UNIT 2
Computer Memory System

Characteristics of 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

- The term **Location** refers to whether memory is Internal or External to the computer.

- An obvious characteristic of memory is its **Capacity.**
- For Internal memory, this is typically expressed in terms of bytes (1 byte = 8 bits) or words. Common word lengths are **8, 16, and 32 bits**.
- External memory capacity is typically expressed in terms of **Bytes, Megabytes, Gigabytes or more**.

- A related concept is the **unit of transfer.** For internal memory, the unit of transfer is equal to the number of data lines into and out of the memory module.

  - **Word:** The "natural" unit of organization of memory. The size of a word is typically equal to the number of bits used to represent an integer and to the instruction length. Eg. Intel processors uses 8 bits to represent a number.

  - **Addressable units:** It is a size of a physical memory. Eg. For 8086, addressable units are of 1 MB.

  - **Unit of transfer:** For main memory, this is the number of bits read out of or written into memory at a time.

- Another distinction among memory types is the **method of accessing.**

  - **Sequential access:** Memory is organized into units of data, called records. Access must be made in a specific **Linear Sequence**. Tape are sequential access.

  - **Direct access:** As with sequential access, direct access involves a **Shared Read–write Mechanism**. Disk units are direct access.

- **Random Access:** Any location can be selected at **Random and Directly addressed and accessed**. Main memory and some cache systems are random access.

- **Associative:** This is a random access type of memory that enables one to **make a comparison of desired bit locations** within a word for a specified match, and to do this for all words simultaneously. Cache memories may employ associative access.

- Three **Performance** parameters are used

  - **Access Time (Latency):** For random-access memory, this is the time it takes to perform a read or write operation.

  - **Memory Cycle Time:** This concept is primarily applied to random-access memory and consists of the access time plus any additional time required before a second access can commence. Note that memory cycle time is concerned with the system bus, not the processor. Eg. Propagation Delays.

- **Transfer Rate:** This is 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).
- For non-random-access memory, the following relationship holds:

$$T_n = T_A + \frac{n}{R}$$

where

T_n = Average time to read or write n bits

T_A = Average access time

n = Number of bits

R = Transfer rate, in bits per second (bps)

- A variety of **Physical Types** of memory have been employed.

- The most common today are semiconductor memory, magnetic surface memory, used for disk and tape, and optical and magneto-optical.

- Several **Physical Characteristics** of data storage are important.

- In a Volatile Memory, information decays naturally or is lost when electrical power is switched off.

- In a Nonvolatile Memory, information once recorded remains without deterioration until deliberately changed; no electrical power is needed to retain information.

- For random-access memory, the **Organization** is a key design issue.

- In this context, organization refers to the physical arrangement of a data in a memory.

The Memory Hierarchy

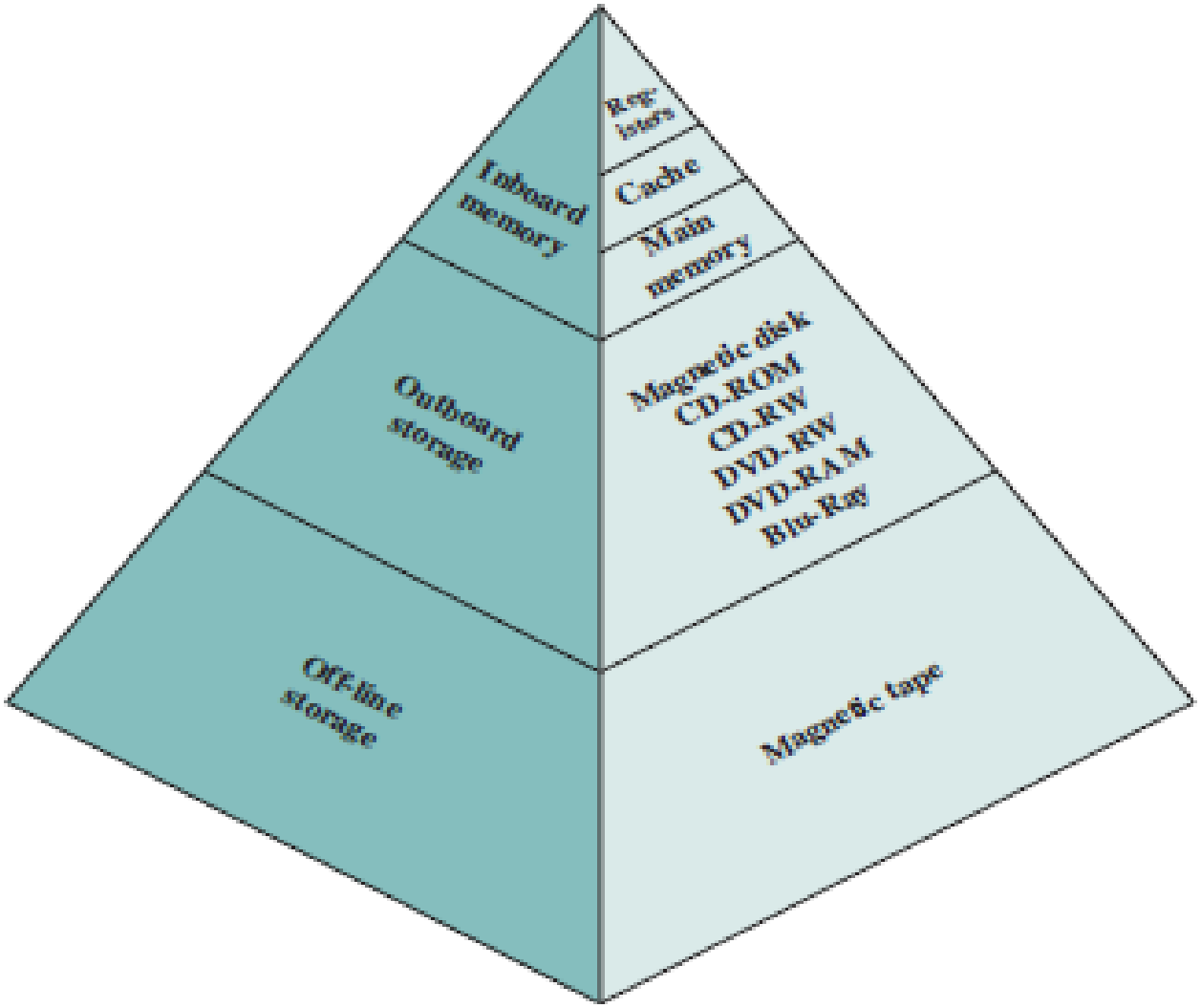
- The design constraints on a computer's memory can be summed up by three questions: **How much? How fast? How expensive?**

- So there are three key characteristics of memory: **Capacity, Access Time, and Cost.**

- According to the characteristics of a memory, A variety of technologies are used to implement memory systems.

- And across this spectrum of technologies, the following relationships hold:

  - **Faster access time, greater cost per bit**
  - **Greater capacity, smaller cost per bit**
  - **Greater capacity, slower access time**

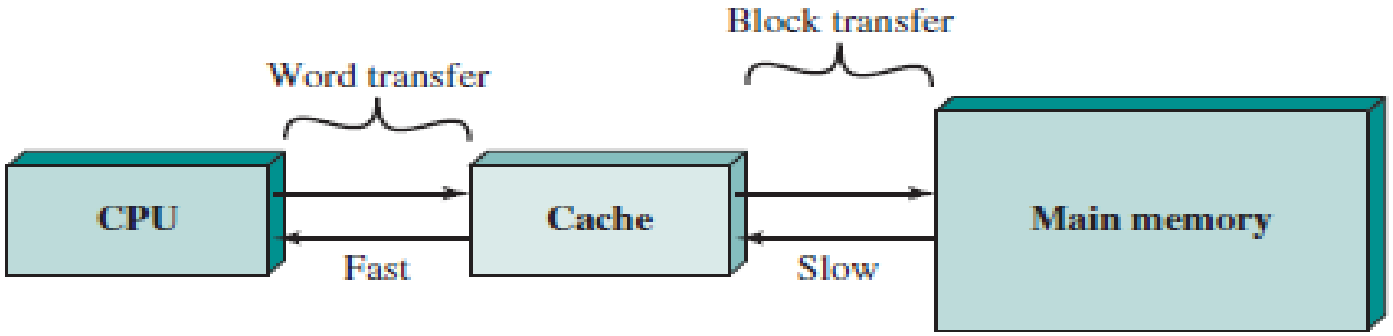

Reg-
isters
Cache
Main
memory
Magnetic disk
CD-ROM
CD-RW
DVD-RW
DVD-RAM
Blu-Ray
Magnetic tape
Inboard
memory
Outboard
storage
Off-line
storage

- As one goes down the hierarchy, the following occur:
  - **Decreasing cost per bit.**
  - **Increasing capacity.**
  - **Increasing access time.**
  - **Decreasing frequency of access of the memory by the processor.**

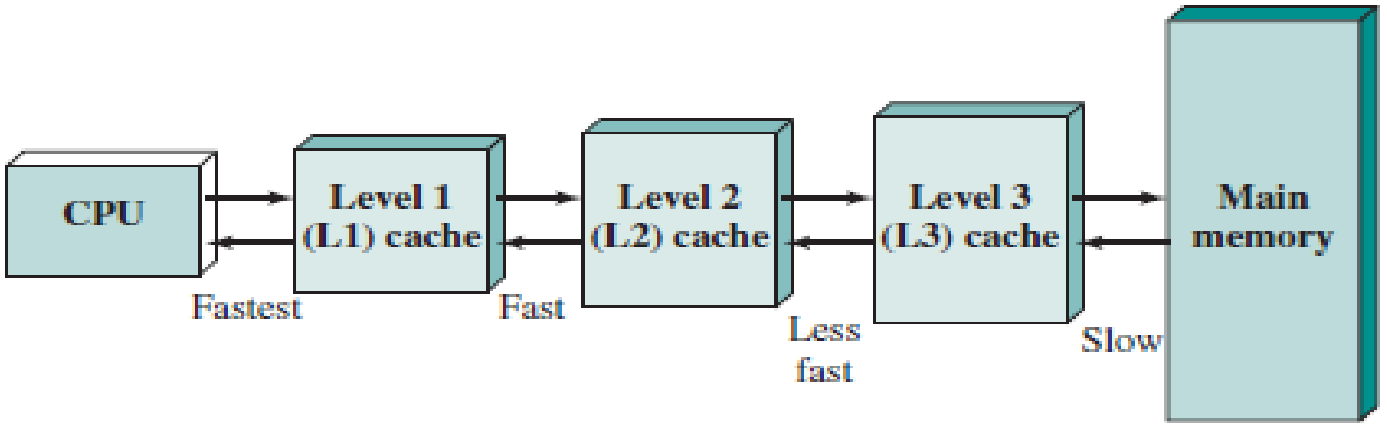
Cache

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

Cache Organization



(a) Single cache



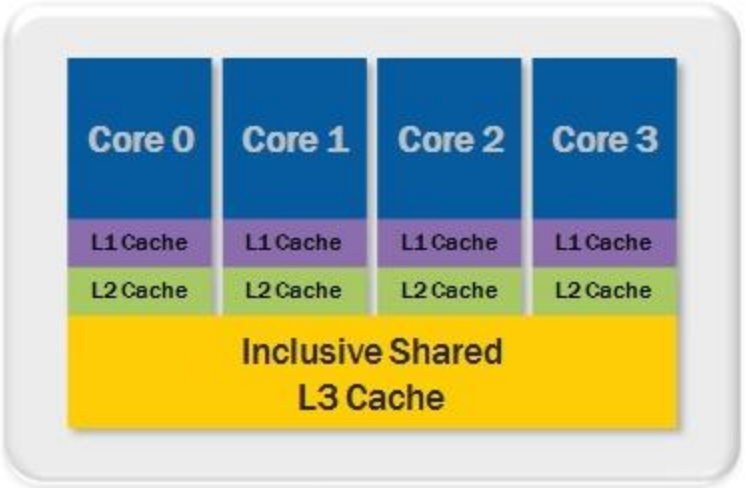
(b) Three-level cache organization

- **Example: Smart Cache of i-Series…**

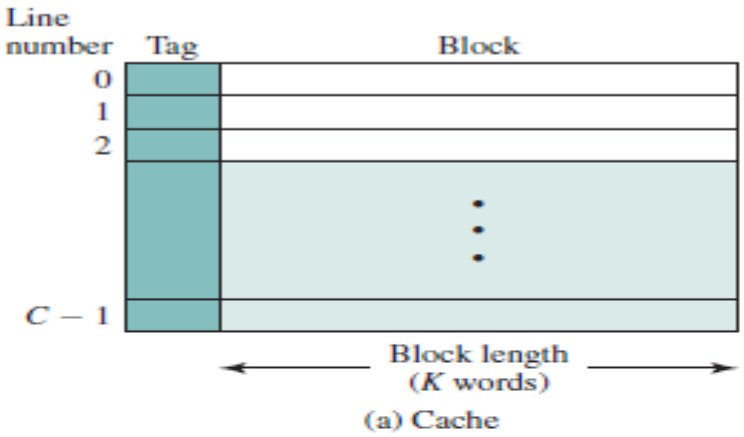
L1 cache: 32 kb Instructions cache +32 kb data cache

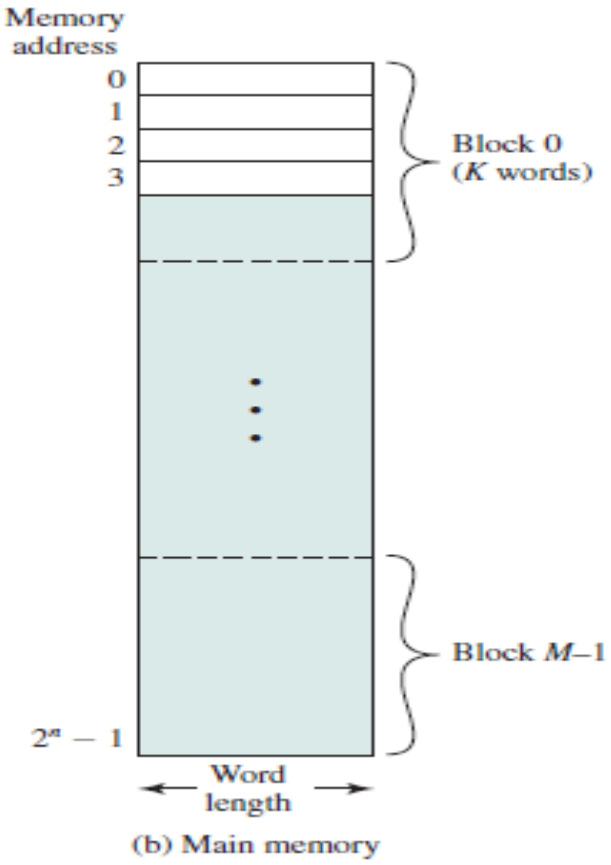
L2 cache: 256 kb (For Instructions and data per core)

L3 cache: 8 mb (For Communication)



