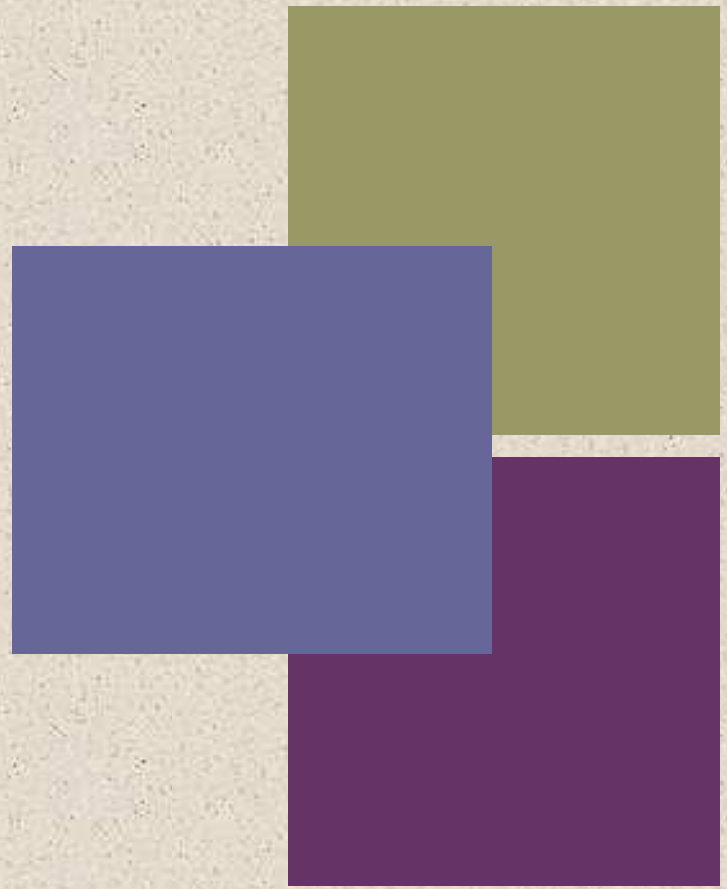
Edited by

Dr. George Lazik

William Stallings Computer Organization and Architecture 10th Edition



Chapter 4
Cache Memory

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

Direct access

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

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
- Main memory and some cache systems are random access

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

Capacity and Performance:

The two most important characteristics of memory

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)

Memory

- The most common forms are:
  - Semiconductor memory
  - Magnetic surface memory
  - Optical
  - Magneto-optical
- Several physical characteristics of data storage are important:
  - 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)
- 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

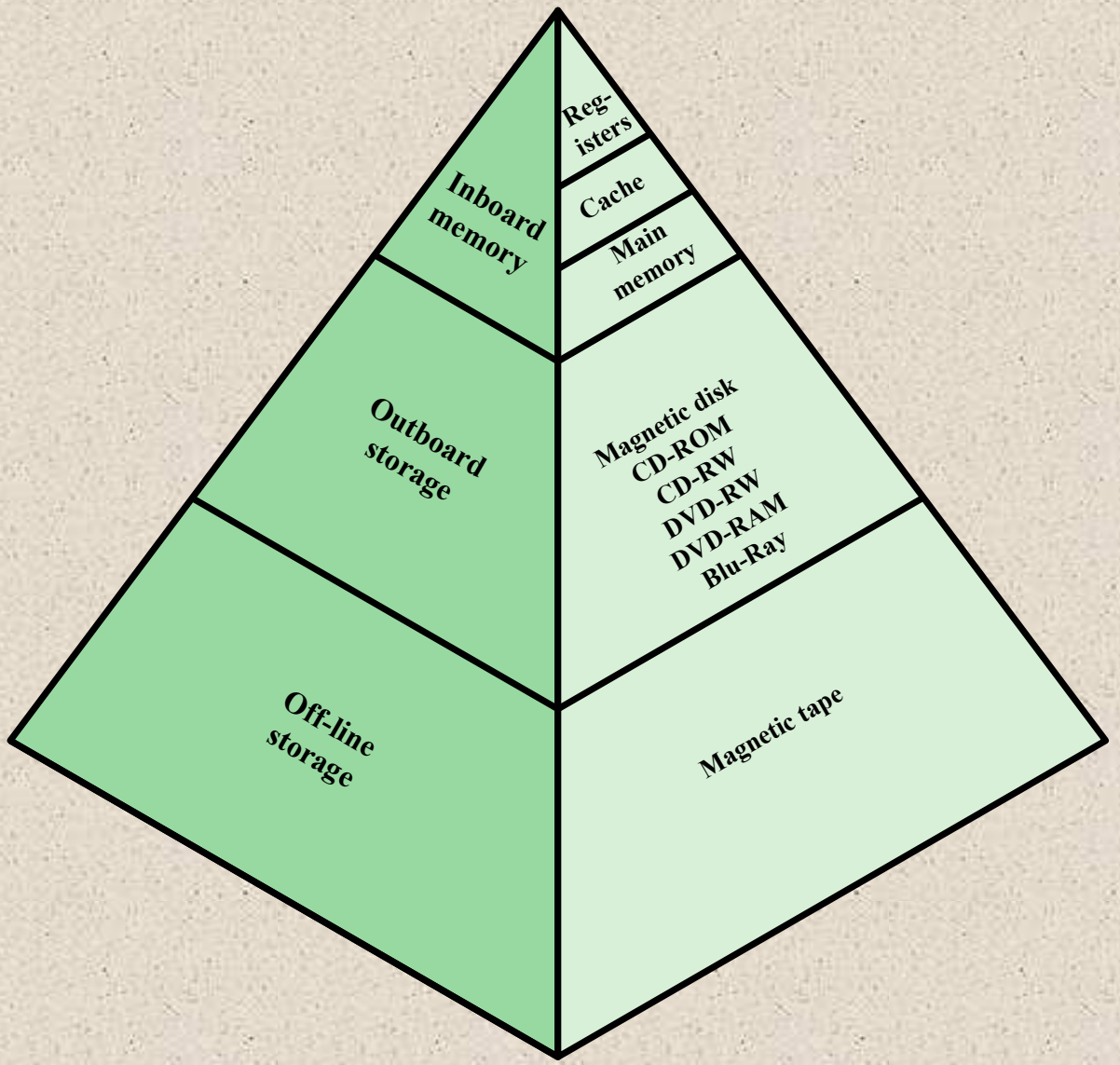


Figure 4.1 The Memory Hierarchy

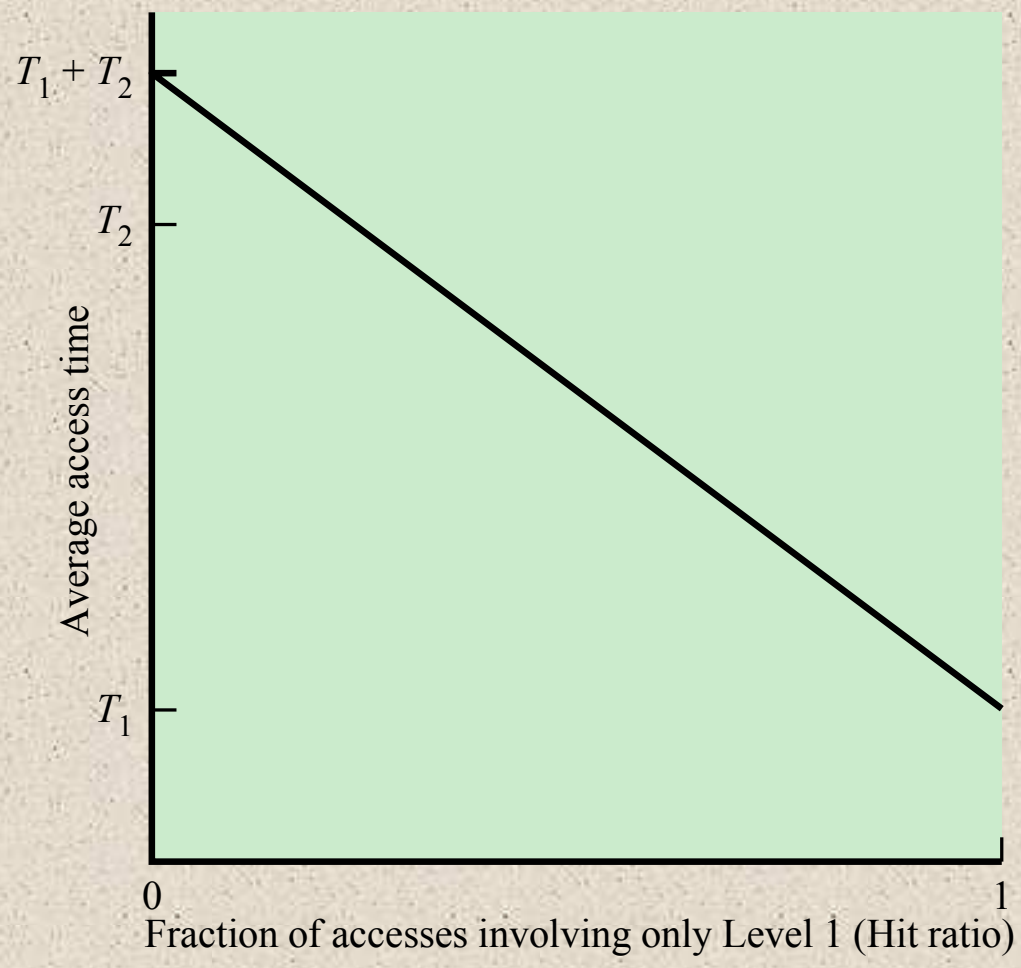
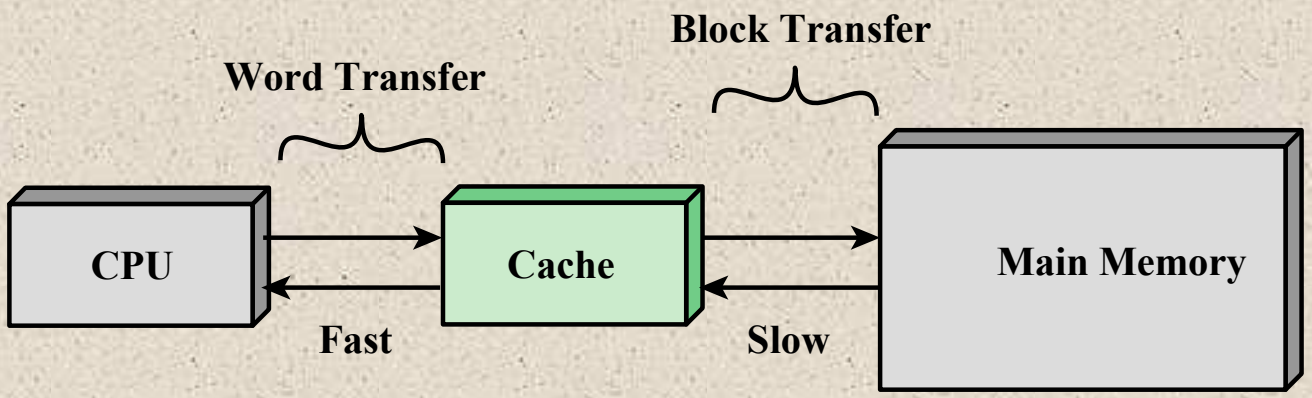


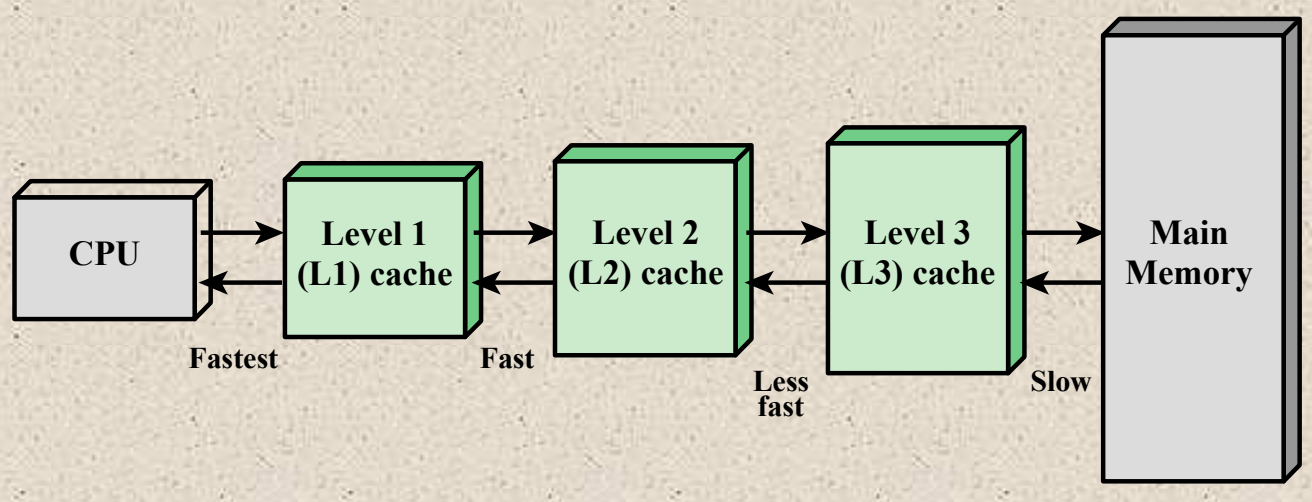
Figure 4.2 Performance of a Simple Two-Level Memory

Memory

- The use of three levels exploits the fact that semiconductor memory comes in a variety of types which differ in speed and cost
- Data are stored more permanently on external mass storage devices
- External, nonvolatile memory is also referred to as **secondary** memory or **auxiliary** memory
- Disk cache
  - A portion of main memory can be used as a buffer to hold data temporarily that is to be read out to disk
  - A few large transfers of data can be used instead of many small transfers of data
  - Data can be retrieved rapidly from the software cache rather than slowly from the disk



