William Stallings
Computer Organization and Architecture
7th Edition

Chapter 5

Memory

4.0	**Memory Organization**.		11	CO3
	4.1	Characteristics of memory system and hierarchy, Main memory, Cache memory principles , Elements of Cache Design		
	4.2	ROM, Types of ROM, RAM, SRAM, DRAM, Flash memory, High speed memories		
	4.3	Cache Memory Organization: Address mapping, Replacement Algorithms, Cache Coherence, MESI protocol, Interleaved and associative memories, Virtual memory, Main memory allocation, Segmentation ,Paging, Secondary storage, RAID levels		

SYLLABUS

Characteristics of memory system and hierarchy,

Main memory ,ROM, Types of ROM, RAM,

SRAM, DRAM, Flash memory, High speed memories

Cache Memory Organization: Address mapping, Replacement Algorithms

Cache Coherence, MESI protocol, Interleaved and associative memories

Virtual memory, main memory allocation, segmentation paging, secondary storage ,RAID levels

Table 4.1 Key Characteristics of Computer Memory Systems

Location	**Performance**
Internal (e.g., processor registers, cache, main memory)	Access time
External (e.g., optical disks, magnetic disks, tapes)	Cycle time
Capacity	Transfer rate
Number of words	**Physical Type**
Number of bytes	Semiconductor
Unit of Transfer	Magnetic
Word	Optical
Block	Magneto-optical
Access Method	**Physical Characteristics**
Sequential	Volatile/nonvolatile
Direct	Erasable/nonerasable
Random	**Organization**
Associative	Memory modules

Characteristics of Memory

1. Location
2. Capacity
3. Unit of transfer
4. Access method
5. Performance(SRAM,DRAM)
6. Physical type
7. Physical characteristics
8. Organisation
    1. Direct Mapping,Associative Mapping

1. Location

•Internal memory/External

•Local mem-Registers

•Cache -internal.

• External memory-peripheral storage devices, such as disk and tape, that are accessible to the processor via I/O controllers

2. Capacity

- Word size
  - —The natural unit of organisation
- Internal mem-bytes/words
- External mem-bytes
- Number of words
  - —or Bytes

Types of various Units of Memory-

Byte

Kilo Byte Mega Byte

Giga Byte Tera Byte

Peta Byte Exa Byte

Zetta Byte Yotta Byte

NAME	EQUAL TO	SIZE(IN BYTES)
Bit	1 bit	1/8
Nibble	4 bits	1/2 (rare)
Byte	8 bits	1
Kilobyte	1024 bytes	1024
Megabyte	1, 024kilobytes	1, 048, 576
Gigabyte	1, 024 megabytes	1, 073, 741, 824
Terrabyte	1, 024 gigabytes	1, 099, 511, 627, 776
Petabyte	1, 024 terrabytes	1, 125, 899, 906, 842, 624
Exabyte	1, 024 petabytes	1, 152, 921, 504, 606, 846, 976
Zettabyte	1, 024 exabytes	1, 180, 591, 620, 717, 411, 303, 424
Yottabyte	1, 024 zettabytes	1, 208, 925, 819, 614, 629, 174, 706, 176

3. Unit of Transfer

•Internal memory, the unit of transfer is equal to the number of electrical lines into and out of the memory module-governed by data bus-width

- •Word-"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.

- •Addressable units-—Smallest location which can be uniquely addressed
- •the relationship between the length in bits A of an address and the number N of addressable units is 2A = N.

- • Units of Transfer-this is the number of bits read out of or written into memory at a time (main memory)
  - •Data is transferred in blocks (external memory)

4. Access Methods (1)

- **Sequential**
  - —Memory is organized into units of data, called records
  - Start at the beginning and read through in order
  - A shared read–write mechanism is used, and this must be moved from its current location to the desired location, passing and rejecting each intermediate record
    - —Access time depends on location of data and previous location
  - —e.g. tape
- **Direct**
  - —Individual blocks have unique address
  - —Access is by jumping to vicinity plus sequential search
  - —involves a shared read–write mechanism

Access time depends on location and previous location

—e.g. disk

- **Random**

—Each addressable location in memory has a unique, physically wired-in addressing mechanism

-Individual addresses identify locations exactly

—Access time is independent of location or previous access

-any location can be selected at random and directly addressed and accessed

—e.g. RAM

Access Methods (2)

- **Associative**
  - —Data is located by a comparison with contents of a portion of the store
  - —Access time is independent of location or previous access
  - -a word is retrieved based on a portion of its contents rather than its address
  - —e.g. cache

5. Performance

- **Access time (latency)**
  - —Time between requesting for operation and the time it is made available at the required location

- **Memory Cycle time**

—Minimum time elapsed between two

consecutive read requests

-consists of the access time plus any additional time required before a second access can commence

- **Transfer Rate**

—Rate at which data can be moved in and out of memory unit

-For RAM, 1\T

-For non-RAM, (PTO)

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

where

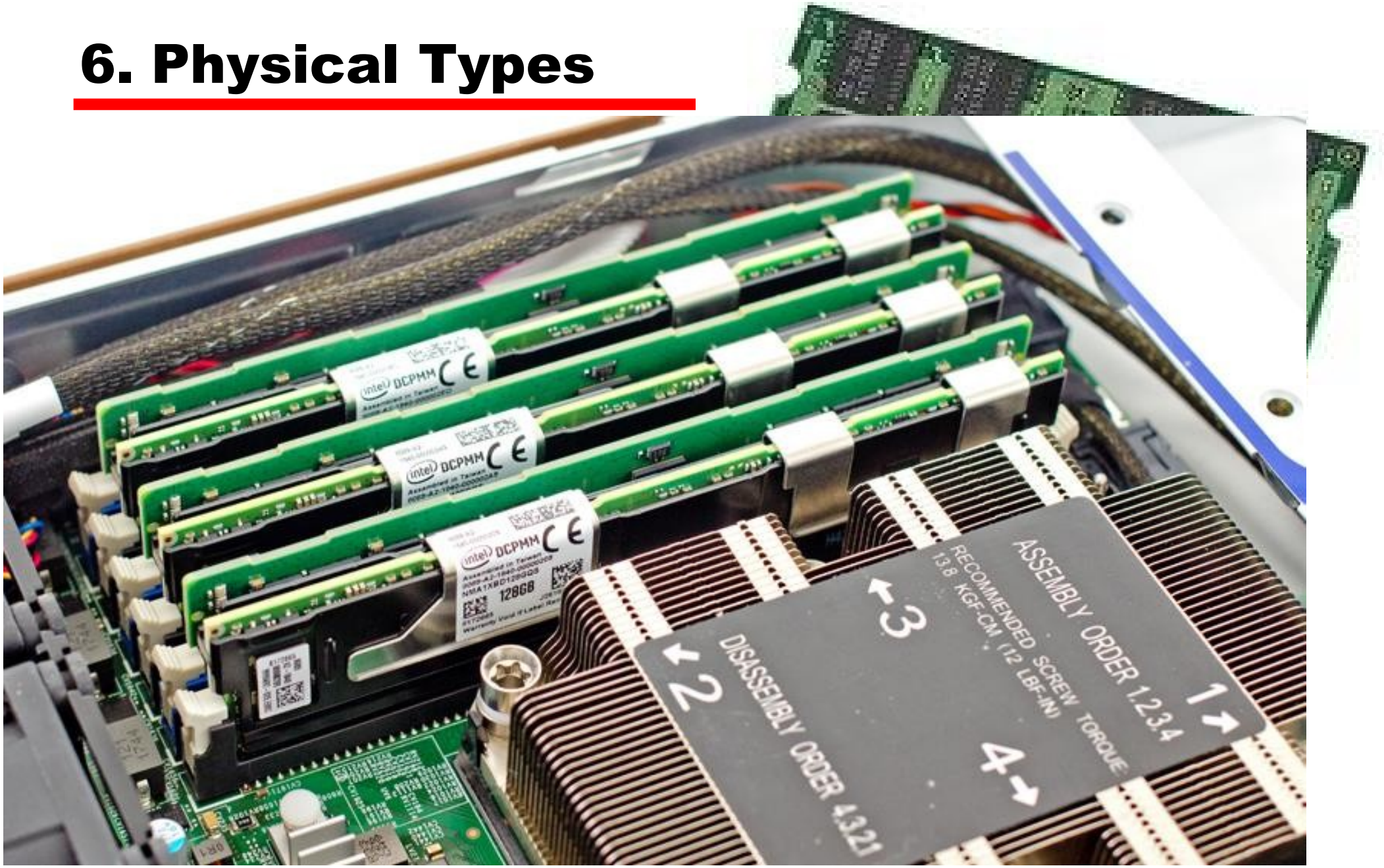
T_n = Average time to read or write n bits

T_A = Average access time

n = Number of bits

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

6. Physical Types

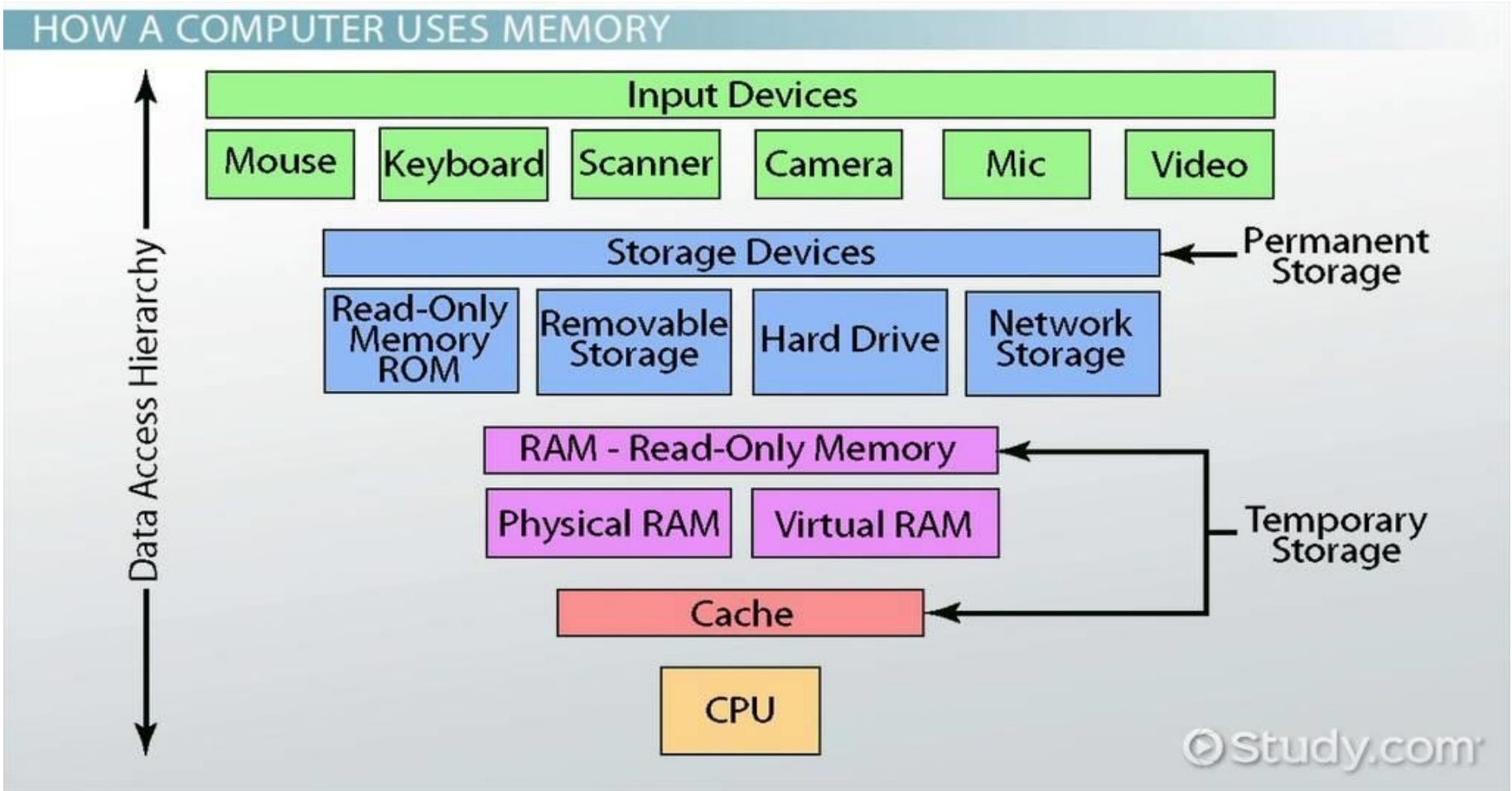


7. Physical Characteristics

- Decay-In a volatile memory, information decays naturally or is lost when electrical power is switched off.
- Volatility- In a nonvolatile memory, information once recorded remains without deterioration until deliberately changed; no electrical power is needed to retain information
- Erasable-Nonerasable memory cannot be altered, except by destroying the storage unit. Semiconductor memory of this type is known as *read-only memory* (ROM)
- Power consumption

8. Organisation

- Physical arrangement of bits into words
- Not always obvious
- e.g. interleaved



Memory Hierarchy

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - “RAM”
- External memory
  - Backing store