Cache/Main Memory Structure

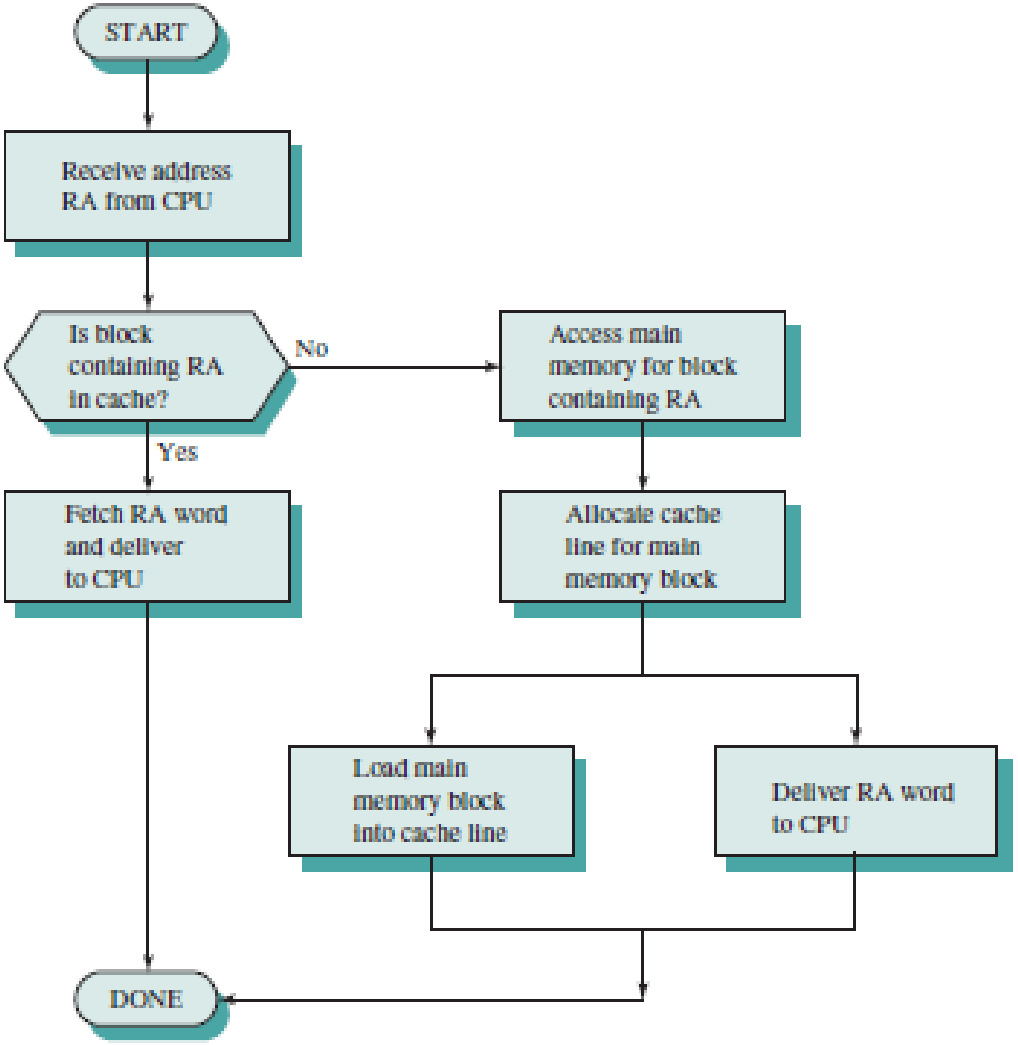




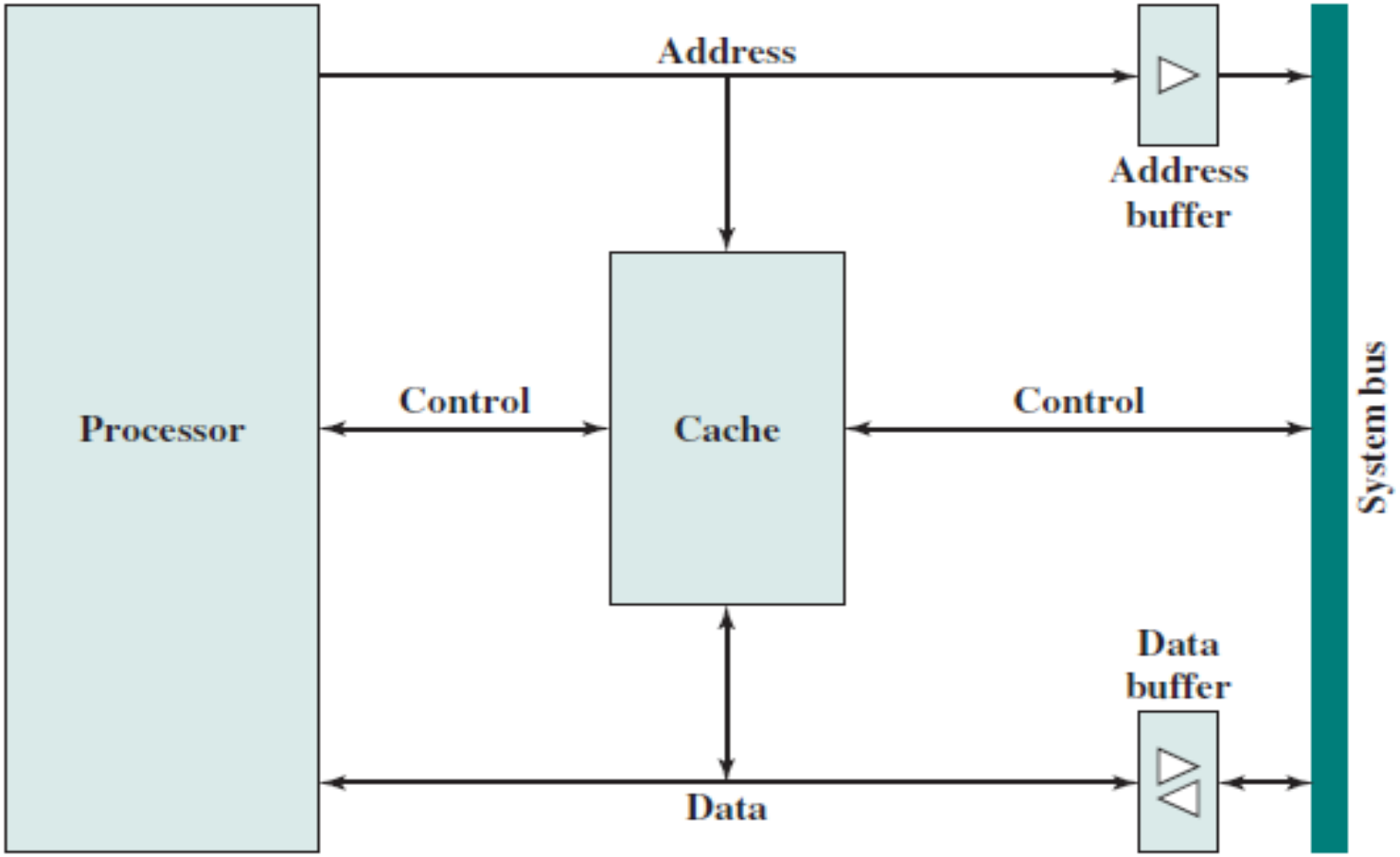
Cache Operation – Overview

- 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 slot.

Cache Read Operation - Flowchart



Typical Cache Organization



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

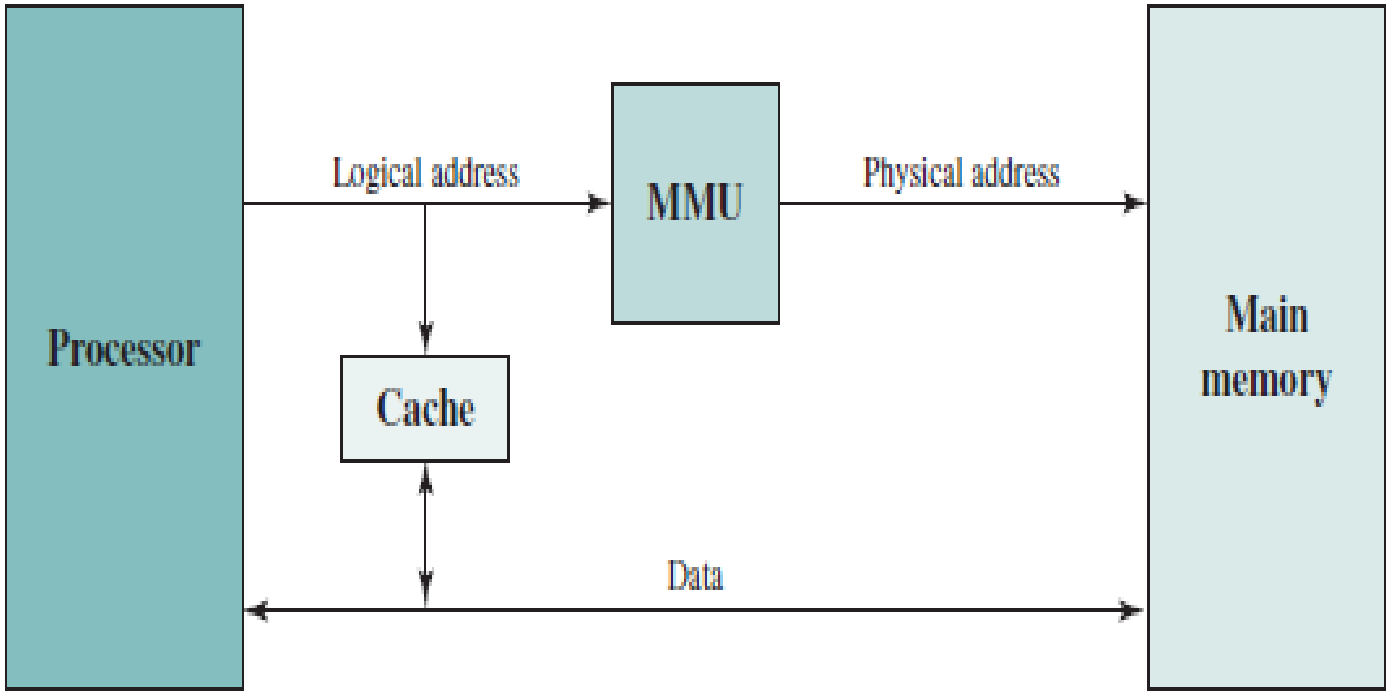
Line Size

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

Cache Addresses

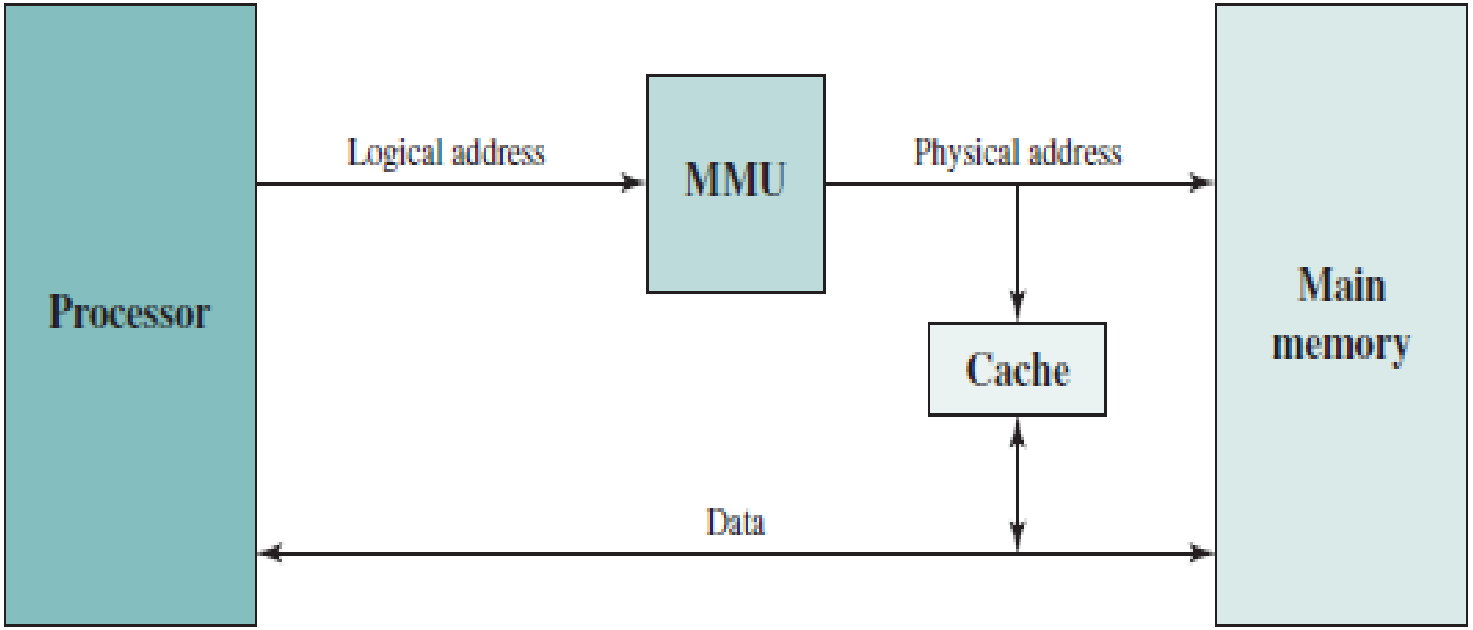
- Two types of Addresses
  - **Logical Cache:** When virtual addresses are used, the system designer may choose to place the cache between the processor and the MMU or between the MMU and main memory. A **Logical Cache**, also known as a **Virtual Cache**, stores data using **Virtual Addresses**.
  - **Physical Cache**: The processor accesses the cache directly, without going through the MMU. A physical cache stores data using main memory **Physical Addresses.**

Logical Cache



(a) Logical cache

Physical Cache



(b) Physical cache

Cache Size

- We would like the size of the cache like this:
  - Size should be small enough so that **the overall average cost per bit is close to that of main memory alone.**
  - Size should be large enough so that **the overall average access time is close to that of the cache alone.**

- **The larger the cache, the larger the number of gates involved in addressing the cache**.

- The result is that **large caches tend to be slightly slower than small ones**—even though they are made up of same integrated circuit technology and put in the same place on chip and circuit board.

- The available chip and board area may also limits cache size.

Processor	Type	Year of Introduction	L1 Cache[a]	L2 Cache	L3 Cache
IBM 360/85	Mainframe	1968	16–32 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	128–256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256–512 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	1 MB	—
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	Workstation/ 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

Mapping Function

- As there are less number of cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines.

- The choice of the mapping function dictates how the cache is organized. Three techniques can be used: **Direct, Associative, and Set Associative.**

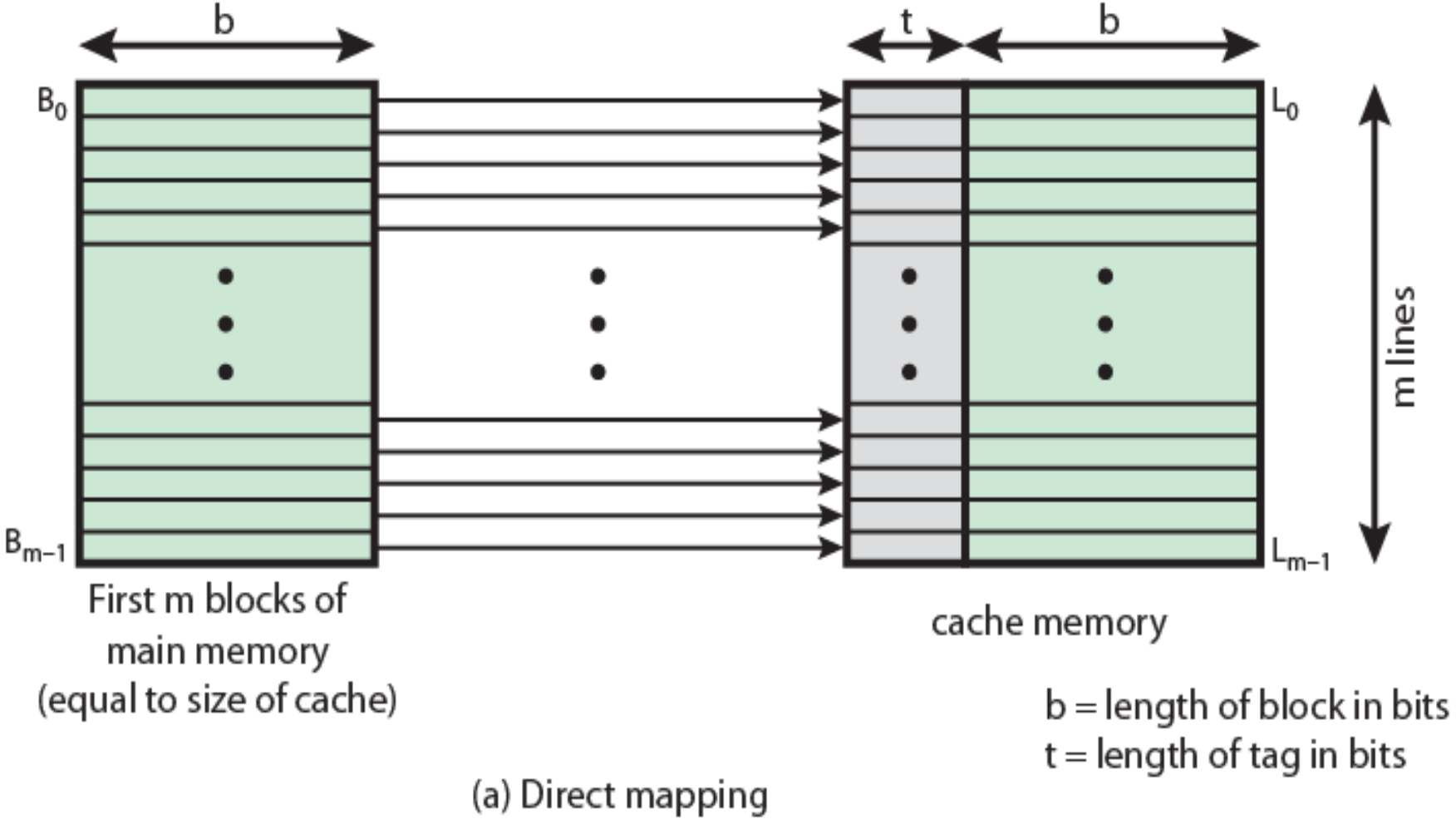
Direct Mapping

- The simplest technique, **maps each block of main memory into only one possible cache line.** The mapping is expressed as

i = j modulo m

  - where
  - *i = cache line number*
  - *j = main memory block number*
  - *m = number of lines in the cache*

