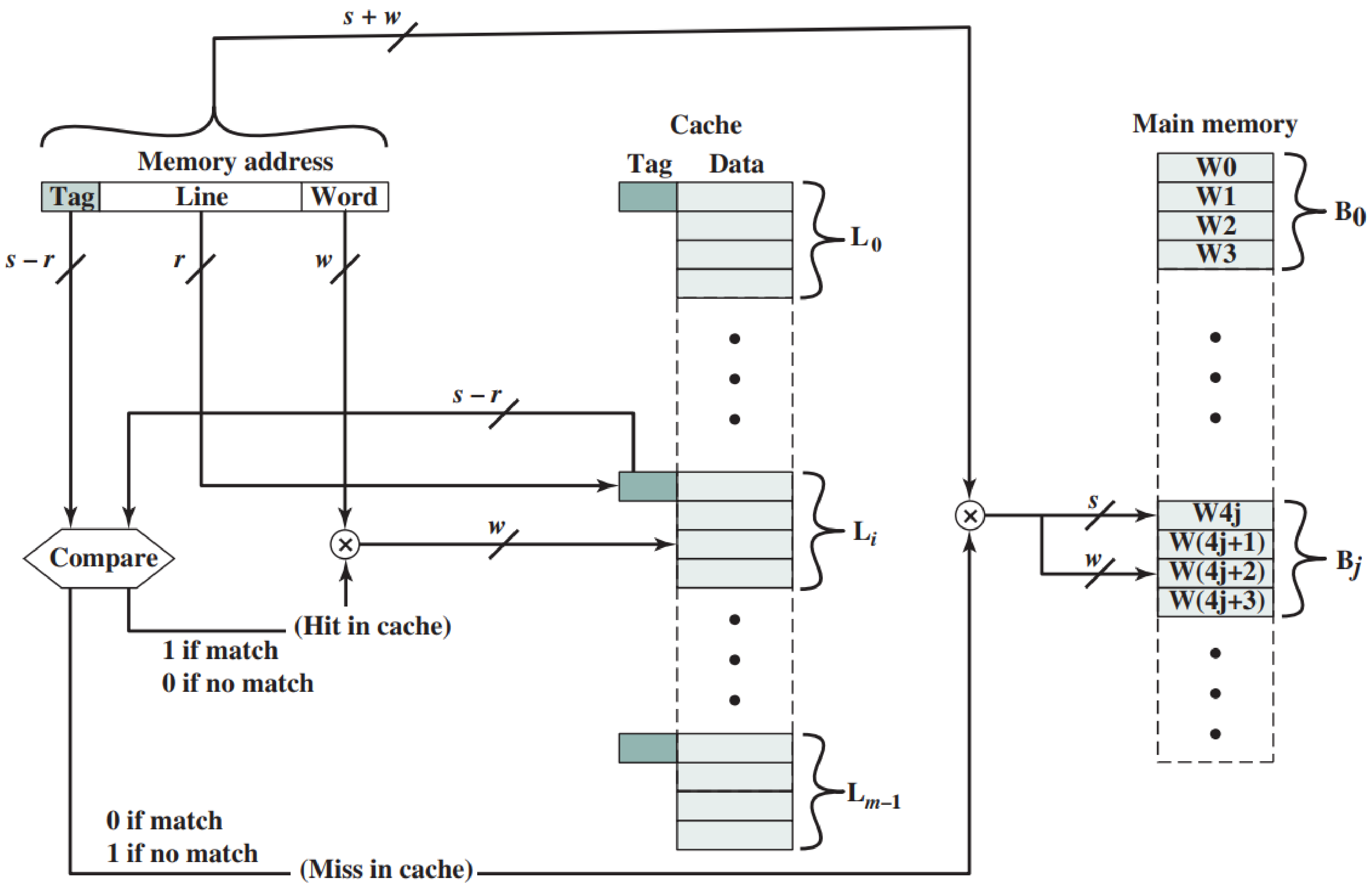
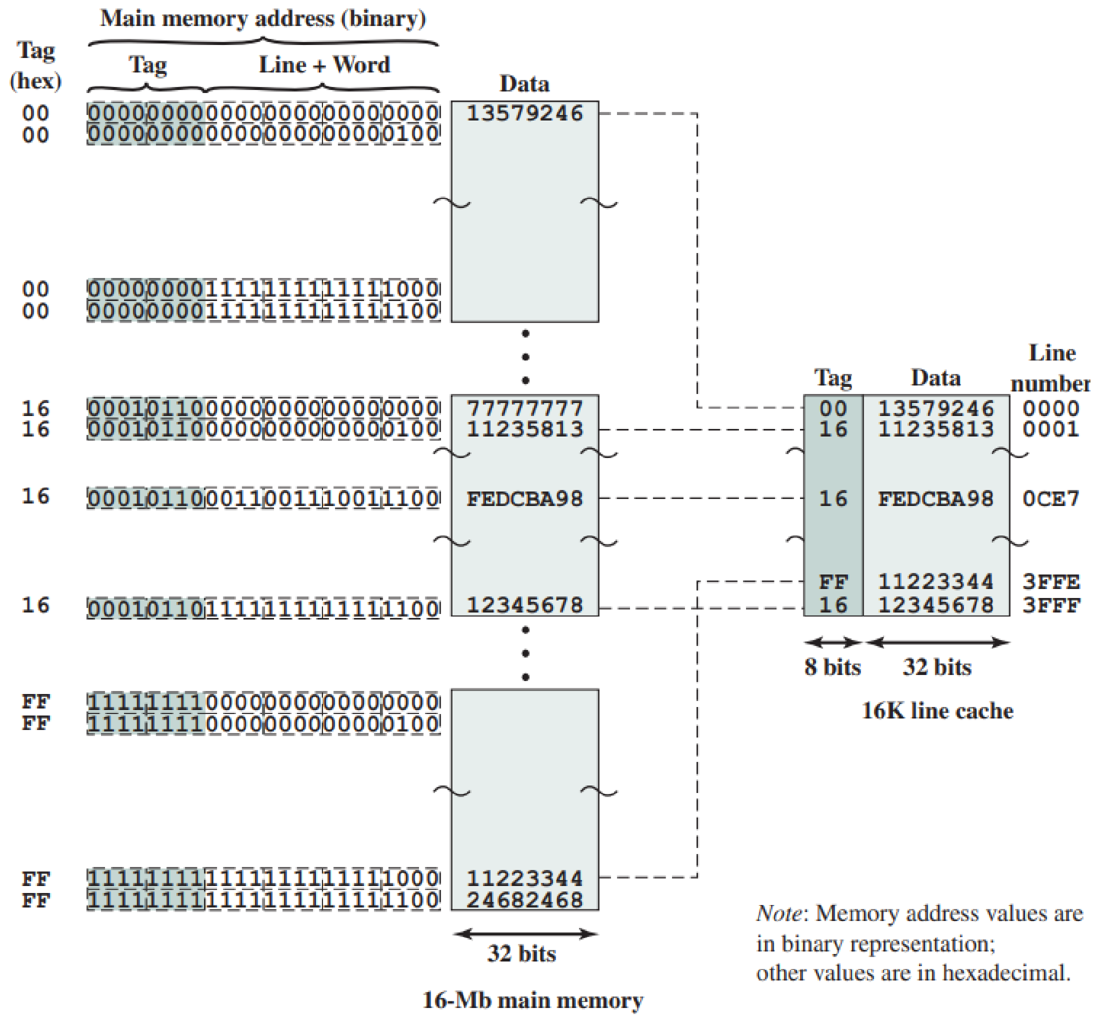


Figure 4.9 Direct-Mapping Cache Organization

Direct Mapping Cache Example



Note: Memory address values are in binary representation; other values are in hexadecimal.