- there is a trade–off among the three key characteristics of memory: **capacity, access time, and cost.**
- 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

Memory hierarchy–

As one goes down the hierarchy, the following occur:

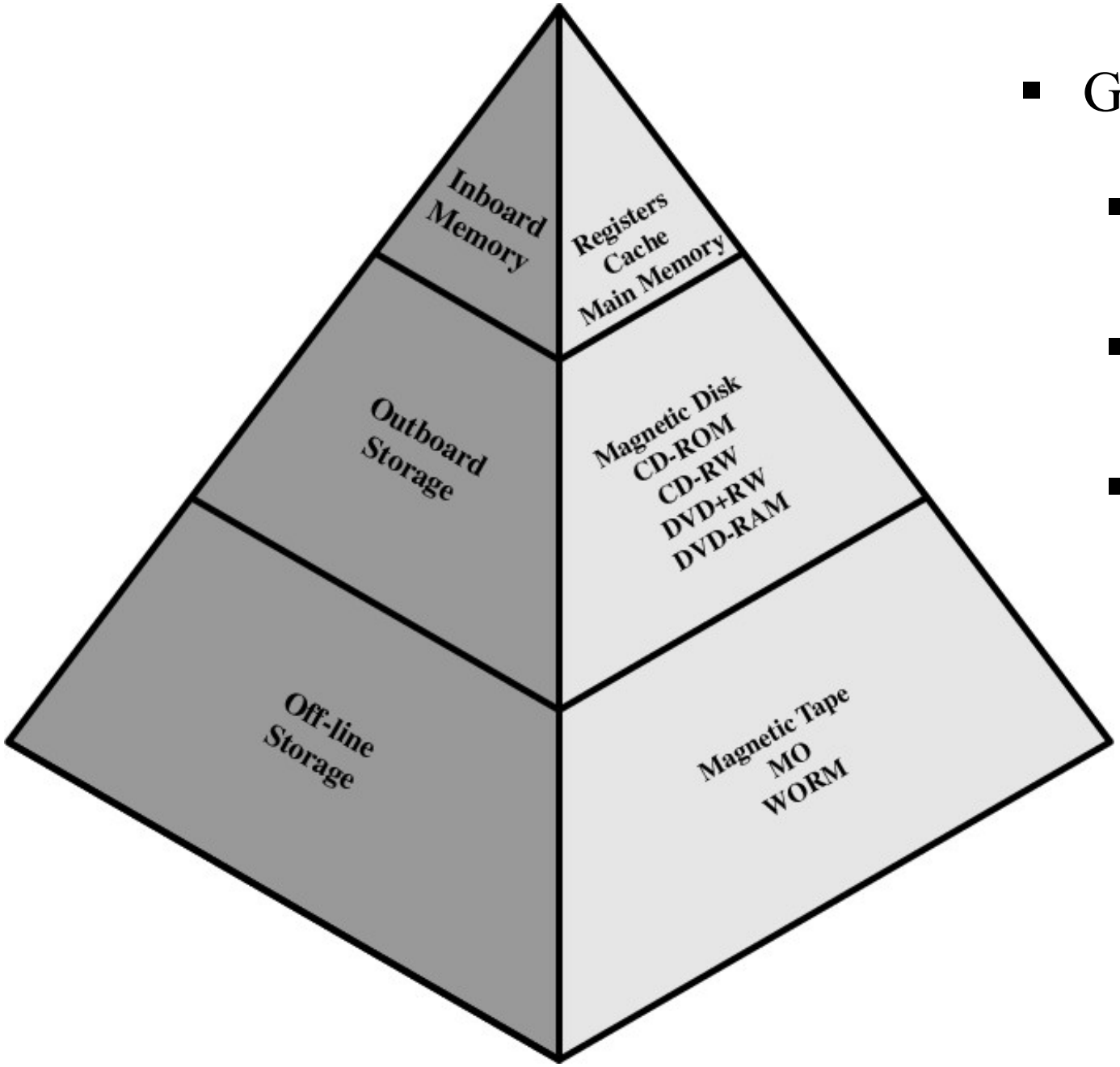
a. Decreasing cost per bit;

b. Increasing capacity;

c. Increasing access time;

d. Decreasing frequency of access of the memory by the processor.

Memory Hierarchy - Diagram



- Going down the hierarchy
  - Decreasing **Cost**
  - Increase **Capacity**
  - Increase **Access Time**

- It is possible to organize data across the hierarchy such that the **percentage of accesses to each successively lower level is substantially less than that of the level above.**
- The fastest, smallest, and most expensive type of memory consists of the **registers internal to the processor.**
- Typically, a processor will contain a few dozen such registers, although some machines contain hundreds of registers.
- **Main memory** is the principal internal memory system of the computer. Each location in main memory has a unique address. Main memory is usually extended with a higher–speed, smaller cache.
- The cache is not usually visible to the programmer or, indeed, to the processor.
- It is a device for staging the **movement of data between main memory and processor registers to improve performance.**
- Data are stored more permanently on external mass storage devices, of which the most common are **hard**

- External, nonvolatile memory is also referred to as **secondary memory** or **auxiliary memory**.
- These are used to store program and data files and are usually visible to the programmer only in terms of files and records, as opposed to individual bytes or words.
- Disk is also used to provide an extension to main memory known as virtual memory

Semiconductor Memory

- Semiconductor memory is a type of digital memory technology that uses semiconductors, such as silicon, to store and retrieve digital data.
- the use of semiconductor chips for main memory is almost universal.
- The basic element of a semiconductor memory is the **memory cell**
- All semiconductor memory cells share certain properties:
  - They exhibit two stable (or semi-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.

- the cell has three functional terminals capable of carrying an electrical signal.
- The select terminal, as the name suggests, selects a memory cell for a read or write operation.
- The control terminal indicates read or write.
- For writing, the other terminal provides an electrical signal that sets the state of the cell to 1 or 0.
- For reading, that terminal is used for output of the cell's state.

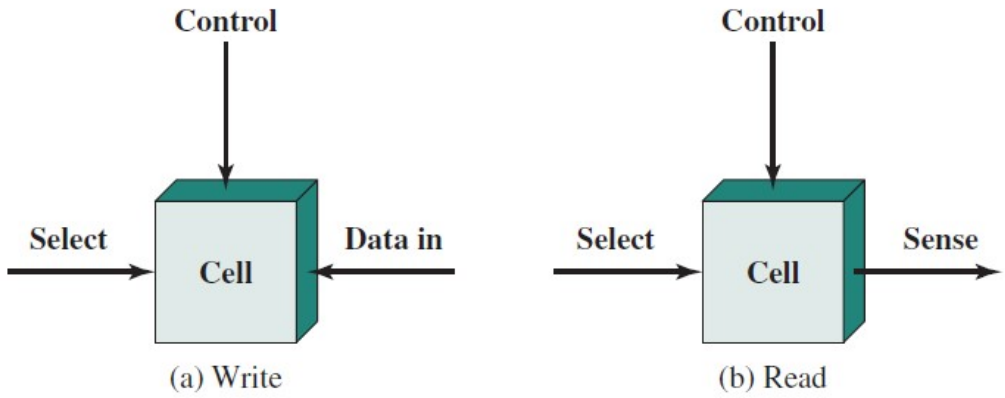


Figure 5.1 Memory Cell Operation

Table 5.1 Semiconductor Memory Types

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		

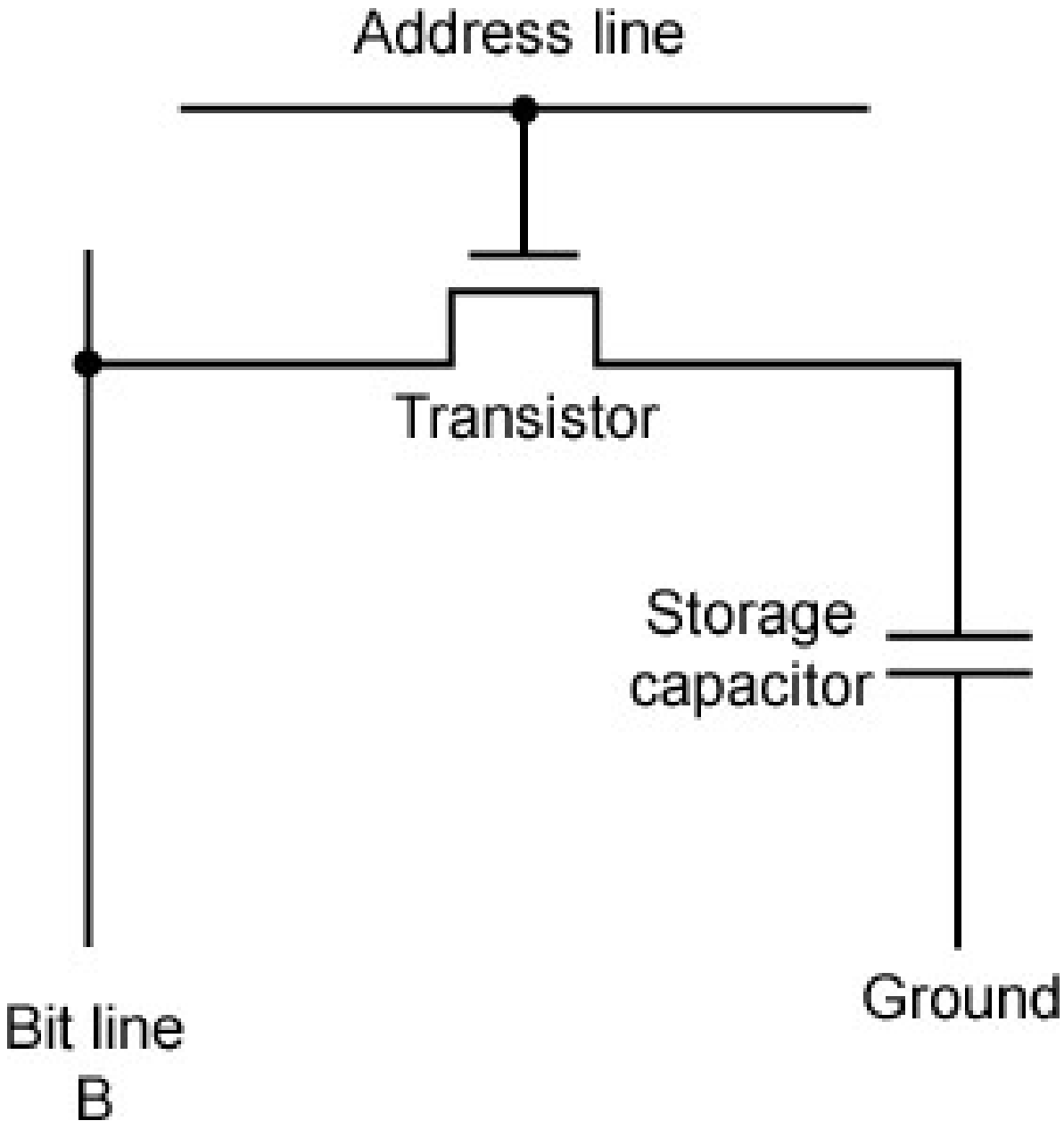
RAM- Random Access Memory

- RAM
  - —Random access- data can be quickly read and modified in any order
  - —Read/Write-it is possible both to read data from the memory and to write new data into the memory easily and rapidly
  - —Volatile-If the power is interrupted, then the data are lost.
  - —Temporary storage
  - —Static or dynamic RAM

Dynamic RAM

- Bits stored as charge in capacitors
- absence of charge in a capacitor is interpreted as a **binary 1 or 0.**
- **Charges leak**-capacitors have a natural tendency to discharge, dynamic RAMs require periodic charge refreshing to maintain data storage
- Need **refreshing** even when powered
- Simpler construction