Direct Mapping from Cache to Main Memory



(a) Direct mapping

Disadvantage

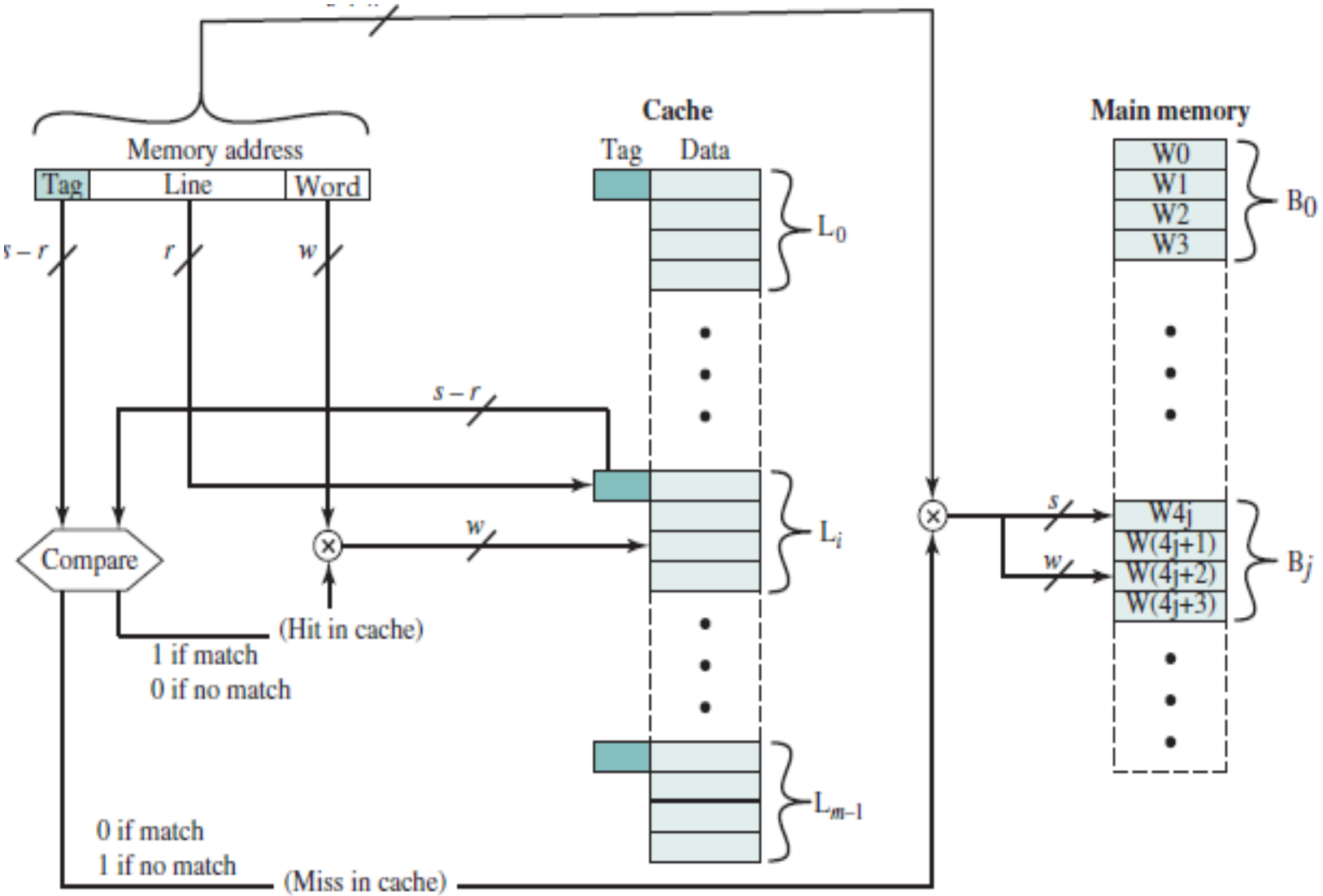
- Its main disadvantage is that **there is a fixed cache location for any given block.**

- Thus, if a program happens to reference words repeatedly from two different blocks that map into the same line, then the blocks will be continually swapped in the cache, and the hit ratio will be low (a phenomenon known as ***Thrashing***).

Example

Example 4.2 For all three cases, the example includes the following elements:

- The cache can hold 64 Kbytes.
- Data are transferred between main memory and the cache in blocks of 4 bytes each. This means that the cache is organized as $16K = 2^{14}$ lines of 4 bytes each.
- The main memory consists of 16 Mbytes, with each byte directly addressable by a 24-bit address ($2^{24} = 16M$). Thus, for mapping purposes, we can consider main memory to consist of 4M blocks of 4 bytes each.

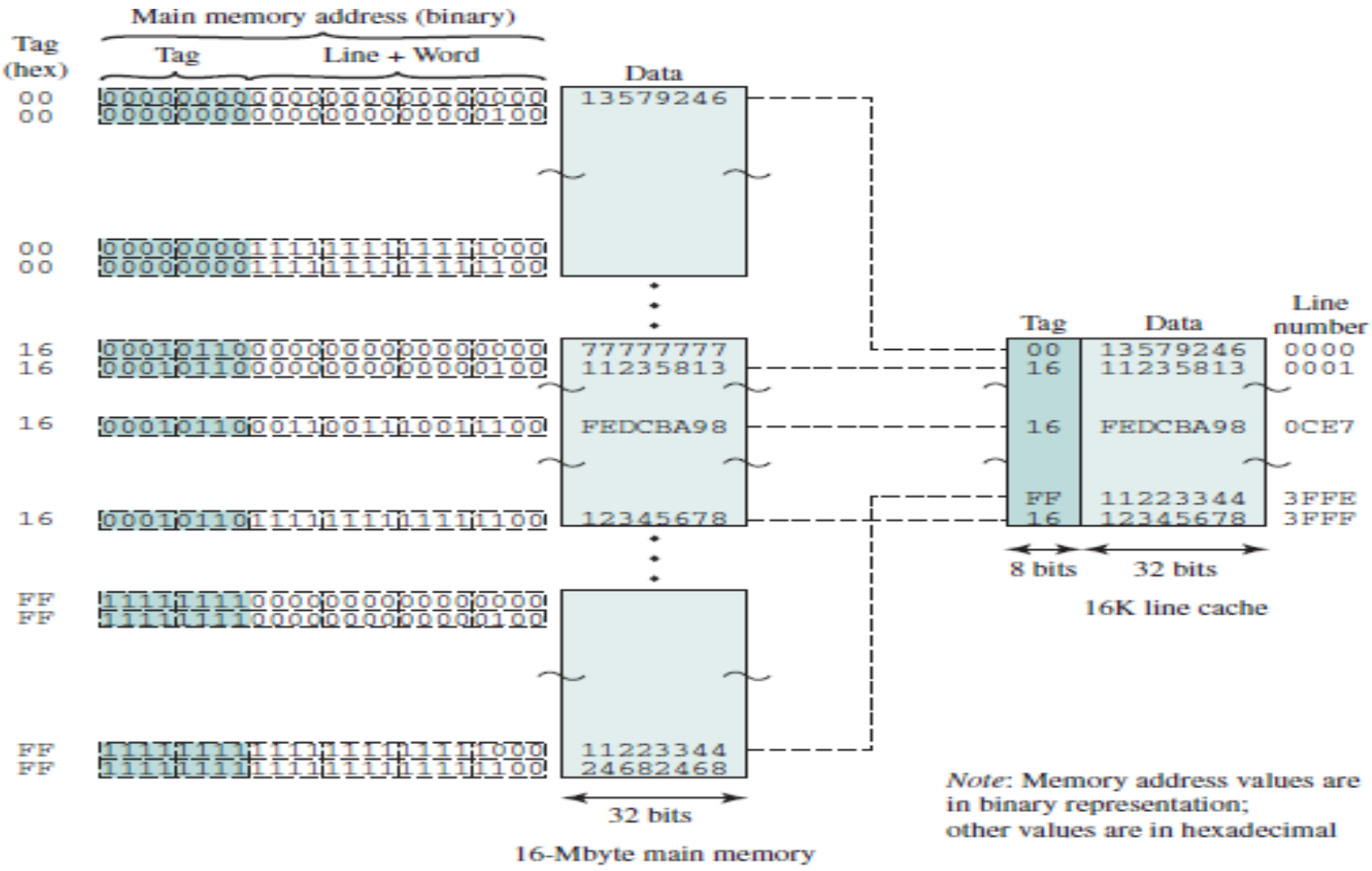

Memory address
Tag
Line
Word
s – r
r
w
Cache
Tag
Data
L0
Li
Lm–1
Main memory
W0
W1
W2
W3
B0
W4j
W(4j+1)
W(4j+2)
W(4j+3)
Bj
s – r
w
s
w
Compare
(Hit in cache)
1 if match
0 if no match
0 if match
1 if no match
(Miss in cache)

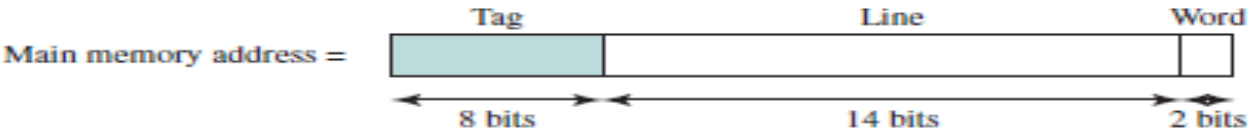
Example 4.2a Figure 4.10 shows our example system using direct mapping.[5] In the example, $m = 16\text{K} = 2^{14}$ and $i = j$ modulo 2^{14}. The mapping becomes

Cache Line	Starting Memory Address of Block
0	000000, 010000, …, FF0000
1	000004, 010004, …, FF0004
⋮	⋮
$2^{14} - 1$	00FFFC, 01FFFC, …, FFFFFC

Note that no two blocks that map into the same line number have the same tag number. Thus, blocks with starting addresses 000000, 010000, …, FF0000 have tag numbers 00, 01, …, FF, respectively.

Referring back to Figure 4.5, a read operation works as follows. The cache system is presented with a 24-bit address. The 14-bit line number is used as an index into the cache to access a particular line. If the 8-bit tag number matches the tag number currently stored in that line, then the 2-bit word number is used to select one of the 4 bytes in that line. Otherwise, the 22-bit tag-plus-line field is used to fetch a block from main memory. The actual address that is used for the fetch is the 22-bit tag-plus-line concatenated with two 0 bits, so that 4 bytes are fetched starting on a block boundary.


Main memory address (binary)
Tag
(hex)
Tag
Line + Word
Data
00
00
13579246
00
00
16
16
77777777
11235813
16
FEDCBA98
16
12345678
FF
FF
FF
FF
11223344
24682468
32 bits
16-Mbyte main memory
Tag
Data
Line number
00
16
13579246
11235813
0000
0001
16
FEDCBA98
0CE7
FF
16
11223344
12345678
3FFE
3FFF
8 bits
32 bits
16K line cache
Note: Memory address values are in binary representation; other values are in hexadecimal

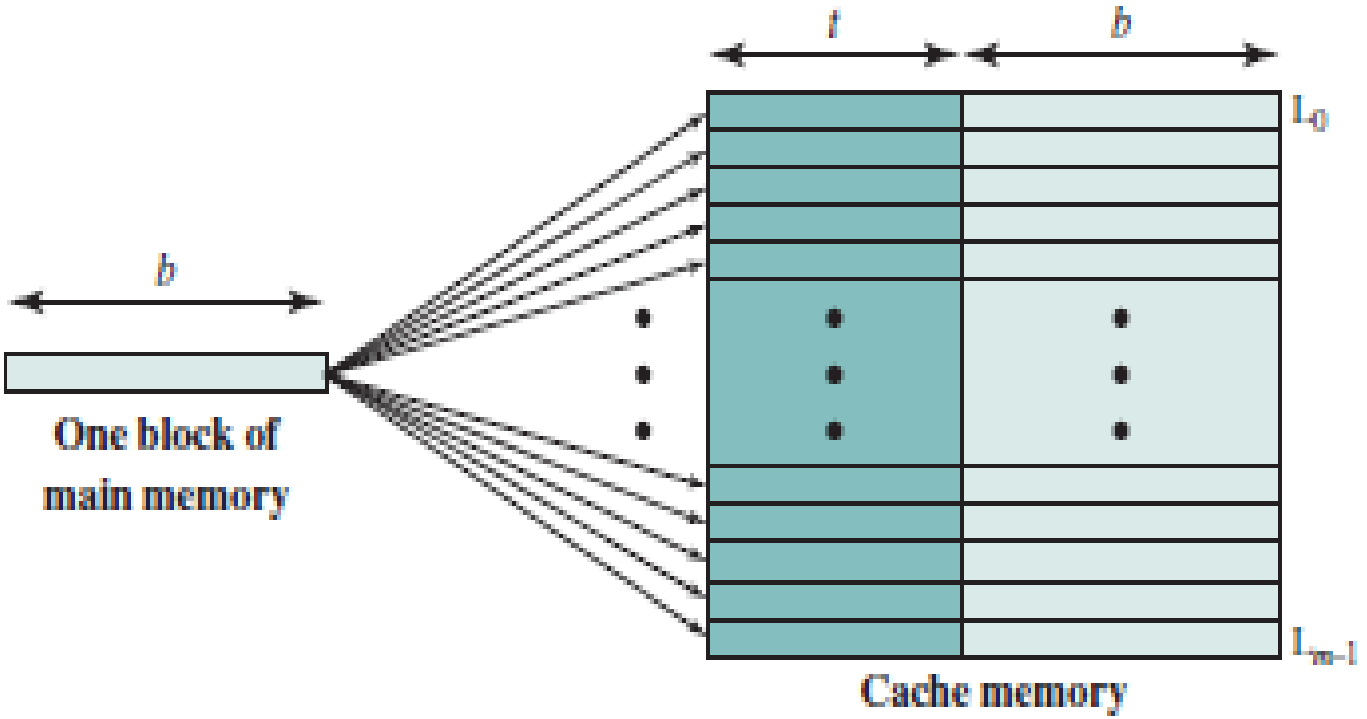

Main memory address =
Tag
Line
Word
8 bits
14 bits
2 bits

Cache line	Main memory blocks assigned
0	$0, m, 2m, \ldots, 2^s - m$
1	$1, m+1, 2m+1, \ldots, 2^s - m + 1$
⋮	⋮
$m-1$	$m-1, 2m-1, 3m-1, \ldots, 2^s - 1$

Associative Mapping

- Associative mapping overcomes the disadvantage of direct mapping by **permitting each main memory block to be loaded into any line of the cache.**