(a) Single cache



(b) Three-level cache organization

Figure 4.3 Cache and Main Memory

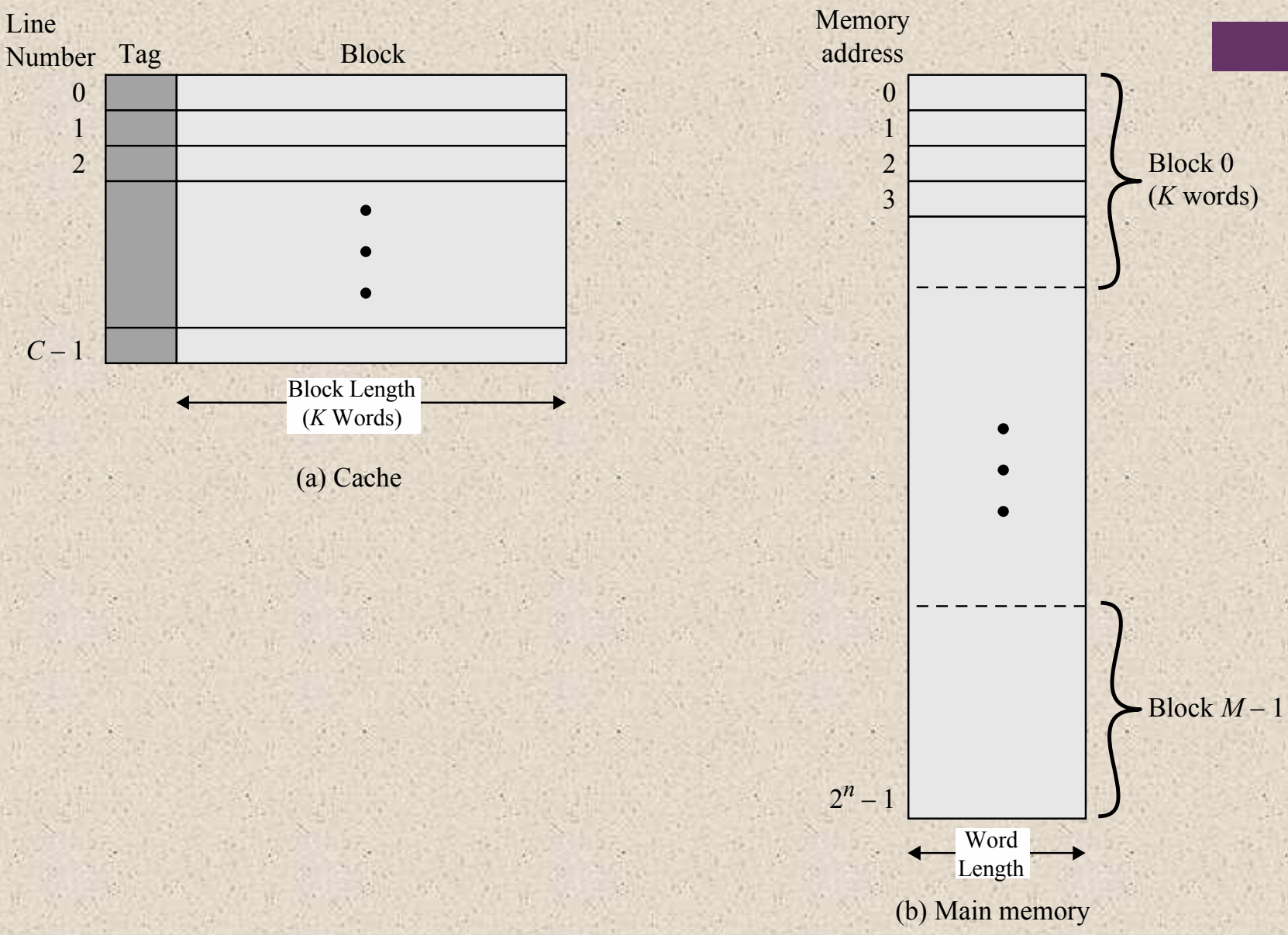


Figure 4.4 Cache/Main-Memory Structure

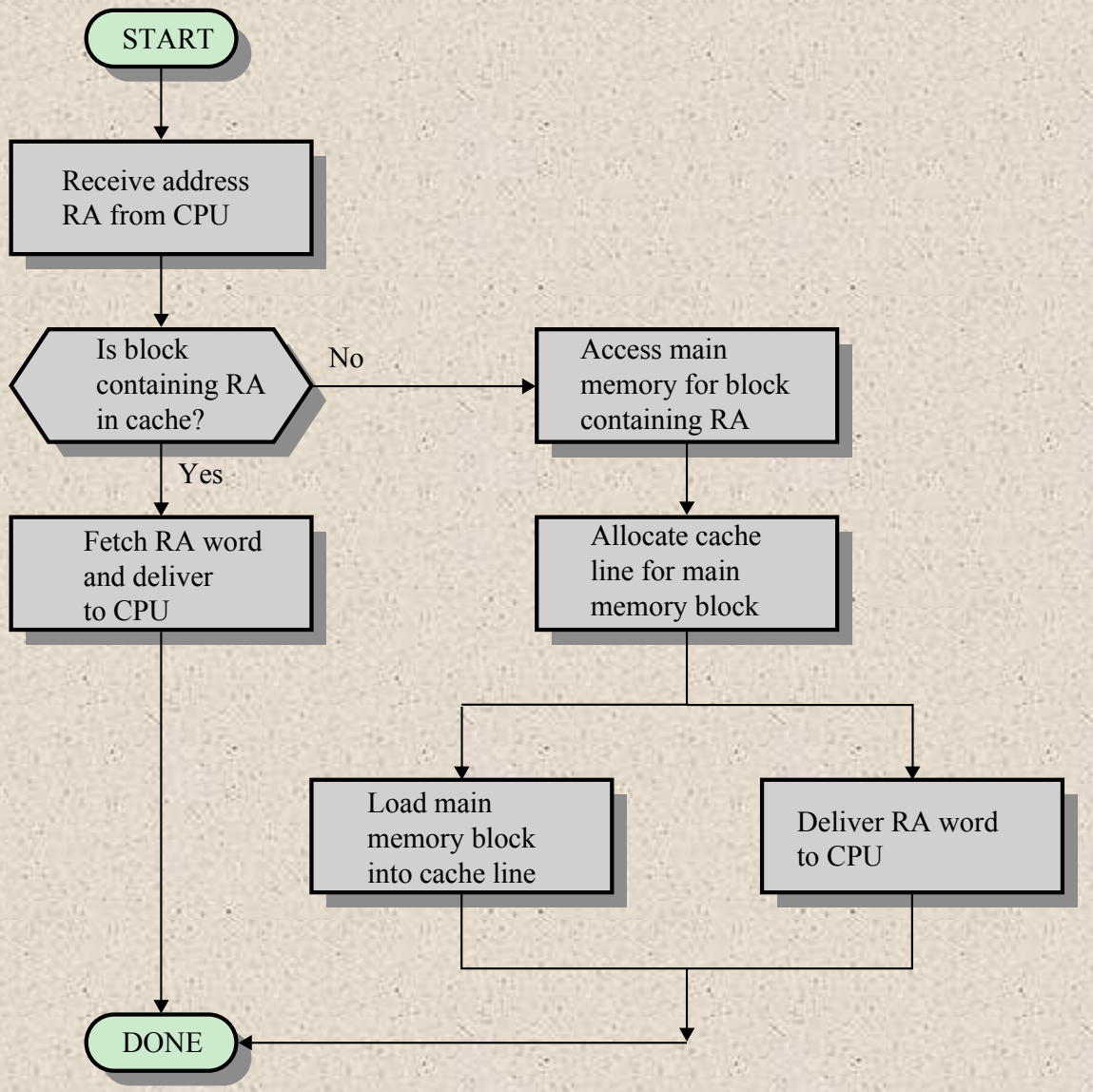


Figure 4.5 Cache Read Operation

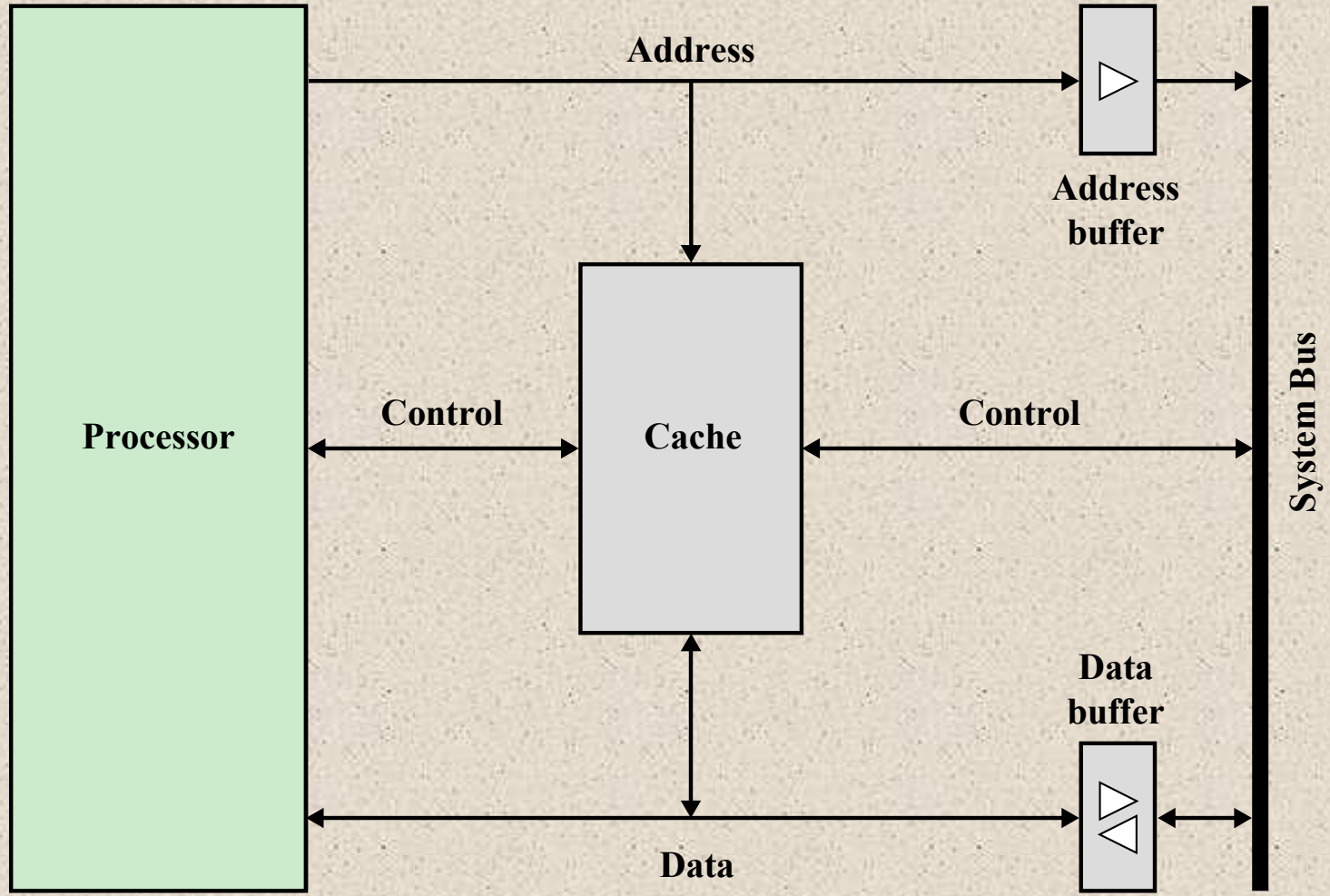


Figure 4.6 Typical Cache Organization

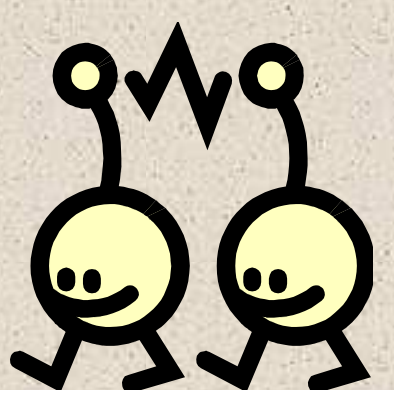
Cache Addresses	**Write Policy**
Logical	Write through
Physical	Write back
Cache Size	**Line Size**
Mapping Function	**Number of caches**
Direct	Single or two level
Associative	Unified or split
Set Associative	
Replacement Algorithm	
Least recently used (LRU)	
First in first out (FIFO)	
Least frequently used (LFU)	
Random	

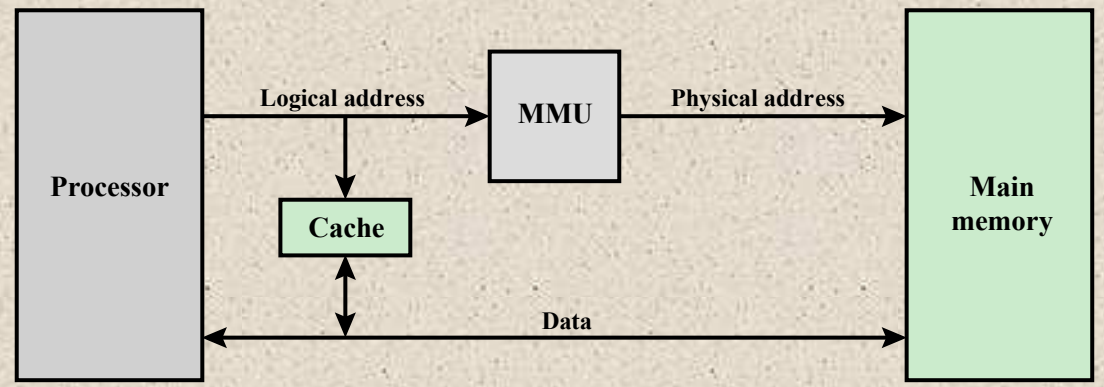
Table 4.2
Elements of Cache Design

Cache Addresses

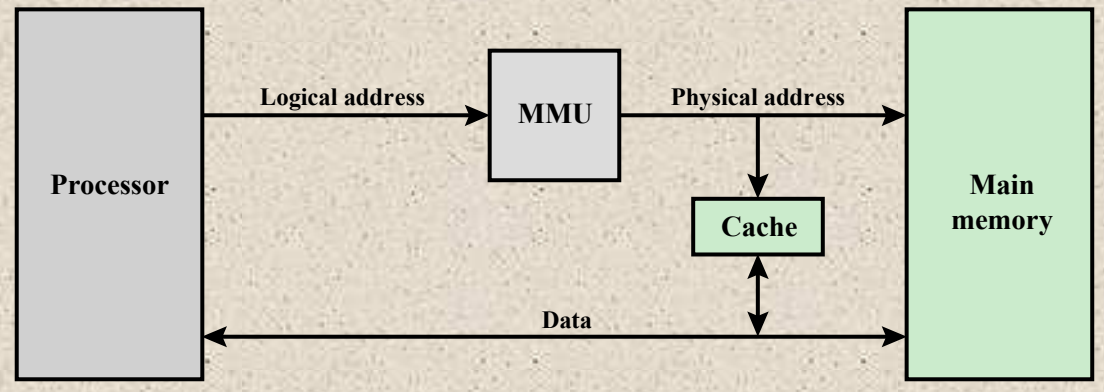
Virtual Memory

- 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





(a) Logical Cache



(b) Physical Cache

Figure 4.7 Logical and Physical Caches

Processor	Type	Year of Introduction	L1 Cachea	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 MTAb	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

Table 4.3

Cache Sizes of Some Processors

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

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

(Table can be found on page 134 in the textbook.)

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

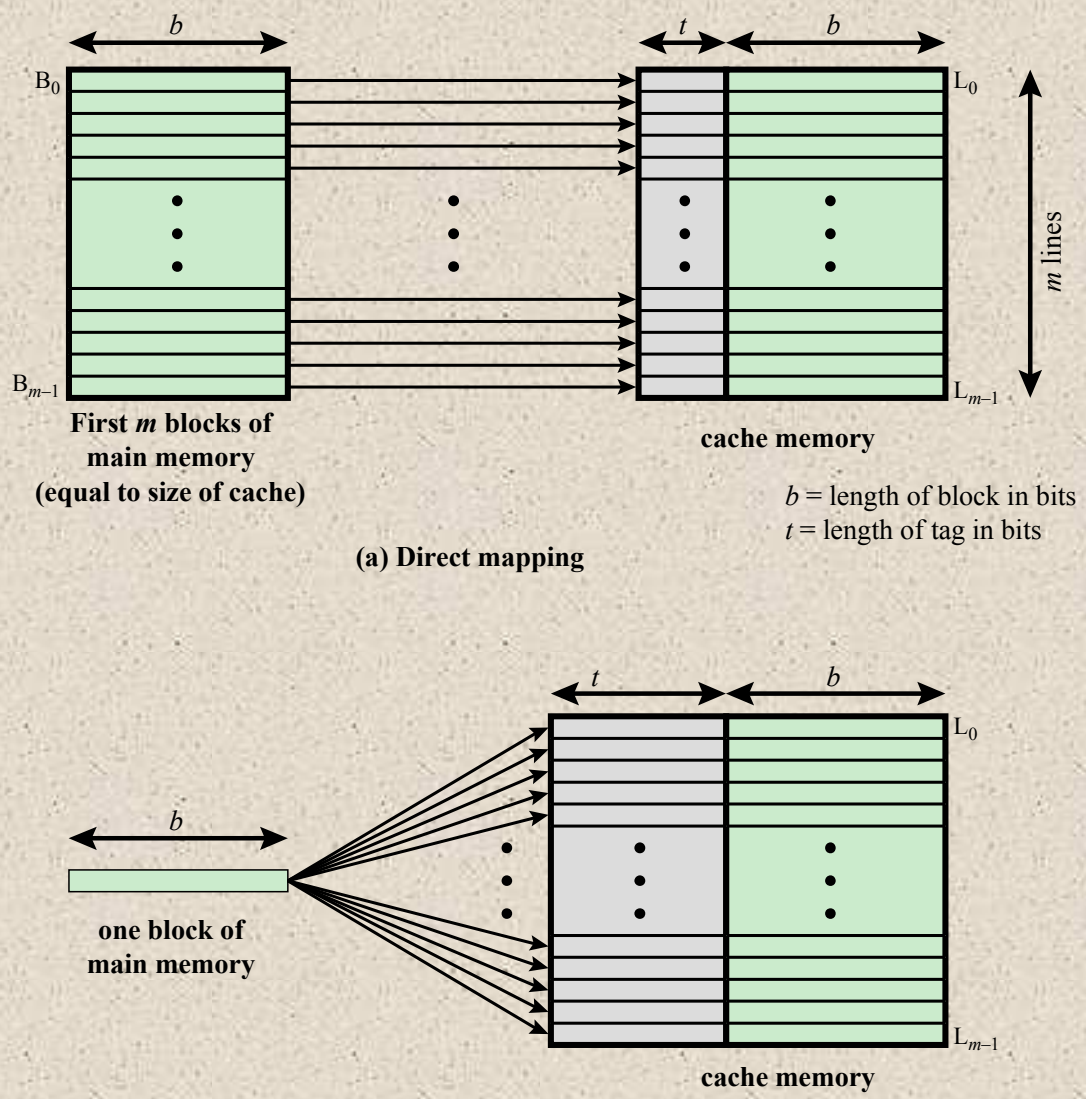


Figure 4.8 Mapping From Main Memory to Cache: Direct and Associative

Simple Direct Mapping Cache

Memory Address Space								
Decimal	Tag		Line			Word	Block
0	0	0	0	0	0	0	Block 0
1	0	0	0	0	0	1	
2	0	0	0	0	1	0	Block 1
3	0	0	0	0	1	1	
4	0	0	0	1	0	0	Block 2
5	0	0	0	1	0	1	
6	0	0	0	1	1	0	Block 3
7	0	0	0	1	1	1	
8	0	0	1	0	0	0	Block 4
9	0	0	1	0	0	1	
10	0	0	1	0	1	0	Block 5
11	0	0	1	0	1	1	
12	0	0	1	1	0	0	Block 6
13	0	0	1	1	0	1	
14	0	0	1	1	1	0	Block 7
15	0	0	1	1	1	1	
16	0	1	0	0	0	0	Block 8
17	0	1	0	0	0	1	
18	0	1	0	0	1	0	Block 9
19	0	1	0	0	1	1	
20	0	1	0	1	0	0	Block 10
21	0	1	0	1	0	1	
22	0	1	0	1	1	0	Block 11
23	0	1	0	1	1	1	
24	0	1	1	0	0	0	Block 12
25	0	1	1	0	0	1	
26	0	1	1	0	1	0	Block 13
27	0	1	1	0	1	1	
28	0	1	1	1	0	0	Block 14
29	0	1	1	1	0	1	
30	0	1	1	1	1	0	Block 15
31	0	1	1	1	1	1	