Dynamic RAM Structure



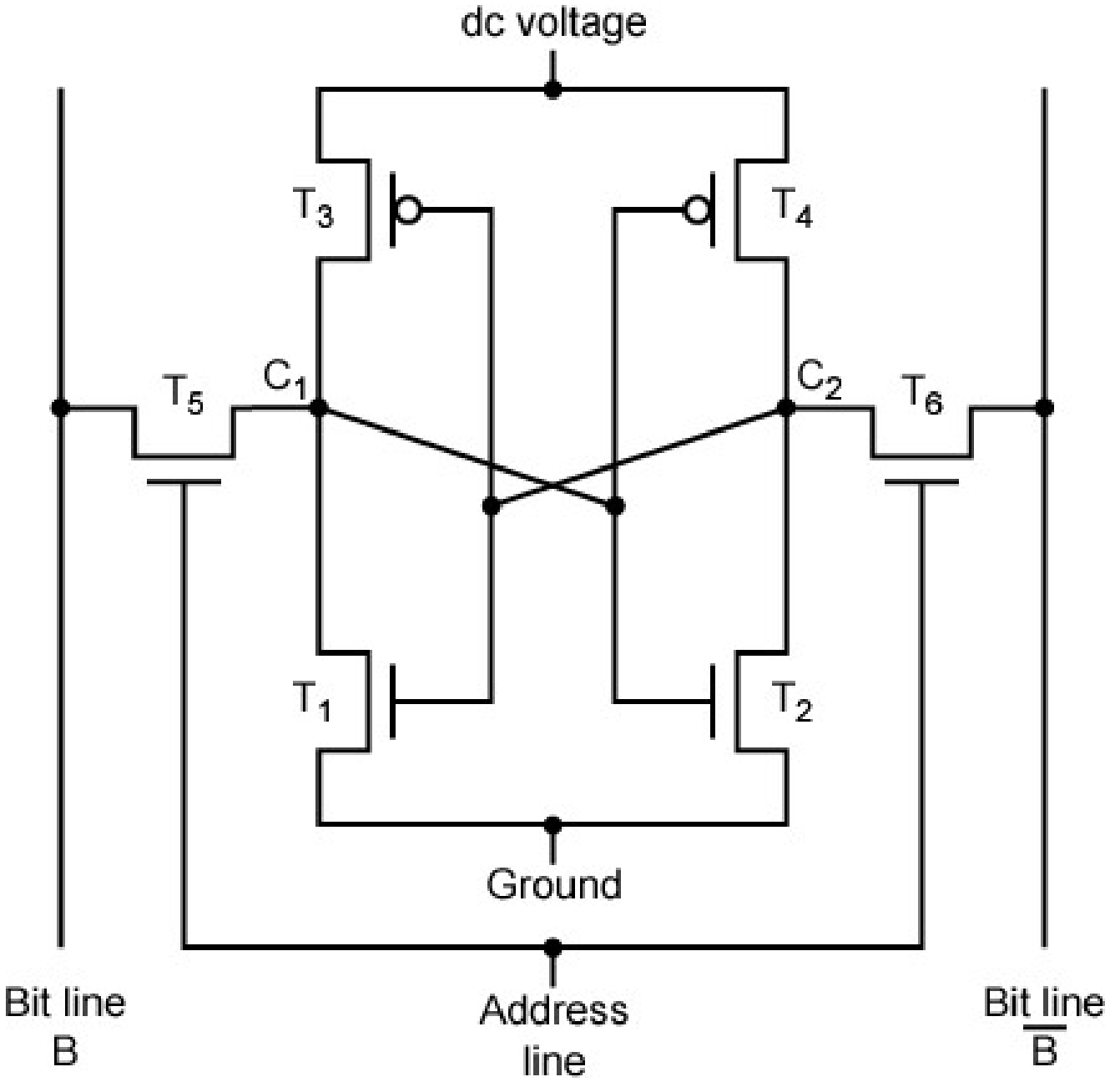
typical DRAM structure for an individual cell that stores one bit.

- The **address line is activated** when the bit value from this cell is to be **read or written.**
- The transistor acts as a switch that is closed (allowing current to flow) if a voltage is applied to the address line and open (no current flows) if no voltage is present on the address line.
  - For the **write operation, a voltage signal is applied to the bit line**; a high voltage represents 1, and a low voltage represents 0.
  - **A signal is then applied to the address line, allowing a charge to be transferred to the capacitor**
  - For the read operation, when the address line is selected, the transistor turns on and the charge stored on the capacitor is fed out onto a bit line and to a sense amplifier
  - The sense amplifier compares the capacitor voltage to a reference value and determines if the cell contains a logic 1 or a logic 0.

Dynamic RAM

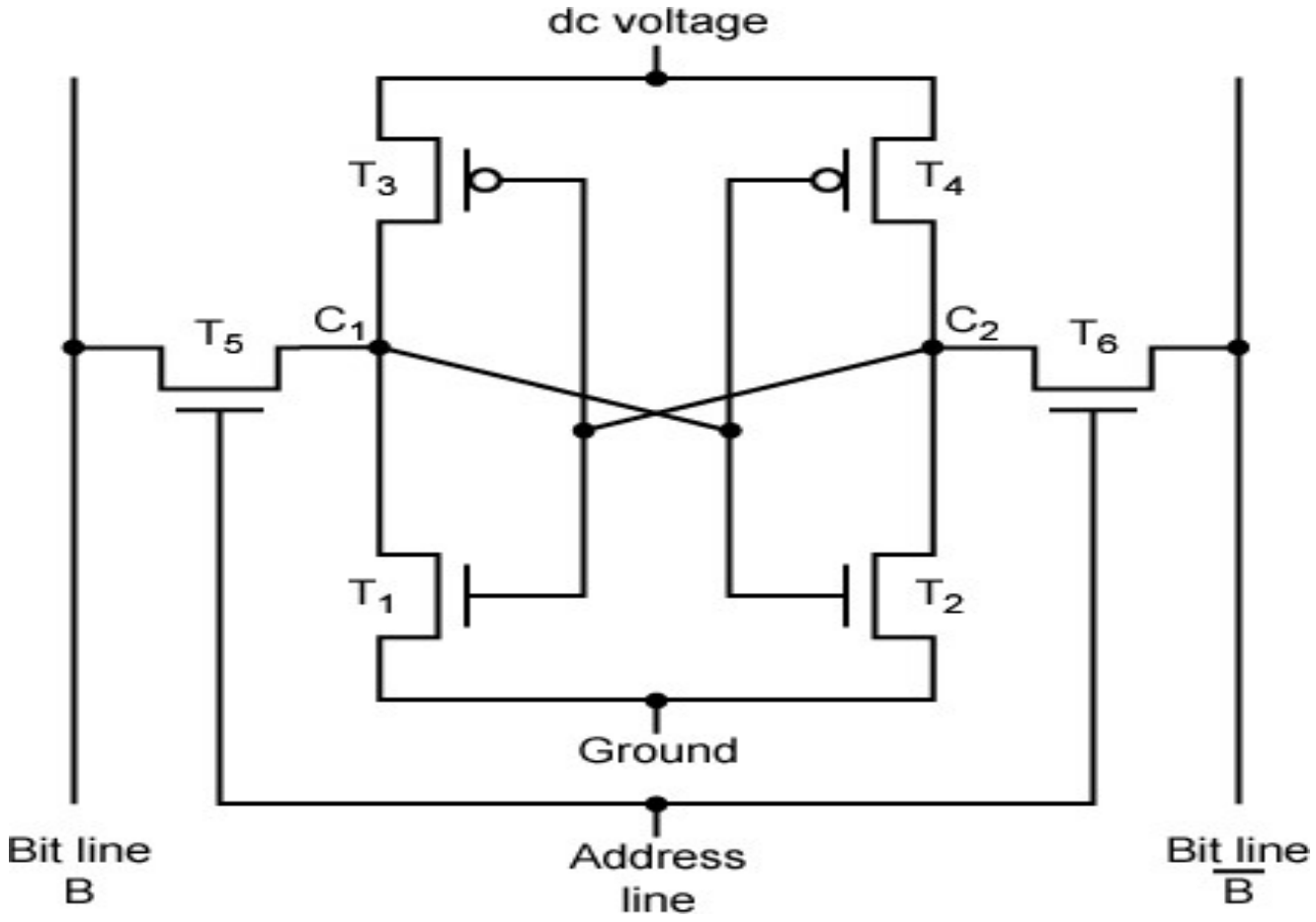
- Less expensive
- Need refresh circuits
- Slower
- Main memory
- Essentially analog
  - Level of charge determines value

Static RAM Structure



Static RAM

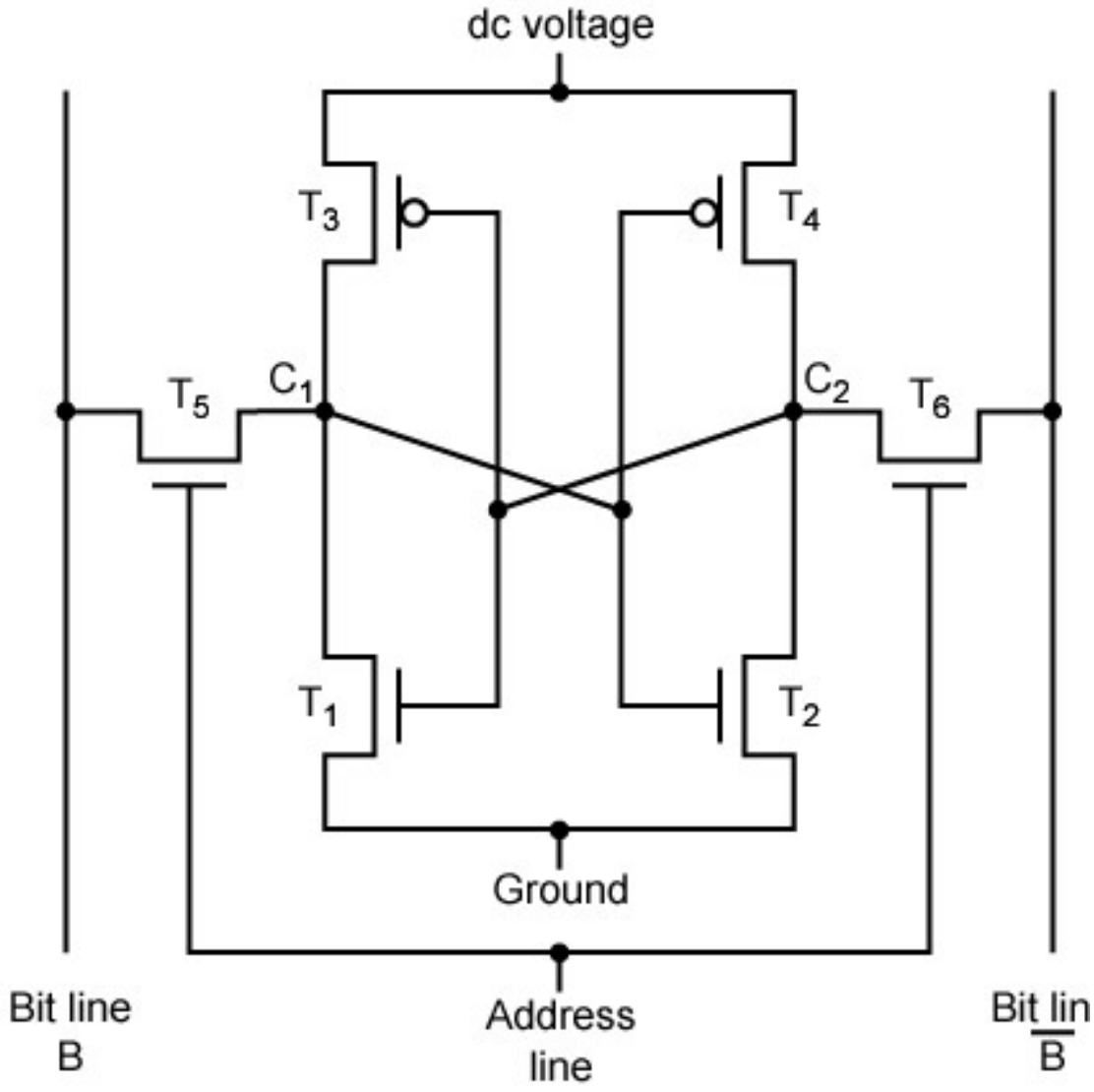
- In a SRAM, binary values are stored using traditional flip-flop logic-gate configurations
- A static RAM will hold its data as long as power is supplied to it.



- SRAM address line is used to open or close a switch.
- The address line controls two transistors (T5 and T6).
- When a signal is applied to this line, the two transistors are switched on, allowing a read or write operation.
- For a write operation, the desired bit value is applied to line B, while its complement is applied to line B.
- This forces the four transistors (T1, T2, T3, T4) into the proper state.
- For a read operation, the bit value is read from line B.

Static RAM Operation

- Transistor arrangement gives stable logic state
- State 1
  - C_1 high, C_2 low
  - T_1 T_4 off, **T_2 T_3 on**
- State 0
  - C_2 high, C_1 low
  - T_2 T_3 off, **T_1 T_4 on**



Static RAM

- Bits stored as on/off switches
- No charges to leak
- No refreshing needed when powered
- More complex construction
- Larger information per bit
- More expensive
- Does not need refresh circuits
- Faster
- Cache
- Digital
  - Uses flip-flops

SRAM v DRAM

- Both volatile
  - —Power needed to preserve data
- Dynamic cell
  - —Simpler & smaller
  - — DRAM is more dense
  - —Less expensive
  - —Needs refresh circuitry
  - —Larger memory requirements
  - -used for main memory
- Static
  - —Faster
  - —Cache memory

Read Only Memory (ROM)

- Permanent storage

  —Non volatile (no power source is required to maintain the bit values in memory)

  —Can read a ROM but cant write new data into it

  -Applications-Microprogramming, System programs

  -data or program is permanently in main memory and need never be loaded from a secondary storage device

- TYPES OF ROM

  —PROM

  —EPROM

  —EEPROM

PROM-Programmable ROM

- Written during manufacture
- Programmable (“once”)
  - —**PROM**
  - —Small amount of data to be written
  - —Less expensive
  - —Non volatile, written only once
  - —Writing performed electrically at the time of chip fabrication

    -provide flexibility and convenience.