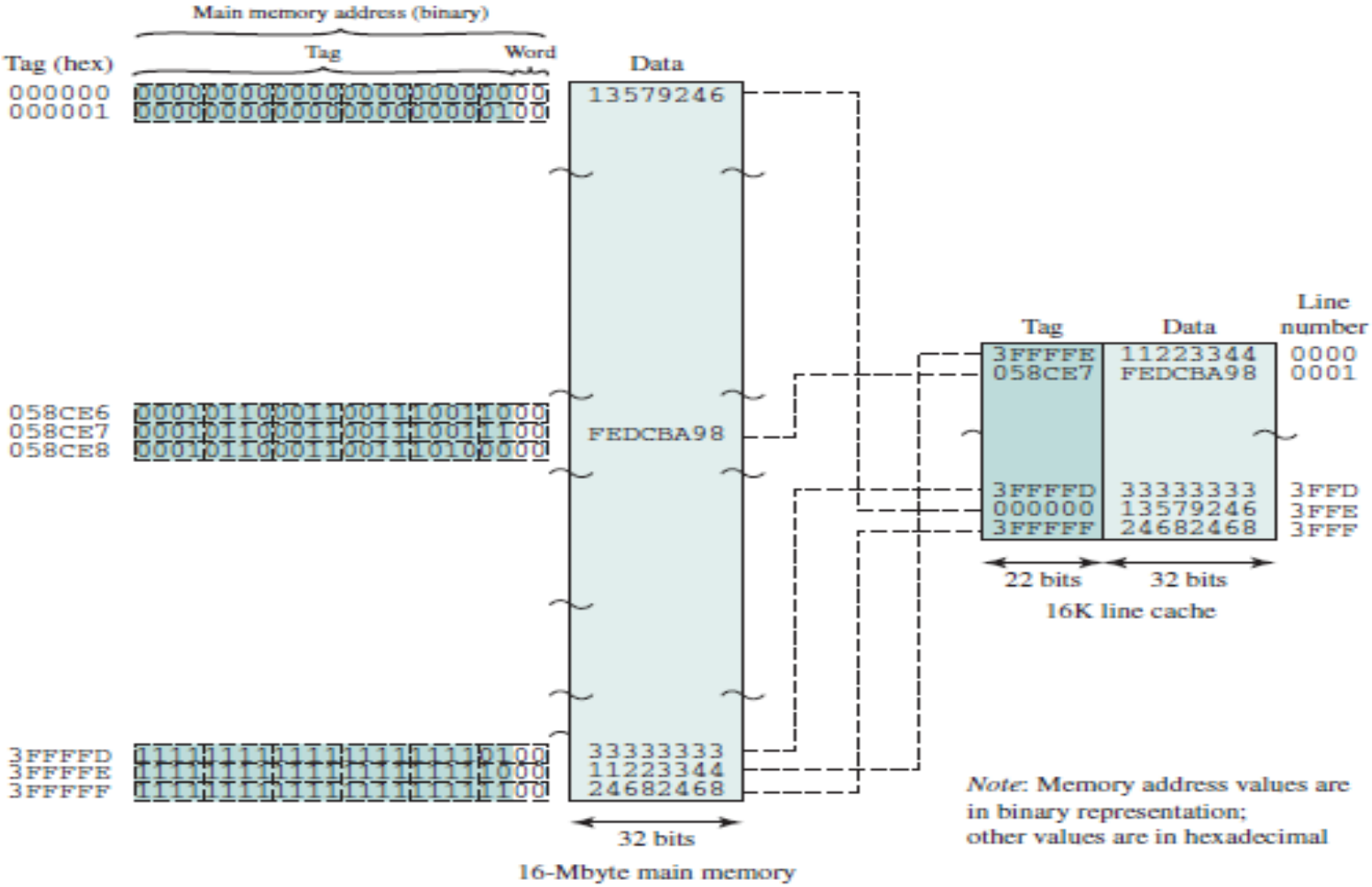
- In this case, the cache control logic interprets a memory address simply as a Tag and a Word field. **The Tag field uniquely identifies a block of main memory.**


t
b
b
One block of
main memory
L_0
L_{m-1}
Cache memory

Example

Example 4.2b Figure 4.12 shows our example using associative mapping. A main memory address consists of a 22-bit tag and a 2-bit byte number. The 22-bit tag must be stored with the 32-bit block of data for each line in the cache. Note that it is the leftmost (most significant) 22 bits of the address that form the tag. Thus, the 24-bit hexadecimal address 16339C has the 22-bit tag 058CE7. This is easily seen in binary notation:

memory address	0001	0110	0011	0011	1001	1100	(binary)
	1	6	3	3	9	C	(hex)
tag (leftmost 22 bits)	00	0101	1000	1100	1110	0111	(binary)
	0	5	8	C	E	7	(hex)


Main memory address (binary)
Tag (hex)
Tag
Word
Data
000000
000001
13579246
058CE6
058CE7
058CE8
FEDCBA98
3FFFFD
3FFFFE
3FFFFF
33333333
11223344
24682468
32 bits
16-Mbyte main memory
Tag
Data
Line number
3FFFFE
058CE7
11223344
FEDCBA98
0000
0001
3FFFFD
000000
3FFFFF
33333333
13579246
24682468
3FFD
3FFE
3FFF
22 bits
32 bits
16K line cache
Note: Memory address values are in binary representation; other values are in hexadecimal
Main memory address =
Tag
Word
22 bits
2 bits

Disadvantage

- The principal disadvantage of associative mapping is the complex circuitry required to examine the tags of all cache lines in parallel.

- So, cost of circuit will be high.

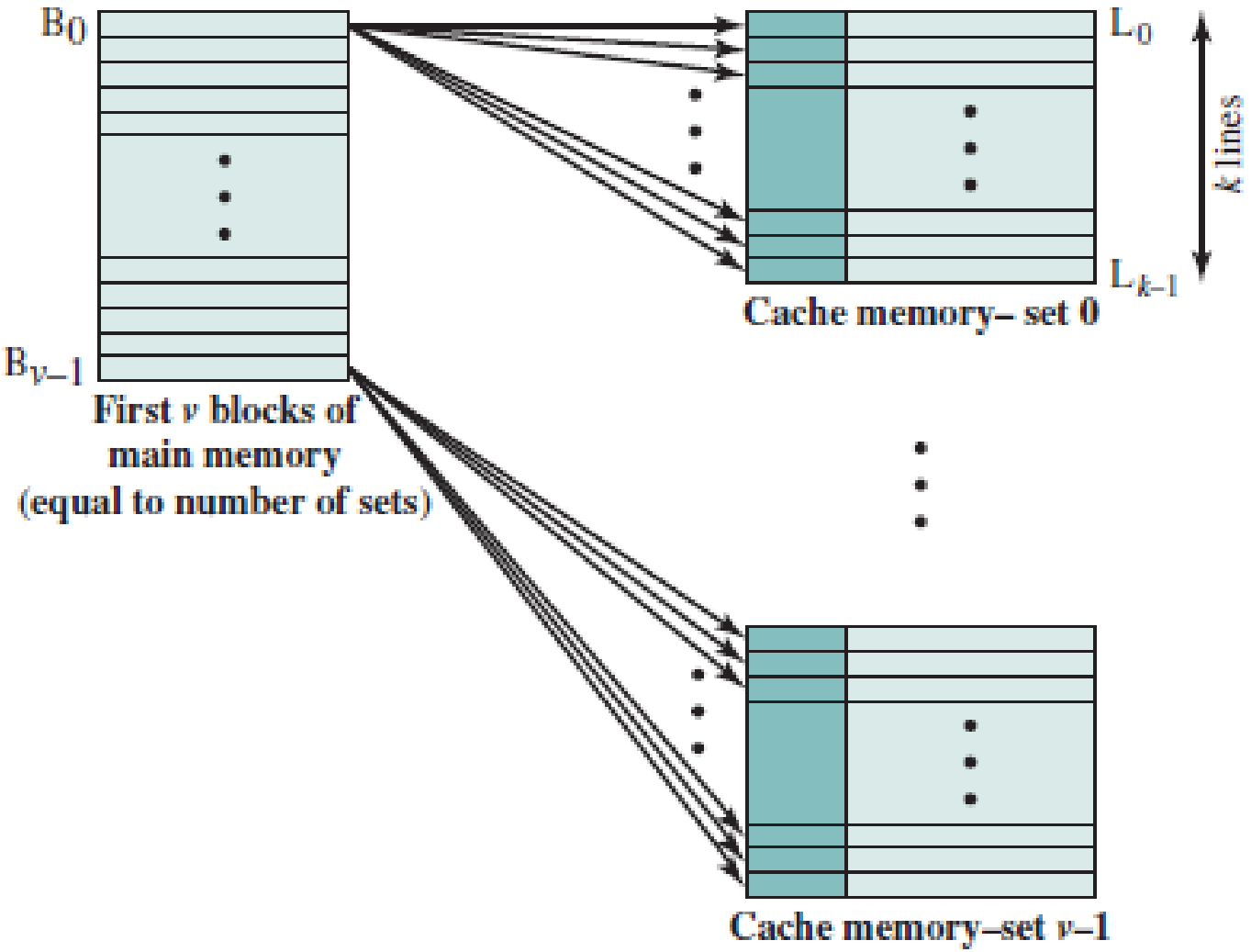
Set-associative Mapping

- Set-associative mapping **uses strengths of both the direct and associative approaches while reducing their disadvantages.**

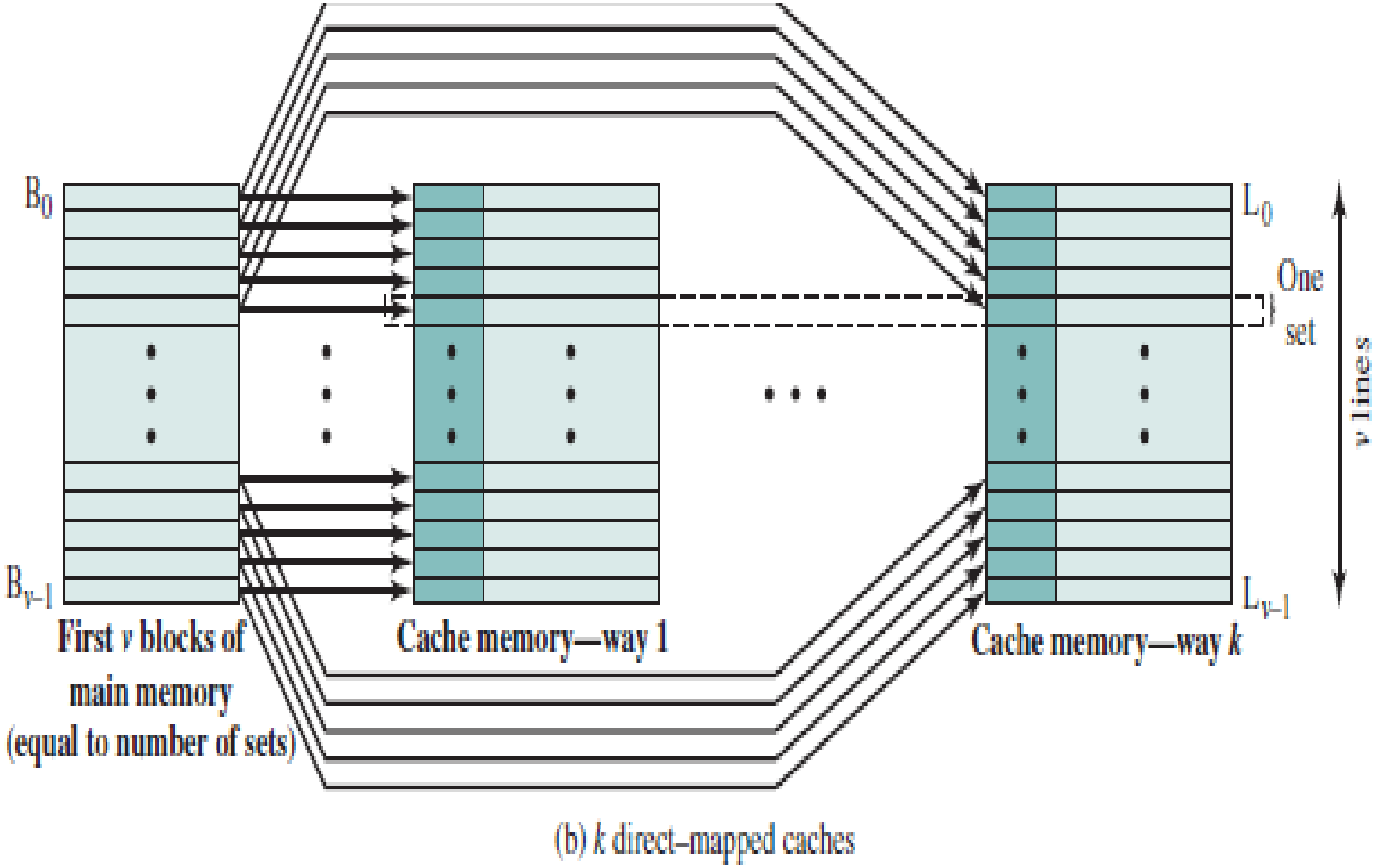
- In this case, the cache consists of a number sets, each of which consists of a number of lines. The relationships are-

$$m = v * k$$
$$i = j \text{ modulo } v$$

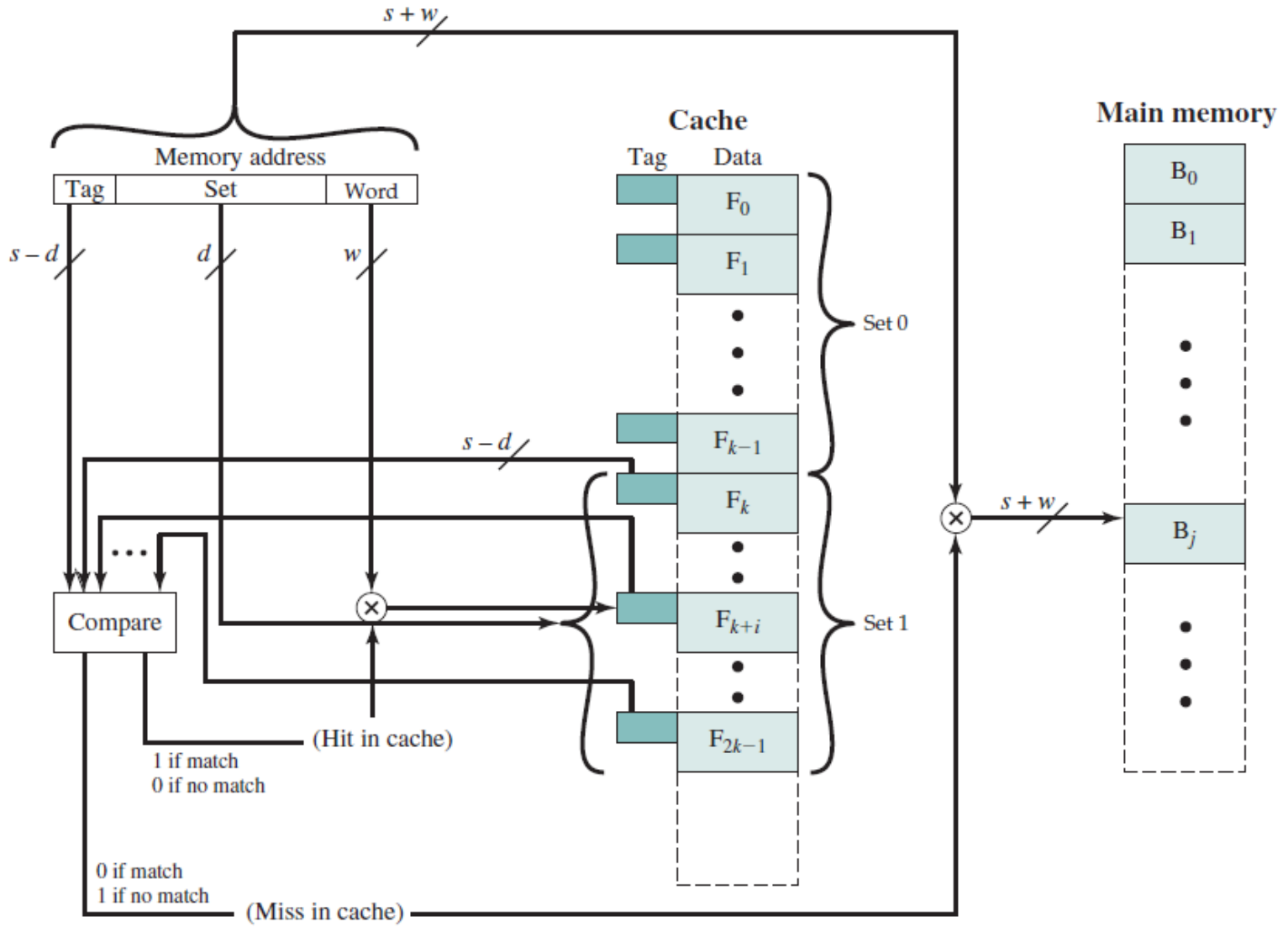
- *i = cache set number*
- *j = main memory block number*
- *m = number of lines in the cache*
- *v = number of sets*
- *k = number of lines in each set*



(a) v associative–mapped caches



(b) k direct–mapped caches


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

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.**

Replacement Algorithms

1. Least Recently Used (LRU)

2. First-in-first-out (FIFO)

3. Least Frequently Used (LFU)

4. Random

1. Least Recently Used (LRU):

- **Replace that block in the set that has been in the cache longest with no reference to it.**

- For **Two-way Set Associative**, this is easily implemented. Each line includes a USE bit. When a line is referenced, its USE bit is set to 1 and the USE bit of the other line in that set is set to 0. When a block is to be read into the set, the line whose USE bit is 0 is used.