Cache		
Line	Word 0	Word 1
0		
1		
2		
3		
4		
5		
6		
7		

SIMPLE EXAMPLE DESIGN PARAMETERS

Word: 1 bit
Line 3 bits
Tag: 2 bits

Block Size: 2^1 or 2 words

Number of Blocks: 16

Number of Lines: 2^3 or 8

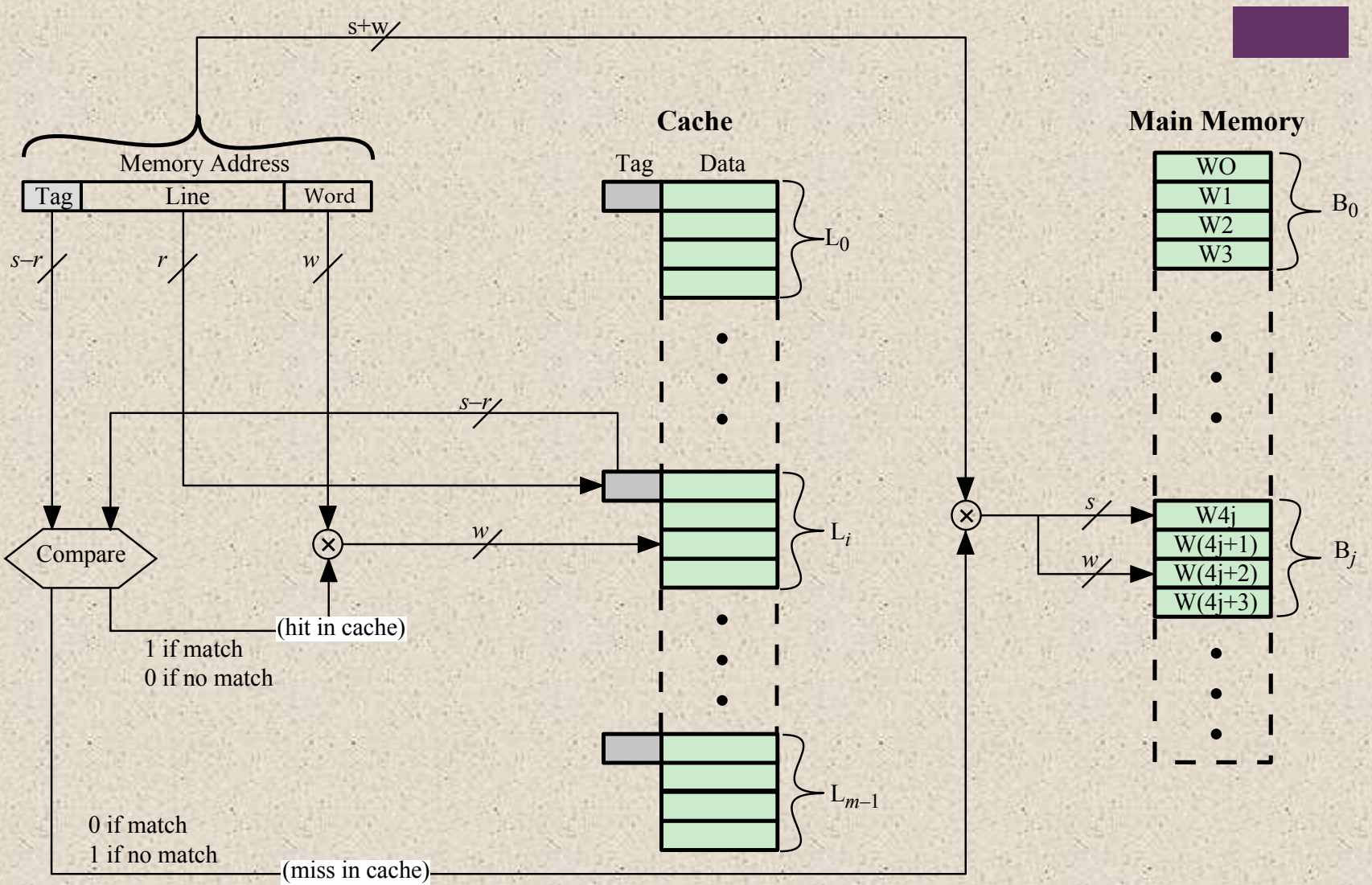
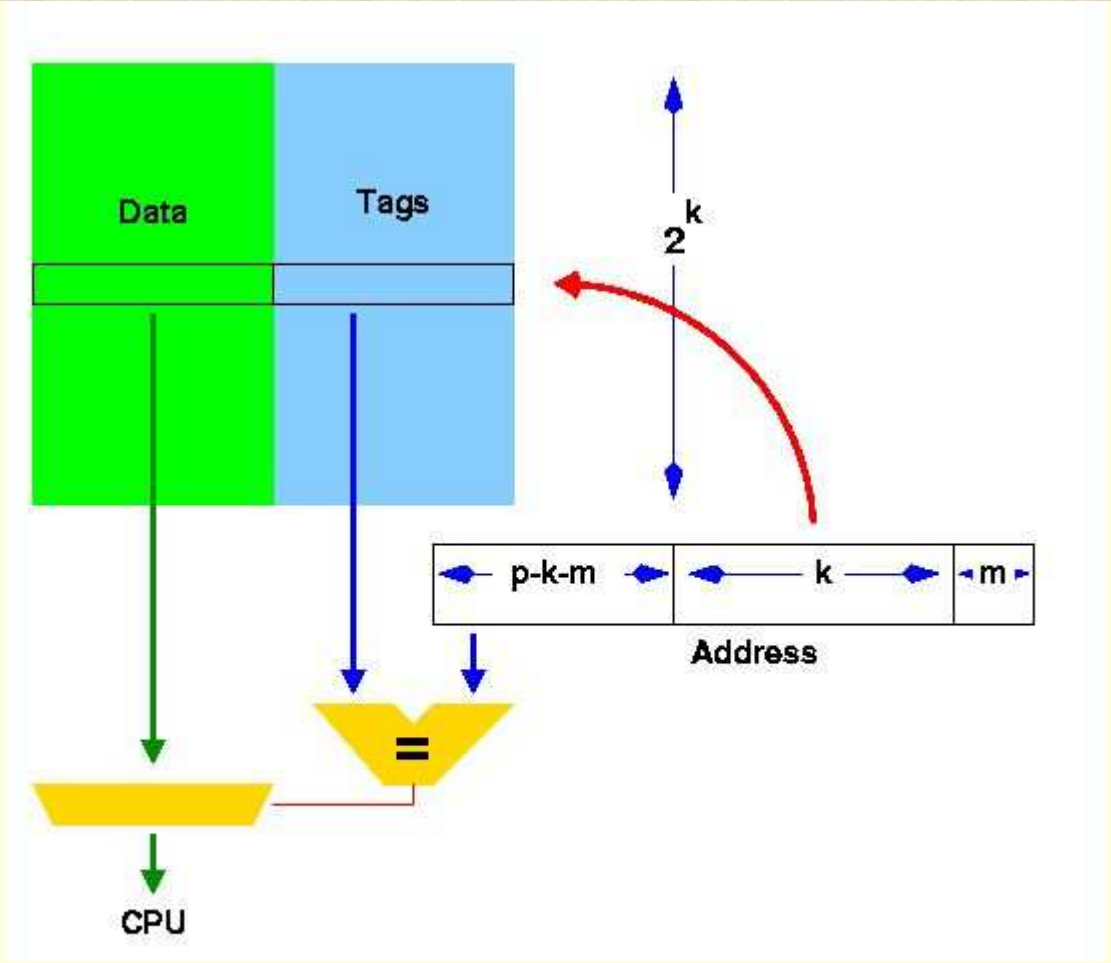


Figure 4.9 Direct-Mapping Cache Organization



Direct Mapped Cache

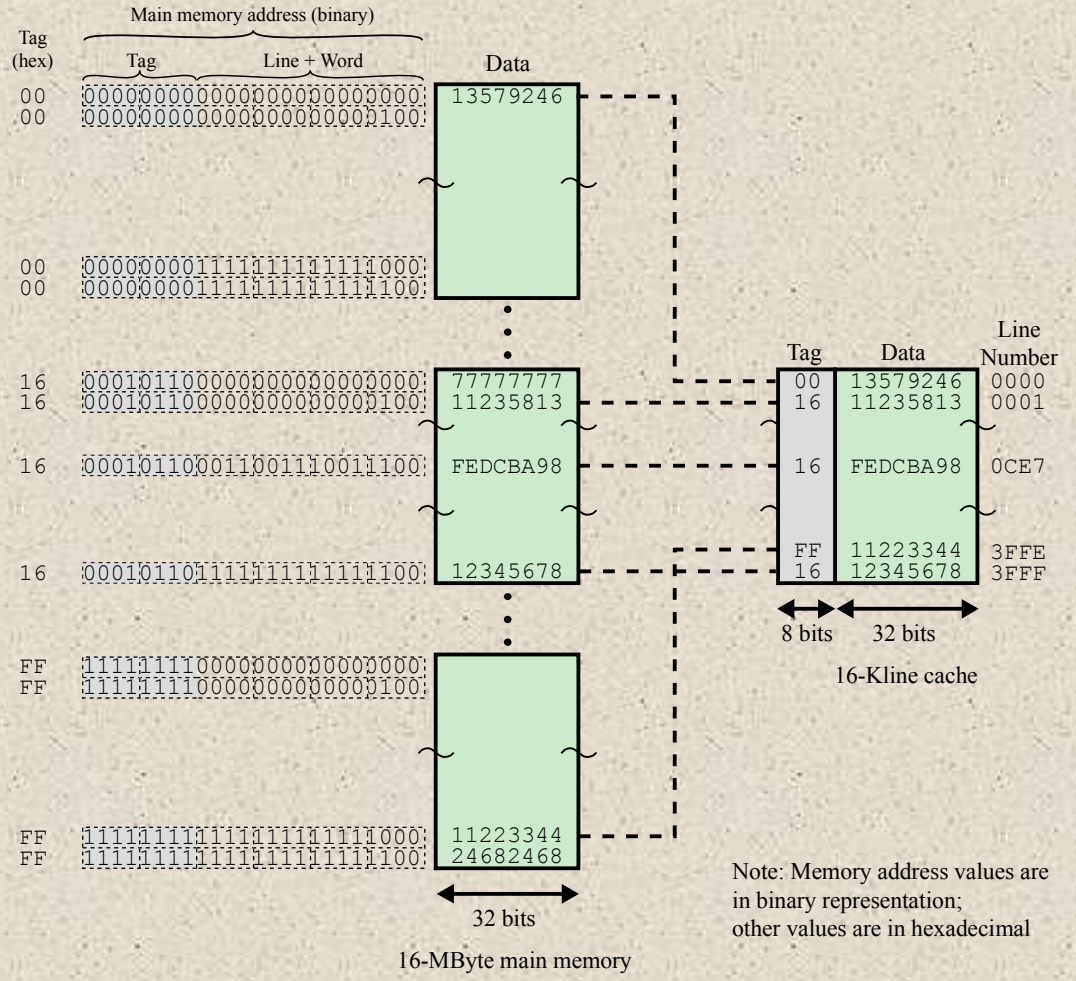


Figure 4.10 Direct Mapping Example

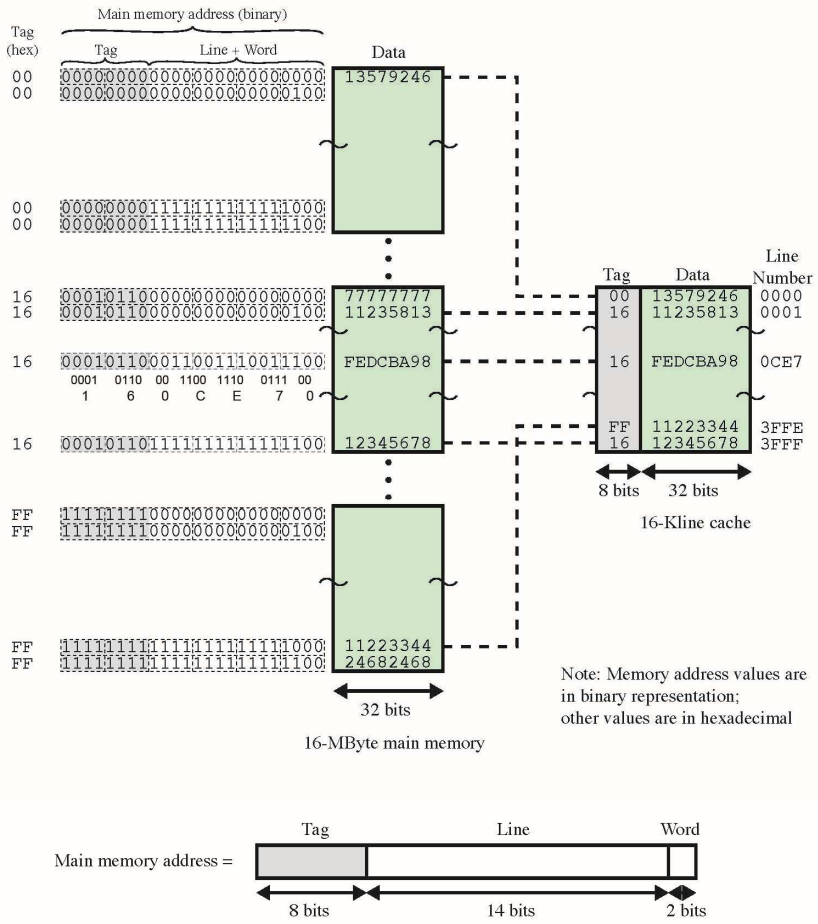


Figure 4.10 Direct Mapping Example

Direct Mapping Summary

- Address length = (s + w) bits **determines memory address space**
- Number of addressable units = 2^{s+w} words or bytes **memory size**
- Block size = line size = 2^{w} words or bytes **specified by designer**
- Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$
- Number of lines in cache = m = 2^{r} **"m" is specified by designer**
- 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