Read “mostly”

—type of memory that can be **read fast, but written to only slowly**.
-read operations are far more frequent than write operations but for which nonvolatile storage is required

Erasable Programmable (**EPROM**)

- Erased by UV

—Electrically Erasable (**EEPROM**)

- Takes much longer to write than read

—Flash memory

- Erase whole memory electrically

EPROM

- Read and written electrically
- All storage cells should be erased electrically to **initial state** by exposure to UV radiation
- Can be erased (altered) multiple times and holds data virtually indefinitely
- More expensive than PROM, multiple update capability

EEPROM

- Can be written anytime without erasing prior contents
- Write operation takes longer than read
- combines the advantage of nonvolatility with the flexibility of being updatable in place, using ordinary bus control, address, and data lines
- More expensive than EPROM, less dense

Types of ROM

1. Programmable Read Only Memory (PROM)

- Empty of data when manufactured
- May be permanently programmed by the user

2. Erasable Programmable Read Only Memory (EPROM)

- Can be programmed, erased and reprogrammed
- The EPROM chip has a small window on top allowing it to be erased by shining ultra-violet light on it
- After reprogramming the window is covered to prevent new contents being erased
- Access time is around 45 – 90 nanoseconds

Types of ROM

3. Electrically Erasable Programmable Read Only Memory (EEPROM)

- Reprogrammed electrically **without** using ultraviolet light
- Must be removed from the computer and placed in a special machine to do this
- Access times between 45 and 200 nanoseconds

4. Flash ROM

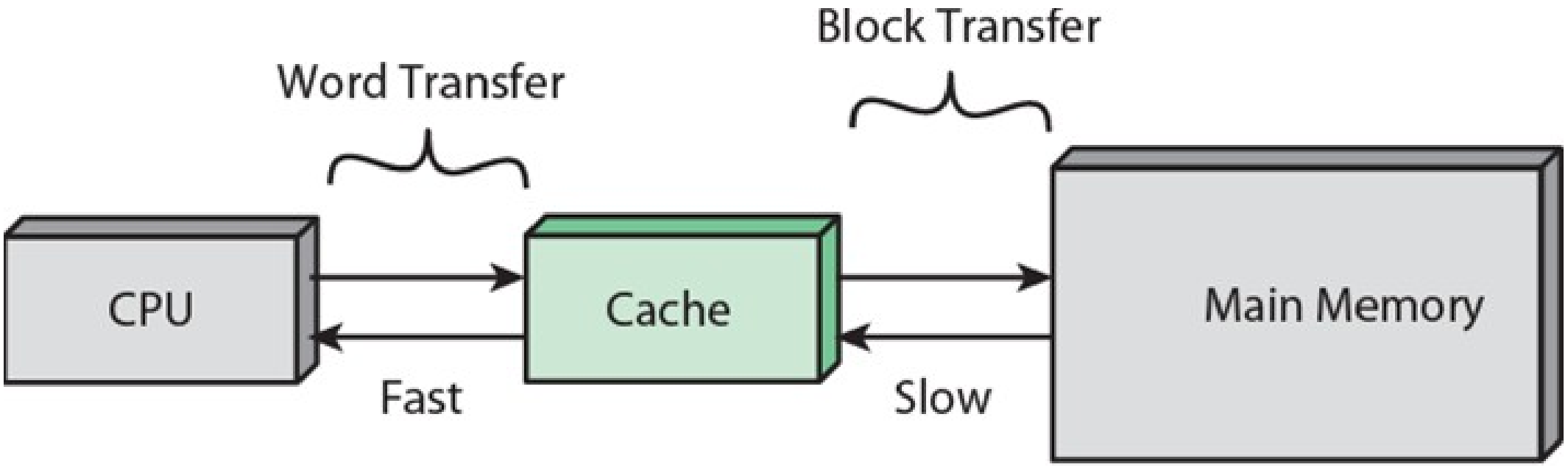
- Similar to EEPROM
- intermediate between EPROM and EEPROM in both cost and functionality.
- uses an electrical erasing technology
- However, can be reprogrammed while still in the computer
- Easier to upgrade programs stored in Flash ROM
- Used to store programs in devices e.g. modems
- Access time is around 45 – 90 nanoseconds
- flash memory uses only one transistor per bit, and so achieves the high density

5. ROM cartridges

- ROM cartridge is a read-only memory chip housed in a plastic casing, with an opening at one end exposing the chip interface.
- The cartridge can be inserted into a computing device, such as a video game console, that loads software or data from the chip.
- Commonly used in games machines
- Prevents software from being easily copied

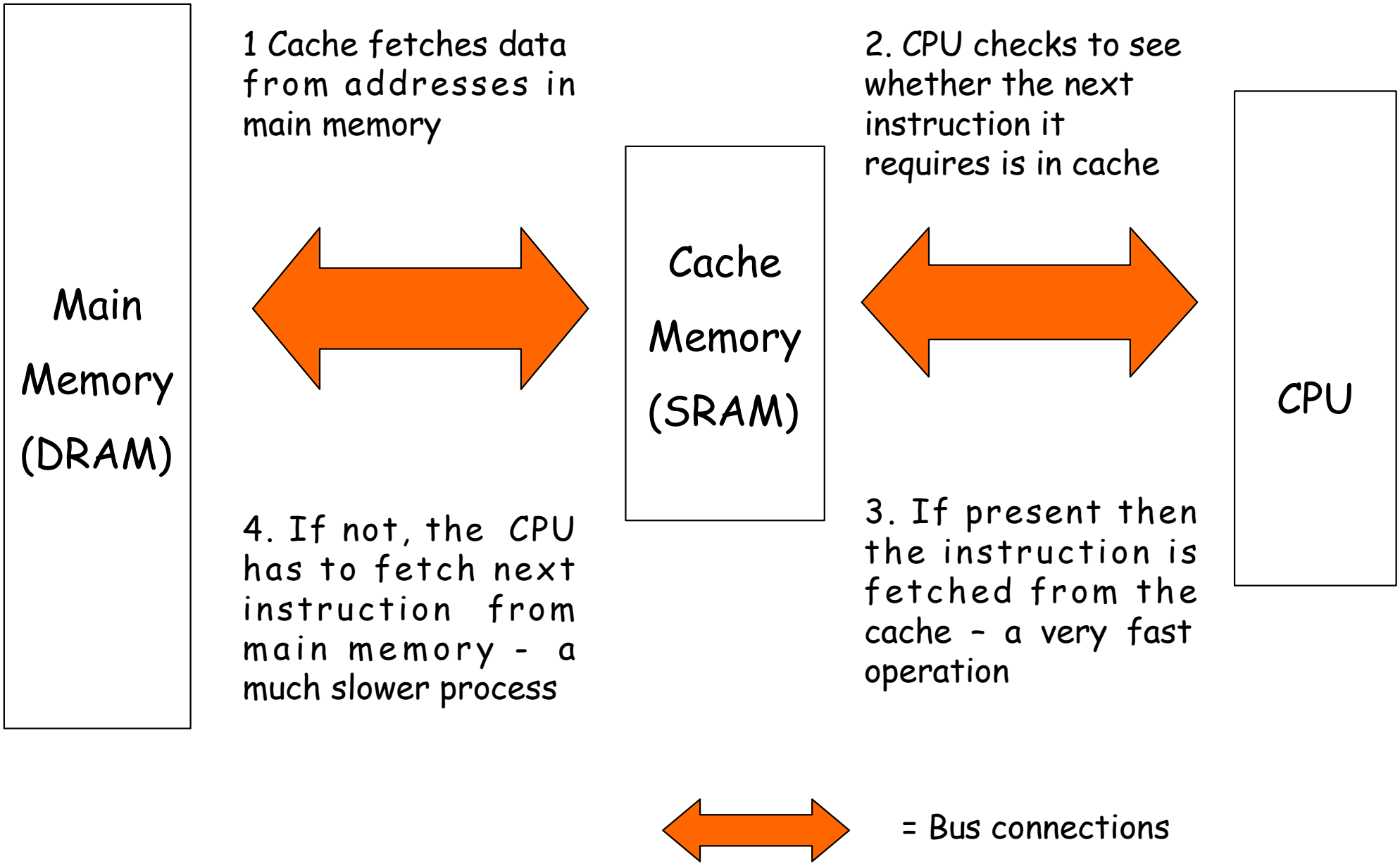
Cache

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



- The cache contains a copy of portions of main memory.
- When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache.
- If so, the word is delivered to the processor.
- If not, a block of main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor.
- Locality of reference, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that there will be future references to that same memory location or to other words in the block.

The operation of cache memory



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

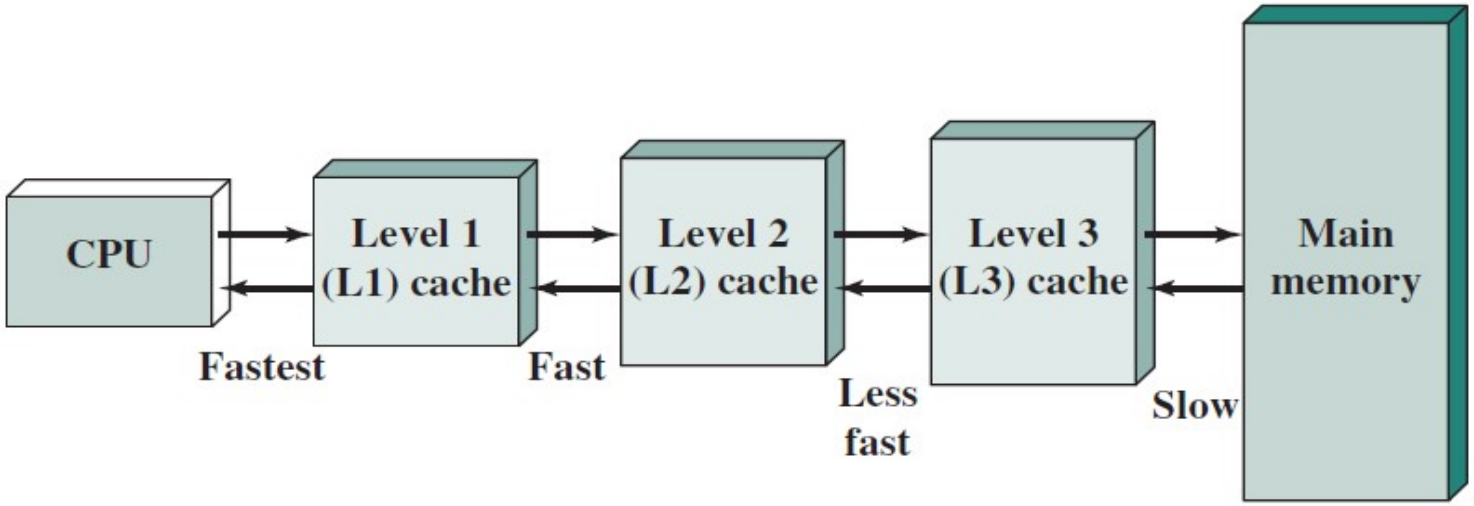
- CPU cache is divided into three main 'Levels', L1, L2, and L3.
  - The hierarchy is according to the speed, and thus, the size of the cache.

➢ **Level 1 (L1) cache** -fast small, **embedded** in the processor chip (CPU).

➢ **Level 2 (L2) cache** more capacity than L1; l**ocated** on the CPU or on a separate chip or coprocessor.

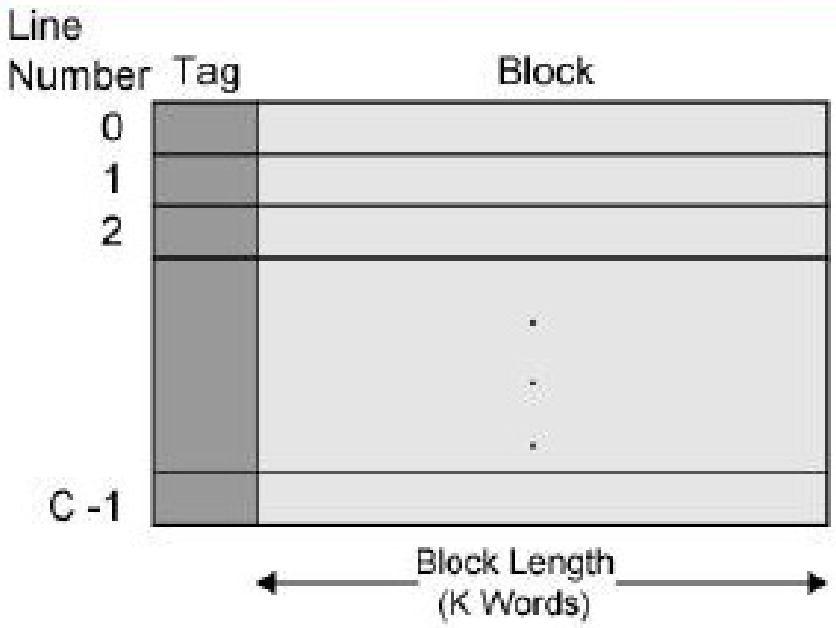
➢ **Level 3 (L3) cache** specialized memory to improve the performance of L1 and L2.