- LRU is also relatively easy to implement for a **Fully Associative Cache**. The cache mechanism maintains a **separate list of indices** to all the lines in the cache. When a line is referenced, it moves to the front of the list.

- **For replacement, the line at the back of the list is used**. Because of its simplicity of implementation, LRU is the most popular replacement algorithm.

2. First-in-first-out (FIFO)

- Replace that block in the set that has been in the cache longest.

- FIFO is easily implemented as a round-robin or circular buffer technique.

3. Least Frequently Used (LFU)

- Replace that block in the set that has experienced the fewest references.

- LFU could be implemented by associating a counter with each line.

4. Random

- Any line from cache memory is randomly chosen and replaced with new value.

Write Policy

- When a block that is resident in the cache is to be replaced, we have to take care that **changes should occur in both, i.e.. Main Memory, Cache.**

- There are two problems to contend with.
  - If a word has been altered only in the cache, then the corresponding memory word is invalid.
  - I/O device has altered main memory, and the cache word is invalid.

- Two ways for write policy:

1. Write Through

2. Write Back

1. Write Through

- The simplest technique is called **Write Through.**
- Using this technique, all write operations are made to main memory as well as to the cache, ensuring that **main memory is always valid.**
- **Processor**, Any **other processor** or **cache module can monitor traffic** to main memory to maintain consistency within its own cache.
- The main disadvantage of this technique is that it **generates large memory traffic and may create a bottleneck.**

2. Write Back

- An alternative technique, known as **Write Back.** This techniques **minimizes memory writes**.

- With write back, updates are made only in the cache. When an update occurs, a **Dirty Bit,** or **Use Bit**, associated with the line is set.

- Then, **when a block is replaced, it is written back to main memory if and only if the dirty bit is set**.

- The problem with write back is that portions of main memory are invalid, and hence **accesses by I/O modules can be allowed only through the cache.**

Many Cache memories and One shared main memory.

- In a bus organization in which **more than one device (typically a processor) has a cache and main memory is shared**, a new problem is introduced.

- If data in one cache are altered, this invalidates not only the corresponding word in main memory, but also that same word in other caches (if any other cache happens to have that same word).

- **Even if a write-through policy is used, the other caches may contain invalid data.**

- Possible approaches to cache coherency include the following:

1. **Bus watching with write through:** Each cache controller monitors the address lines to detect write operations to memory by other bus masters. If another master writes to a location in shared memory that also resides in the cache memory, the cache controller invalidates that cache entry.

2. **Hardware transparency:** Additional hardware is used to ensure that all updates to main memory via cache are reflected in all caches. Thus, if one processor modifies a word in its cache, this update is written to main memory. In addition, any matching words in other caches are similarly updated.

3. **Noncacheable memory:** Only a portion of main memory is shared by more than one processor, and this is designated as noncacheable. In such a system, all accesses to shared memory are cache misses.

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 from very small to larger sizes, the hit ratio will at first increase because of the **Principle Of Locality**, which states that data next to the referenced word are likely to be referenced in the near future.

- But actually this will not happened, hit ratio starts decreasing.

- The relationship between block size and hit ratio is complex, depending on the locality characteristics of a particular program. **So no definitive optimum value has been found.**

- A size of from **8 to 64 bytes** seems reasonably close to optimum.

Number of Cache

- When caches were originally introduced, the typical system had a single cache. More recently, the use of multiple caches has become the norm. Two aspects of this design
  - **Multilevel cache**
  - **Unified Vs Split Cache**

1. Multilevel Cache

- Inclusion of on-chip cache will help your system to increase performance.

- But when external devices want to access memory, they have to scan memories (may be RAM/ROM) directly.

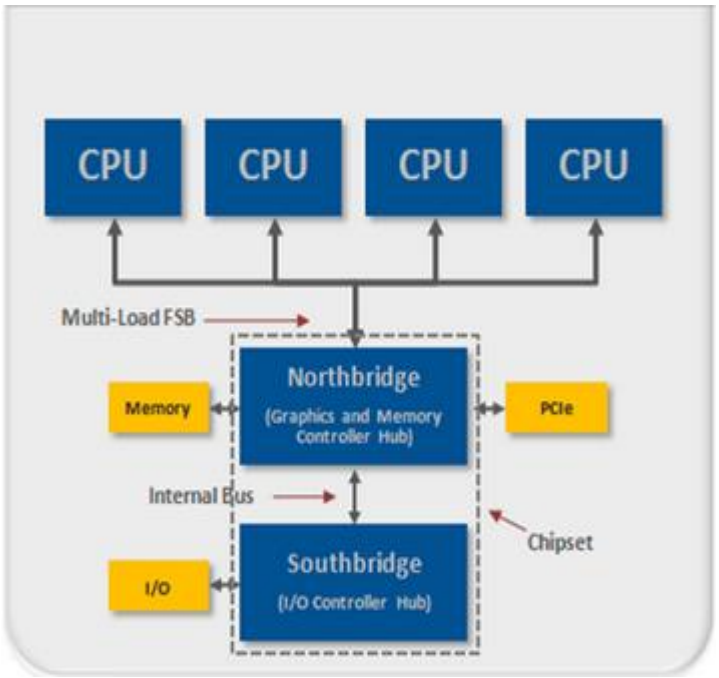
- As these memories are slower compared to cache, will require more time to get data from memory.

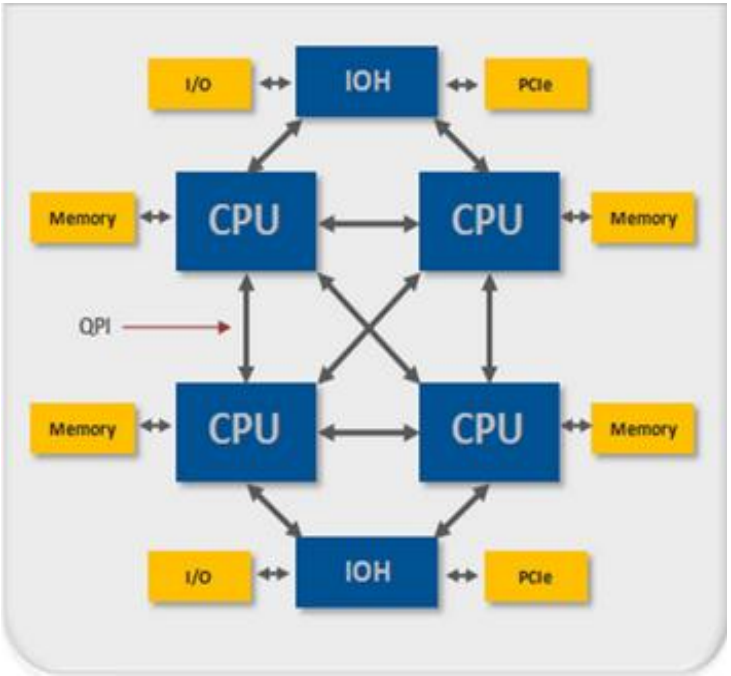
- **So inclusion of off-chip cache will help external devices to get most recent or frequent data.**

- The simplest such organization is known as a two-level cache, with the **internal cache designated as Level 1 (L1) and the external cache designated as Level 2 (L2).**

- Also if we use different bus (external bus) to access data from L2 cache, it will help to reduce burden on a system. **Eg. Intel QPI.**

- **Quick Path Interconnect:** QPI is the new point-to-point interconnect for connecting a CPU to either a chipset or another CPU. It provides up to 25.6 GB/s of total bidirectional data throughput per link.





- ***PCI Express* (Peripheral Component Interconnect Express)**, officially abbreviated as *PCIe*, is a high-speed serial computer expansion bus standard.

2. Unified Vs. Split Caches

- Many of the designs consisted of a **single cache used to store references to both data and instructions. (Unified Cache)**

- More recently, it has become common to **split the cache into two: one dedicated to Instructions and one dedicated to Data. (Split Cache)**

- These two caches both exist at the same level, typically as two L1 caches.

- When the processor attempts to fetch an instruction from main memory, it first consults the instruction L1 cache, and when the processor attempts to fetch data from main memory, it first consults the data L1 cache.

- **Advantages of Unified Cache:**
  - For a given cache size, a unified cache has a higher hit rate than split caches because it balances the load between instruction and data fetches automatically. That is, **if an execution pattern involves many more instruction fetches than data fetches, then the cache will tend to fill up with instructions, and if an execution pattern involves relatively more data fetches, the opposite will occur.**
  - **Only one cache needs to be designed** and implemented.

- **Advantage of Split Cache:**

  - The key advantage of the split cache design is that **it eliminates confusion for the cache between the instruction fetch/decode unit and the execution unit.**

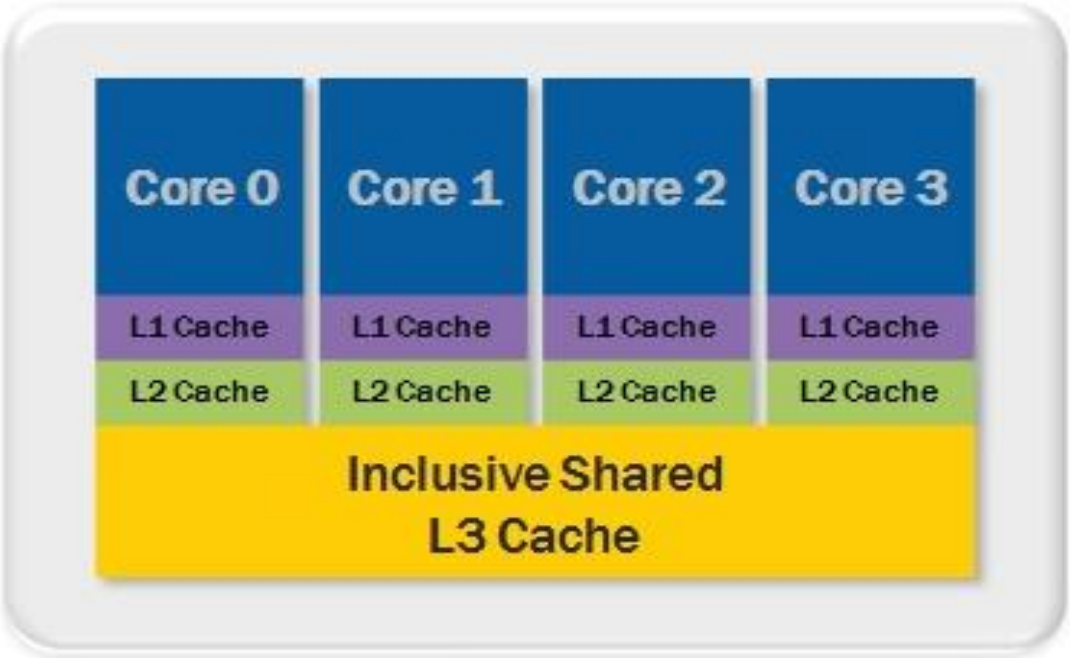
  - This is **important in any design that relies on the Pipelining of instructions**. Typically, the processor will fetch instructions ahead of time and fill a buffer, or pipeline, with instructions to be executed.

- **Example: Smart Cache of i-Series…**

L1 cache: 32 kb Instructions cache +32 kb data cache **{Split }**

L2 cache: 256 kb (For Instructions and data per core) **{Unified}**

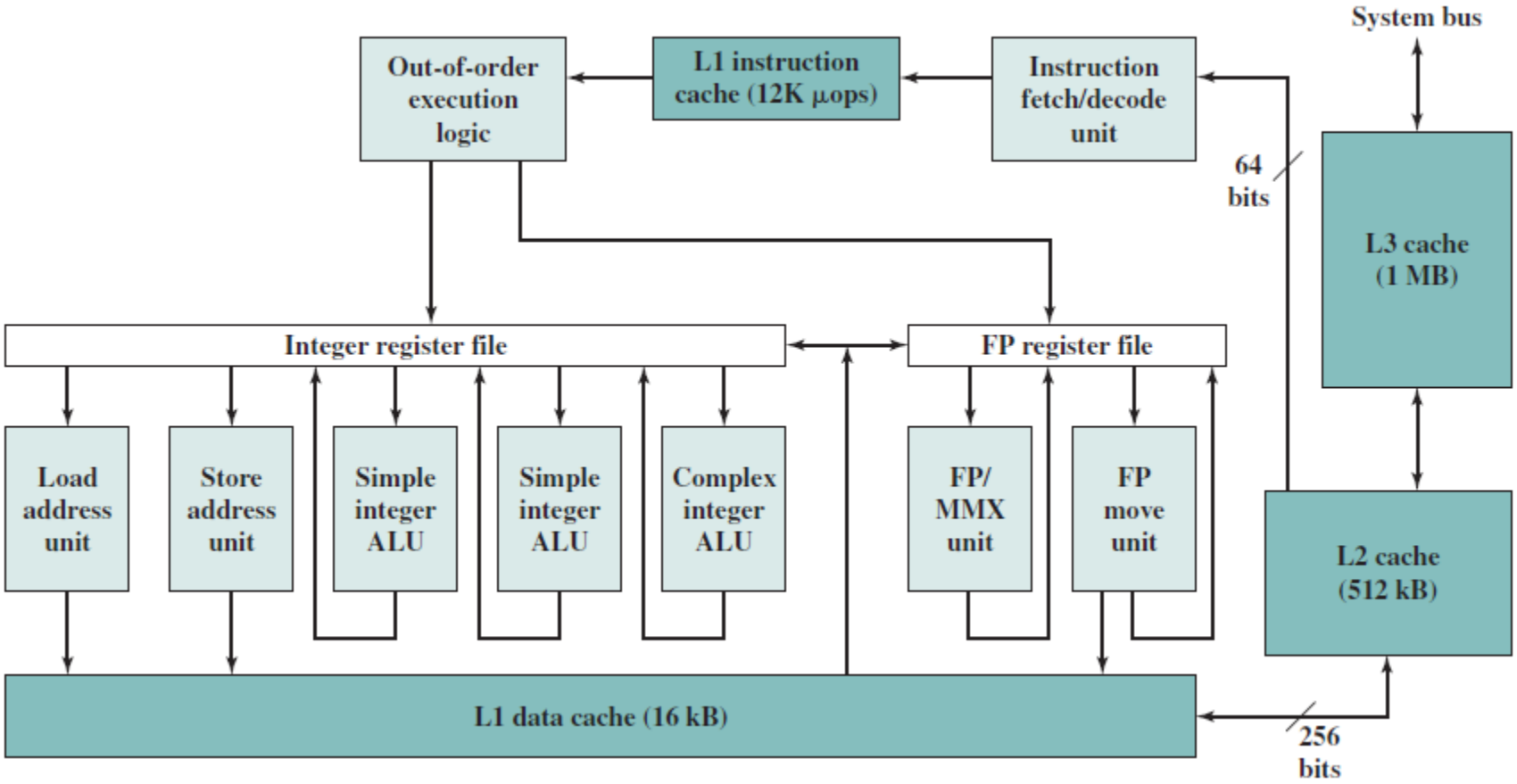
L3 cache: 8 mb (For Communication)



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 Cache Organization



- **Fetch-Decode Unit:** Read instruction from L2 cache, decodes it into set of micro-operations and stores it to L1 instruction cache.