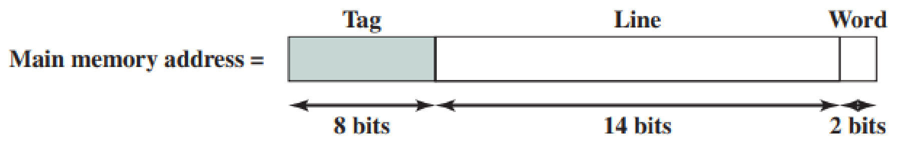
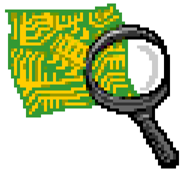


Figure 4.10 Direct Mapping Example

+ Direct Mapping Summary

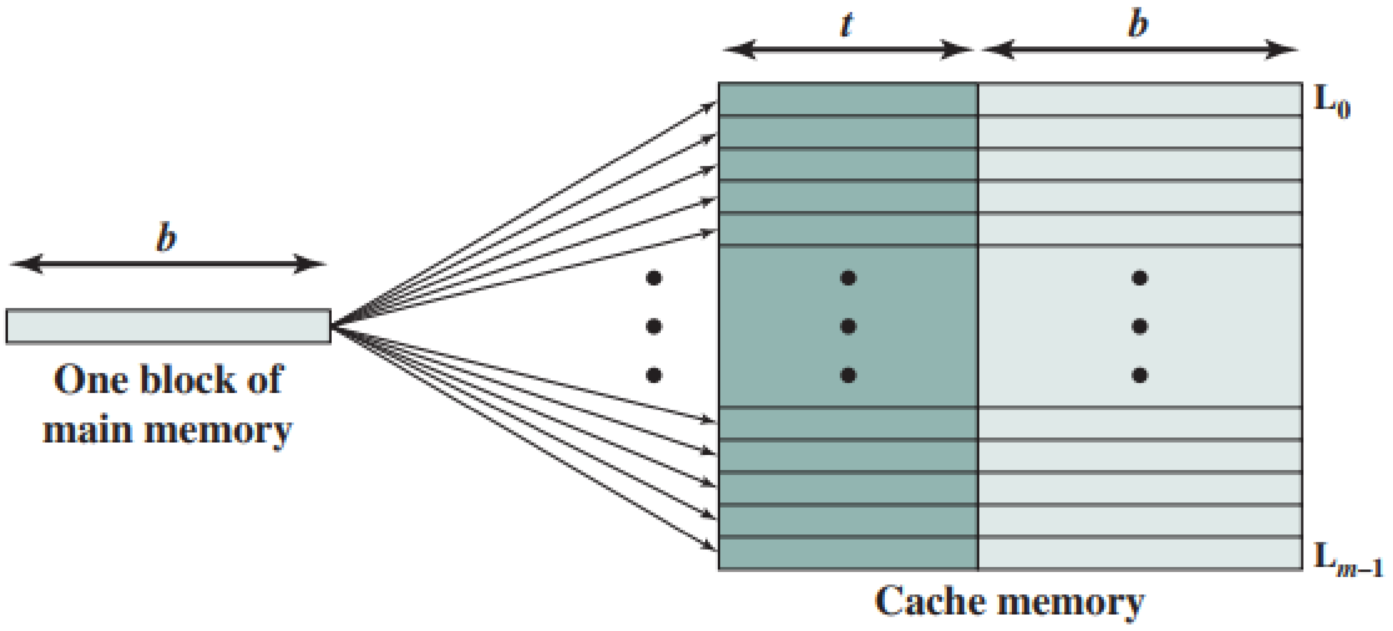
- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$
- Number of lines in cache = m = 2^{r}
- Size of tag = (s – r) bits



Victim Cache

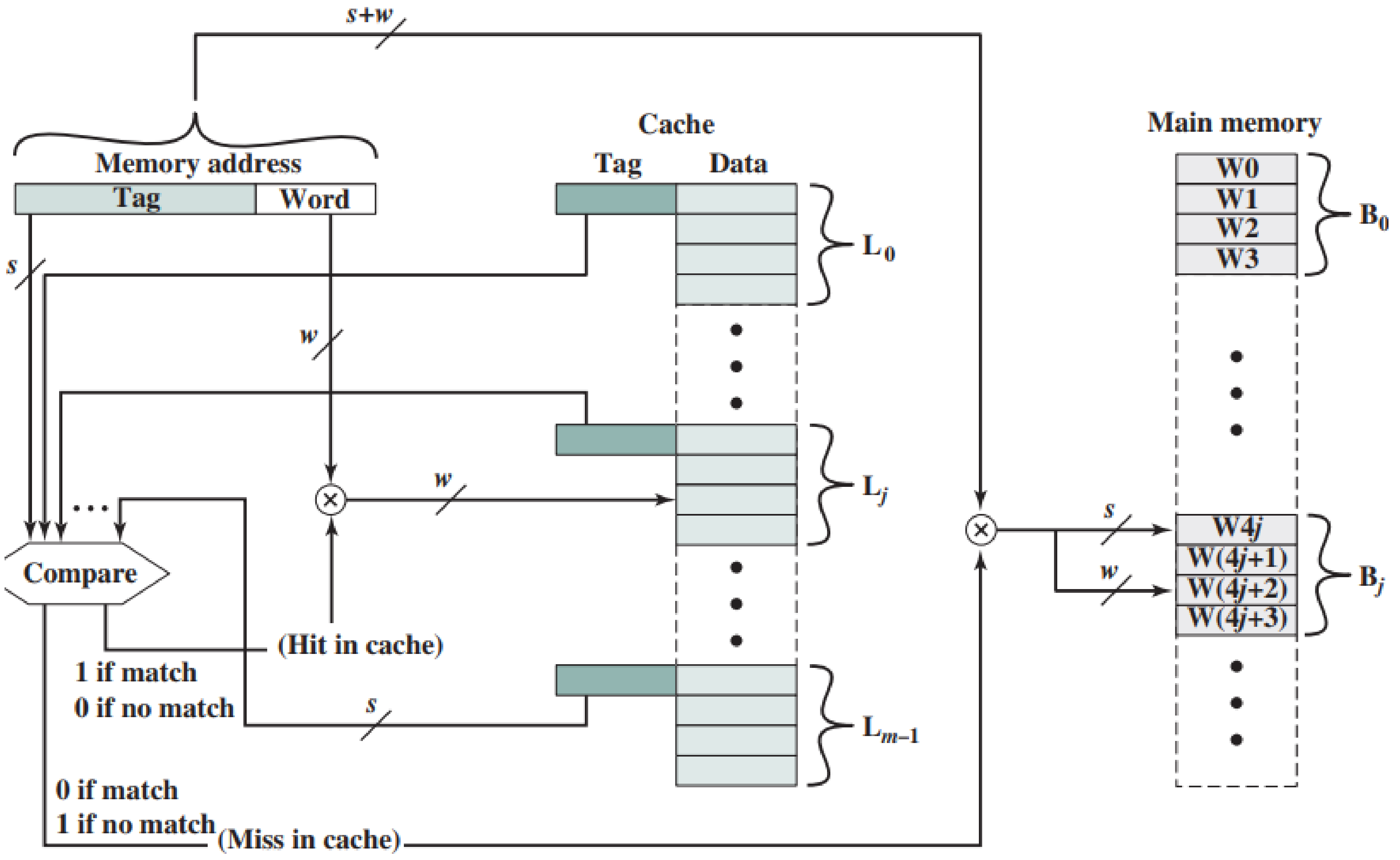
- Originally proposed as an approach to reduce the conflict misses of direct mapped caches without affecting its fast access time
- Fully associative cache
- Typical size is 4 to 16 cache lines
- Residing between direct mapped L1 cache and the next level of memory