  ➢ Slower than L1 or L2, double the speed of RAM

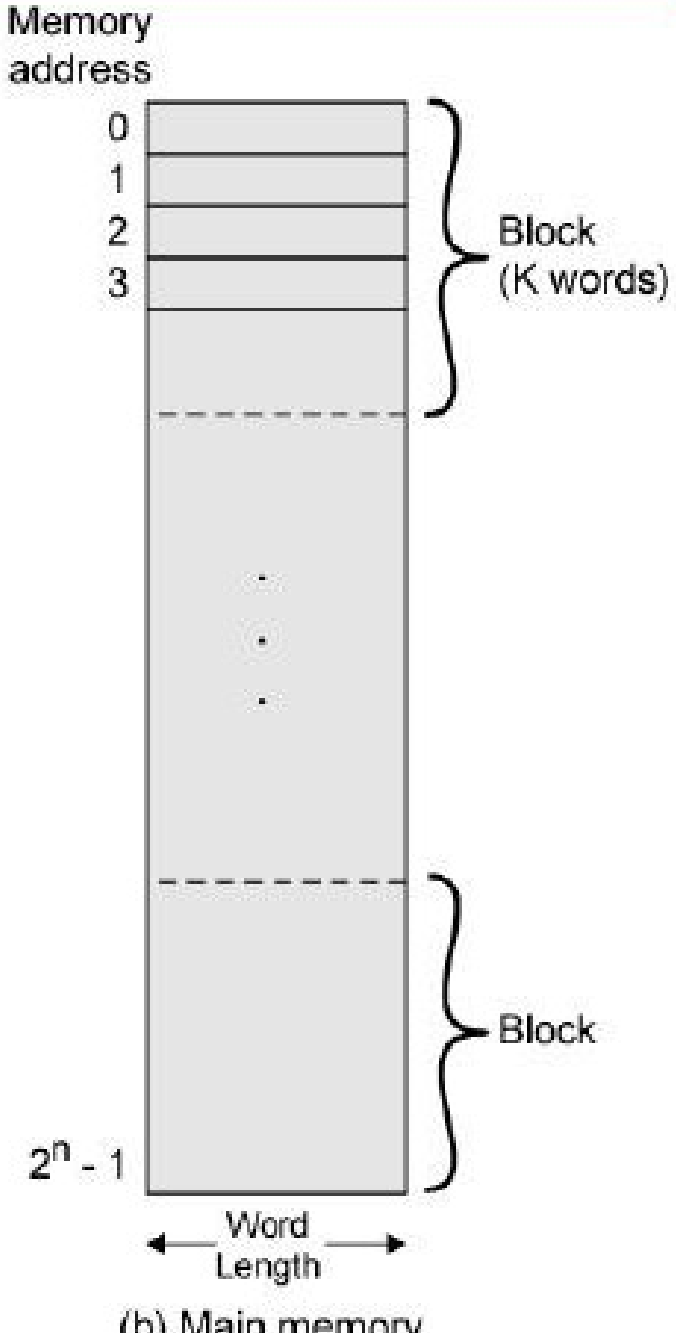


(b) Three-level cache organization

Cache/Main Memory Structure



(a) Cache



(b) Main memory

TAG - A unique identifier for a group of data. Since different regions of memory may be mapped into a block, the tag is used to differentiate between them.

- Main memory consists of up to 2*n* addressable words, with each word having a unique *n*-bit address.
- For mapping purposes, this memory is considered to consist of **a number of fixed-length blocks of *K* words each**.
- That is, there are *M* = 2^*n*/*K* blocks in main memory.

- The cache consists of *m* blocks, called **lines**.
- Each line contains *K* words plus a tag of a few bits
- The number of lines is considerably less than the number of main memory blocks (m <<M).
- each line includes a tag that identifies which particular block is currently being stored.

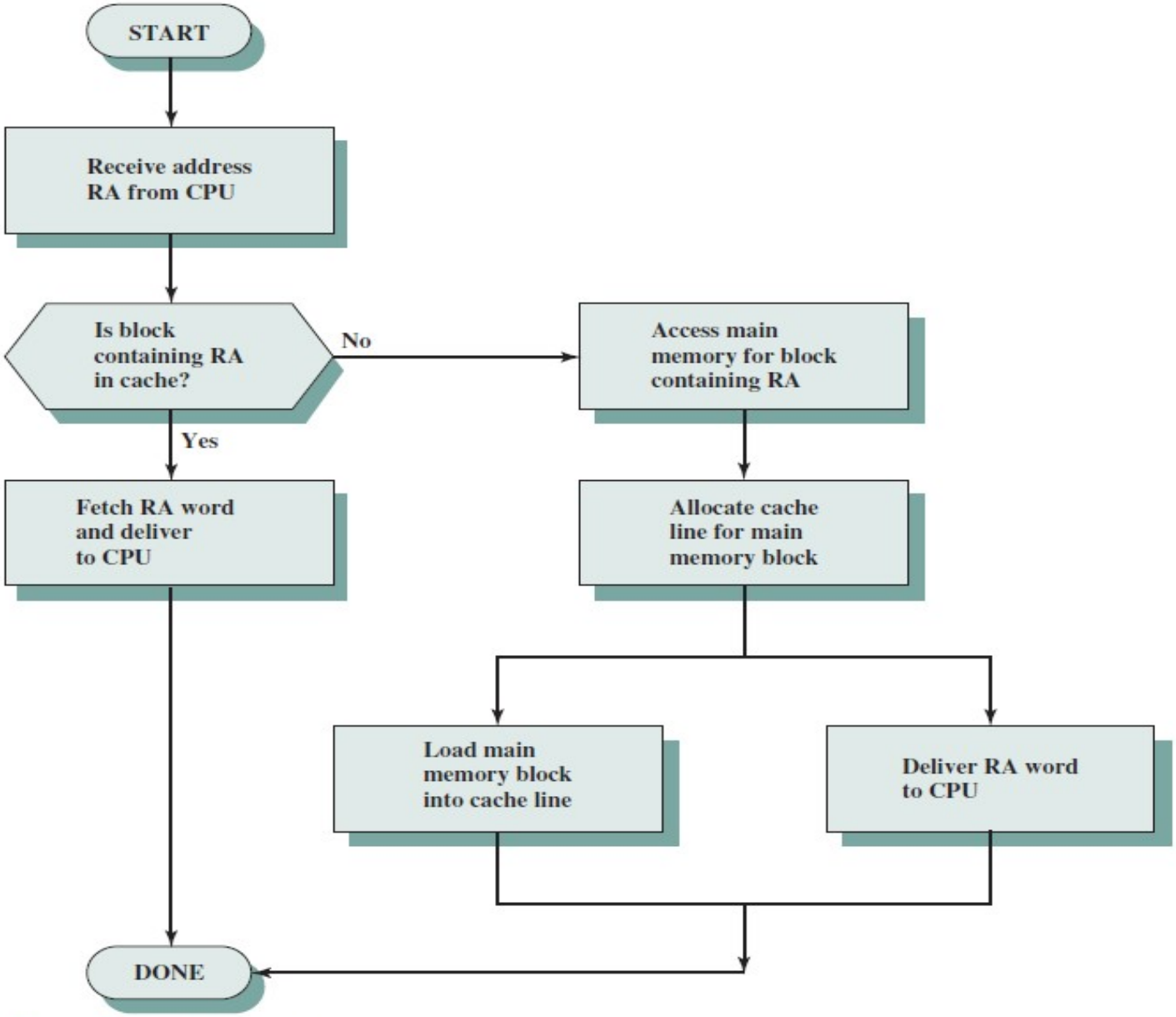


Figure 4.5 Cache Read Operation

- The processor generates the read address (RA) of a word to be read. If the word is contained in the cache, it is delivered to the processor.
- Otherwise, the block containing that word is loaded into the cache, and the word is delivered to the processor.

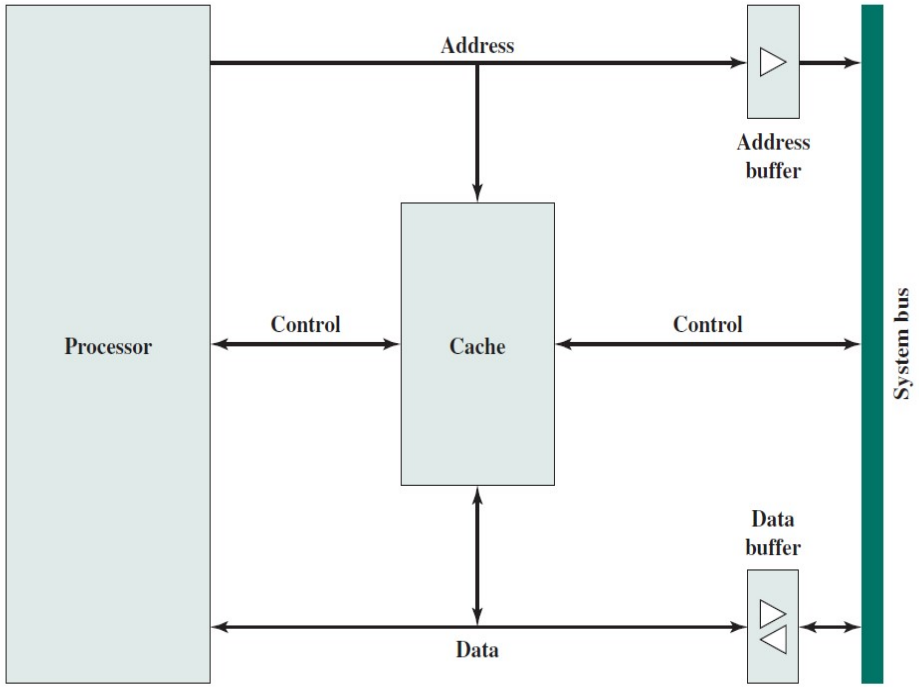


Figure 4.6 Typical Cache Organization

- the cacheconnects to the processor via data, control, and address lines.
- When a cache hit occurs, the data and address buffers are disabled and communication is only between processor and cache, with no system bus traffic.
- When a cache miss occurs, the desired address is loaded onto the system bus and the data are returned through the data buffer to both the cache and the processor.

Elements of Cache Design

- Cache address & Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Cache Address

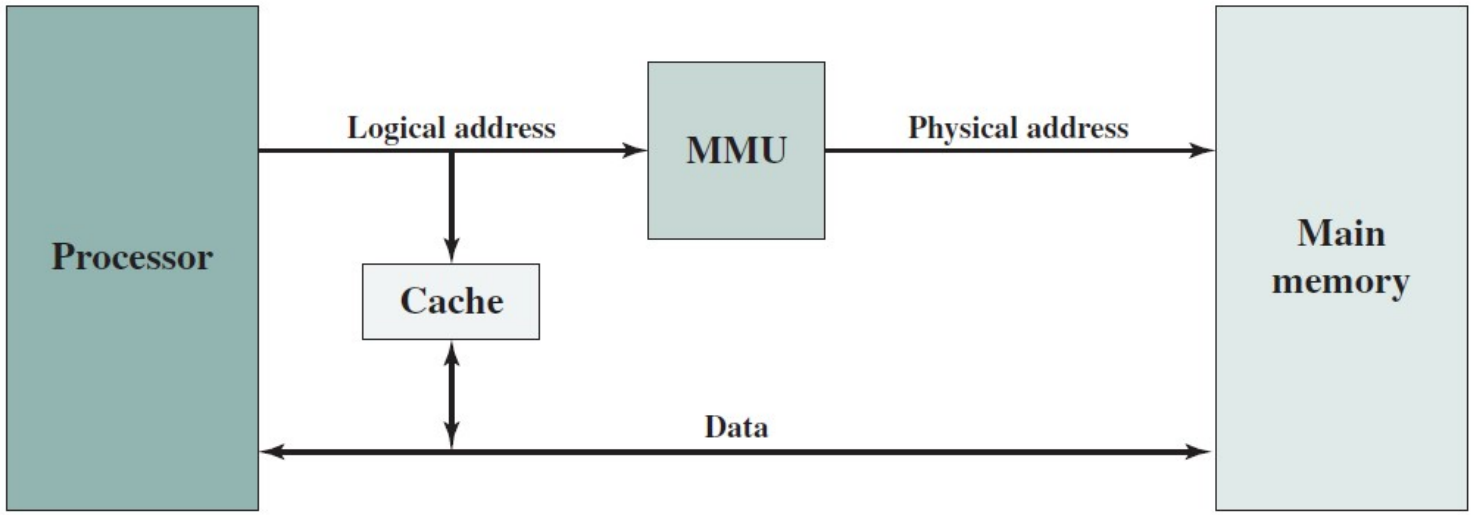
Table 4.2 Elements of Cache Design

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	

- Cache addresses
- virtual memory is a facility

that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available

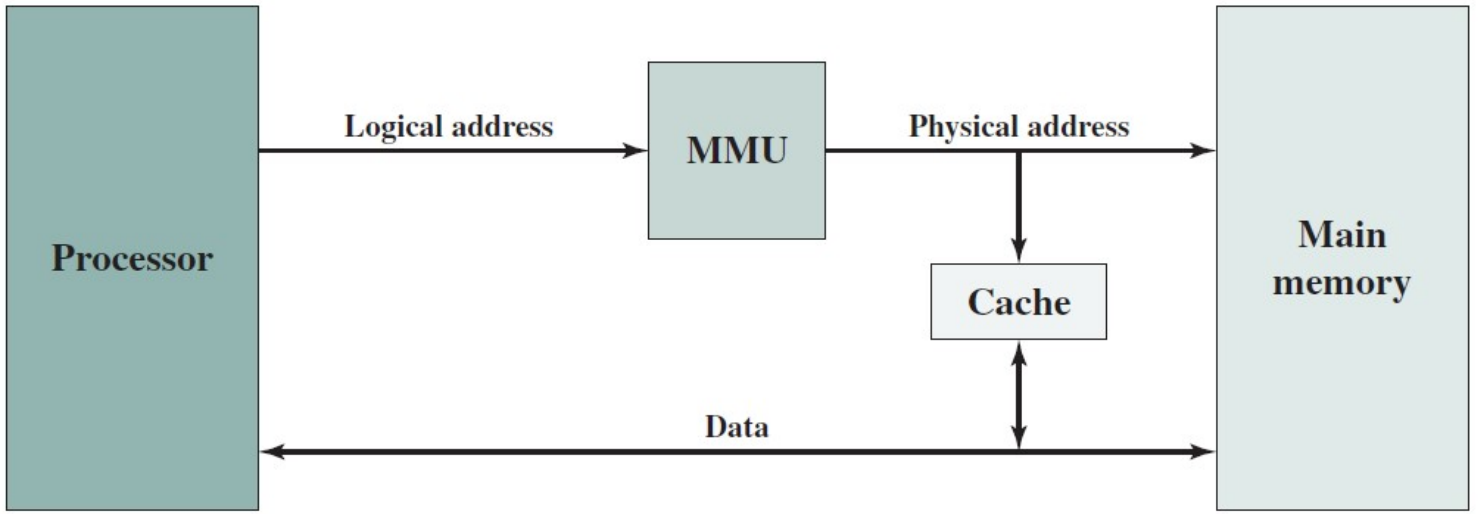
- a hardware memory management unit (MMU) translates each virtual address into a physical address in main memory