- **Out-of-order execution logic:** Schedules execution of the micro-operations subject to data dependencies and resource availability; thus, micro-operations may be scheduled for execution in a different order than they were fetched from the instruction stream.

- **Execution units:** These units executes micro-operations, fetching the required data from the L1 data cache and temporarily storing results in registers.

- **Memory subsystem:** This unit includes the L2 and L3 caches and the system bus, which is used to access main memory when the L1 and L2 caches have a cache miss and to access the system I/O resources.

Internal Memory

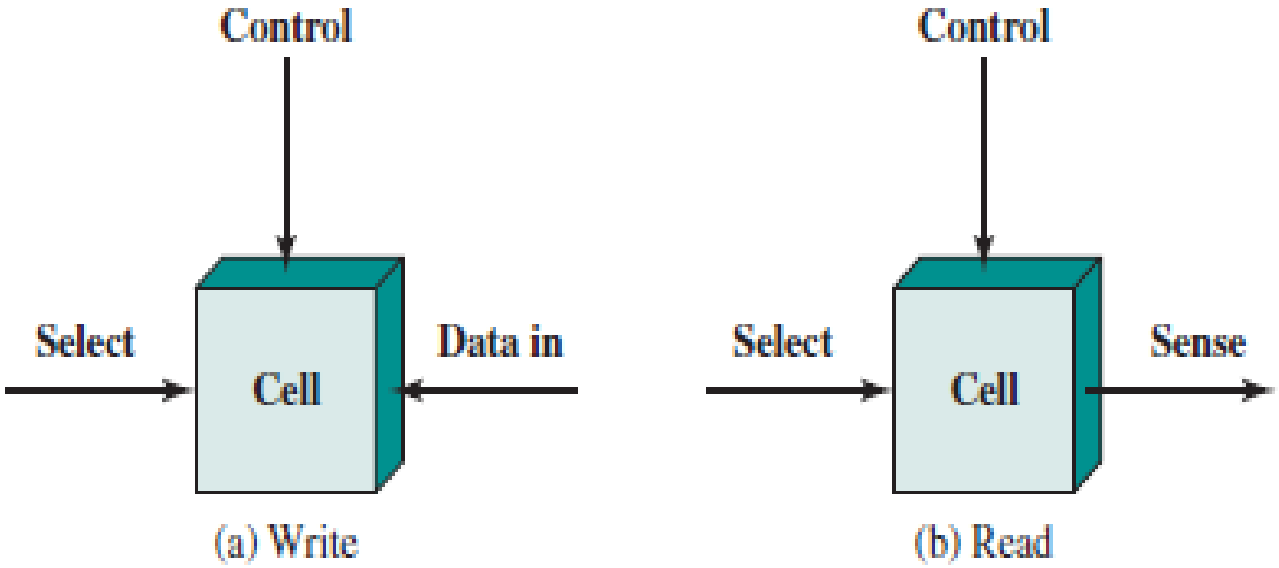
- **Semiconductor Main Memory**
  1. Organization
  2. DRAM and SRAM
  3. Types of ROM
  4. Chip Logic
  5. Chip Packaging
  6. Module Organization
  7. Interleaved Memory

1. Organization

- The basic element of a semiconductor memory is the **Memory Cell.** Although a variety of electronic technologies are used, all semiconductor memory cells share certain properties.

- Properties:
  - They exhibit **two stable states**, which can be used to represent **binary 1 and 0**.
  - They are **capable of being written into** (at least once), to set the state.
  - They are **capable of being read to** sense the state.

Memory Cell Operation



- **Select:** This selects memory to read data or to write data.

- **Control:** It will specify the operation; whether is it read or write.

- Third signal, when cell is writing data, it is set to 1. And when cell is reading data terminal provides Cell's state.

2. DRAM and SRAM

- **Dynamic RAM (DRAM)** is made with cells that store data as charge on capacitors.

- The **presence or absence of charge** in a capacitor is interpreted as a **binary 1 or 0.**

- Because capacitors have a natural tendency to discharge, **dynamic RAMs require periodic charge refreshing** to maintain data storage.

- The term **dynamic refers to this tendency of the stored charge to leak away**, even with power continuously applied.

- **Static RAM (SRAM)** is a digital device that uses the same logic elements used in the processor.

- In a SRAM, **binary values are stored using traditional flip-flop logic-gate configurations.**

- A static RAM will hold its data as long as power is supplied to it.

3. Types of ROM

- As the name suggests, a **Read-only Memory (ROM)**.
- It **contains a permanent pattern of data** that cannot be changed.
- A ROM is **nonvolatile**; that is, no power source is required to maintain the bit values in memory.

- **Applications of ROMs**

1. Microprogramming (TSR's)
2. Library Subroutines for frequently wanted functions (ISR's)
3. System Programs (Drivers etc)
4. Function Tables (GDT, LDT, IDT)

- When only a small number of ROMs with a particular memory content is needed, a less expensive alternative is the **Programmable ROM (PROM).** Like the ROM, the PROM is nonvolatile and may be written into only once.

- Data is written electrically once by manufacturer or Customer.

- Another variation on read-only memory is the **Read-mostly Memory**, which is useful for applications in which read operations are far more frequent than write operations but for which nonvolatile storage is required.

- There are three common forms of read-mostly memory: **EPROM, EEPROM,** and **Flash Memory**.

- The optically **Erasable Programmable Read-only Memory (EPROM)** is read and written electrically, as with PROM.
- However, before a write operation, all the storage cells must be erased to the same initial state.
- Thus, the EPROM can be altered multiple times and, like the ROM and PROM, holds its data virtually indefinitely.
- A more attractive form of read-mostly memory is **Electrically Erasable Programmable Read-only Memory (EEPROM).**
- This is a read-mostly memory that can be written into at any time without erasing prior contents; only the byte or bytes addressed are updated.

- Another form of semiconductor memory is **Flash Memory.**

- An entire flash memory can be erased in one or a few seconds, which is much faster than EPROM.

- In addition, it is possible to erase just blocks of memory rather than an entire chip.

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

4. Chip Logic

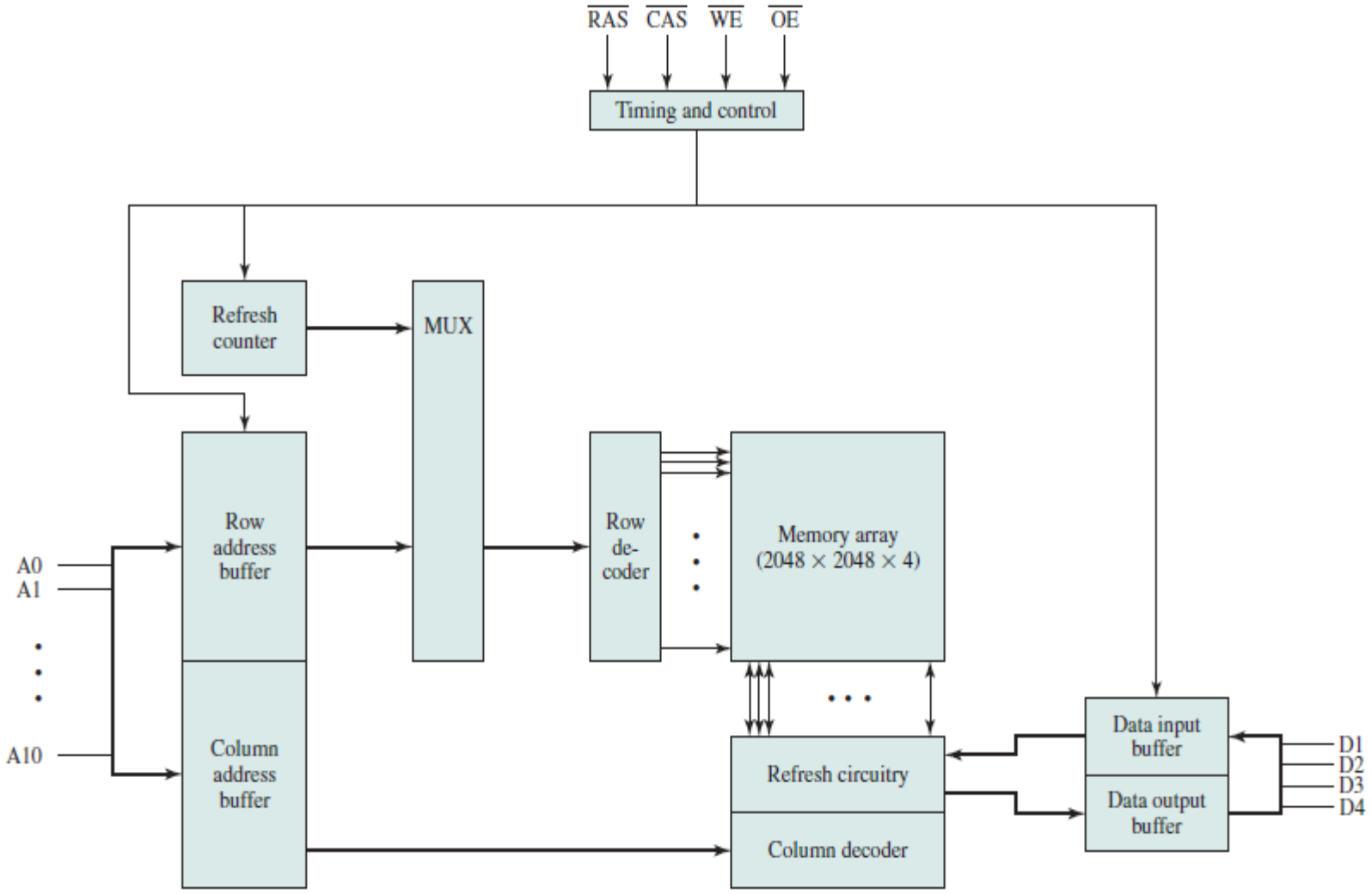
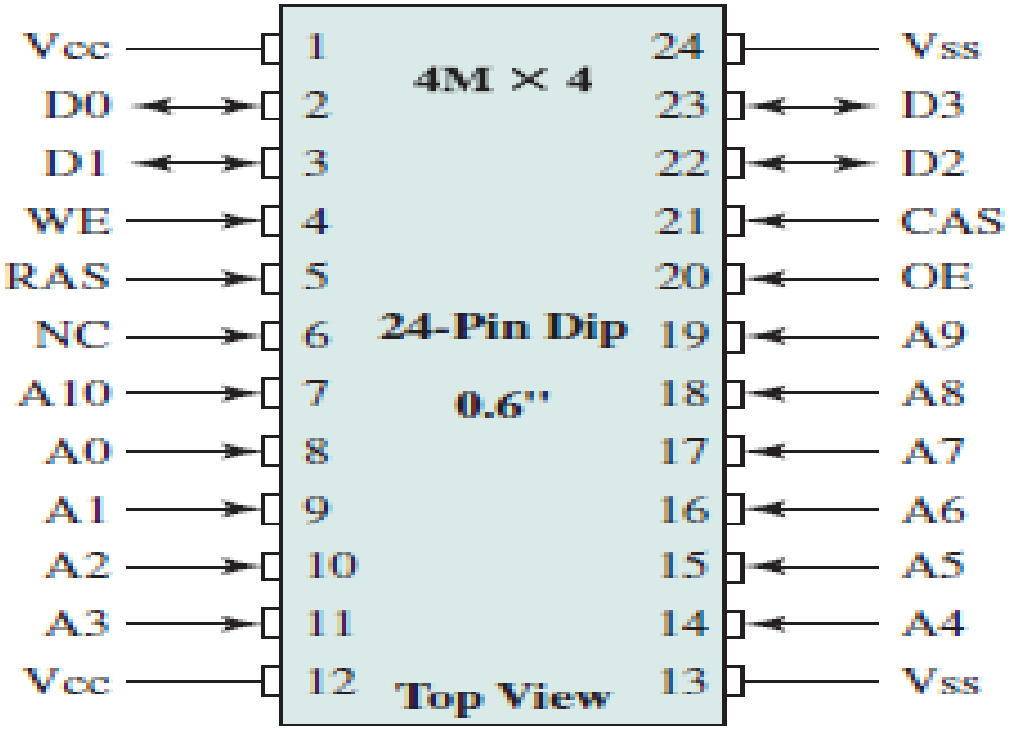


Figure 5.3 Typical 16 Megabit DRAM (4M × 4)

5. Chip Packaging



(b) 16-Mbit DRAM

Advanced DRAM Organization

- Because of high cost of SRAM, mainly DRAM's are used widely.

- Following are the different kinds of DRAM's available
  - **SDRAM (Synchronous DRAM)**
  - **RDRAM (Rambus DRAM)**
  - **DDR-DRAM (Double Data Rate DRAM)**
  - **CDRAM (Cache DRAM)**

Performance Comparison of Some DRAM Alternatives

	Clock Frequency (MHz)	Transfer Rate (GB/s)	Access Time (ns)	Pin Count
SDRAM	166	1.3	18	168
DDR	200	3.2	12.5	184
RDRAM	600	4.8	12	162

Working of Traditional DRAM

- In typical DRAM, Processor gives addresses and control levels to the memory, to read or write data into the DRAM.

- When addresses and control levels are received memory, Processor goes to waiting state.

Waiting Time of Processor= Access time of RAM.

- During the waiting time of processor, the DRAM performs various internal functions, such as required calculations of the row and column lines, sensing the data, and routing the data out through the output buffers.

- **The processor must simply wait through this delay**, slowing system performance.

Synchronous DRAM (SDRAM)

- One of the most widely used form of DRAM.
- Traditional DRAM is **Asynchronous**, where SDRAM is **Synchronous**.
- The SDRAM exchanges data with the processor, which is synchronized to an external clock signal.
- And also runs at the full speed of the processor/memory bus without imposing wait states.

- With synchronous access, the DRAM performs data read or write operations under control of the system clock.

- The processor gives required information to DRAM.

- The DRAM then responds after a set number of clock cycles.

- Meanwhile, the master can safely do other tasks while the SDRAM is processing the request.

Rambus DRAM

- RDRAM, developed by Rambus, later used by Intel for its Pentium and Itanium processors.

- RDRAM chips are vertical packages, with all pins on one side. The chip exchanges data with the processor over 28 wires no more than 12 centimeters long.

- The bus can address up to 320 RDRAM chips and is rated at 1.6 GBps.

- The special RDRAM bus delivers address and control information using an **Asynchronous Block-oriented Protocol.**

- After an initial 480 ns access time, this produces the 1.6 GBps data rate.

- What makes this speed possible is the bus itself, which defines all signals very precisely. Rather than being controlled by the explicit RAS(#), CAS(#), R/W(#), and CE(#) signals used in conventional DRAMs.

DDR SDRAM

- SDRAM is limited by the fact that it can only send data to the processor once per bus clock cycle.

- A new version of SDRAM, referred to as **Double-data-rate SDRAM** can send data twice per clock cycle, once on the rising edge of the clock pulse and once on the falling edge.

- There have been two generations of improvement to the DDR technology.

- DDR2 increases the data transfer rate by increasing the operational frequency of the RAM chip and by increasing the **Prefetch Buffer** from 2 bits to 4 bits per chip.