Associative Cache Mapping



(b) Associative mapping

Fully Associative Cache Organization



Associative Mapping Example

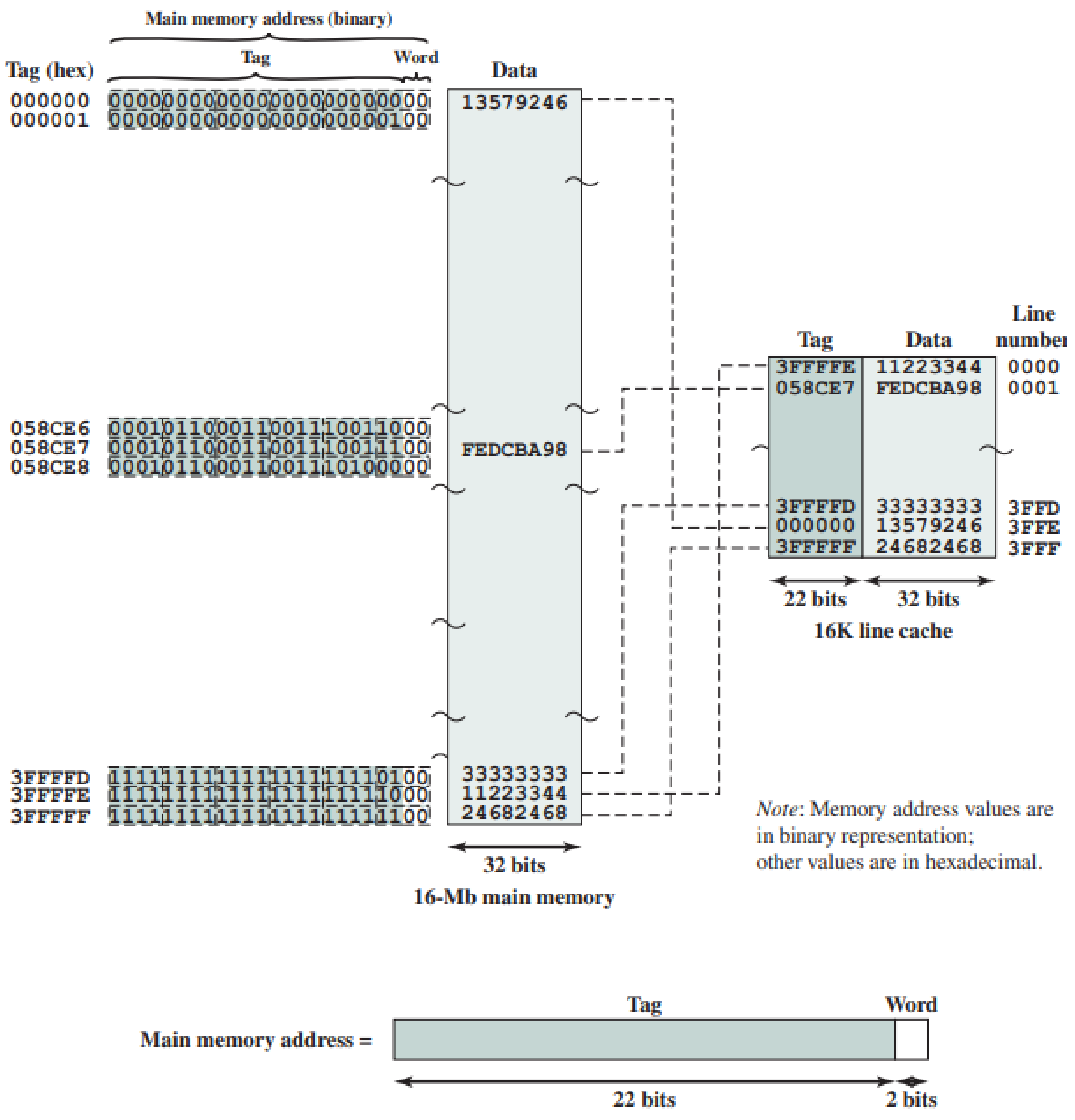
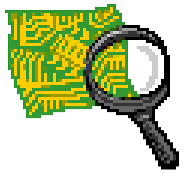


Figure 4.12 Associative Mapping Example

+ Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2w 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



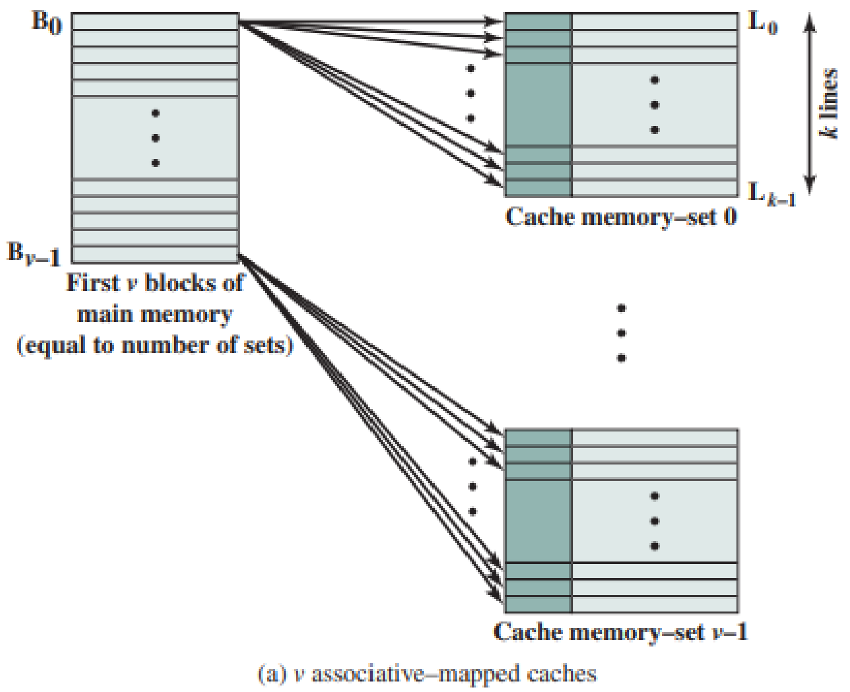
Set Associative Mapping

- Compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages
- Cache consists of a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