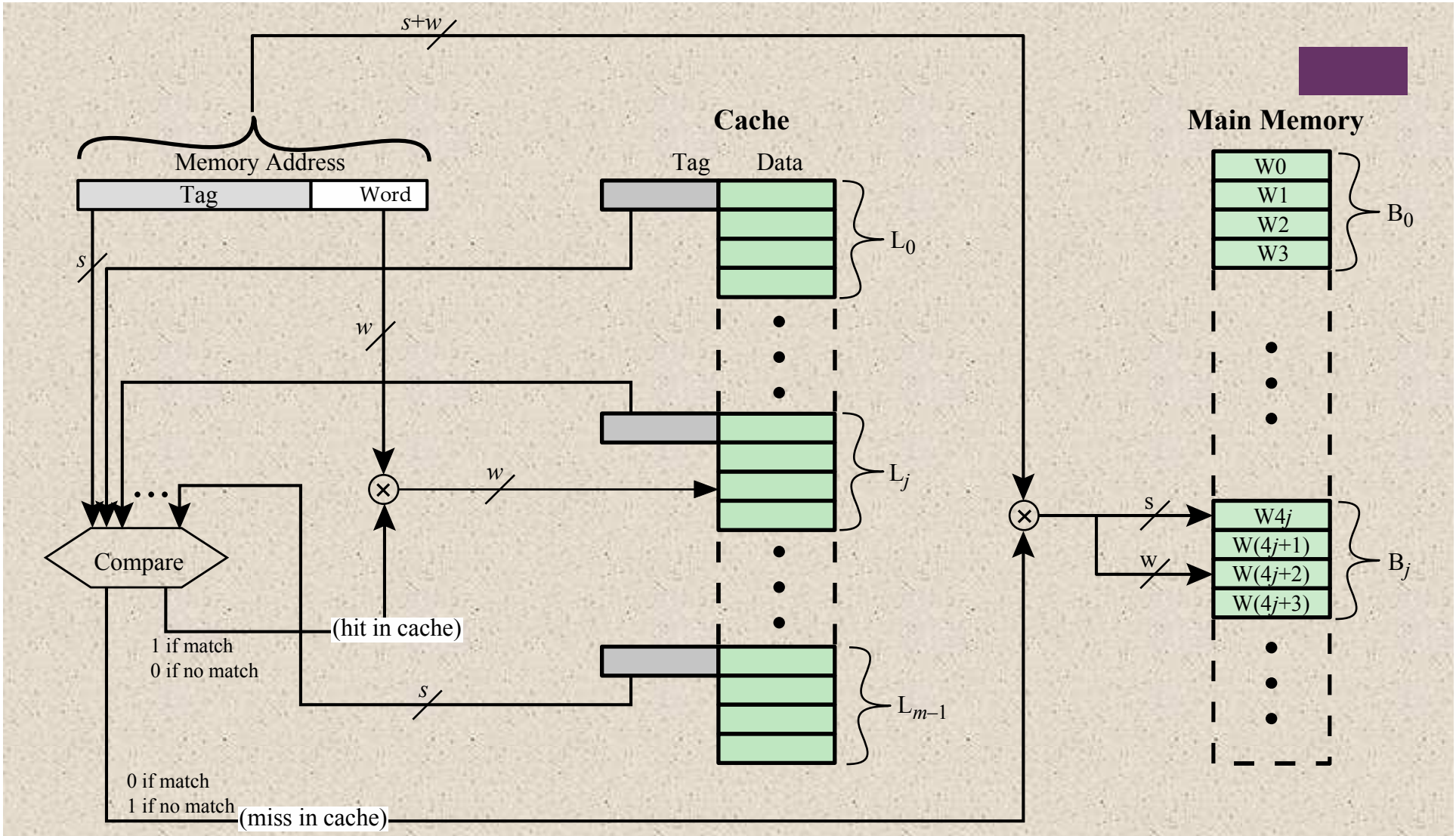
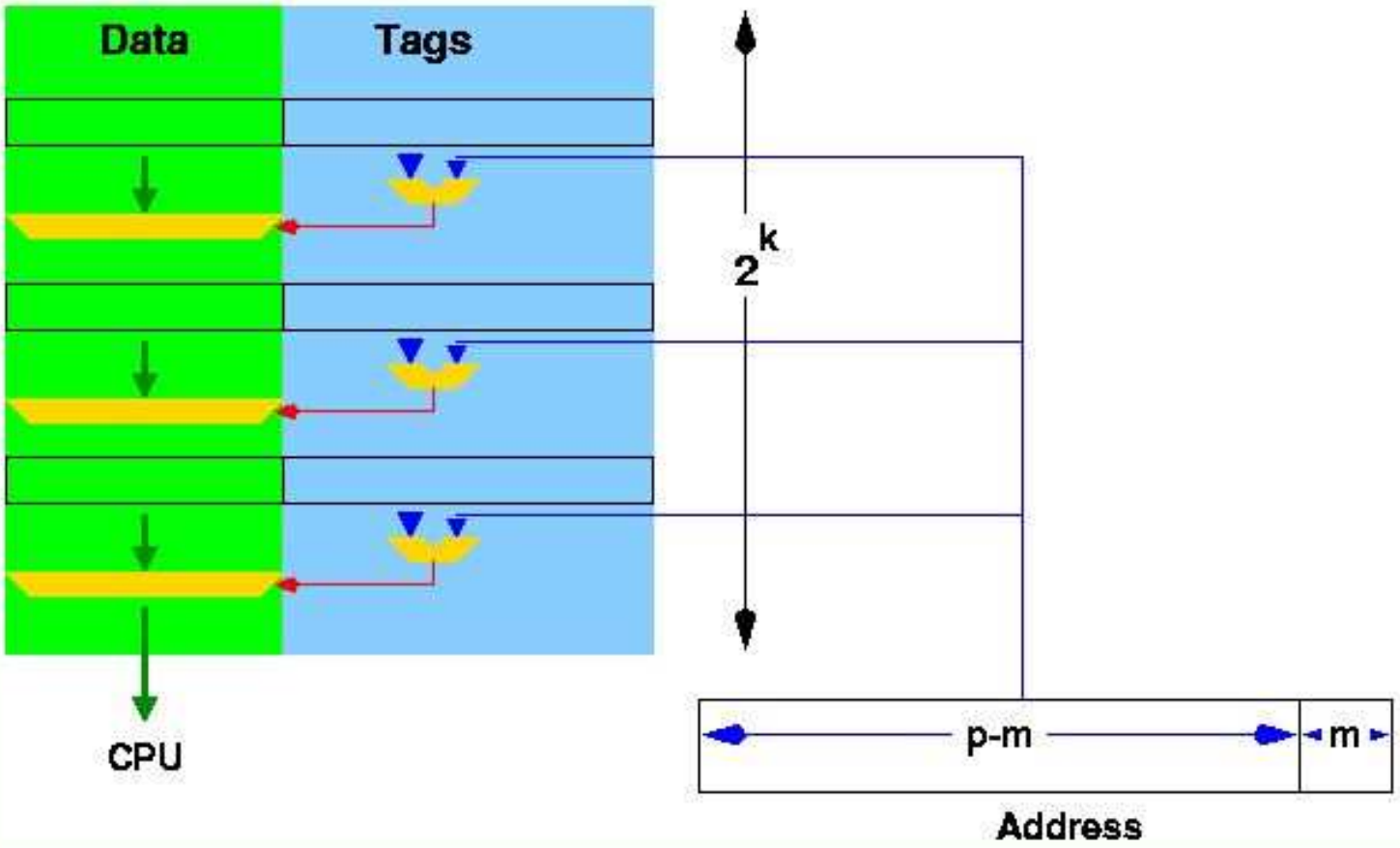
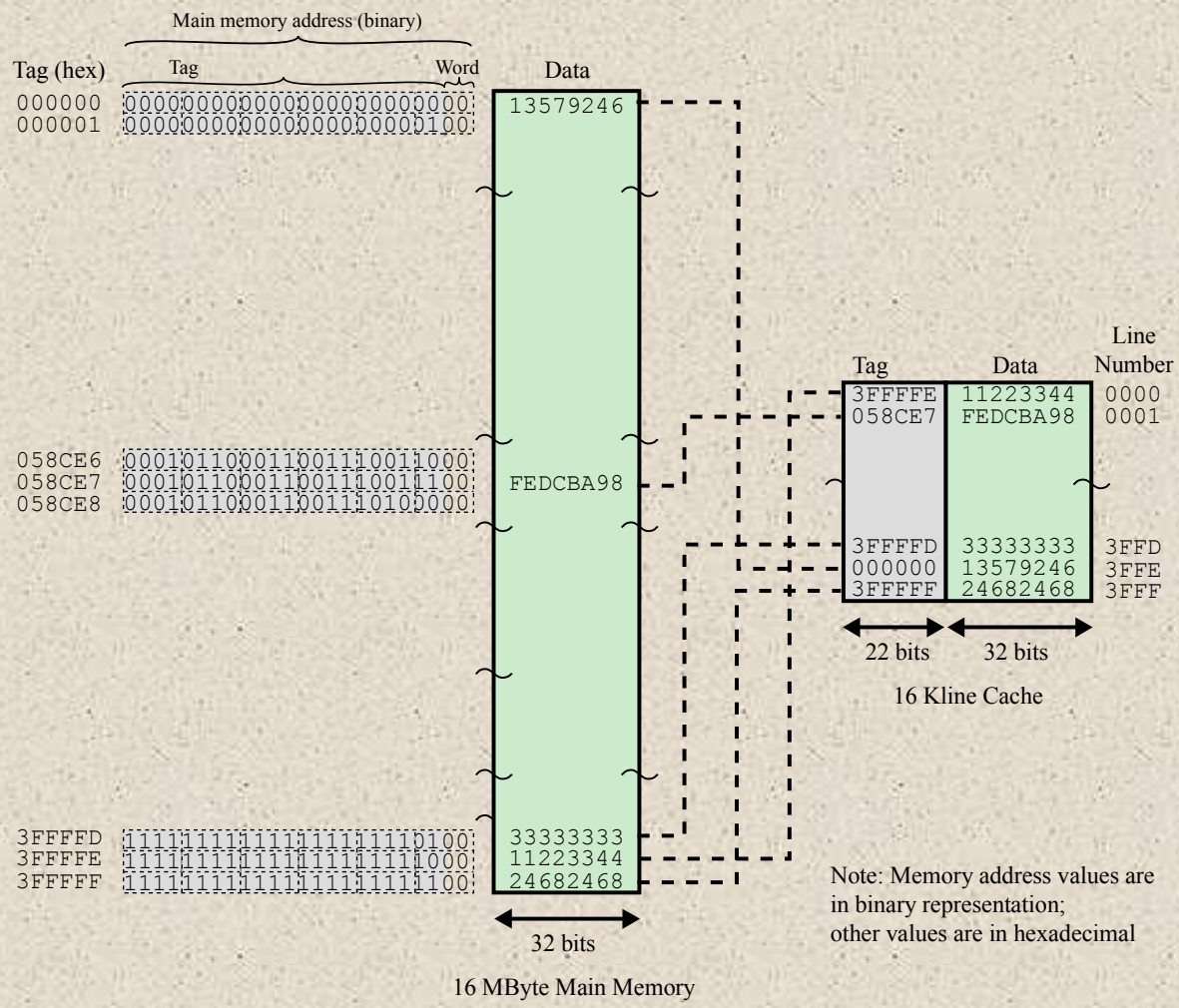


Figure 4.11 Fully Associative Cache Organization



Fully Associative Cache



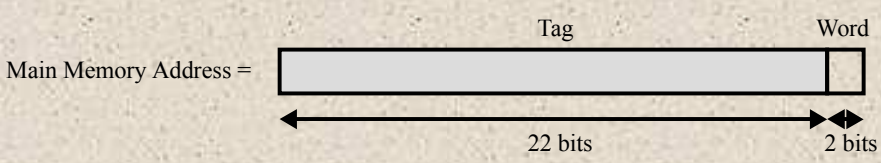


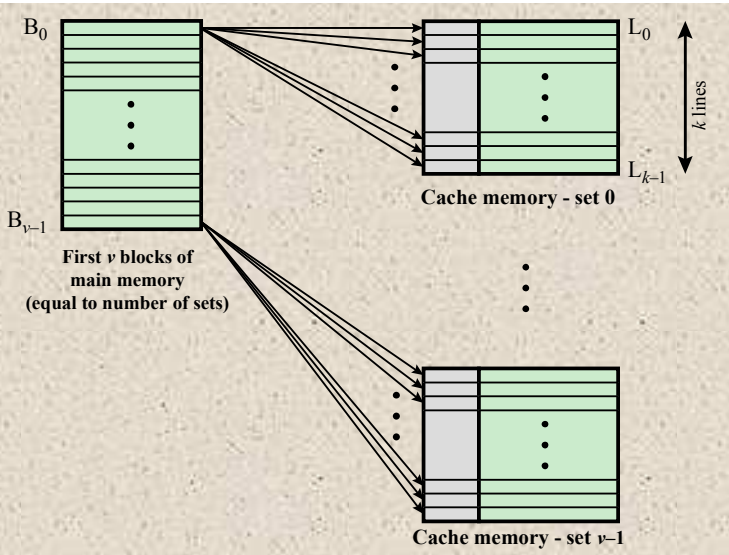
Figure 4.12 Associative Mapping Example

Associative Mapping Summary

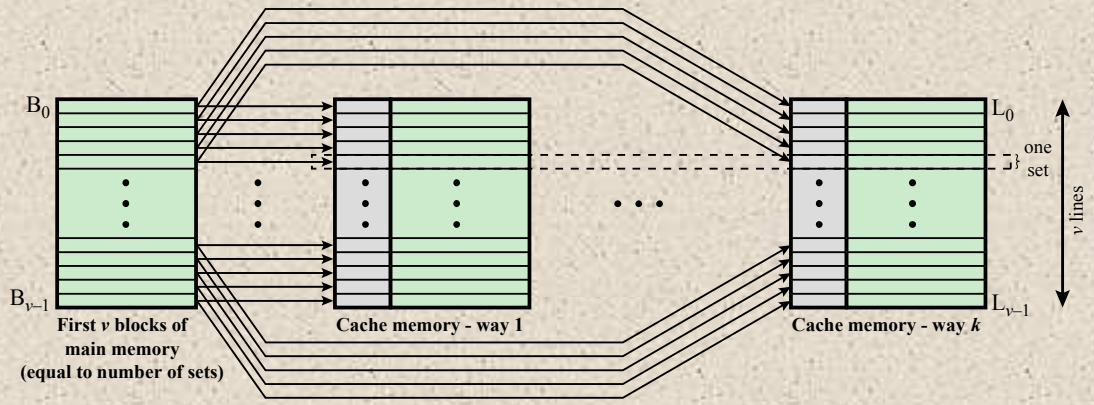
- Address length = (s + w) bits **determines memory address space**
- Number of addressable units = 2^{s+w} words or bytes **memory size**
- Block size = line size = 2^{w} words or bytes **specified by designer**
- Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$
- Number of lines in cache = undetermined **specified by designer**
- 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



(a) v associative-mapped caches



(b) k direct-mapped caches

Figure 4.13 Mapping From Main Memory to Cache: k-way Set Associative

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 **k is determined by designer**
- Number of sets = v = 2^{d} **d is determined by designer**
- 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

Simple Set Associative Cache

Memory Address Space								
Memory Address	Tag		Set			Word	Block
0	0	0	0	0	0	0	Block 0
1	0	0	0	0	0	1	
2	0	0	0	0	1	0	Block 1
3	0	0	0	0	1	1	
4	0	0	0	1	0	0	Block 2
5	0	0	0	1	0	1	
6	0	0	0	1	1	0	Block 3
7	0	0	0	1	1	1	
8	0	0	1	0	0	0	Block 4
9	0	0	1	0	0	1	
10	0	0	1	0	1	0	Block 5
11	0	0	1	0	1	1	
12	0	0	1	1	0	0	Block 6
13	0	0	1	1	0	1	
14	0	0	1	1	1	0	Block 7
15	0	0	1	1	1	1	
16	0	1	0	0	0	0	Block 8
17	0	1	0	0	0	1	
18	0	1	0	0	1	0	Block 9
19	0	1	0	0	1	1	
20	0	1	0	1	0	0	Block 10
21	0	1	0	1	0	1	
22	0	1	0	1	1	0	Block 11
23	0	1	0	1	1	1	
24	0	1	1	0	0	0	Block 12
25	0	1	1	0	0	1	
26	0	1	1	0	1	0	Block 13
27	0	1	1	0	1	1	
28	0	1	1	1	0	0	Block 14
29	0	1	1	1	0	1	
30	0	1	1	1	1	0	Block 15
31	0	1	1	1	1	1	

K = 1		Set	K = 2	
Word 0	Word 1		Word 0	Word 1
		0		
		1		
		2		
		3		
		4		
		5		
		6		
		7		

SIMPLE EXAMPLE DESIGN PARAMETERS

Word: 1 bit
Set: 3 bits
Tag: 2 bits
Keys: 2 with 2 words/line

Block Size: 2^1 or 2 words

Number of Blocks: 16

Number of Sets: 2^3 or 8

Simple Set Associative Cache Worksheet

Memory Address Space

Memory Address	Tag		Set			Word	
0	0	0	0	0	0	0	Block
1	0	0	0	0	0	1	
2	0	0	0	0	1	0	Block
3	0	0	0	0	1	1	
4	0	0	0	1	0	0	Block
5	0	0	0	1	0	1	
6	0	0	0	1	1	0	Block
7	0	0	0	1	1	1	
8	0	0	1	0	0	0	Block
9	0	0	1	0	0	1	
10	0	0	1	0	1	0	Block
11	0	0	1	0	1	1	
12	0	0	1	1	0	0	Block
13	0	0	1	1	0	1	
14	0	0	1	1	1	0	Block
15	0	0	1	1	1	1	
16	0	1	0	0	0	0	Block
17	0	1	0	0	0	1	
18	0	1	0	0	1	0	Block
19	0	1	0	0	1	1	
20	0	1	0	1	0	0	Block
21	0	1	0	1	0	1	
22	0	1	0	1	1	0	Block
23	0	1	0	1	1	1	
24	0	1	1	0	0	0	Block
25	0	1	1	0	0	1	
26	0	1	1	0	1	0	Block
27	0	1	1	0	1	1	
28	0	1	1	1	0	0	Block
29	0	1	1	1	0	1	
30	0	1	1	1	1	0	Block
31	0	1	1	1	1	1	