(a) Logical cache

A **logical cache**, also known as a **virtual cache**, stores data using **virtual addresses**.

The processor accesses the cache directly, without going through the MMU.



(b) Physical cache

- A **physical cache** stores data using main memory **physical addresses**.
- Advantage & disadvantage-refer text

Size does matter

- Cost

  —More cache is expensive

- Speed

  —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—

  More cache is faster (up to a point)

  —Checking cache for data takes time

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.
- 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, hit ratio will begin to decrease**
- probability of using the newly fetched information becomes **less** than the probability of reusing the information that has to be replaced.
- Larger blocks reduce the number of blocks that fit into a cache
- As a block becomes larger, each additional word is farther from the requestedword

Number of caches

- Single cache vs multi-level caches
- two-level cache, with the internal level 1 (L1) and the external cache designated as level 2 (L2).
  - If there is no L2 cache and the processor makes an access request for a memory location not in the L1 cache, then the processor must access DRAM or ROM memory across the bus.
  - Due to the typically slow bus speed and slow memory access time-poor performance

- L1,L2 & L3 caches

- Split the cache into two: one dedicated to instructions and one dedicated to data. 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

- unified cache has a higher hit rate than split caches
- Only one cache needs to be designed and implemented

MAPPING TECHNIQUES

- An algorithm is needed for mapping main memory blocks into cache lines because **there are fewer cache lines than main memory blocks**
- Technique using which we bring the main memory into the cache memory.
- Determine which main memory block currently occupies a cache line.

MAPPING TECHNIQUES

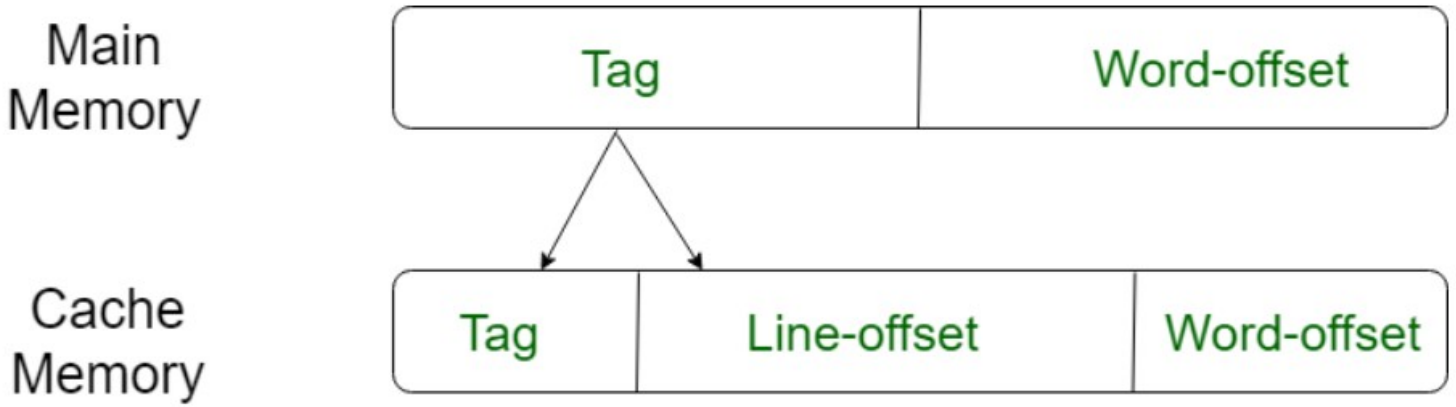
- **DIRECT MAPPING**
- ASSOCIATIVE MAPPING
  - —FULLY ASSOCIATIVE MAPPING
  - —SET ASSOCIATIVE MAPPING
    - **– 2-WAY SET ASSOCIATIVE MAPPING**

Direct Mapping

- Assign each memory block to a **specific line in the cache**.
- If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed.
- An address space is split into two parts **index field and a tag field.**
- The cache is used to store the tag field whereas the rest is stored in the main memory.
- Direct mapping`s performance is directly proportional to the Hit ratio.

- Direct mapping is the most efficient cache mapping scheme, but it is also the least effective in its utilization of the cache - that is, it may leave some cache lines unused.

```
i = j modulo m
where
i = cache line number
j = main memory block number
m = number of lines in the cache
```



Direct Mapping

Working of Direct Mapping

- **Word**" field selects one from among the 16 addressable words in a line.
- The "**Line**" field defines the cache line where this memory line should reside.
- The "**Tag**" field of the address is then compared with that cache line's 5-bit tag to determine whether there is a hit or a miss.
  - If there's a miss, we need to swap out the memory line that occupies that position in the cache and replace it with the desired memory line.

DIRECT MAPPING

- Cache- 128 blocks of 16 words each
  - —128 * 16 = 2048 **approx 2 KB**

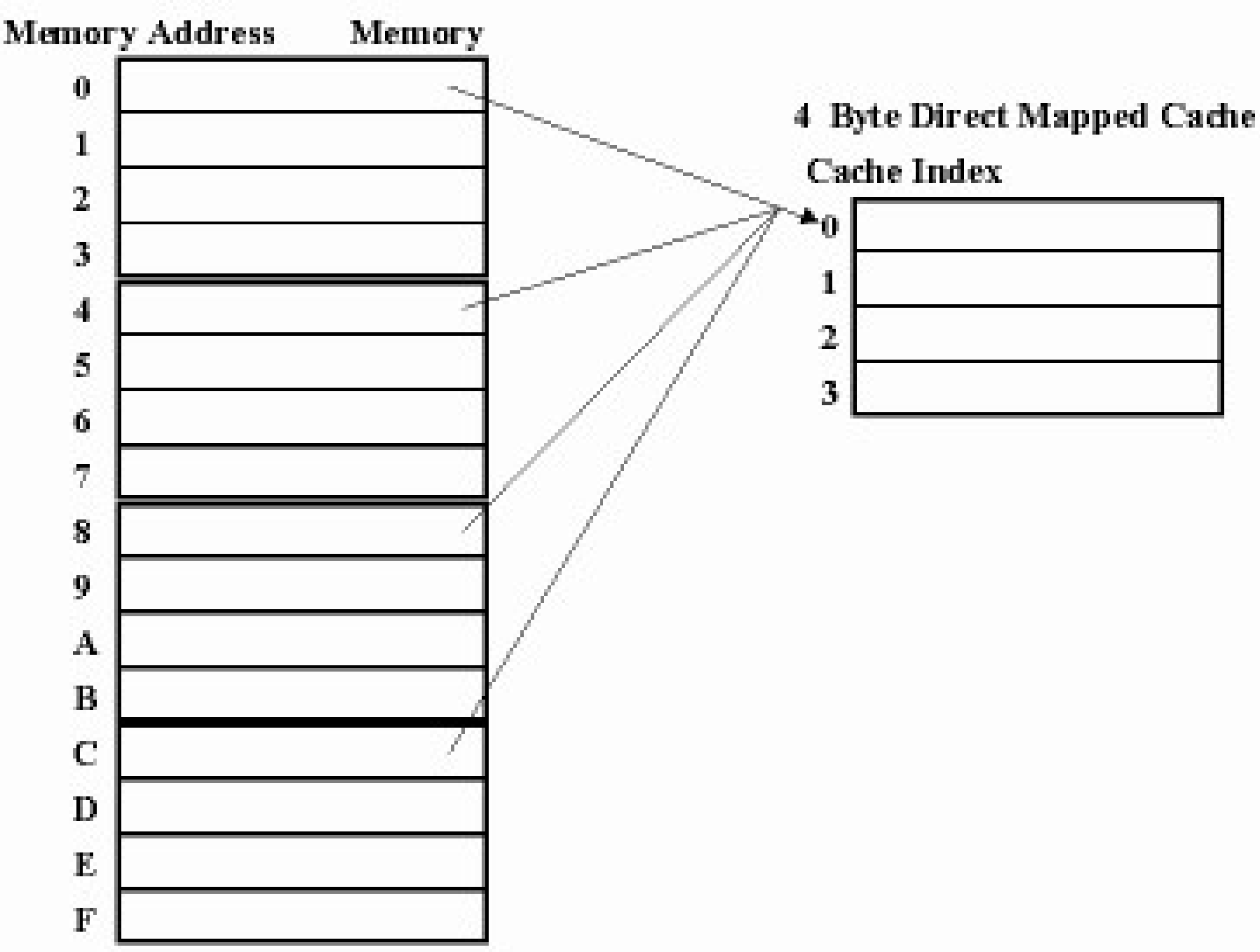
0	16 words
1	16 words
…	16 words
127	16 words