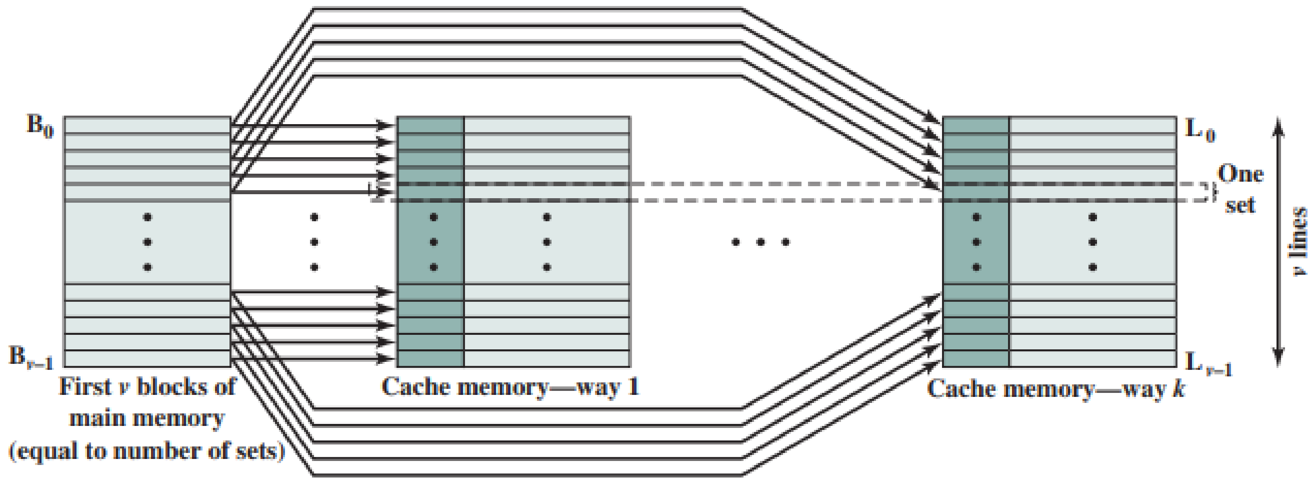
+

Mapping From Main Memory to Cache:

k-Way Set Associative



(a) v associative–mapped caches



(b) k direct–mapped caches

Figure 4.13 Mapping from Main Memory to Cache: *k*-Way Set Associative

k-Way Set Associative Cache Organization

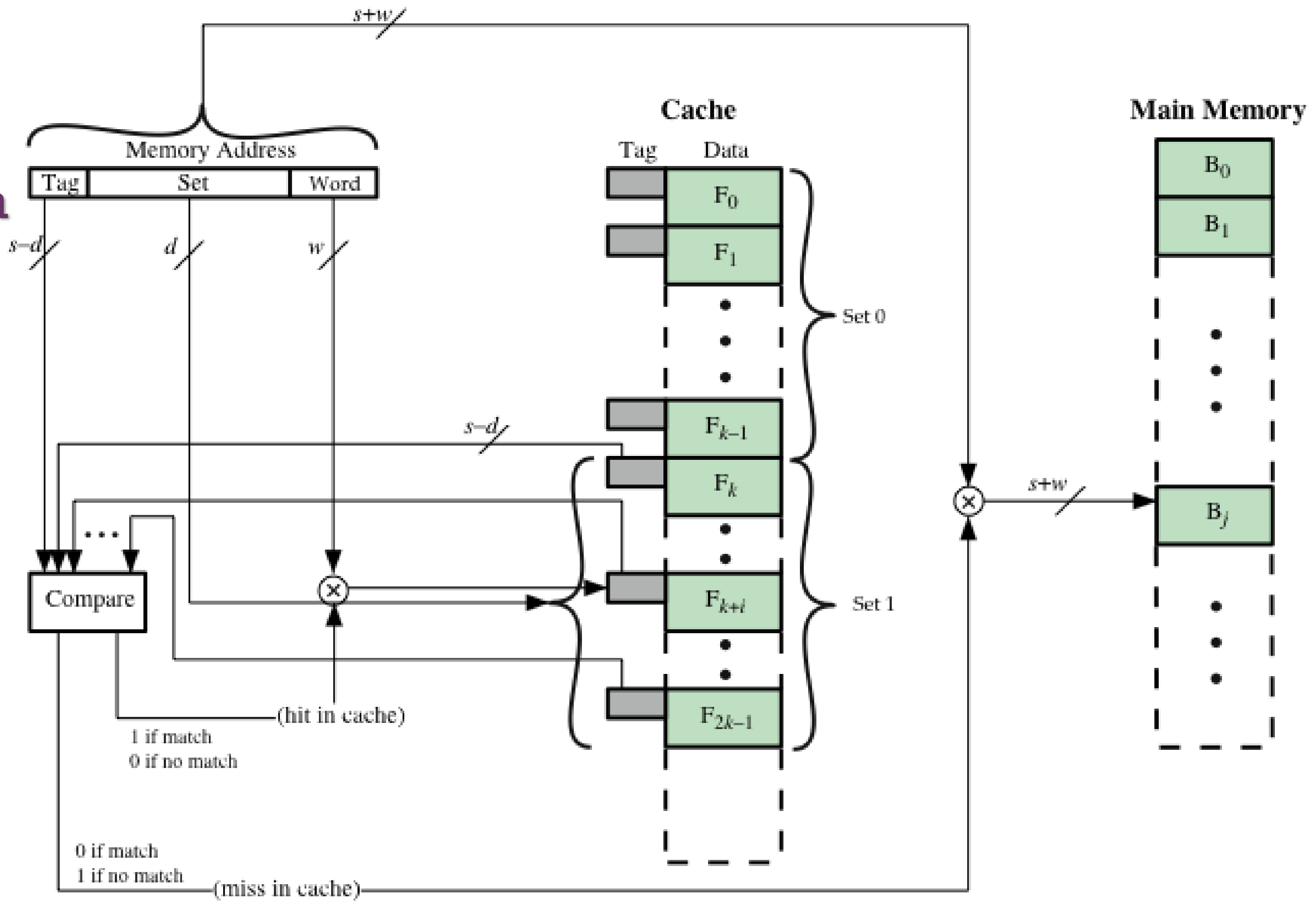
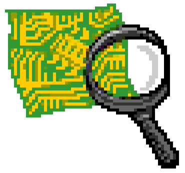


Figure 4.14 *k*-Way Set Associative Cache Organization

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 = $2^{s+w}/2^w = 2^s$
- Number of lines in set = k
- Number of sets = $v = 2^d$
- Number of lines in cache = $m=kv = k * 2^d$
- Size of cache = $k * 2^{d+w}$ words or bytes
- Size of tag = $(s – d)$ bits