Key 1				
v	d	Tag	Word 0	Word 1

Set
0
1
2
3
4
5
6
7

Key 2				
v	d	Tag	Word 0	Word 1

DATA WORDS NOTATION

W0-A	W1-A

W0-A	W1-A

WHERE "A" IS THE MEMORY ADDRESS

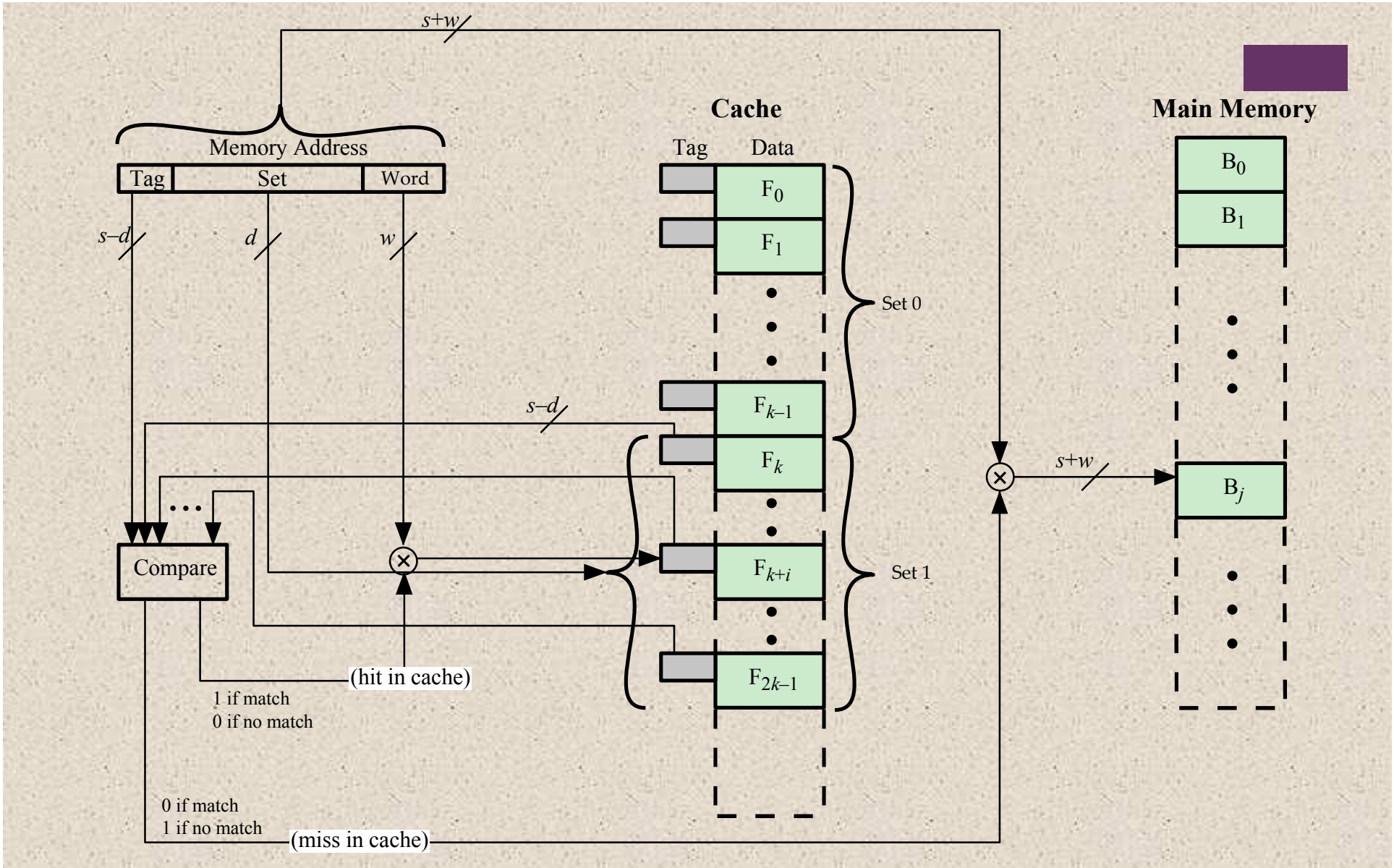
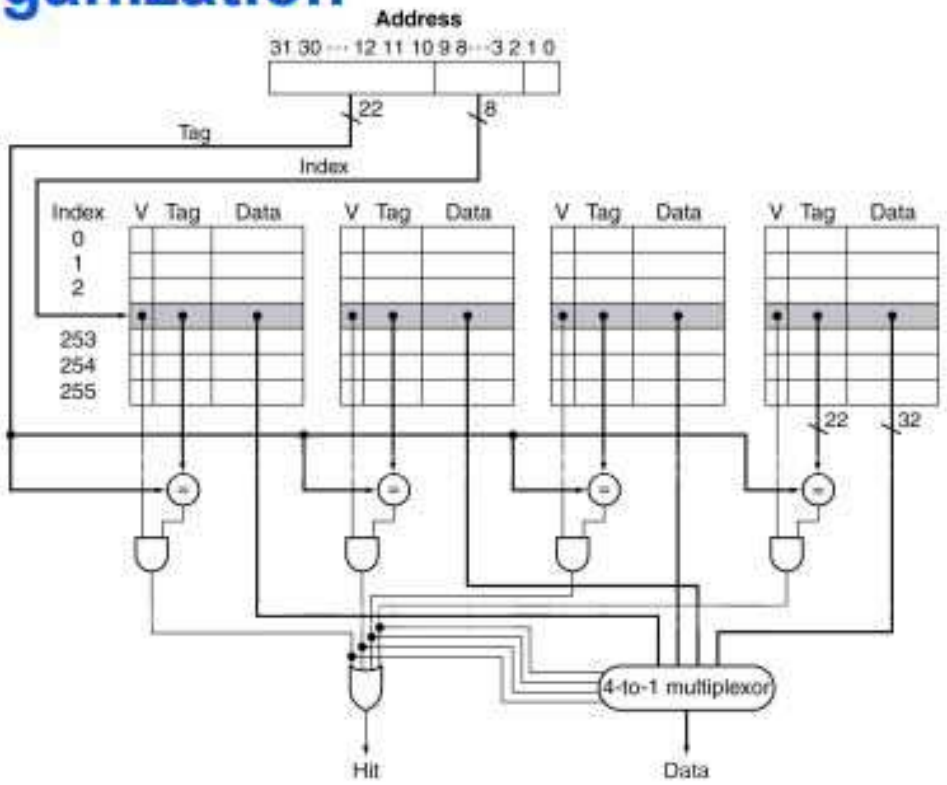


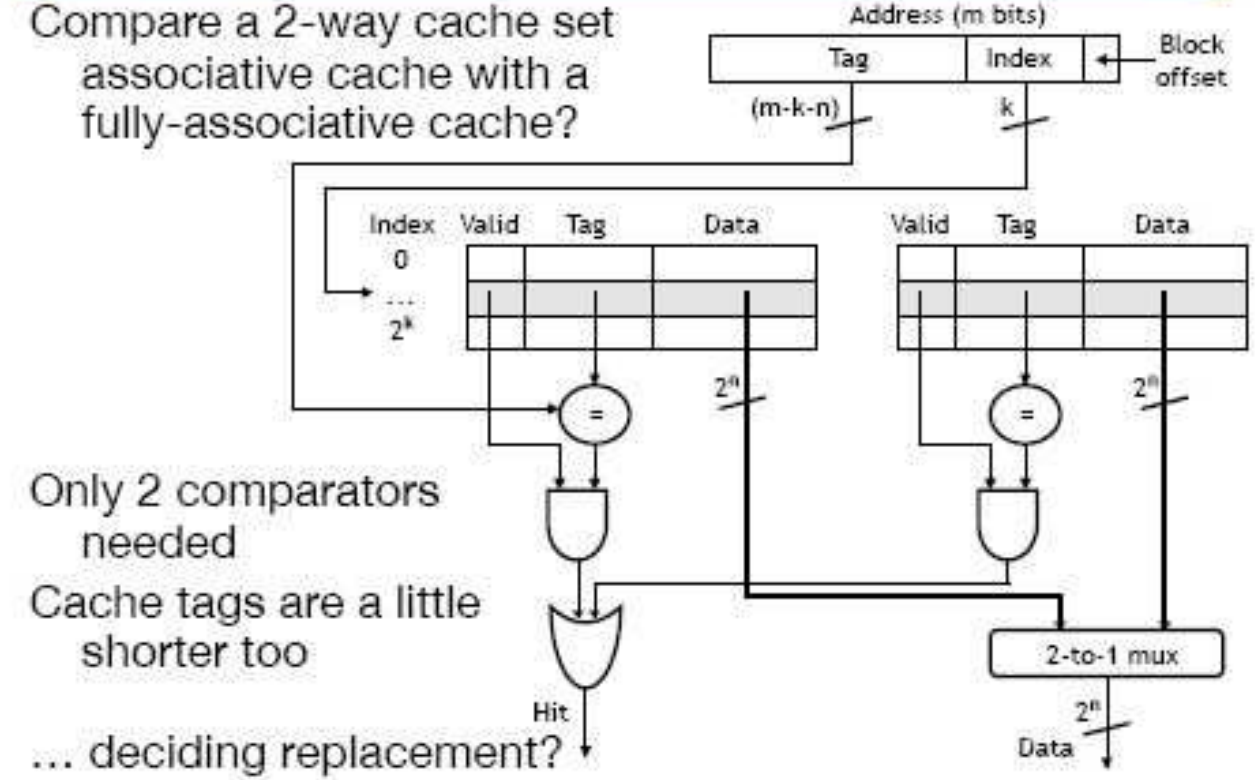
Figure 4.14 *k*-Way Set Associative Cache Organization

Set Associative Cache Organization



2-way set associative implementation

Compare a 2-way cache set associative cache with a fully-associative cache?

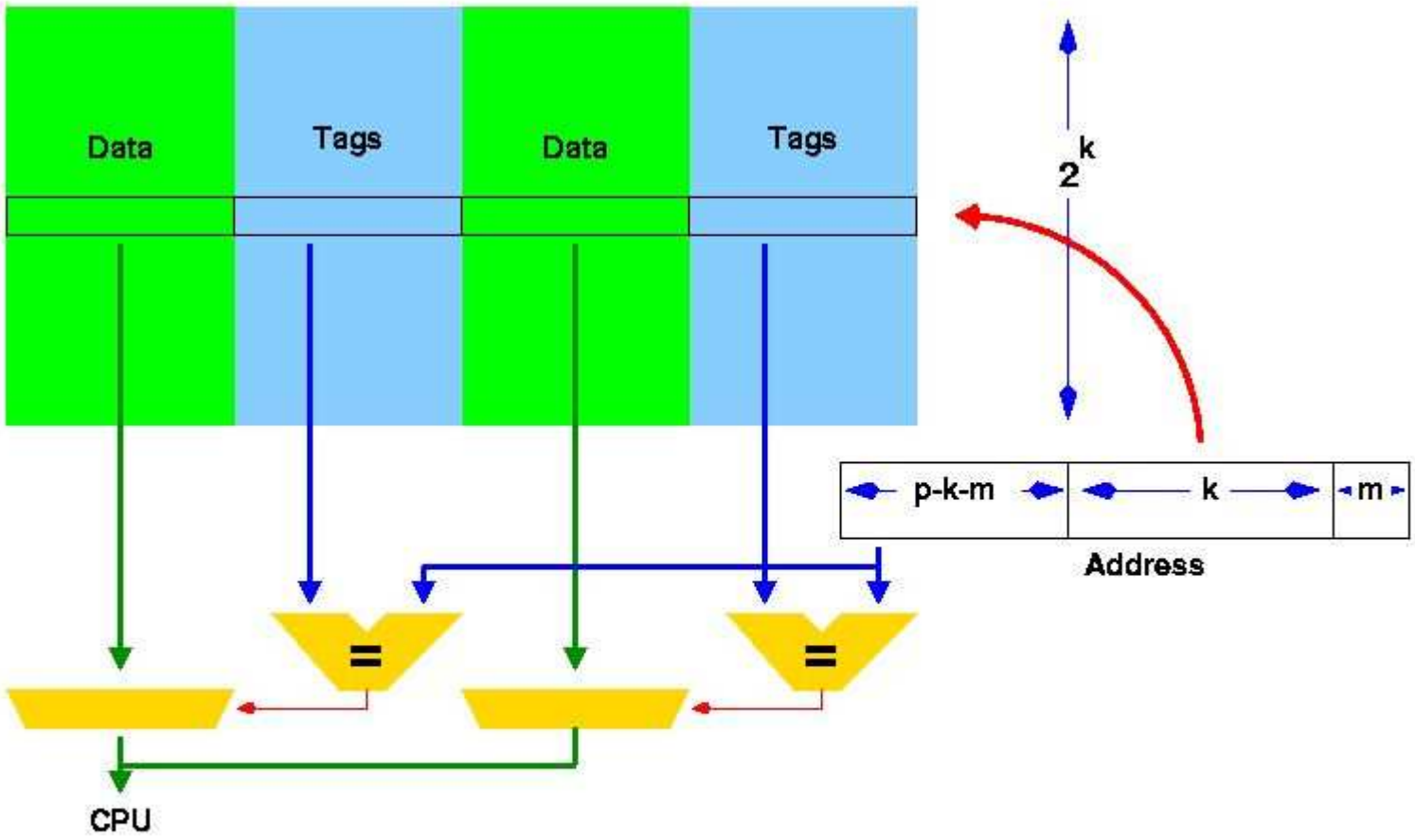


Only 2 comparators needed

Cache tags are a little shorter too

... deciding replacement?

2



Set Associative Cache

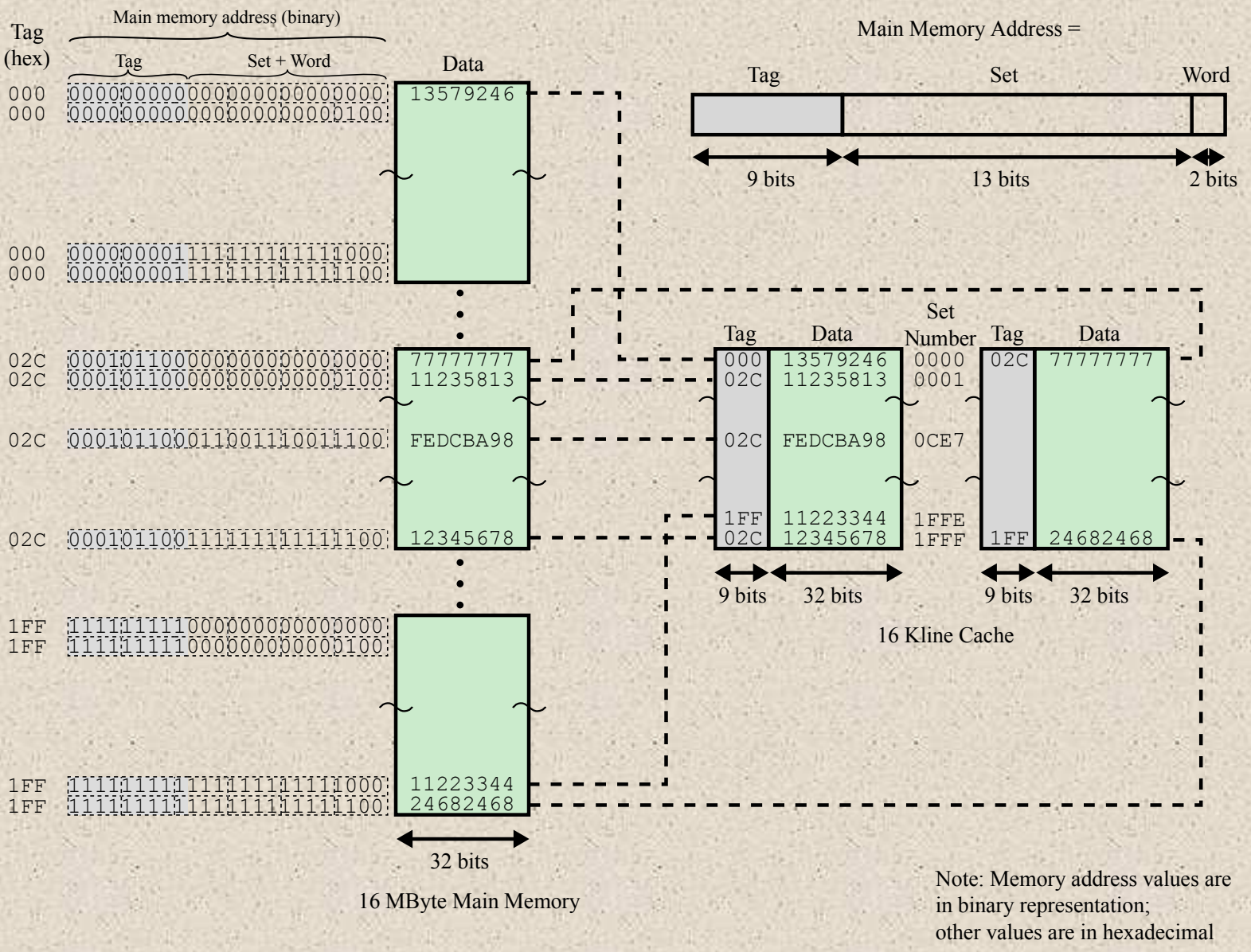


Figure 4.15 Two-Way Set Associative Mapping Example

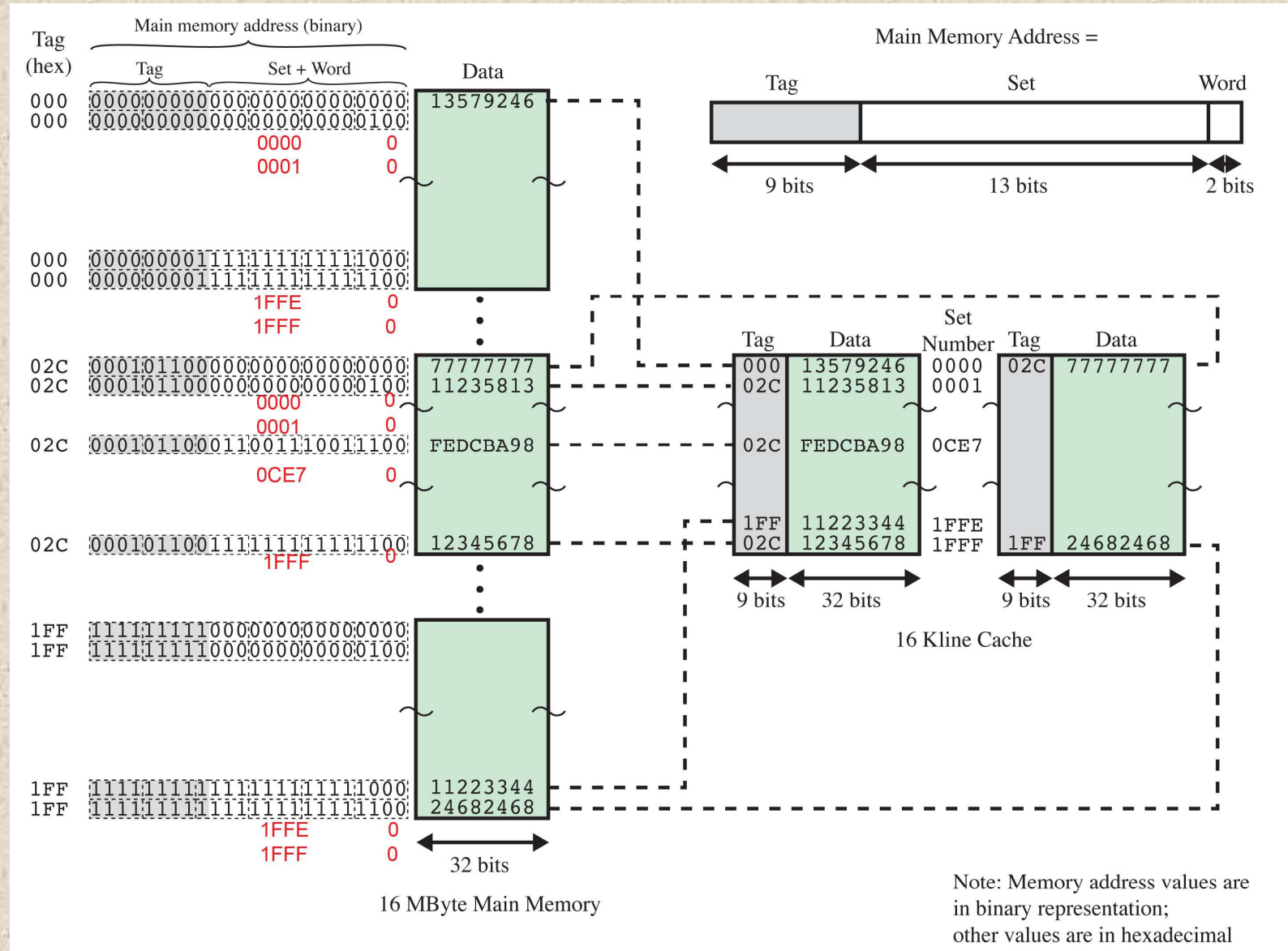


Figure 4.15 Two-Way Set Associative Mapping Example

Figure 4.15 shows our example using set-associative mapping with two lines in each set, referred to as two-way set-associative.