- Main Memory – 4K blocks of 16 words each
  - —4K * 16 = 64000 **approx 64 KB**

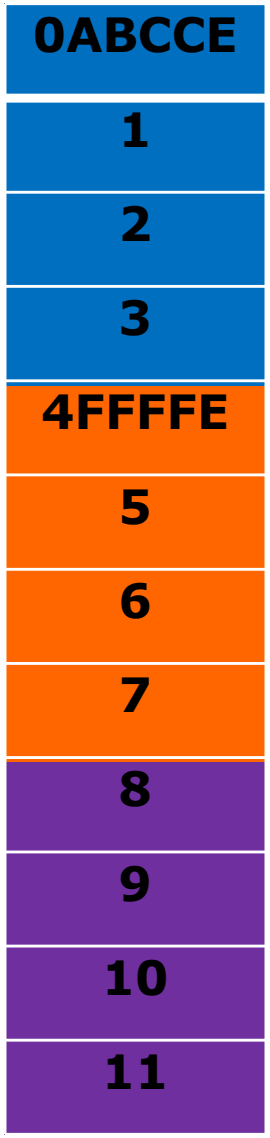
- **TOTAL ADDRESS SIZE 16 bit**

Tag	Block/Line(128)	Word(16)
5(16-(7+4))	**7(2^7)**	**4 (2^4)**

DIRECT MAPPING CONCEPT



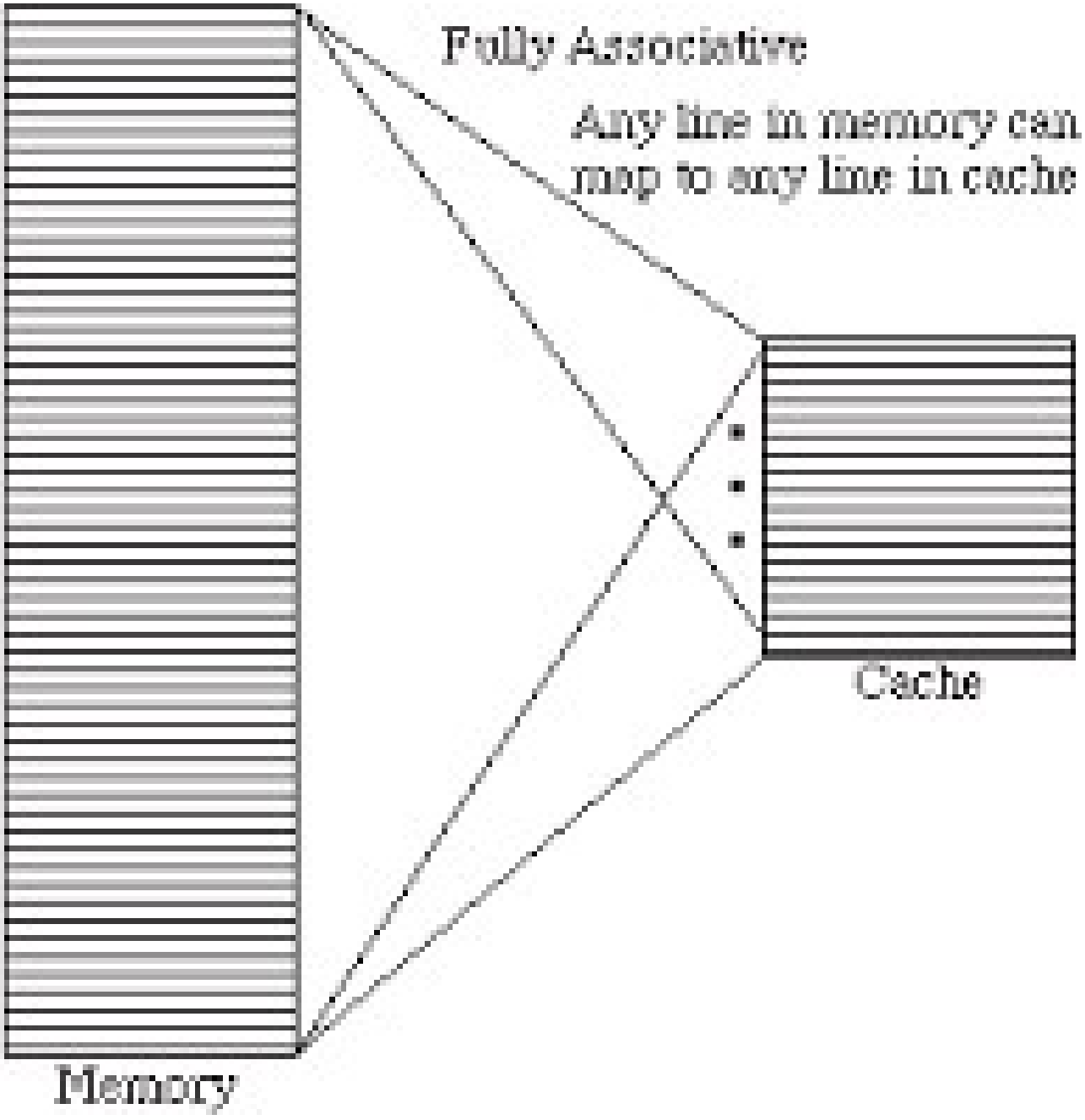
Main Memory



Cache

0 / 4 / 8 [4FFFFEE]
1 /5 / 9 [9]
2 /6 / 10 [2]
3 /7/ 11 [7]

FULLY ASSOCIATIVE MAPPING



- store the content and addresses of the memory word.
- Any block can go into any line of the cache
- placement of any word at any place in the cache memory.
- It is considered to be the fastest and most flexible mapping form.

ASSOCIATIVE MAPPING

- Cache- 128 blocks of 16 words each
- 128 * 16 = 2048 **approx 2 KB**

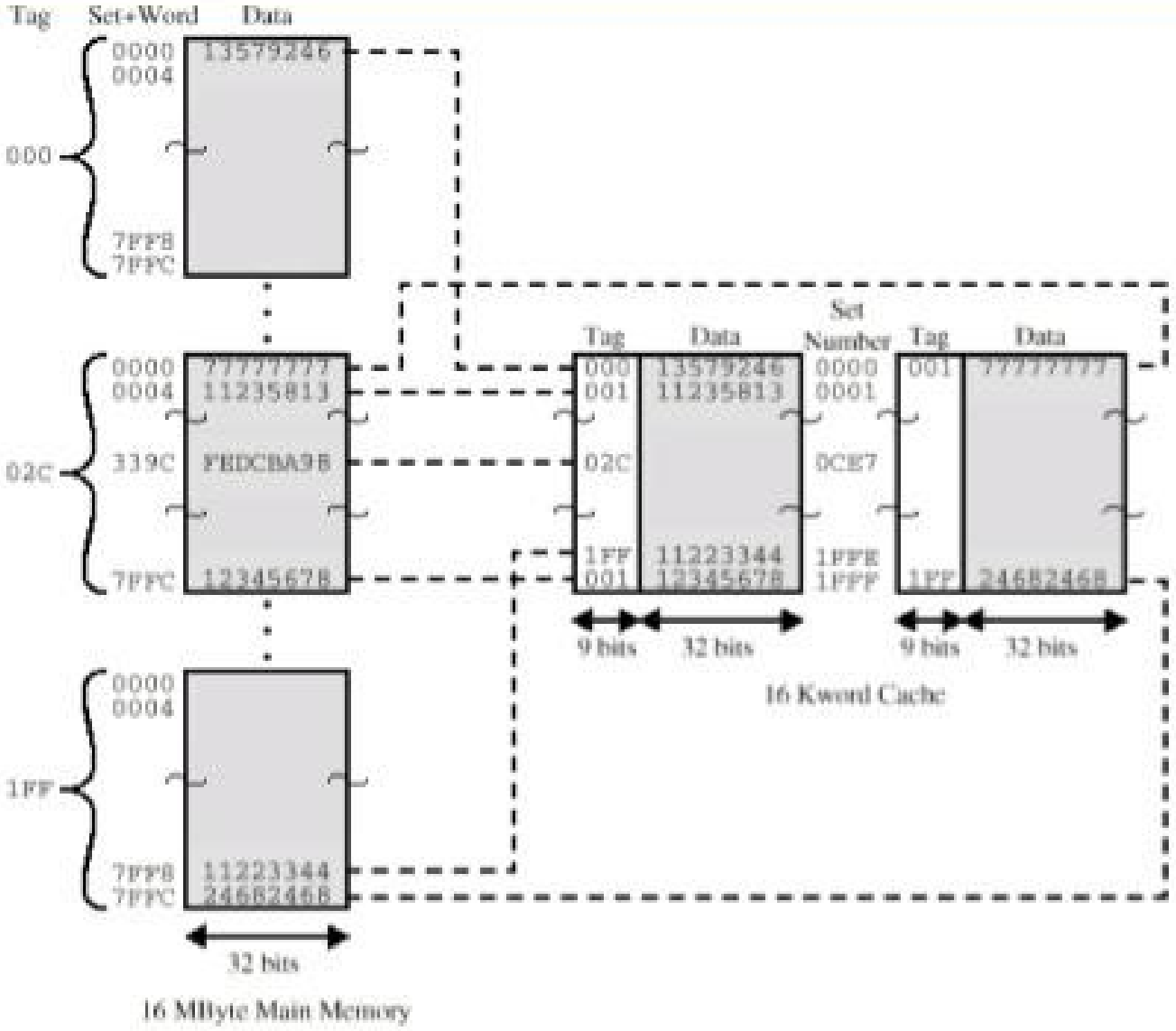
- Main Memory – 4K blocks of 16 words each
- 4K * 16 = 64000 **approx 64 KB**
- **TOTAL ADDRESS SIZE 16 bit**

Tag	Word
12(16-4)	**4 (2^4)**

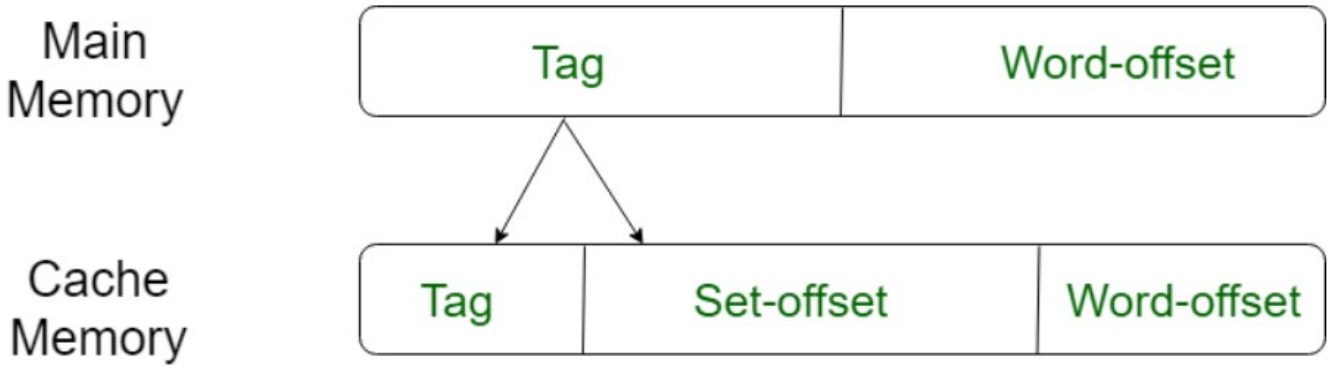
Working of Fully Associative Mapping

- "Tag" field identifies one of the 2^{12} = 4096 memory lines;
- All the cache tags are searched to find out whether or not the Tag field matches one of the cache tags.
- If so, we have a hit, and if not there's a miss and we need to replace one of the cache lines by this line before reading or writing into the cache.
- (The "Word" field again selects one from among 16 addressable words (bytes) within the line.)

Two Way Set Associative Mapping Example



- Cache miss & time of searching-more in previous
- instead of having exactly one line that a block can map to in the cache, we will group a few lines together creating a ***set***.
- Then a block in memory can map to any one of the lines of a specific set.
- the cache lines get grouped into various sets where all the individual sets consist of k number of lines.
- Here, a certain main memory block can map to only a particular cache set.
- However, within this very set, the memory block can be mapped to a freely available line of cache.



Set-Associative Mapping

SET ASSOCIATIVE MAPPING(2-way)

- Cache- 128 blocks of 16 words each
- 128 * 16 = 2048 **approx 2 KB**
- Set divided into 2 (128 /2) =64
- Main Memory – 4K blocks of 16 words each
- 4K * 16 = 64000 **approx 64 KB**

Tag	Set	Word
6(16-(4+6))	6(2^6)	4 (2^4)

Working

- "Tag" field identifies one of the $2^6 = 64$ different memory lines in each of the $2^6 = 64$ different "Set" values.
- Since each cache set has room for only two lines at a time, the search for a match is limited to those two lines (rather than the entire cache).
- If there's a match, we have a hit and the read or write can proceed immediately.
- Otherwise, there's a miss and we need to replace one of the two cache lines by this line before reading or writing into the cache. (The "Word" field again select one from among 16 addressable words inside the line.)
- In set-associative mapping, when the number of lines per set is *n*, the mapping is called *n*-way associative. For instance, the above example is 2-way associative.