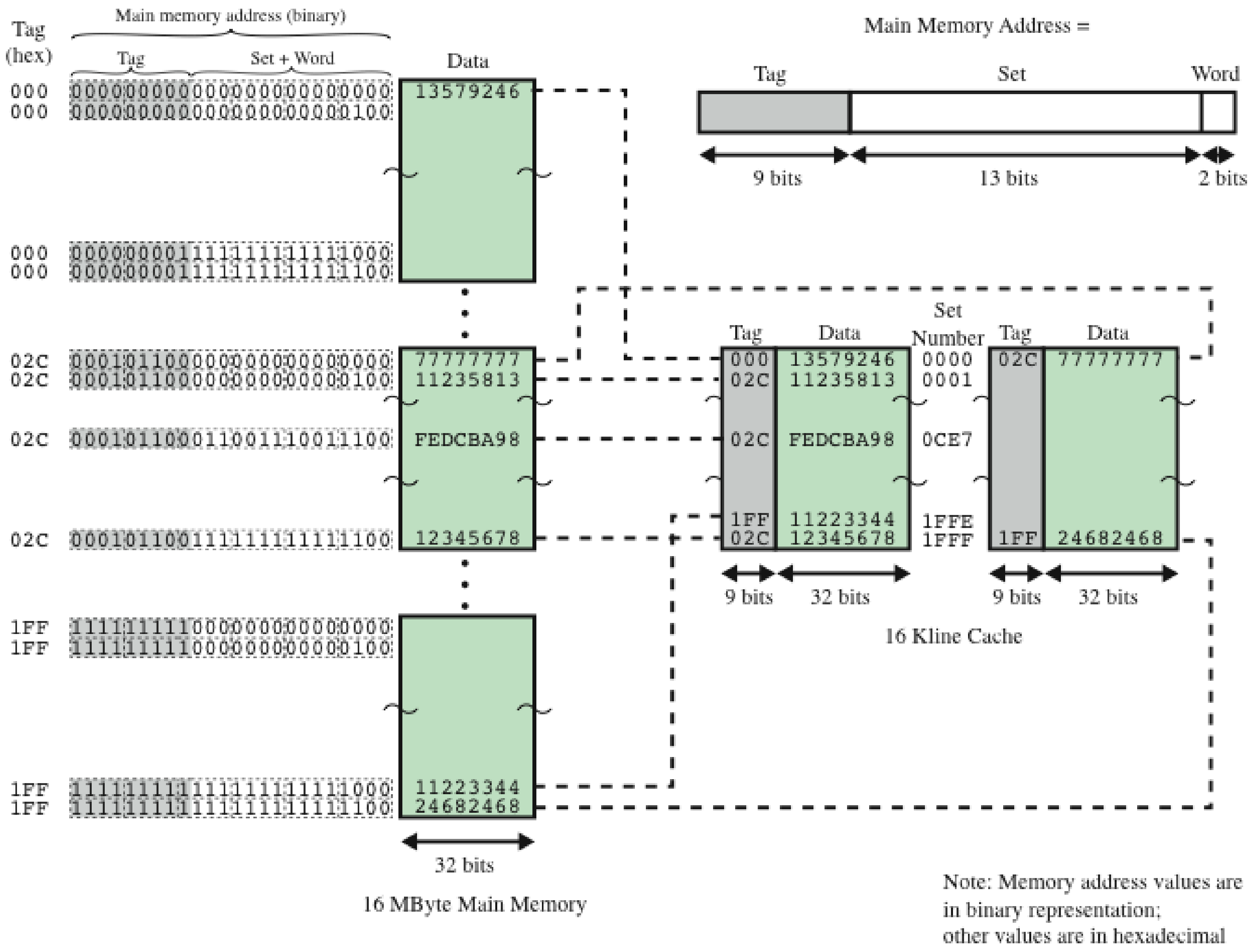


Figure 4.15 Two-Way Set Associative Mapping Example

+

Varying Associativity Over Cache Size

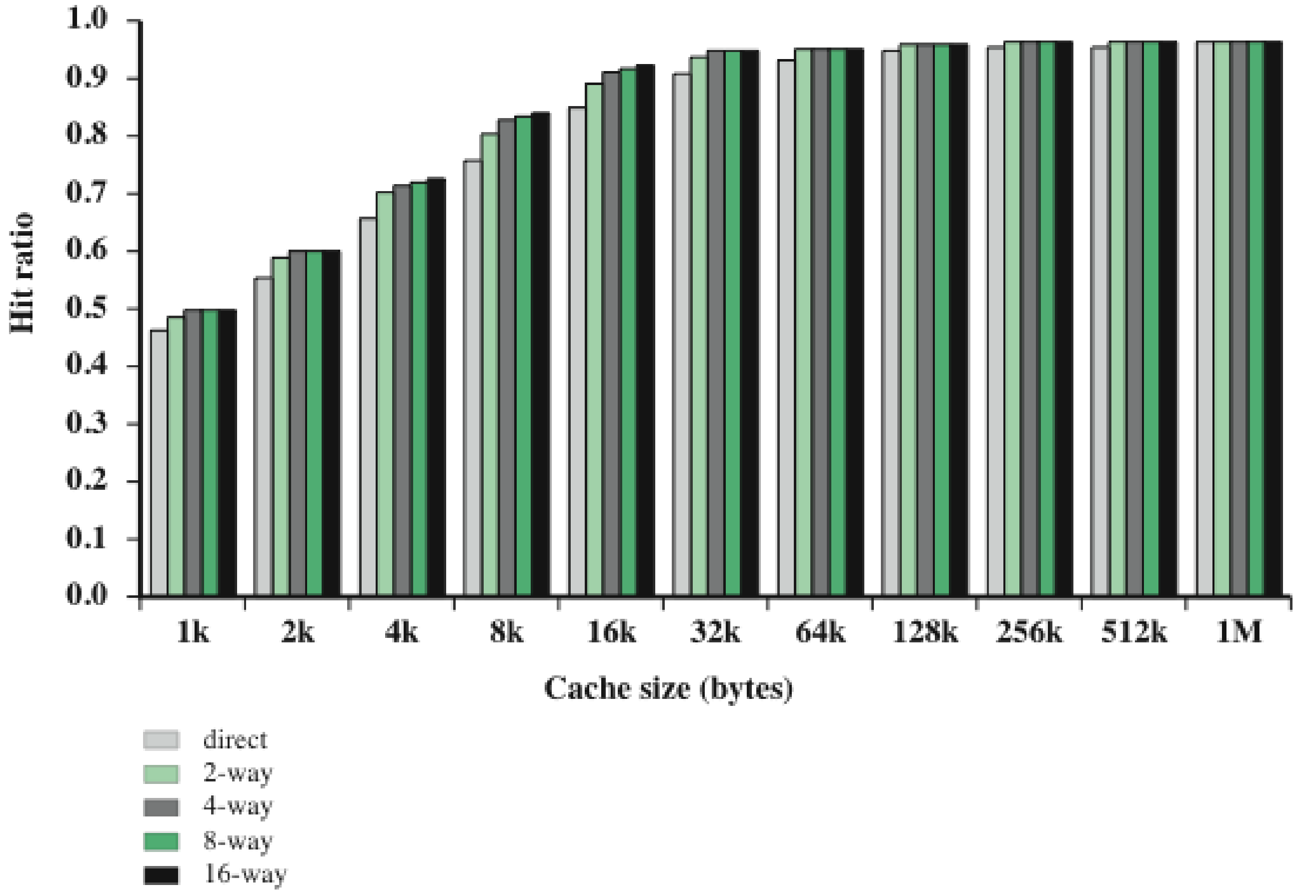


Figure 4.16 Varying Associativity over Cache Size

+
Problem

- Assume the size of a main memory in a computer is **1Mbytes**. The block size of the main memory is **16 Bytes**. The size of each word is 1Byte. The size of the cache memory is **64KBytes**.

- Draw the Main memory format for direct mapping, associative mapping and two way set associative mapping.

- direct mapping

4 bit tag	12 bits	4 bit word

- associative mapping

16 bit Tag	4 bit word

- 2 way set associative mapping

5 bit tag	11 bit set	4 bit word

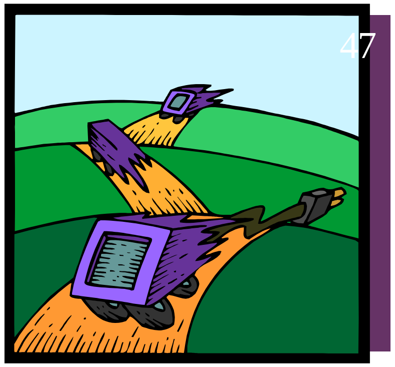
+ Problem

- 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.