- **The 13-bit set number identifies a unique set of two lines within the cache. It also gives the number of the block in main memory, modulo 2^{13}. This determines the mapping of blocks into lines.**
- **Thus, blocks 000000, 008000, …, FF8000 of main memory map into cache set 0.**

- **Any of those blocks can be loaded into either of the two lines in the set. Note that no two blocks that map into the same cache set have the same tag number.**

- **For a read operation, the 13-bit set number is used to determine which set of two lines is to be examined.**

- **Both lines in the set are examined for a match with the tag number of the address to be accessed. In the extreme case of v = m , k = 1, the set-associative technique reduces to direct mapping, and for v = 1, k = m , it reduces to associative mapping.**

- **The use of two lines per set (v = m /2, k = 2) is the most common set-associative organization. It significantly improves the hit ratio over direct mapping.**

- **Four- way set associative (v = m /4, k = 4) makes a modest additional improvement for a relatively small additional cost [MAYB84, HILL89]. Further increases in the number of lines per set have little effect.**

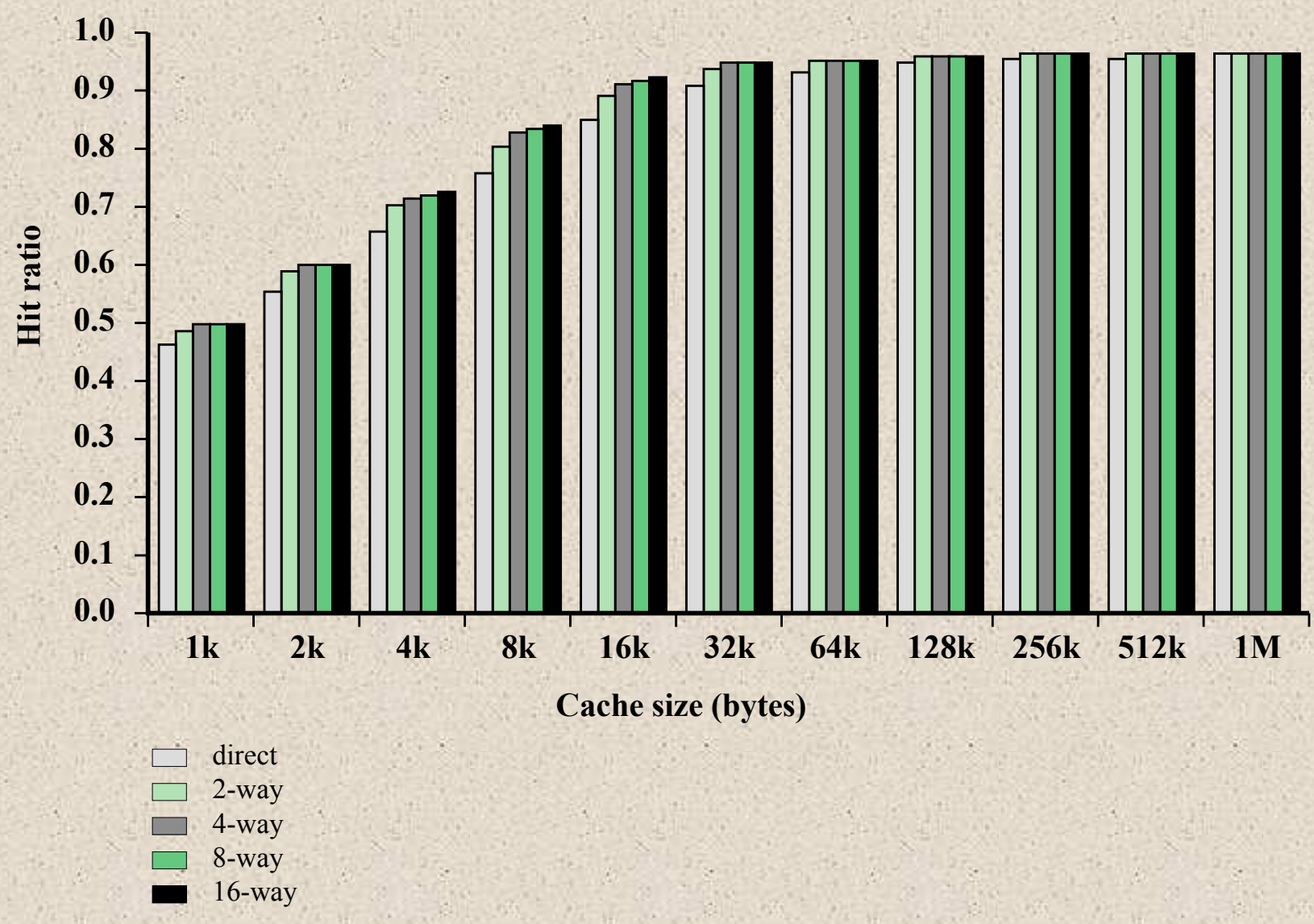
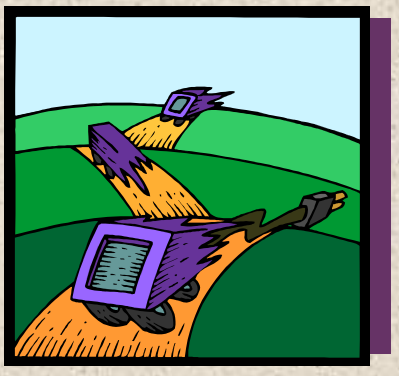


Figure 4.16 Varying Associativity over Cache Size

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 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

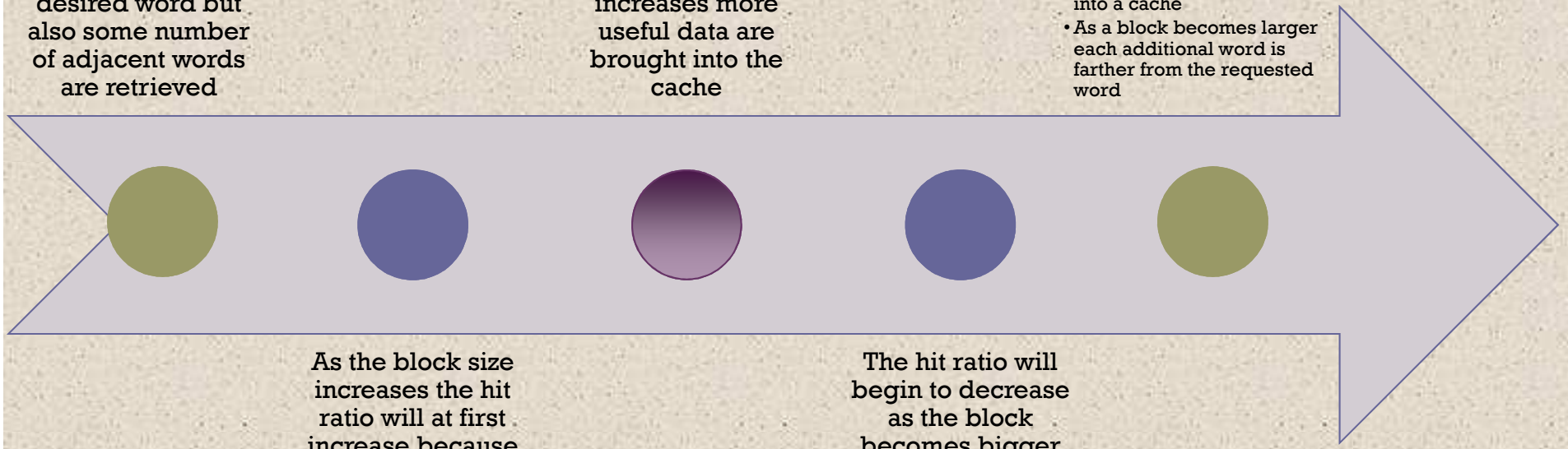
Line Size

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 more useful data are brought into the cache

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



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

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

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

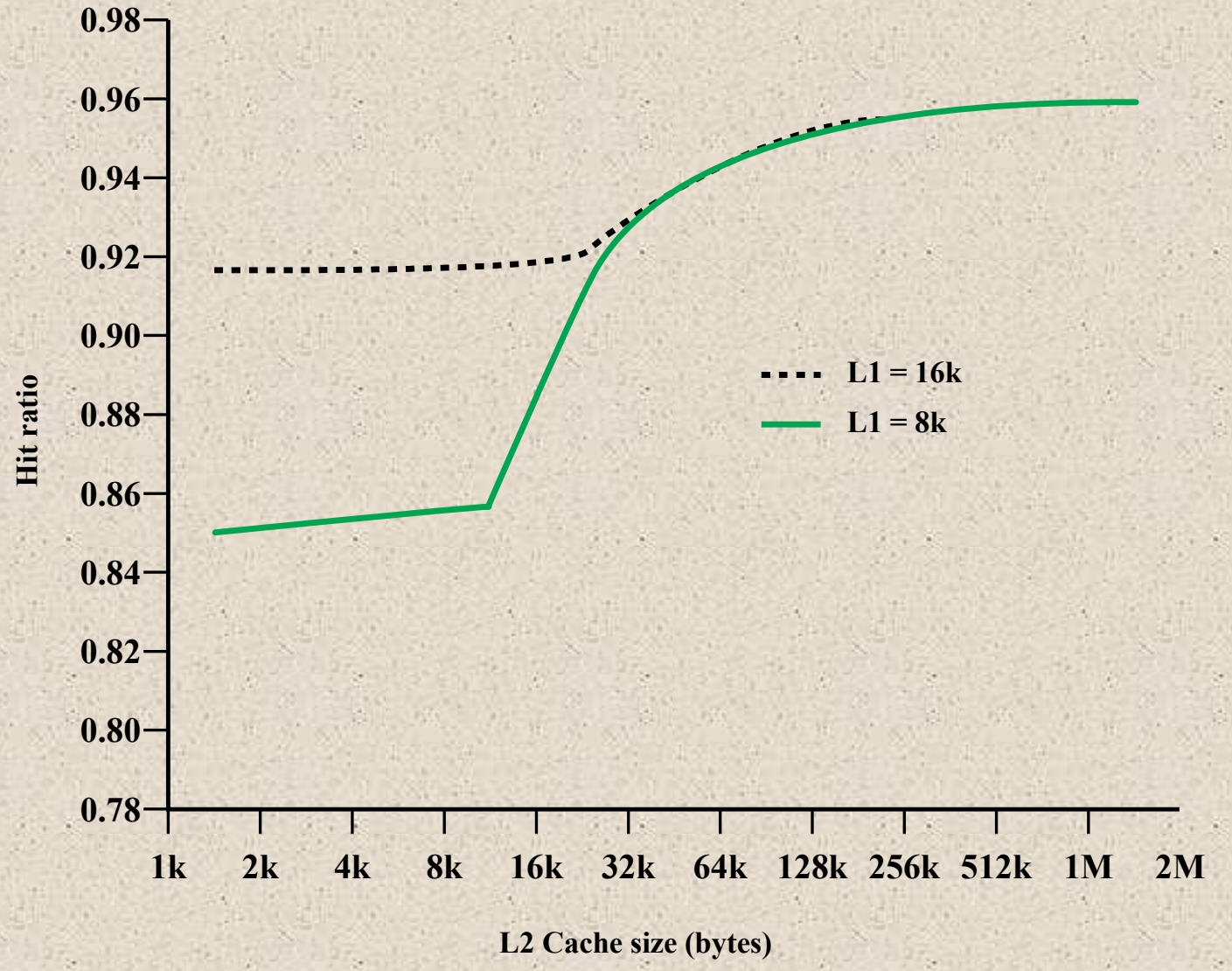


Figure 4.17 Total Hit Ratio (L1 and L2) for 8 Kbyte and 16 Kbyte L1

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

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

Table 4.4

Intel Cache Evolution

(Table is on page 150 in the textbook.)

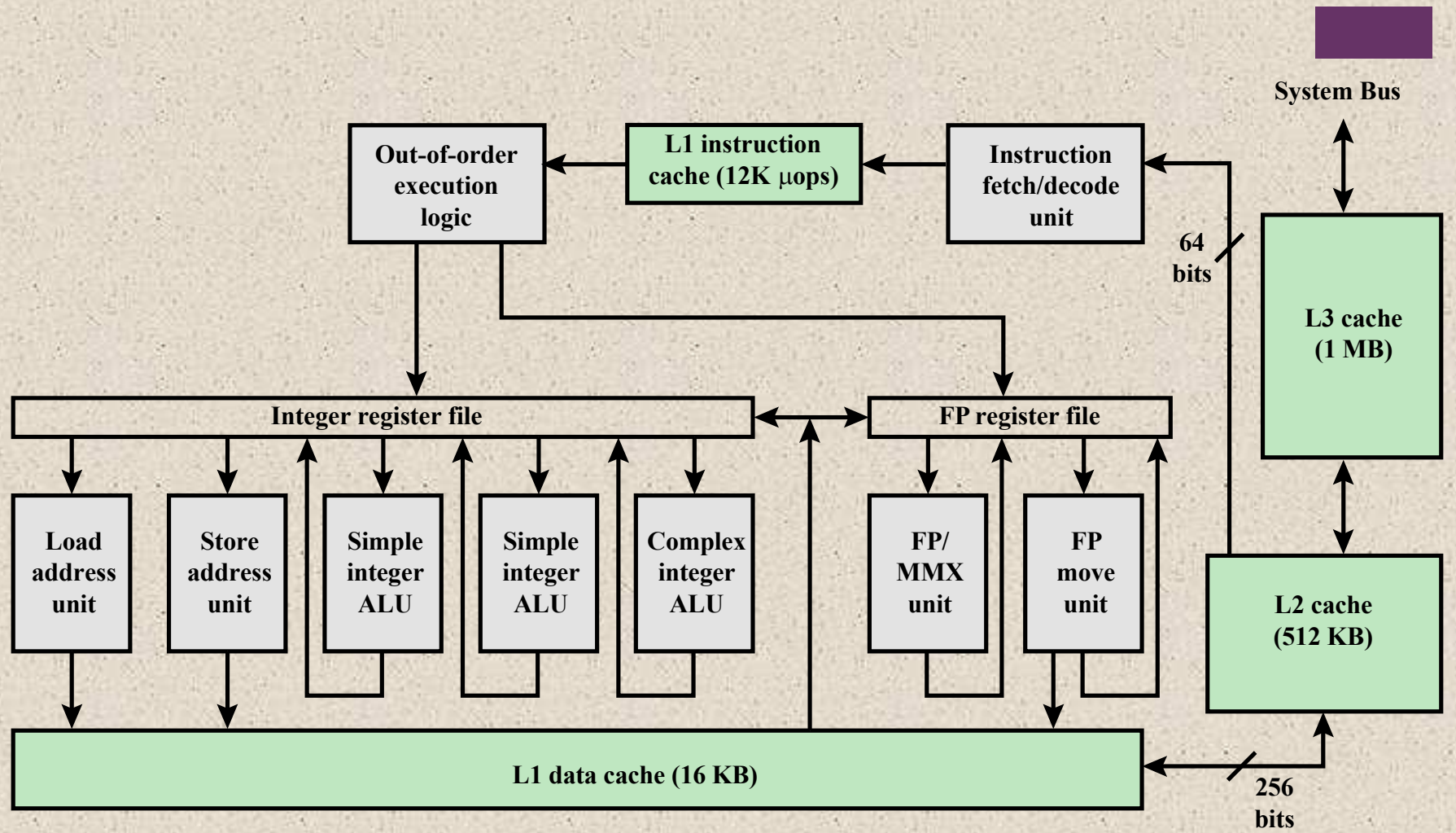


Figure 4.18 Pentium 4 Block Diagram

Table 4.5 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.