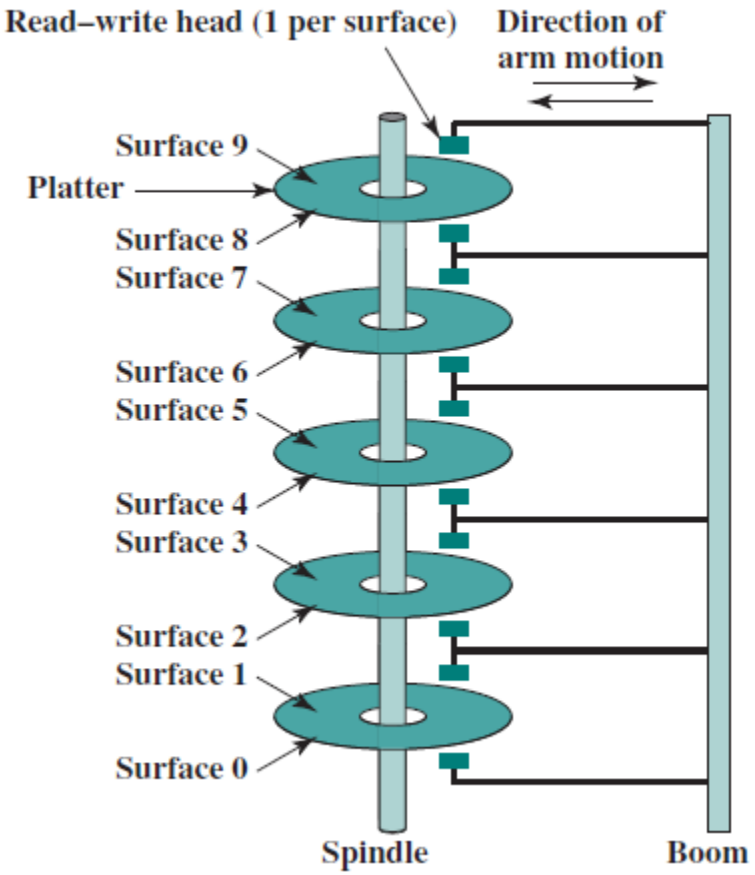
- The **Prefetch Buffer** is a memory cache located on the RAM chip. The buffer enables the RAM chip to preposition bits to be placed on the data bus as rapidly as possible.

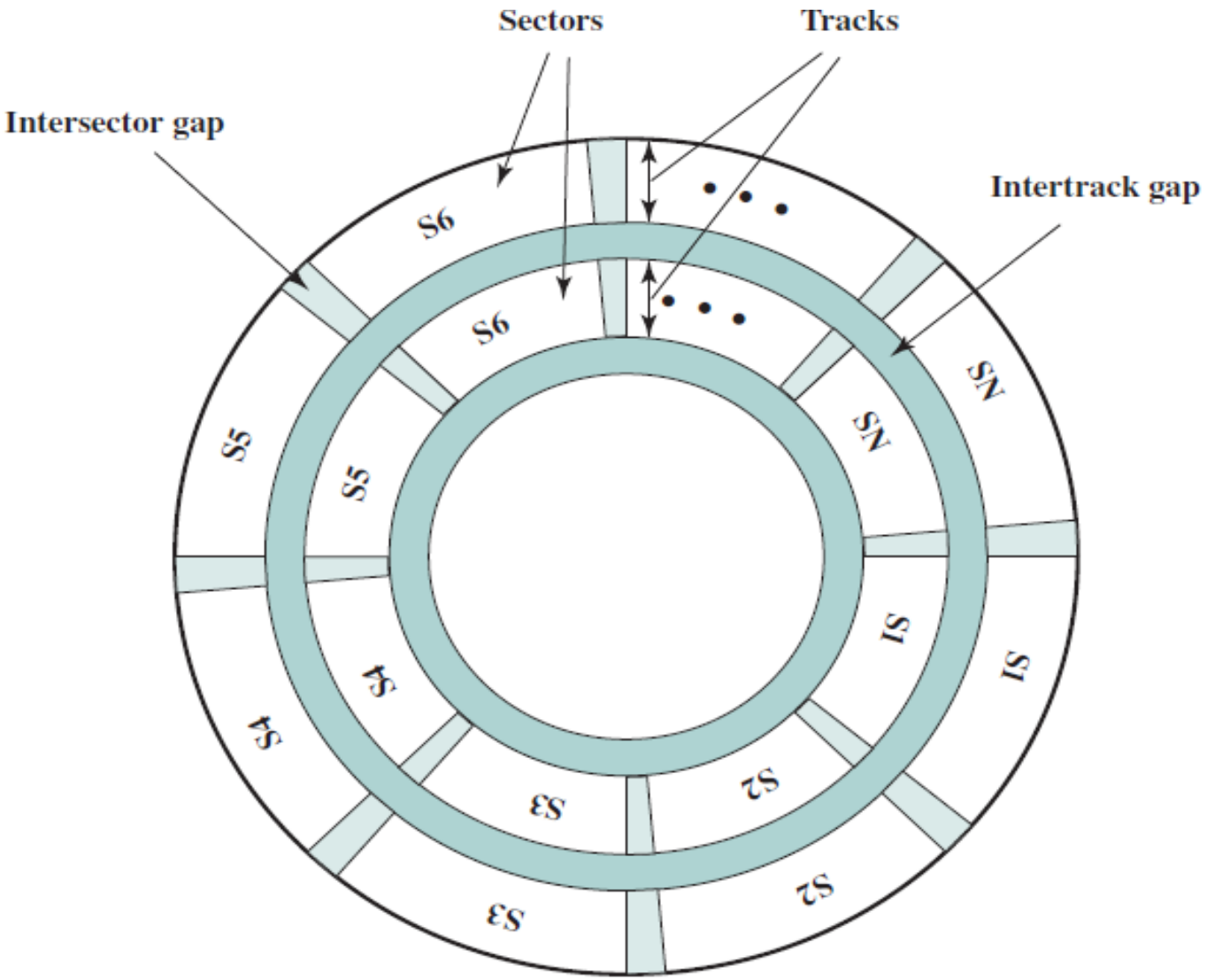
- DDR3, introduced in 2007, increases the prefetch buffer size to 8 bits.

Cache DRAM

- Cache DRAM (CDRAM) integrates a small SRAM cache (16 Kb) onto a generic DRAM chip.
- The SDRAM on the CDRAM can be used in two ways.
- First, it can be used as a **True Cache**, consisting of a number of 64-bit lines.
- The cache mode of the CDRAM is effective for ordinary random access to memory.

Hard Disk Organization




Sectors
Tracks
Intersector gap
Intertrack gap
S6
S6
S5
S5
S4
S4
S3
S3
S2
S2
S1
S1
N5
N5

- Data are recorded on and later retrieved from the disk via a conducting coil named the **Head**;
- In many systems, **there are two heads, a Read Head and a Write Head**.
- During a read or write operation, the **head is stationary while the platter rotates beneath it**.
- The head is a relatively small device capable of reading from or writing to a portion of the platter rotating beneath it.
- This gives rise to the organization of data on the platter in a concentric set of rings, called **Tracks**.

- Each track is the same width as the head. **There are thousands of tracks per surface**.
- Adjacent tracks are separated by **Gaps**.
- This prevents, or at least minimizes, errors due to misalignment of the head or data overwriting.
- Data are transferred to and from the disk in **Sectors.**
- There are **typically hundreds of sectors per track**, and these **may be of either fixed or variable length.**

RAID

- RAID is a technology that is used to increase the performance and/or reliability of data storage.
- The abbreviation stands for ***Redundant Array of Inexpensive Disks***.
- A RAID system consists of two or more drives working in parallel.
- These disks can be hard disks.

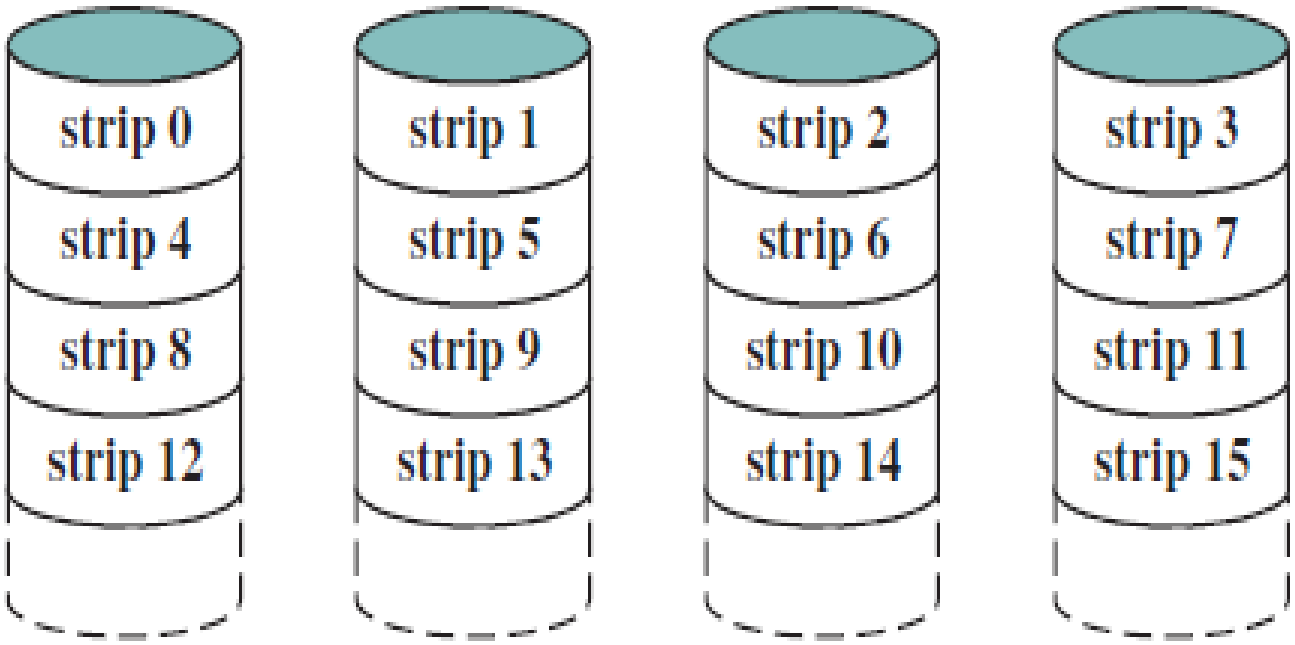
- The RAID scheme consists of seven levels,0 to 6.

- These levels have different design architectures that share three common characteristics:

  1. RAID is a set of physical disk drives viewed by the operating system as a **Single Logical Drive**.
  2. Data are distributed across the physical drives of an array in a scheme known as **Striping**.
  3. Redundant disk capacity is used to store parity information, which guarantees **Data Recoverability** in case of a disk failure.

It is not necessary that, each RAID level supports all these three characteristics.

RAID 0 (Non-Redundant)



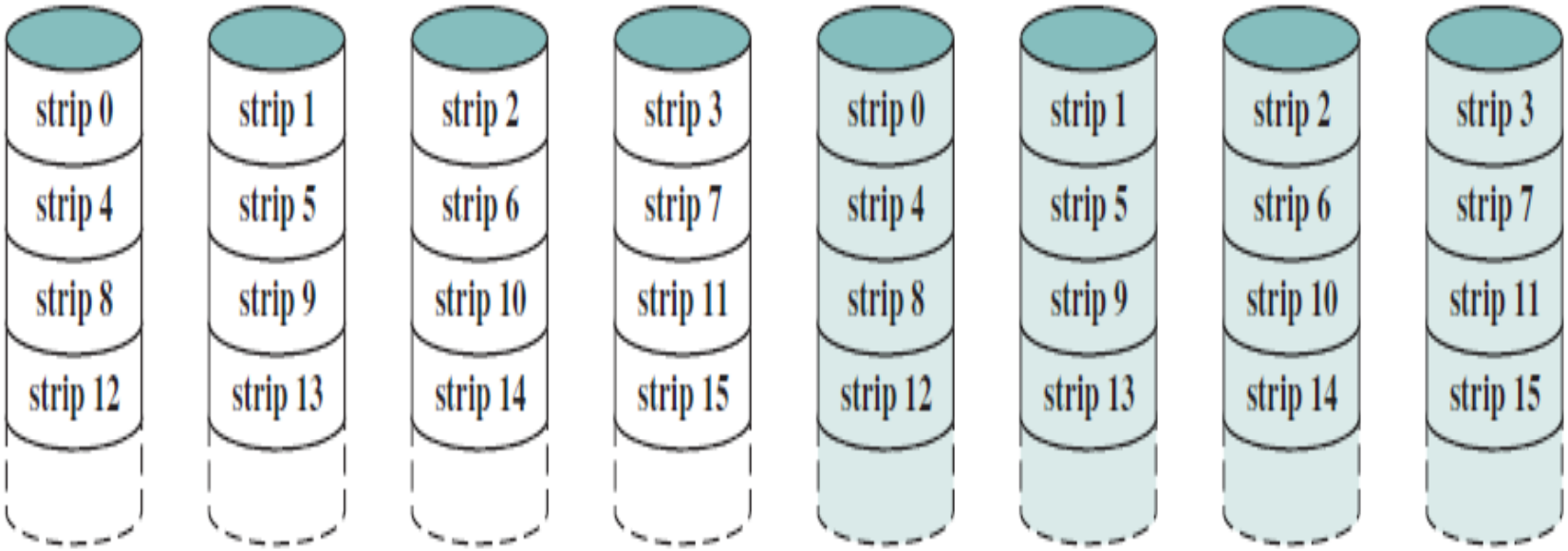
(a) RAID 0 (Nonredundant)

- **Advantages**
  1. Two different requests can be issued on same time.
  2. As **data is stored in a Round Robin manner in stripes**, single I/O request which want to access contiguous memory blocks, can access data easily.

- **Disadvantages**
  1. RAID 0 is **Not Fault-tolerant**.

- **Use**
  1. RAID 0 is **ideal for non-critical storage** of data that have to be read/written at a high speed, such as on an image retouching or video editing station.

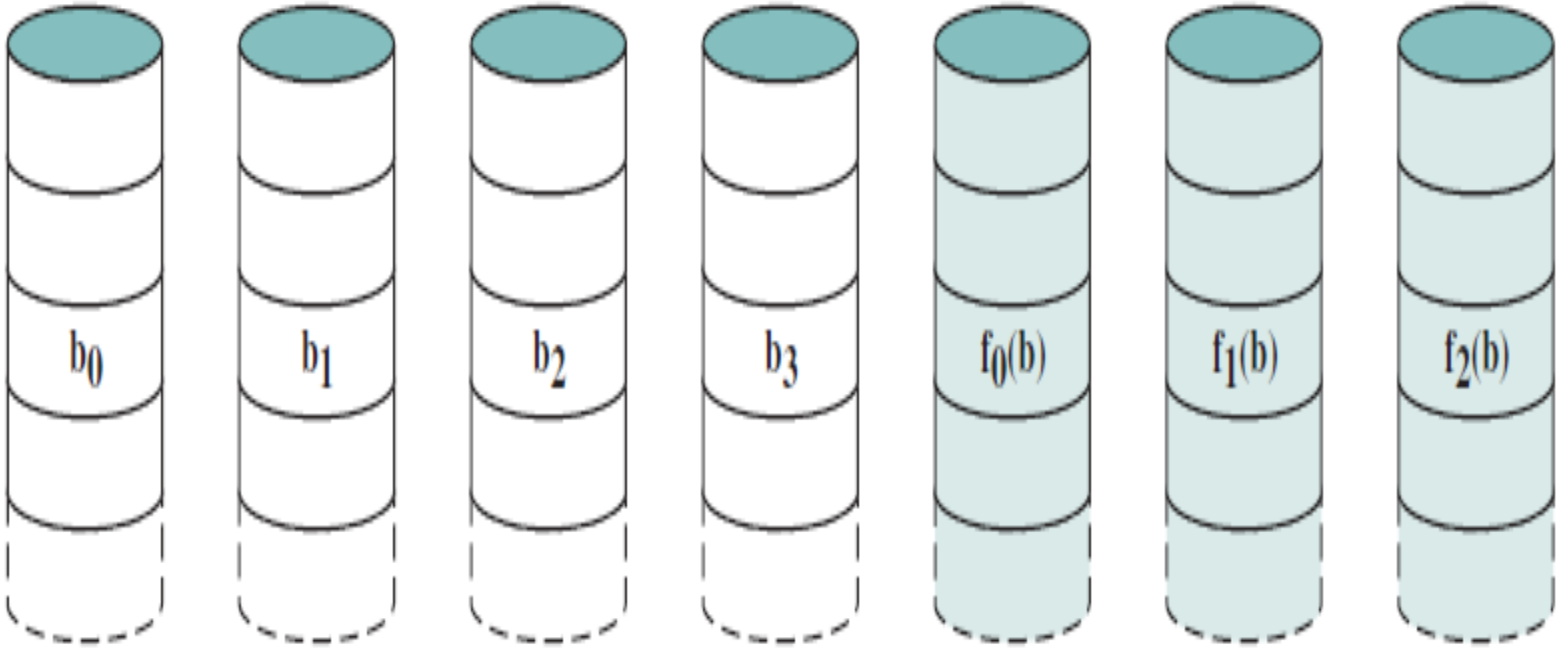
RAID 1 (Mirrored)



(b) RAID 1 (Mirrored)

- **Advantages**
  1. Very **simple techn**ique.
  2. If data is lost, **can be easily be re-retrived**.
  3. RAID 1 offers **excellent read speed and a write-speed** that is comparable to that of a single drive.
- **Disadvantages**
  1. **Effective storage capacity is only half** of the total drive capacity.
  2. Failed drive can only be replaced after powering down the computer it is attached to (**Hot Swapping**). For servers that are used simultaneously by many people, this may not be possible.
- **Use**
  1. RAID-1 is **ideal for mission critical storage**, for example accounting systems.