+

Replacement Algorithms



- Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced
- For direct mapping there is only one possible line for any particular block and no choice is possible
- For the associative and set-associative techniques a replacement algorithm is needed
- To achieve high speed, an algorithm must be implemented in hardware

+

The four most common replacement algorithms are:

- Least recently used (LRU)
  - Most effective
  - Replace that block in the set that has been in the cache longest with no reference to it
  - Because of its simplicity of implementation, LRU is the most popular replacement algorithm
- First-in-first-out (FIFO)
  - Replace that block in the set that has been in the cache longest
  - Easily implemented as a round-robin or circular buffer technique
- Least frequently used (LFU)
  - Replace that block in the set that has experienced the fewest references
  - Could be implemented by associating a counter with each line

Write Policy

When a block that is resident in the cache is to be replaced there are two cases to consider:

If the old block in the cache has not been altered then it may be overwritten with a new block without first writing out the old block

If at least one write operation has been performed on a word in that line of the cache then main memory must be updated by writing the line of cache out to the block of memory before bringing in the new block

There are two problems to contend with:

More than one device may have access to main memory

A more complex problem occurs when multiple processors are attached to the same bus and each processor has its own local cache - if a word is altered in one cache it could conceivably invalidate a word in other caches

Write Through and Write Back

- Write through
  - Simplest technique
  - All write operations are made to main memory as well as to the cache
  - The main disadvantage of this technique is that it generates substantial memory traffic and may create a bottleneck
- Write back
  - Minimizes memory writes
  - Updates are made only in the cache
  - Portions of main memory are invalid and hence accesses by I/O modules can be allowed only through the cache
  - This makes for complex circuitry and a potential bottleneck

Cache coherency

Even if a write-through policy is used, the other caches may contain invalid data. A system that prevents this problem is said to maintain cache coherency. Possible approaches to cache coherency include the following:

1. **Hardware transparency:** Additional hardware is used to ensure all updates

2. **Non-cacheable memory:** A portion of main memory is shared by more than one processor, and non-cacheable.

3. **Bus watching with write through:** Each cache controller monitors the address lines to detect write operations to memory by other bus masters.

Line Size

When a block of data is retrieved and placed in the cache not only the desired word but also some number of adjacent words are retrieved

As the block size increases the hit ratio will at first increase because of the principle of locality

As the block size increases more useful data are brought into the cache

The hit ratio will begin to decrease as the block becomes bigger and the probability of using the newly fetched information becomes less than the probability of reusing the information that has to be replaced

Two specific effects come into play:

- Larger blocks reduce the number of blocks that fit into a cache
- As a block becomes larger each additional word is farther from the requested word

+
Multilevel Caches

- As logic density has increased it has become possible to have a cache on the same chip as the processor
- The on-chip cache reduces the processor's external bus activity and speeds up execution time and increases overall system performance
  - When the requested instruction or data is found in the on-chip cache, the bus access is eliminated
  - On-chip cache accesses will complete appreciably faster than would even zero-wait state bus cycles
  - During this period the bus is free to support other transfers
- Two-level cache:
  - Internal cache designated as level 1 (L1)
  - External cache designated as level 2 (L2)
- Potential savings due to the use of an L2 cache depends on the hit rates in both the L1 and L2 caches
- The use of multilevel caches complicates all of the design issues related to caches, including size, replacement algorithm, and write policy

Hit Ratio (L1 & L2) For 8 Kbyte and 16 Kbyte L1

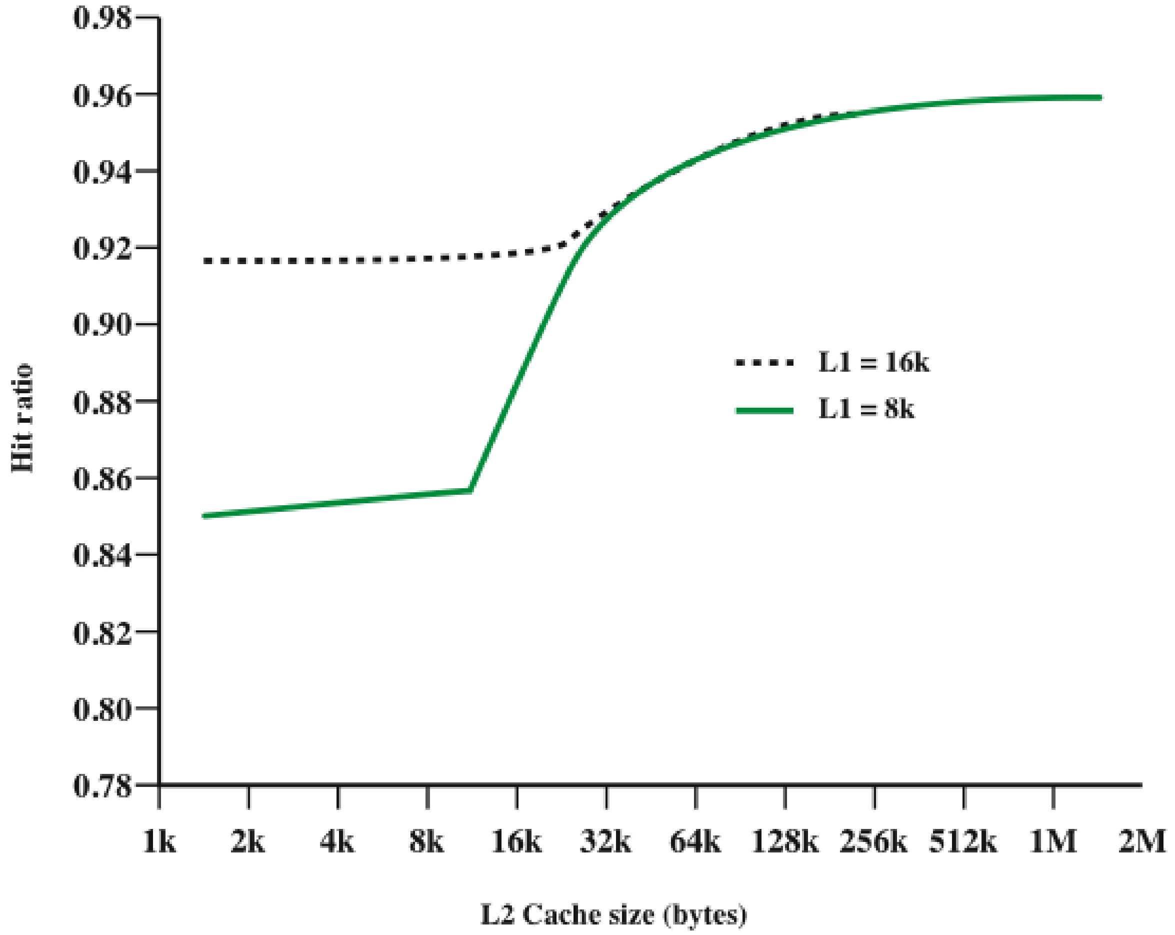


Figure 4.17 Total Hit Ratio (L1 and L2) for 8 Kbyte and 16 Kbyte L1

Unified Versus Split Caches

- Has become common to split cache:
  - One dedicated to instructions
  - One dedicated to data
  - Both exist at the same level, typically as two L1 caches
- Advantages of unified cache:
  - Higher hit rate
    - Balances load of instruction and data fetches automatically
    - Only one cache needs to be designed and implemented
- Trend is toward split caches at the L1 and unified caches for higher levels
- Advantages of split cache:
  - Eliminates cache contention between instruction fetch/decode unit and execution unit
    - Important in pipelining

PENTIUM 4 CACHE ORGANIZATION

Intel Cache Evolution

Problem	Solution	Processor on Which Feature First Appears
External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip.	Add external L2 cache using faster technology than main memory.	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Pentium 4 Block Diagram