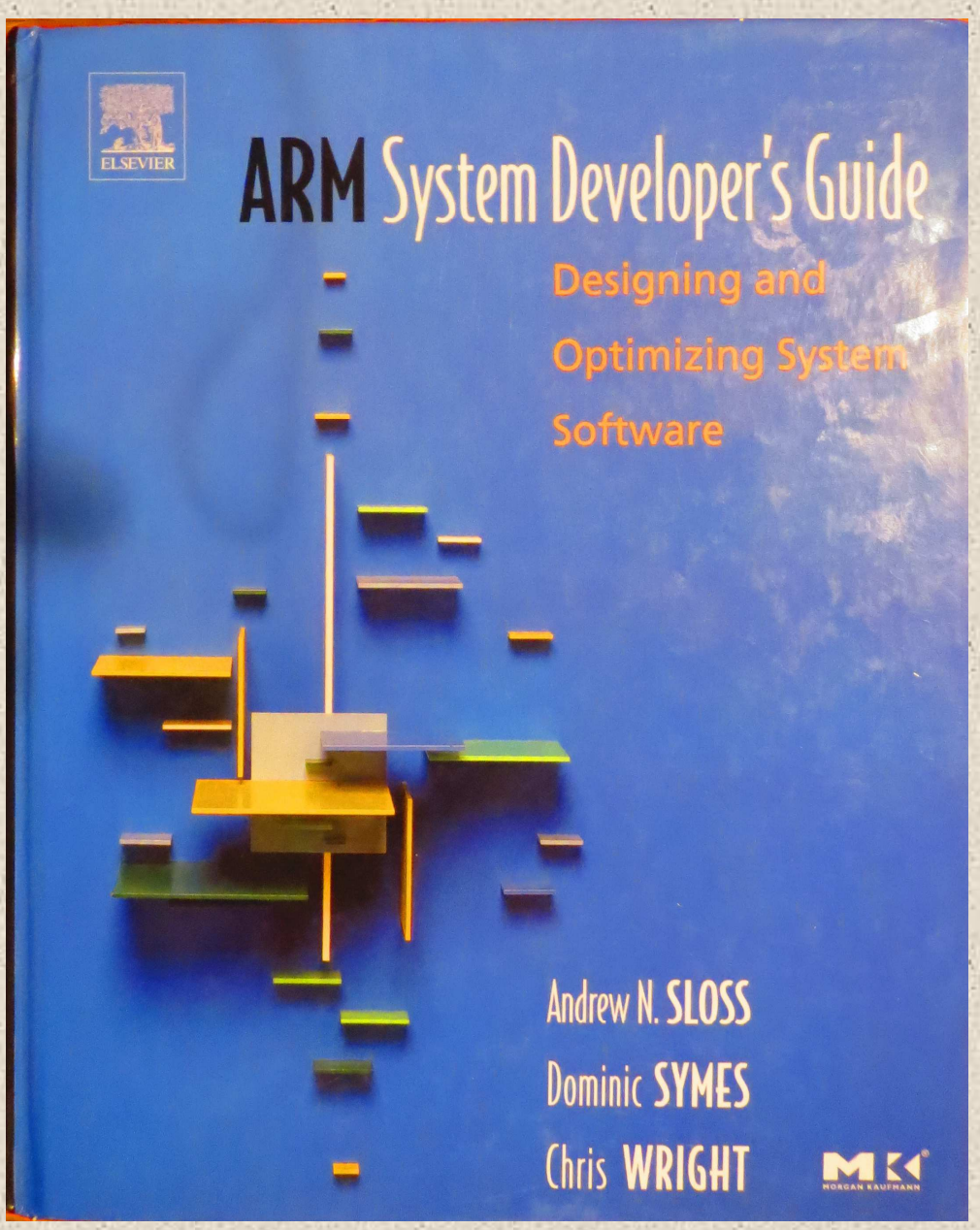
Summary

Chapter 4

Cache Memory

- Computer memory system overview
  - Characteristics of Memory Systems
  - Memory Hierarchy
- Cache memory principles
- Pentium 4 cache organization

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



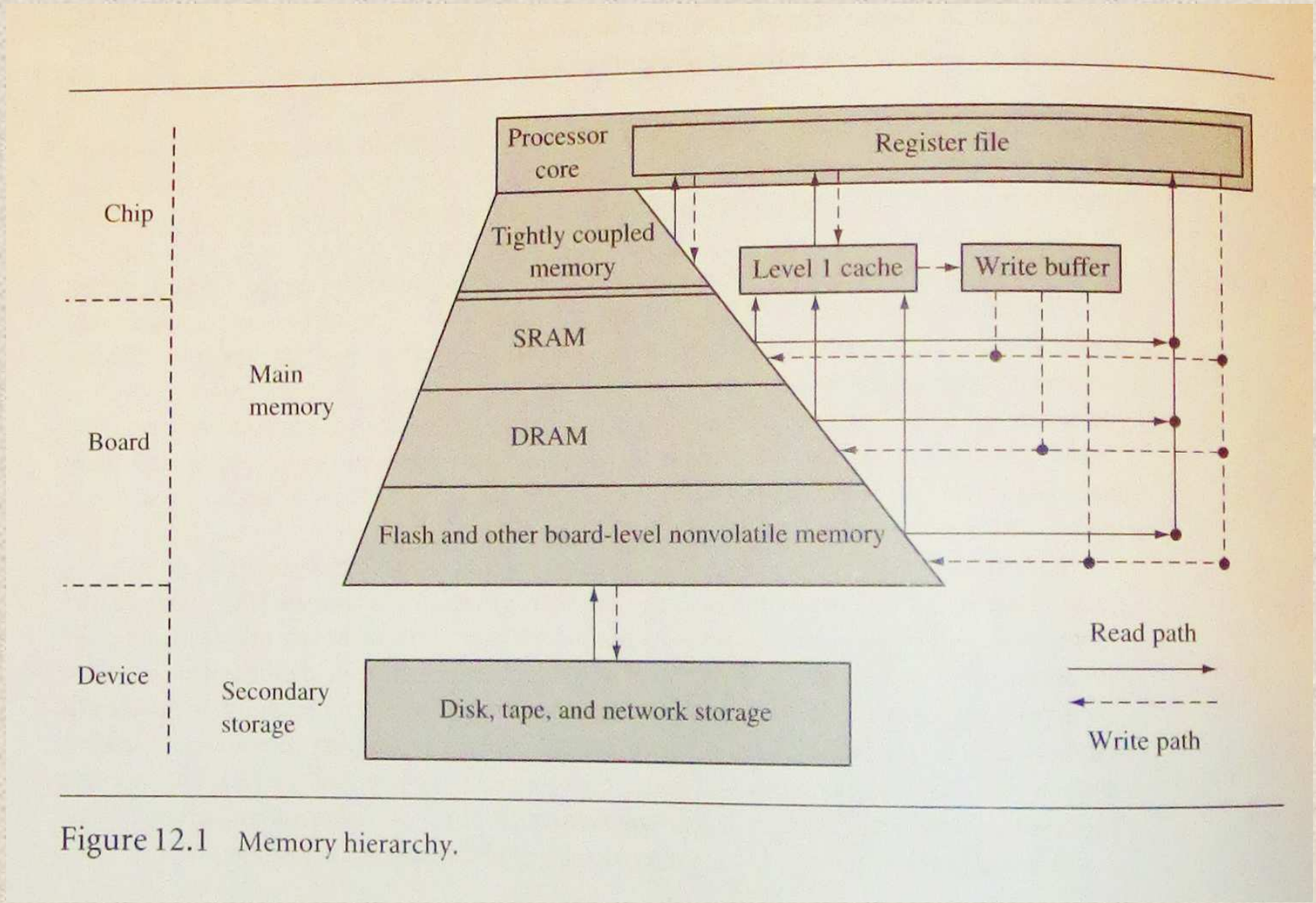


Figure 12.1 Memory hierarchy.

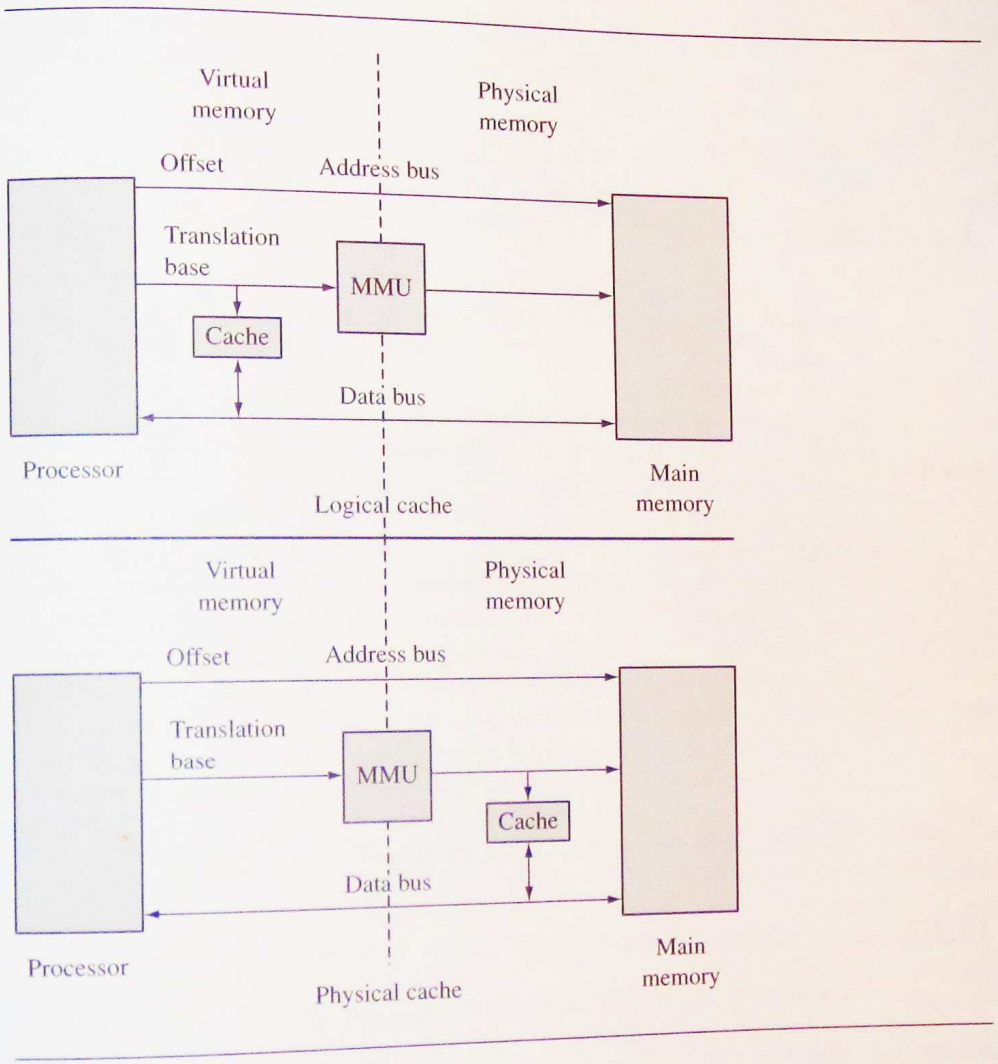


Figure 12.3 Logical and physical caches.

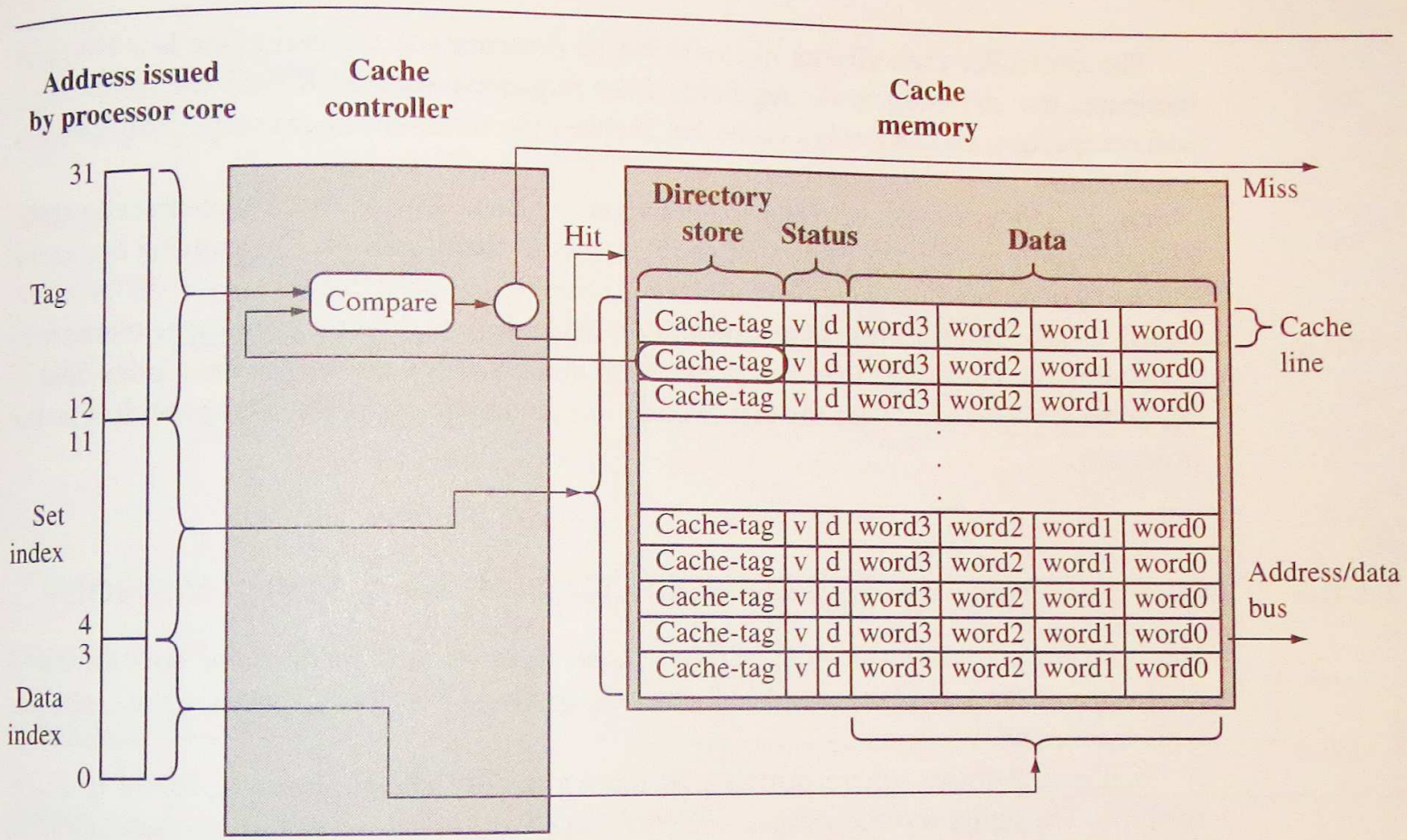


Figure 12.4 A 4 KB cache consisting of 256 cache lines of four 32-bit words.

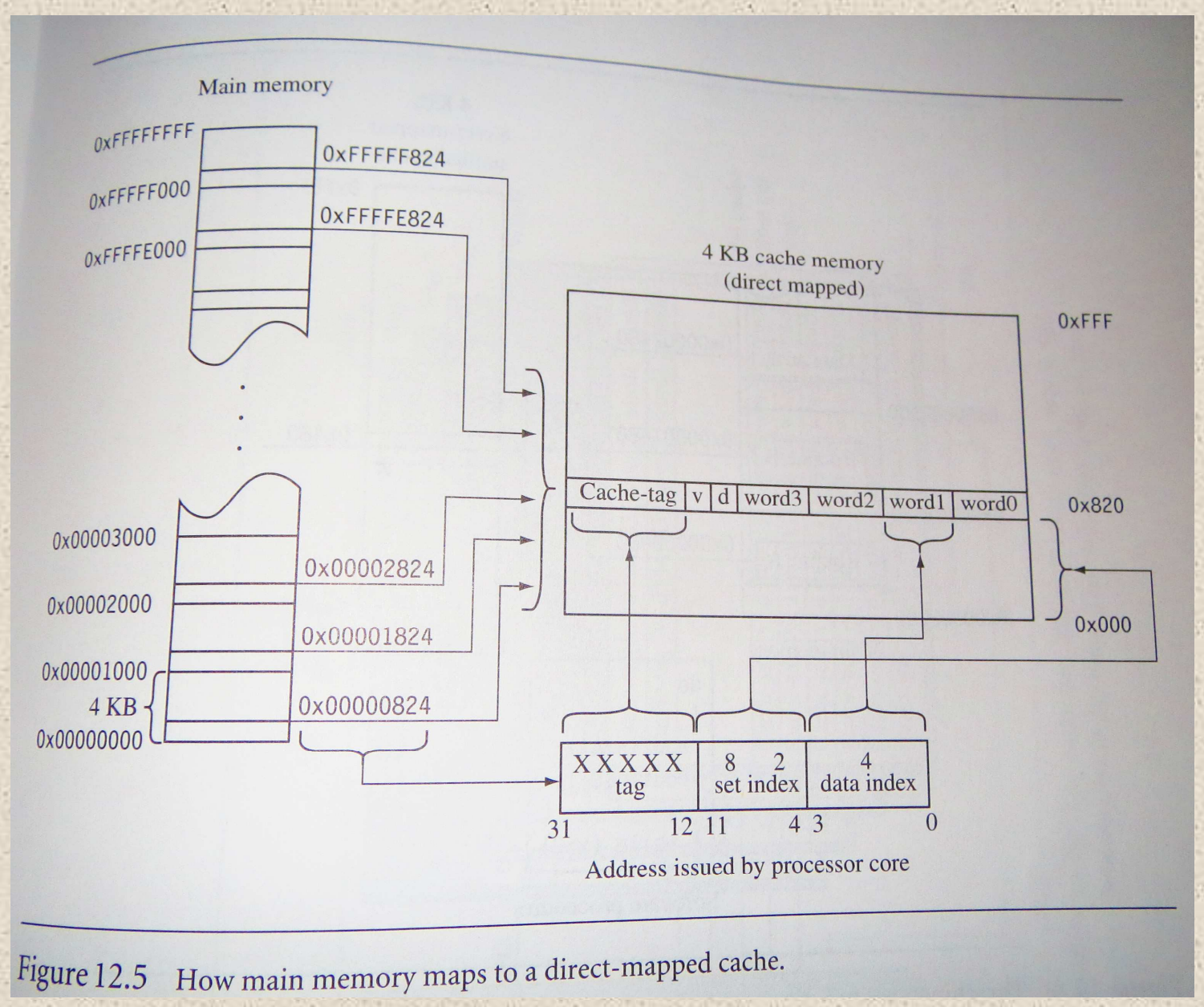


Figure 12.5 How main memory maps to a direct-mapped cache.