RAID 2 (Redundancy through Hamming Code)



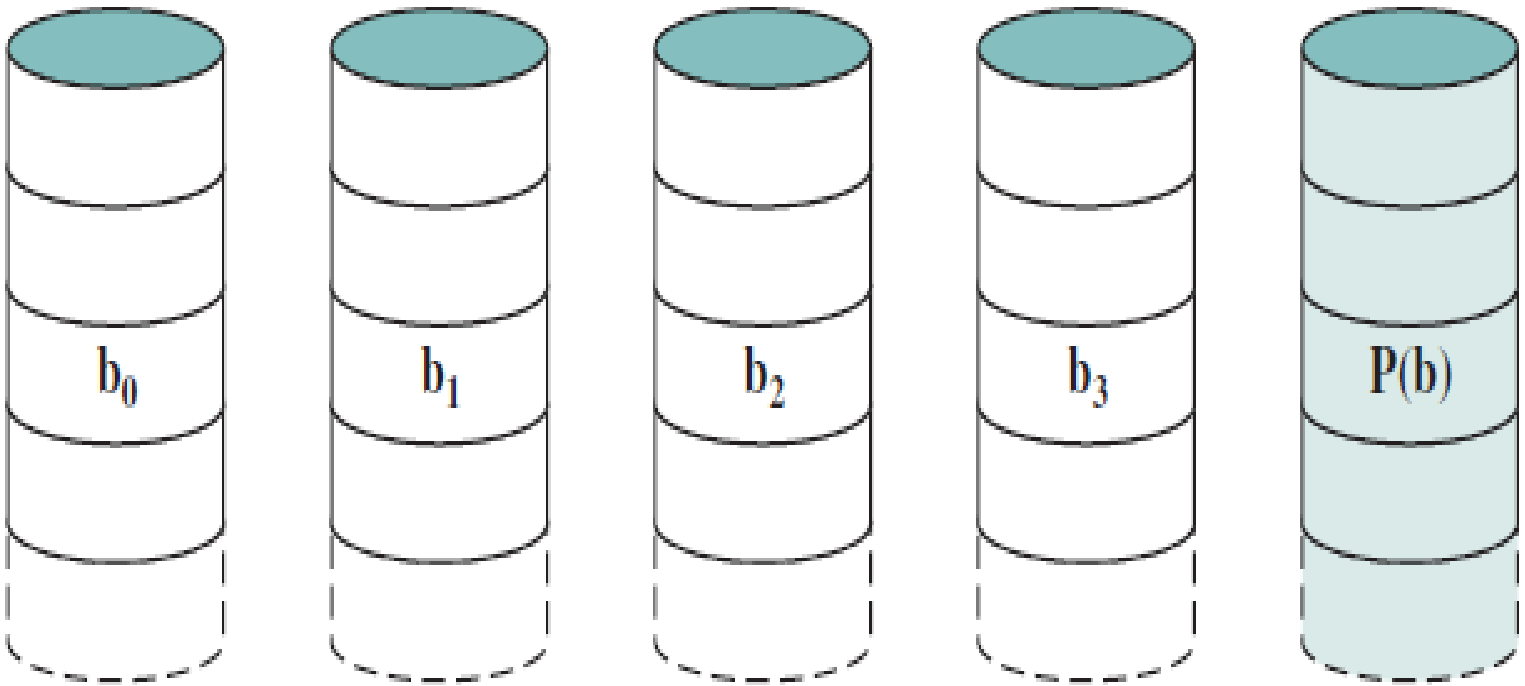
(c) RAID 2 (Redundancy through Hamming code)

- **Advantages**
  1. **Hamming code is used**, which is able to correct single-bit errors and detect double-bit errors.
  2. If data is lost, **can be easily be re-retrived**.

- **Disadvantages**
  1. Costly.
  2. Read access time is slower.

- **Use**
  1. Effective choice in an **environment in which many disk errors occur.**

RAID 3 (Bit-interleaved Parity)



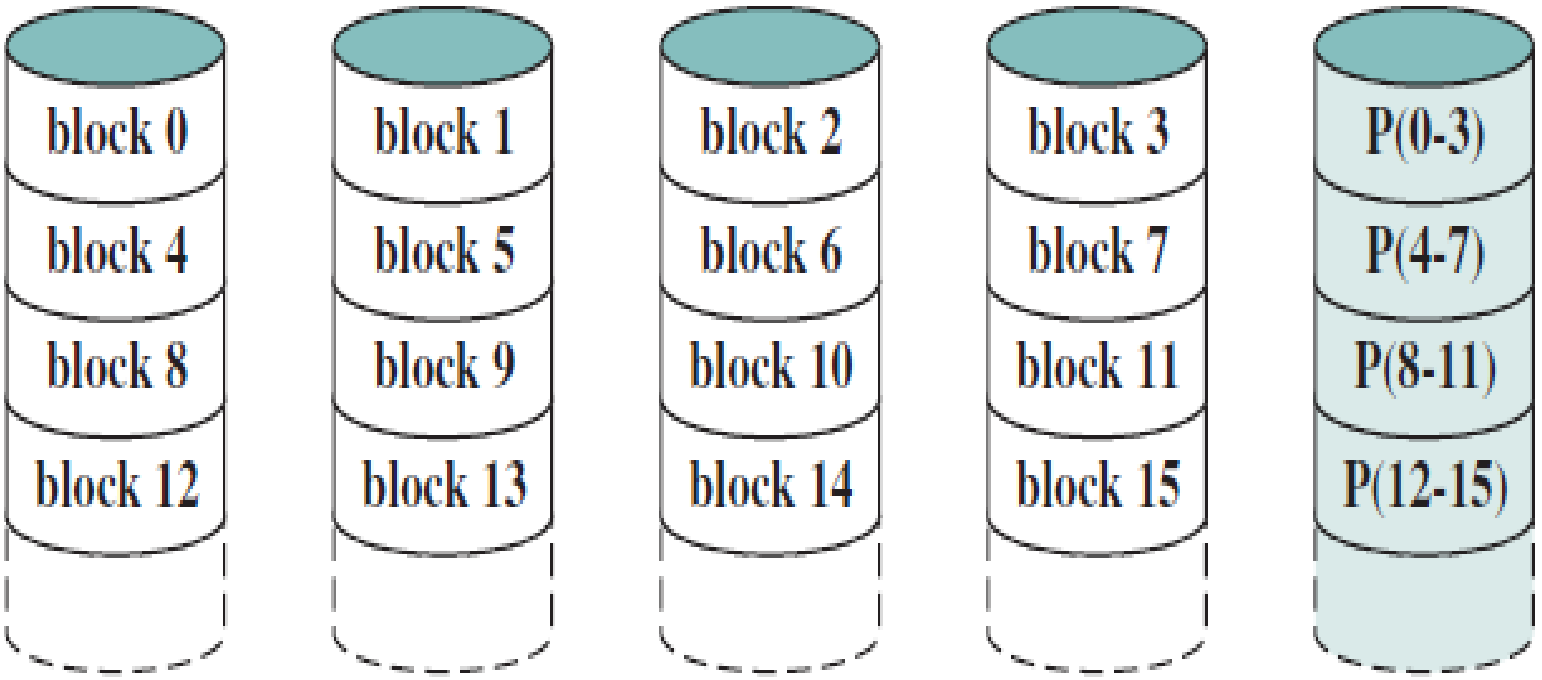
(d) RAID 3 (Bit-interleaved parity)

- **Advantages**
  1. Because data are striped in very small strips, RAID 3 can achieve very high data transfer rates.
  2. For large data transfers, performance is very good.

- **Disadvantages**
  1. Only one I/O request can be executed at a time. Thus, in a transaction-oriented environment, performance suffers.

- **Use**
  1. This makes it suitable for applications that demand the **highest transfer rates in long sequential reads and writes**, for example uncompressed video editing.

RAID 4 (Block-level Parity)



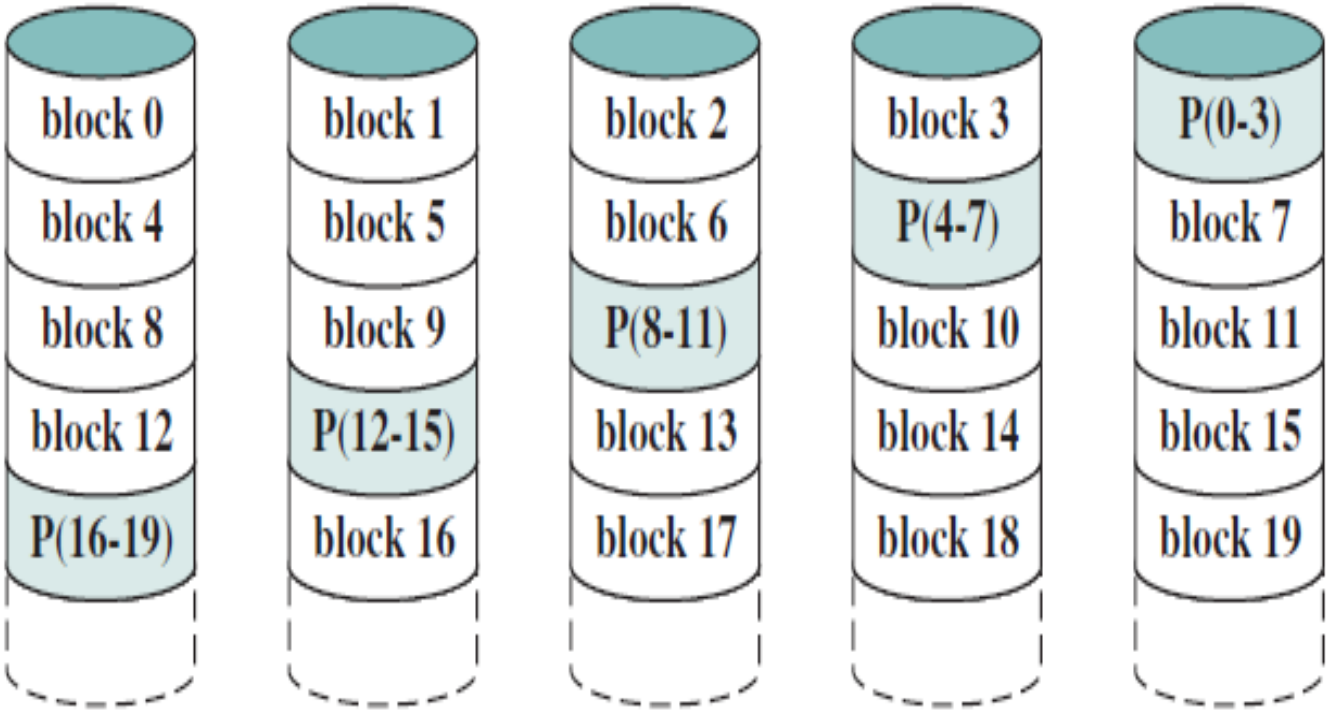
(e) RAID 4 (Block-level parity)

- **Advantages**
  1. **Separate parity disk**, where parity bit of each block is stored.

- **Disadvantages**
  1. With change in a data, every time parity disk is need to be updated.
  2. Every write operation must involve the parity disk, which therefore can become a bottleneck.

- **Use**
  1. No commercial implementations exist/ not commercially viable.

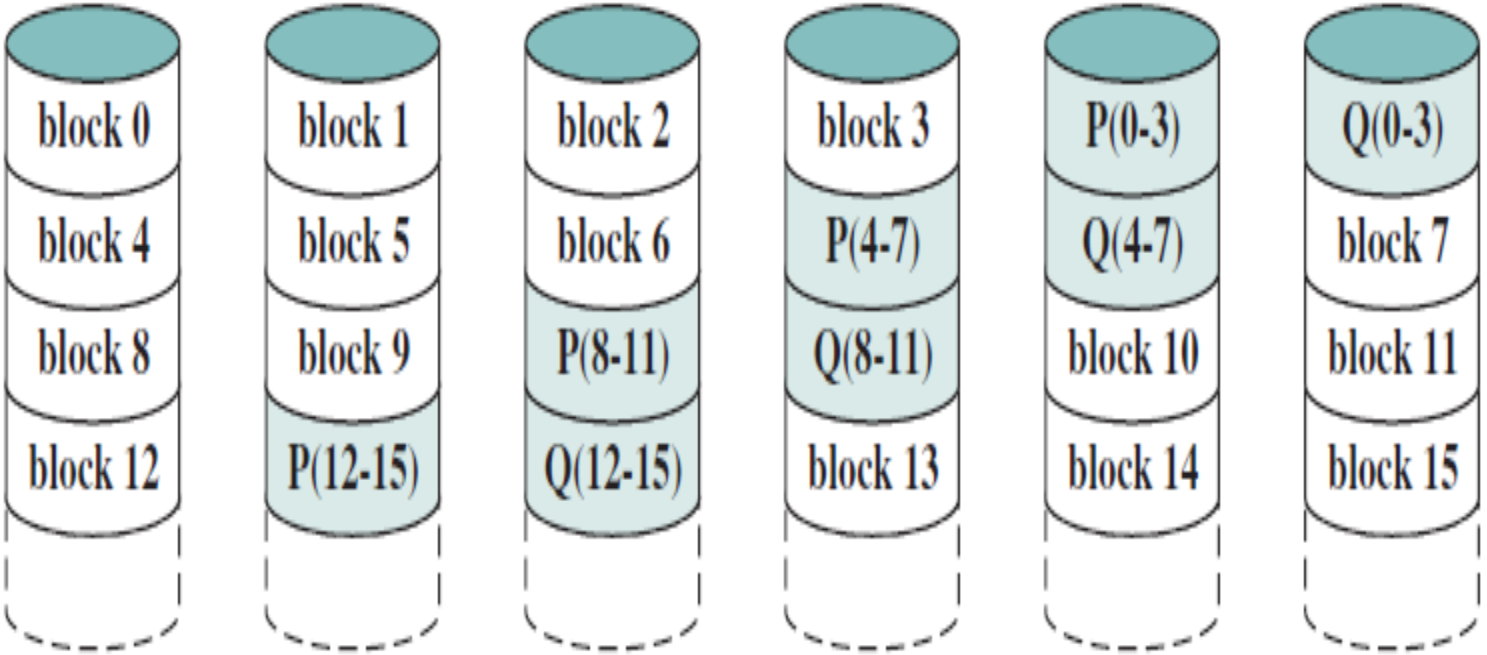
RAID 5 (Block-level Distributed Parity)



(f) RAID 5 (Block-level distributed parity)

- **Advantages**
  1. Read data transactions are very fast while write data transactions are somewhat slower (due to the parity that has to be calculated).
  2. If a drive fails, **you still have access to all data**, even while the failed drive is being replaced and the storage controller rebuilds the data on the new drive.
- **Disadvantages**
  1. Drive failures have an effect on throughput, although this is still acceptable.
  2. This is complex technology, so time required to recover data may be longer.
- **Use**
  1. RAID 5 is a good all-round system that combines efficient storage with excellent security and decent performance. It is **ideal for file and application servers** that have a limited number of data drives.

RAID 6 (Dual Redundancy)



(g) RAID 6 (Dual redundancy)

- **Advantages**
  1. Like with RAID 5, read data transactions are very fast.
  2. If two drives fail, you still have access to all data, even while the failed drives are being replaced. So RAID 6 is more secure than RAID 5.
- **Disadvantages**
  1. Write data transactions are slowed down due to the parity that has to be calculated.
  2. Drive failures have an effect on throughput, although this is still acceptable.
  3. This is complex technology. Rebuilding an array in which one drive failed can take a long time.
- **Use**
  1. RAID 6 is a good all-round system that combines efficient storage with excellent security and decent performance. It is preferable over RAID 5 in file and application servers that use many large drives for data storage.