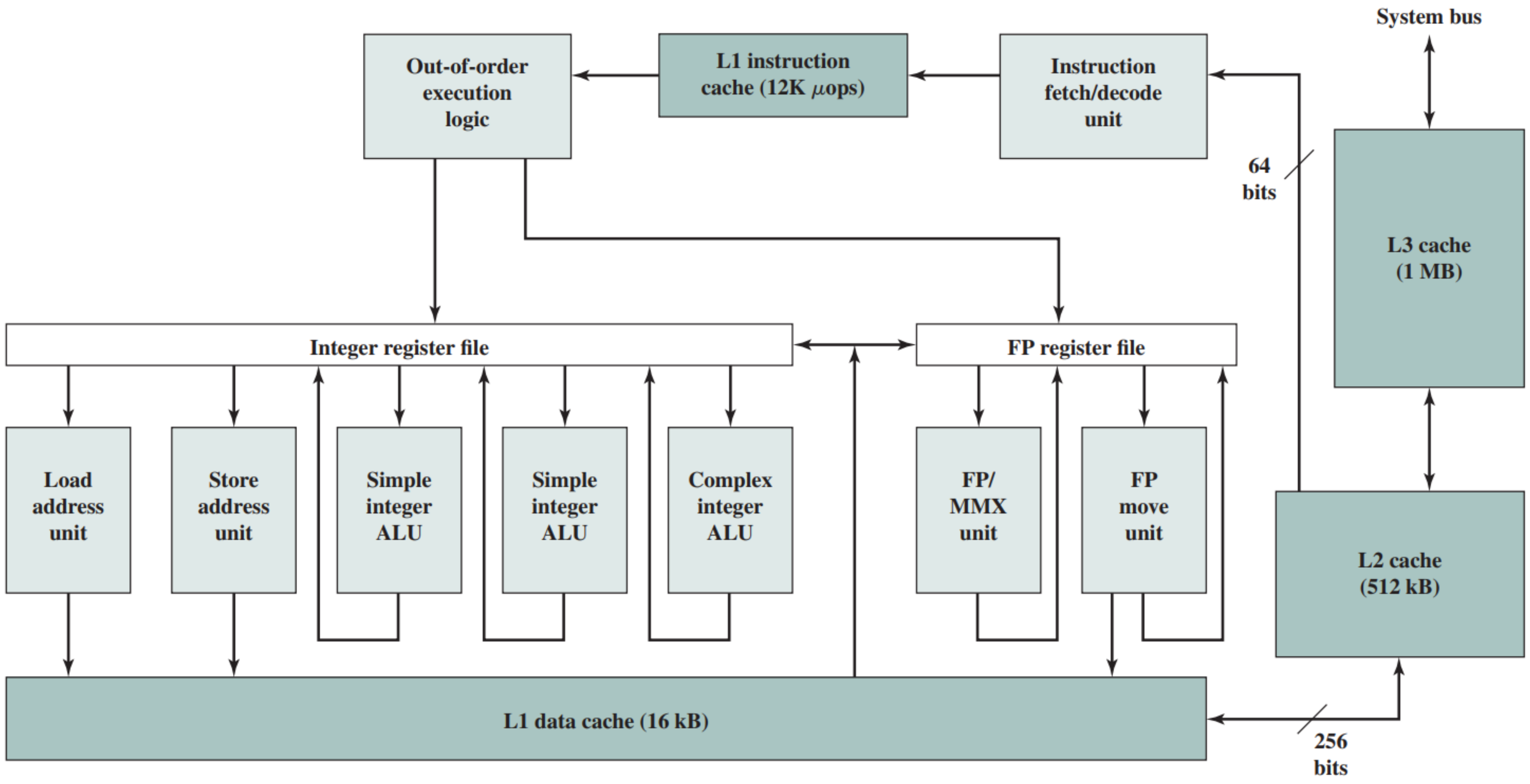
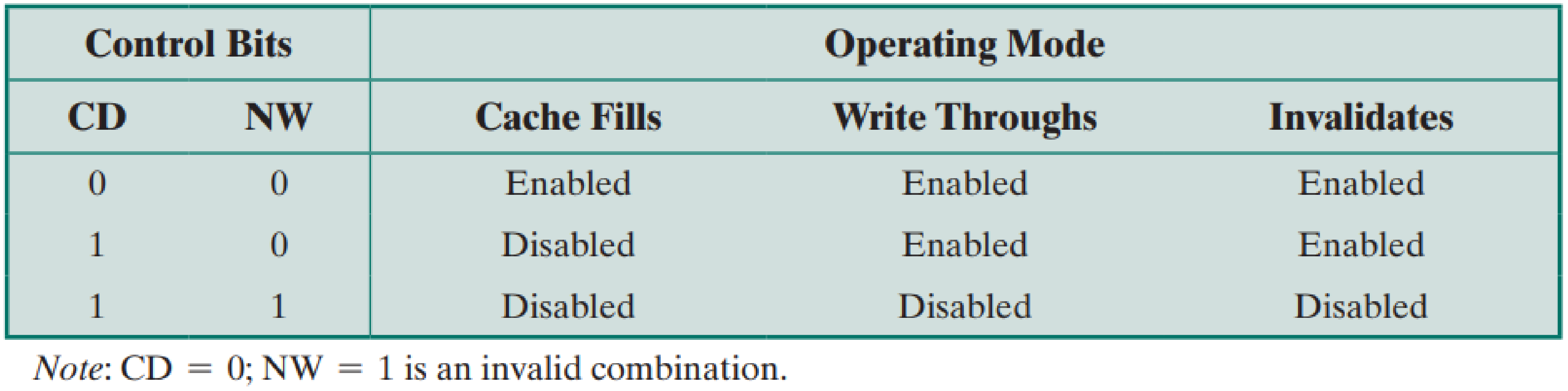