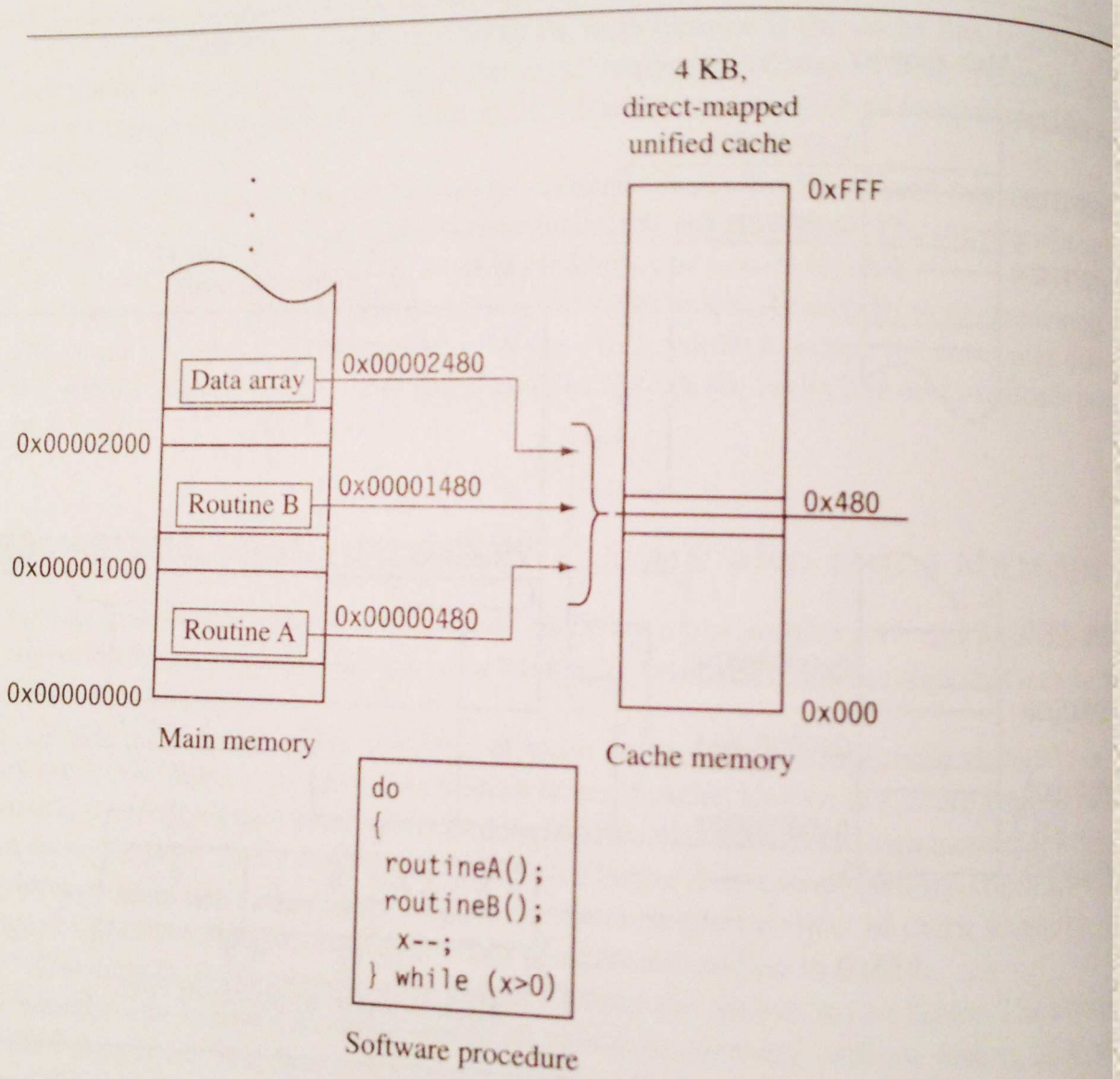


Figure 12.6 Thrashing: two functions replacing each other in a direct-mapped cache.

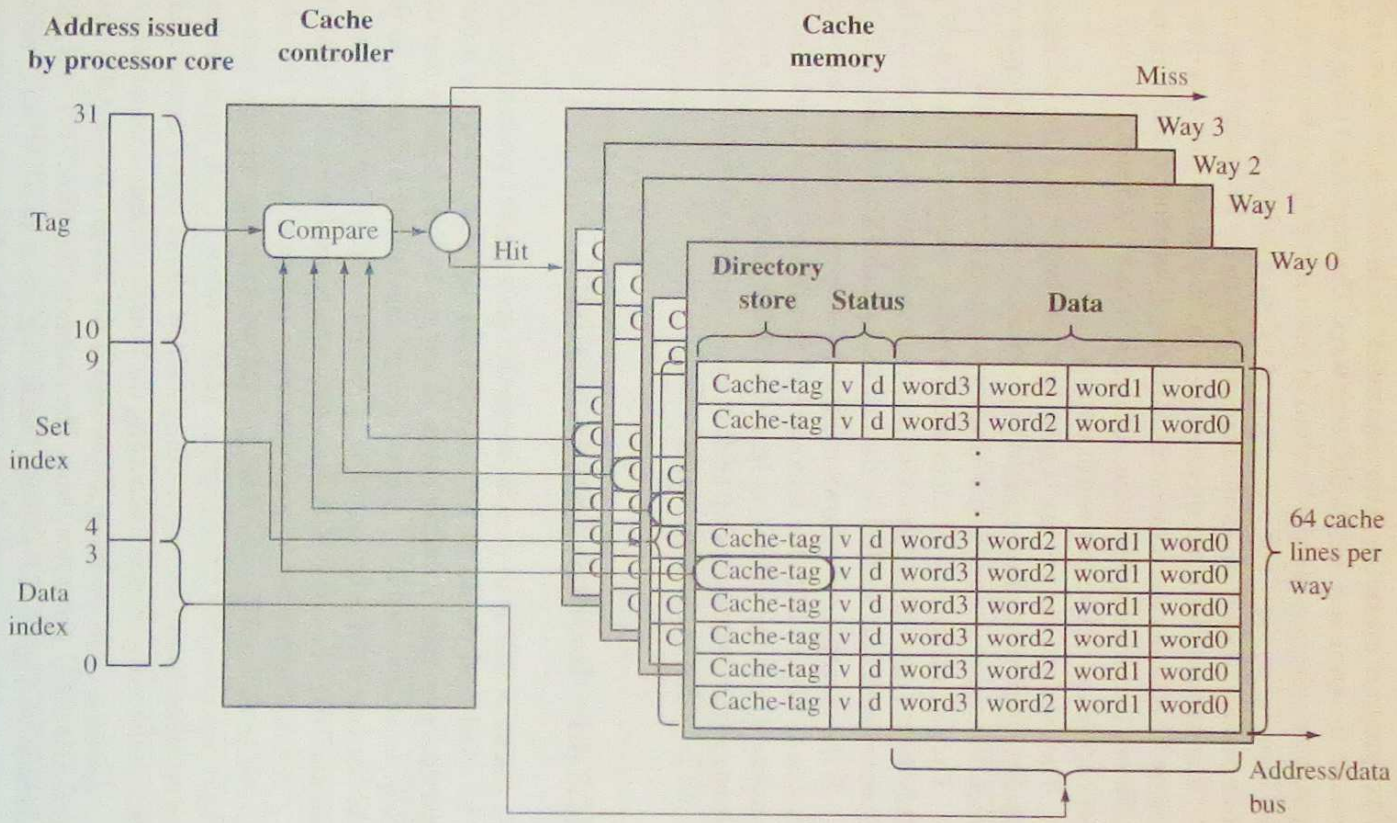


Figure 12.7 A 4 KB, four-way set associative cache. The cache has 256 total cache lines, which are separated into four ways, each containing 64 cache lines. The cache line contains four words.

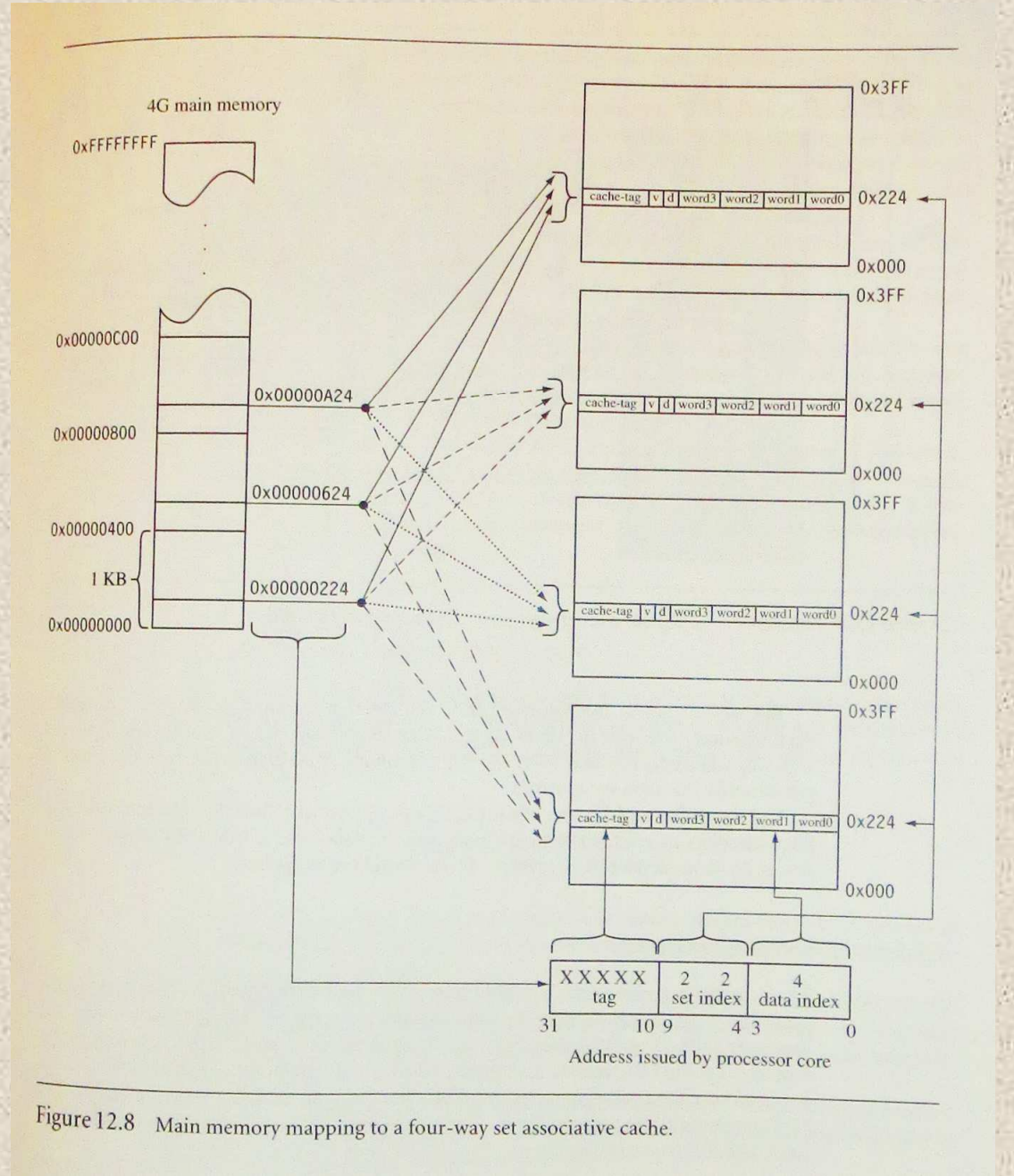


Figure 12.8 Main memory mapping to a four-way set associative cache.

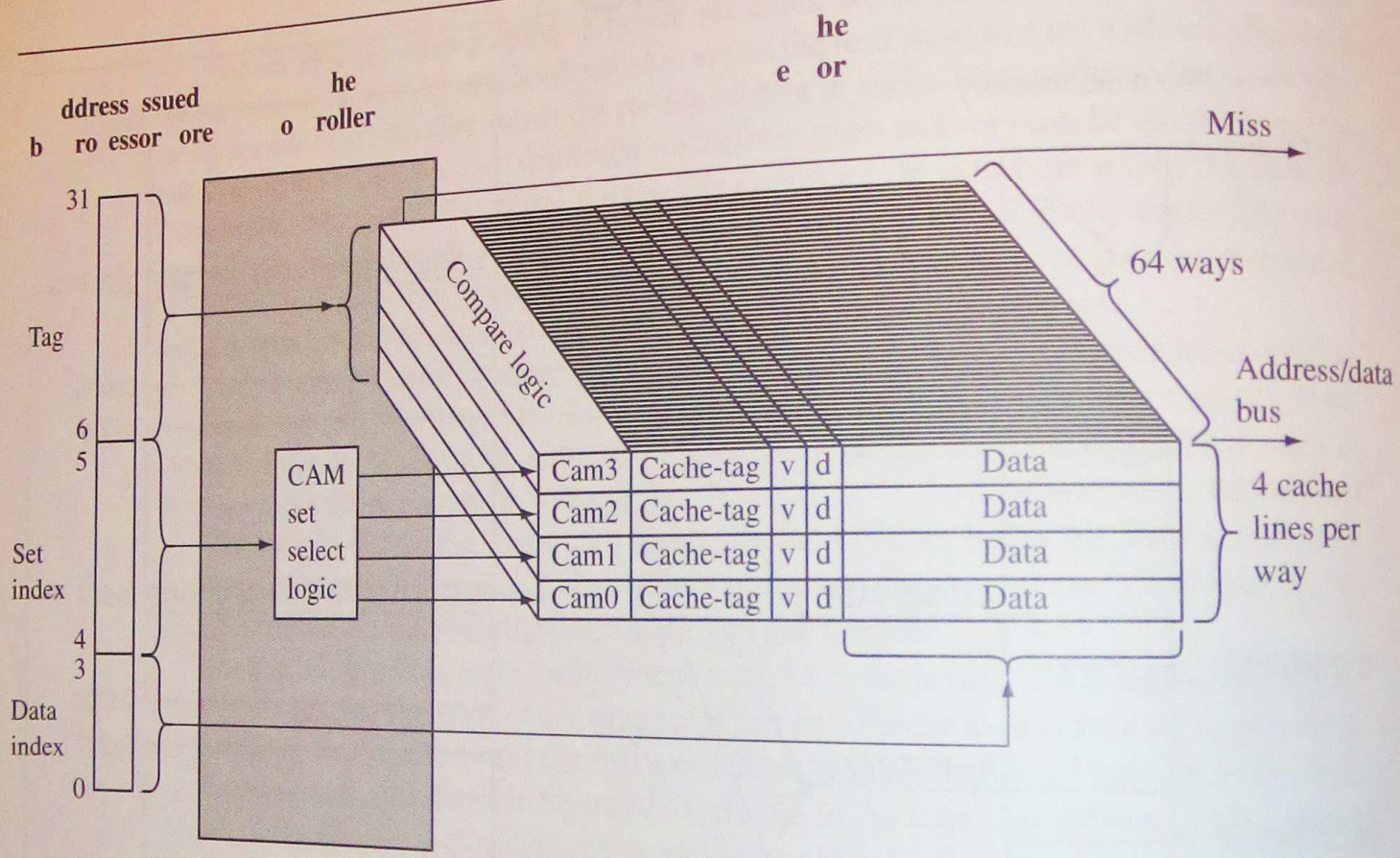


Figure 12.9 ARM940T—4 KB 64-way set associative D-cache using a CAM.