Figure 4.18 Pentium 4 Block Diagram

Pentium 4 Cache Operating Modes



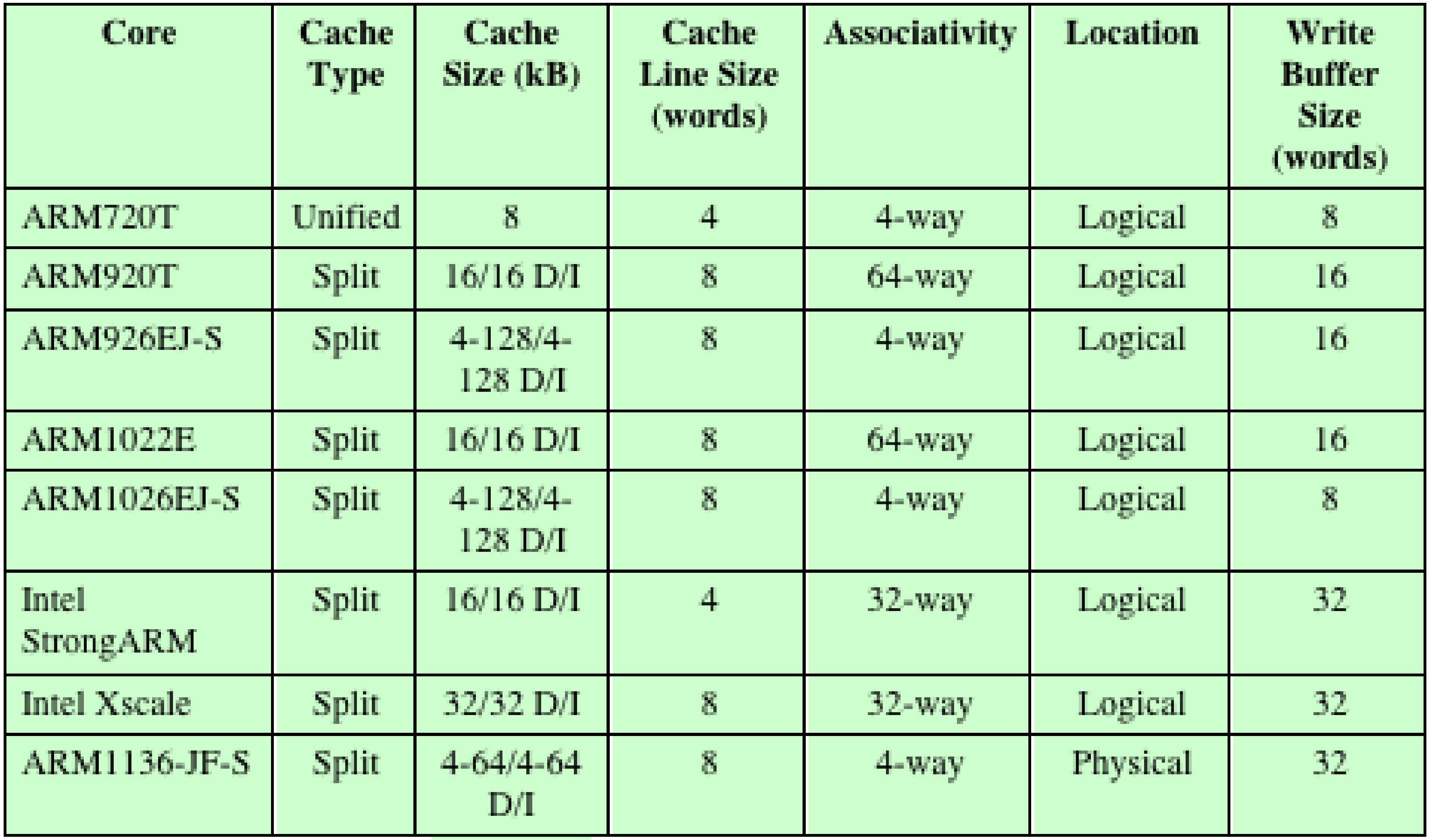
Control Bits		Operating Mode		
CD	**NW**	**Cache Fills**	**Write Throughs**	**Invalidates**
0	0	Enabled	Enabled	Enabled
1	0	Disabled	Enabled	Enabled
1	1	Disabled	Disabled	Disabled

Note: CD = 0; NW = 1 is an invalid combination.

CD (cache disable)
NW (not write-through)

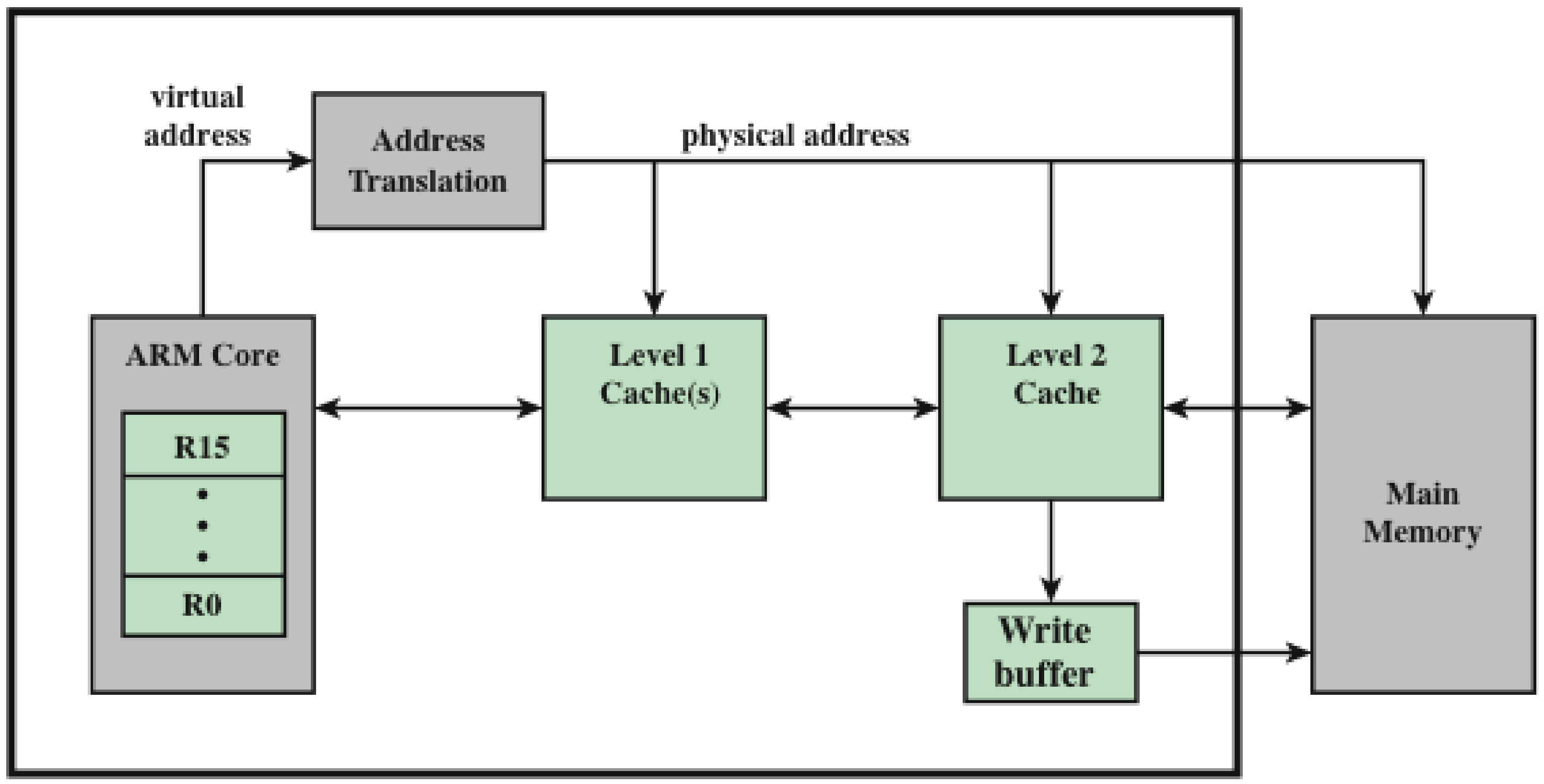
Table 4.5 Pentium 4 Cache Operating Modes

ARM Cache Features



Core	Cache Type	Cache Size (kB)	Cache Line Size (words)	Associativity	Location	Write Buffer Size (words)
ARM720T	Unified	8	4	4-way	Logical	8
ARM920T	Split	16/16 D/I	8	64-way	Logical	16
ARM926EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	16
ARM1022E	Split	16/16 D/I	8	64-way	Logical	16
ARM1026EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	8
Intel StrongARM	Split	16/16 D/I	4	32-way	Logical	32
Intel Xscale	Split	32/32 D/I	8	32-way	Logical	32
ARM1136-JF-S	Split	4-64/4-64 D/I	8	4-way	Physical	32

ARM Cache and Write Buffer Organization



+
Summary

Chapter 4

Cache Memory

- Characteristics of Memory Systems
  - Location
  - Capacity
  - Unit of transfer
- Memory Hierarchy
  - How much?
  - How fast?
  - How expensive?
- Cache memory principles

- Elements of cache design
  - Cache addresses
  - Cache size
  - Mapping function
  - Replacement algorithms
  - Write policy
  - Line size
  - Number of caches
- Pentium 4 cache organization
- ARM cache organization

Thank you for the patience :)