Relationships in the Set-Associative Mapping can be defined as:

```
m = v * k
i= j mod v

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

Direct Mapping
Cache Line Table

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

Direct Mapping pros & cons

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

Associative Mapping

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line’s tag is examined for a match
- Cache searching gets expensive

Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

Refer problems from mapping
Refer merits & demerits of each mapping schemes

Problem statement

Consider a cache consisting of 256 blocks of 16 words each for a total of 4096(4KB) words and assume that the main memory is addressable by a 16 bit address and it consists of 4KB blocks of 16 words.

How many bits are there in each of the TAG,BLOCK/SET and WORD field for Direct Mapping , Fully Associative and 2-way set associative techniques?

Problem statement

A block set associative cache memory consists of 128 blocks divided into four block sets. The main memory consists of 16,384 blocks and each block contains 256 words.

i) How many bits are required for addressing the main memory?

ii) How many bits are needed to represent the TAG, SET and WORD fields?

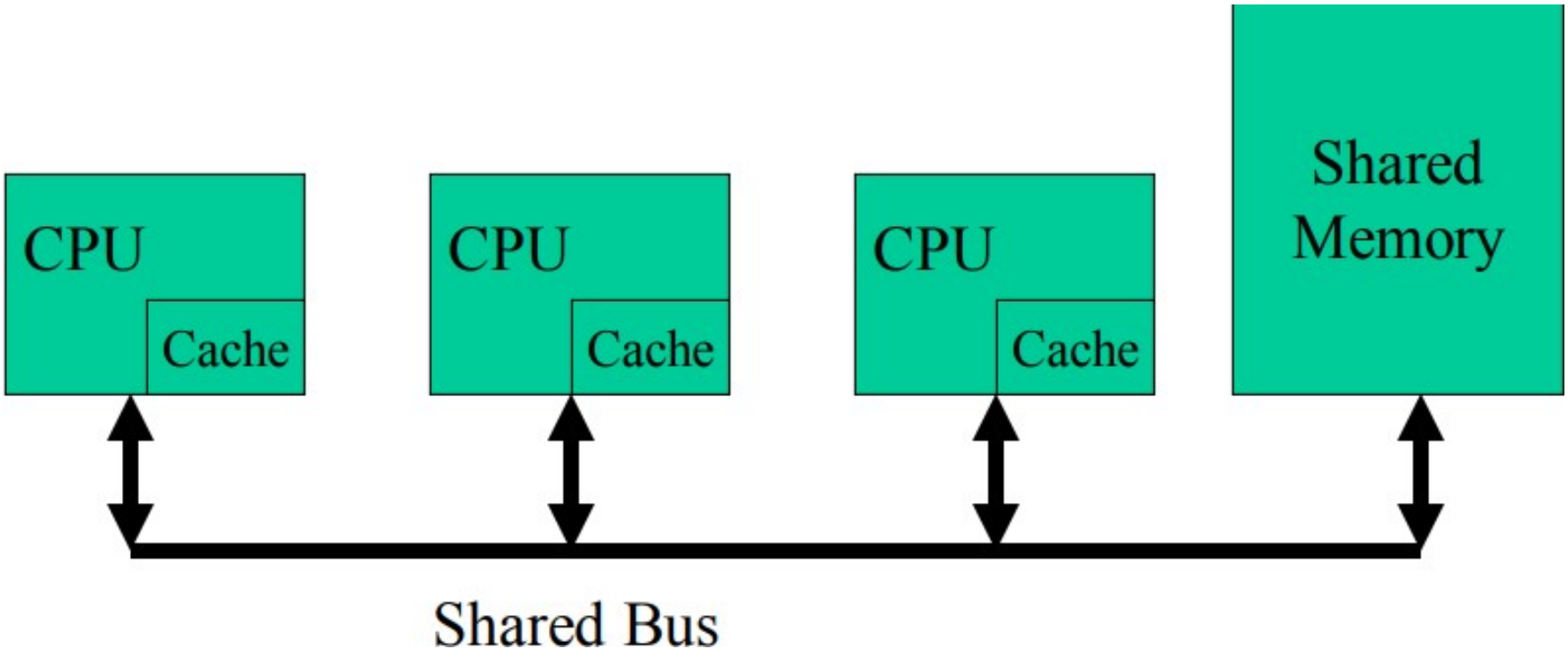
Problem statement

A block set associative cache memory consists of 64 blocks divided into four block sets. The main memory consists of 4096 blocks and each block contains 128 words.

i) How many bits are there in main memory?

ii) How many bits are needed to represent the TAG, SET and WORD fields?

CACHE COHERENCE



- In a multiprocessor system, **data inconsistency** may occur among adjacent levels or within the same level of the memory hierarchy.
- it is possible to have **many copies of any one instruction operand: one copy in the main memory and one in each cache memory**.
- When one copy of an operand is changed, the other copies of the operand must be changed also.

- 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**.

Cache Coherence

- Problem - multiple copies of same data in different caches
- Can result in an **inconsistent** view of memory
  - —Write through
  - —Write back policy
  - —Write invalidate
  - —Write Update

Write through

- Every memory write operation updates the main memory.
- If the word is present in the cache memory at the requested address, the cache memory is also updated simultaneously with the main memory.

Write back policy

- Write operations are usually made only to the cache.
- Main memory is only updated when the corresponding cache line is evicted from the cache.
- Write invalidate
- the immediate sending of the updated cache block to the other cache is not done.
- Simply, an invalidate command is sent to all the other cache copies & also to the original version in the shared memory so that all of their copies of data become invalid. It is denoted by 'I' and I specifies invalid / dirty data.
- Write Update
- if a processor updates its cache data, it **immediately updates all the other cached copies also.**
- The broadcast mechanism is used to send the new data block to all the cache having the copies.

Cache coherence approaches-Software Solutions

- **Compiler and operating system** deal with problem
- Overhead of detecting potential problems transferred to compile time
- Complier marks data likely to be changed and OS prevents such data from being cached
- Design complexity transferred from hardware to software

Cache coherence approaches-Hardware Solution

- **Cache coherence protocols**
- Dynamic recognition at run time of potential problems
- More **efficient** use of cache-improved performance
- **Transparent** to programmer & compiler-reducing the software development burden.
- **Directory protocols & Snoopy protocols**(to maintain cache consistency)

Directory Protocols

- Directory protocols **collect and maintain information about where copies of lines reside.**
- there is a centralized controller that is part of the main memory controller, and a directory that is stored in main memory.
- The directory contains global state information about the contents of the various local caches.
- When an individual cache controller (The cache controller is hardware that copies code or data from main memory to cache memory automatically) makes a request, the centralized controller checks and issues necessary commands for data transfer between memory and caches or between caches.
- Before a processor can write to a local copy of a line, it must request exclusive access to the line from the controller.
- Before granting this exclusive access, the controller sends a message to all processors with a cached copy of this line, forcing each processor to invalidate its copy.
- After receiving acknowledgments back from each such processor, the controller grants exclusive access to the requesting processor.

- When another processor tries to read a line that is exclusively granted to another processor, it will send a miss notification to the controller.
- The controller then issues a command to the processor holding that line that requires the processor to do a write back to main memory.

- Demerits-Refer from text

Snoopy Protocols

- Distribute cache coherence responsibility among **cache controllers**
- Cache recognizes that a line (that it holds) is shared
- Updates are announced to other caches also
- Each cache controller is able to "snoop" on the network to observe these broadcasted notifications, and react accordingly.
- **Suited to bus based multiprocessor** (shared bus provides a simple means for broadcasting and snooping)
- Increases bus traffic

Write through

- **All writes go to main memory as well as cache**
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Lots of traffic
- Slows down writes

Write back

- **Updates initially made in cache only**
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if **update bit is set**
- Other caches get out of sync
- I/O must access main memory through cache

Write Invalidate-commonly used

- **Multiple readers, one writer**
- a line may be shared among several caches for reading purposes.
- When a write is required, all other caches of the line are invalidated
- When one of the caches wants to perform a write to the line, it first issues a notice that invalidates that line in the other caches, making the line exclusive to the writing cache.
- Once the line is exclusive, the owning processor can make cheap local writes until some other processor requires the same line.
- State of every line is marked as modified, exclusive, shared or invalid
- MESI protocol

Write Update

- Multiple readers and writers
- Updated word is distributed to all other processors
- Some systems use an adaptive mixture of both solutions

MESI Protocol

Commonly implemented for Cache coherence

MESI protocol -four states that a cache line may be in:

- **Modified**
- **Exclusive**
- **Shared**
- **Invalid**

MESI protocol

- **I**nvalid: This cache line is not valid
- **E**xclusive: The line in the cache is the same as that in main memory and is not present in any other cache.
- **S**hared: The line in the cache is the same as that in main memory and may be present in another cache.
- **M**odified: The line has been modified. The **memory** copy is invalid.

Table 17.1 MESI Cache Line States

	M **Modified**	**E** **Exclusive**	**S** **Shared**	**I** **Invalid**
This cache line valid?	Yes	Yes	Yes	No
The memory copy is …	out of date	valid	valid	—
Copies exist in other caches?	No	No	Maybe	Maybe
A write to this line …	does not go to bus	does not go to bus	goes to bus and updates cache	goes directly to bus

MESI State Transition Diagram

Refer Text

Interleaved
And
Associative Memory

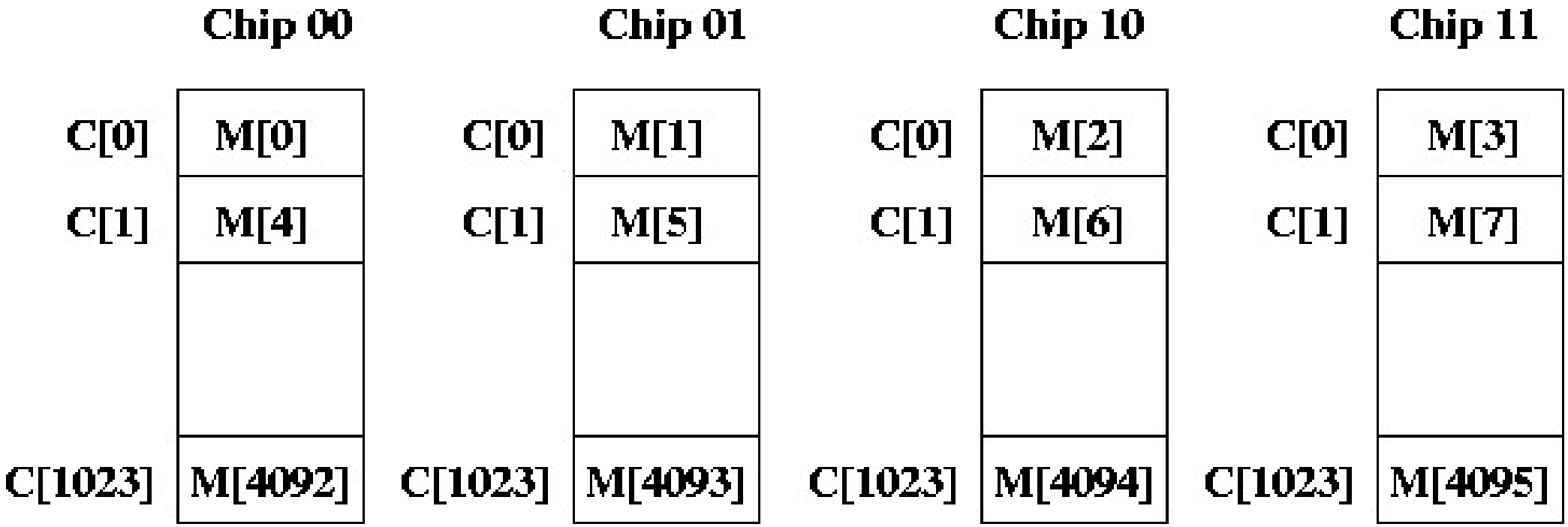
Interleaved Memory

- Main memory is composed of a collection of DRAM memory chips.
- A number of chips can be grouped together to form a ***memory bank***.
- It is possible to organize the memory banks in a way known as interleaved memory.
- Each bank is independently able to service a memory read or write request, so that a system with **K banks can service K requests simultaneously, increasing memory read or write rates by a factor of K**.
- If consecutive words of memory are stored in different banks, then the transfer of a block of memory is **speeded up**
- **Interleaved memory** is a design made to compensate for the relatively slow speed of DRAM

- **Spreads memory** addresses evenly across memory banks

- Contiguous memory reads and writes-Resulting in **higher memory throughputs** due to reduced waiting.
- Increase BW, performance
- 2 way & 4 way interleaving

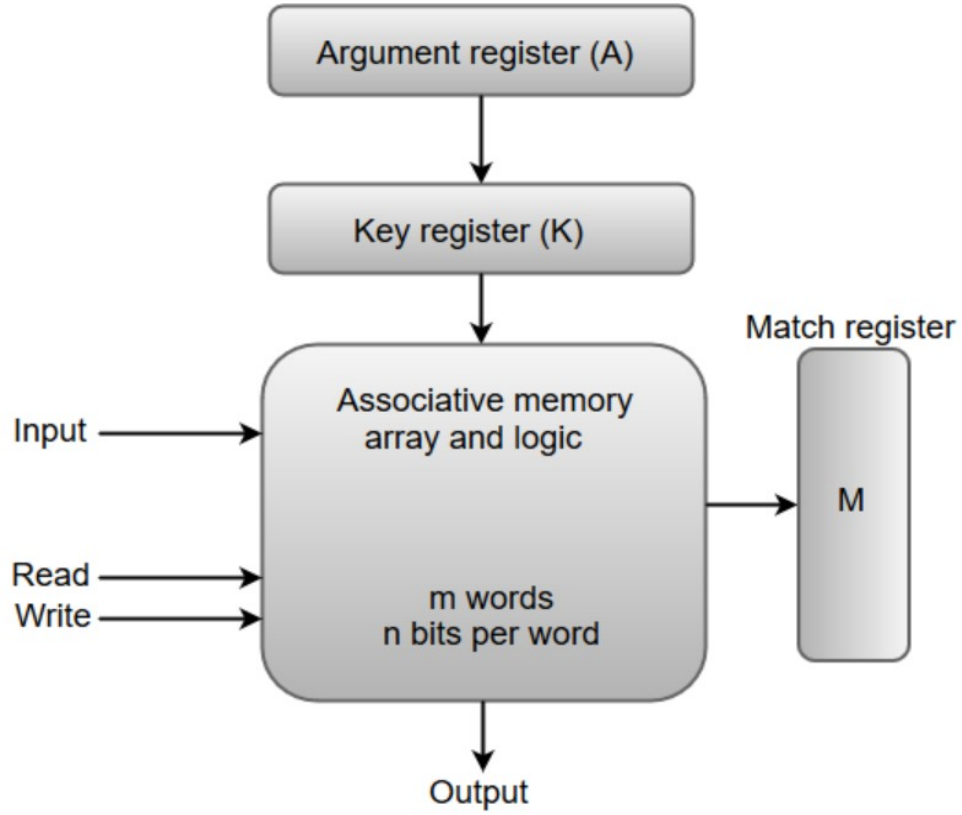
4 way interleaving



Associative Memory

- An associative memory can be considered as a memory unit whose stored data can be identified for access by the **content of the data itself rather than by an address or memory location.**
- Associative memory is often referred to as **Content Addressable Memory (CAM).**
- Content-addressed or associative memory-memory is **accessed by its content** (as opposed to an explicit address).
- When a write operation is performed on associative memory, no address or memory location is given to the word. The memory itself is capable of finding an empty unused location to store the word.
- when the word is to be read from an associative memory, the content of the word, or part of the word, is specified. The words which match the specified content are located by the memory and are marked for reading.

It takes very less time to find a particular word based on the content

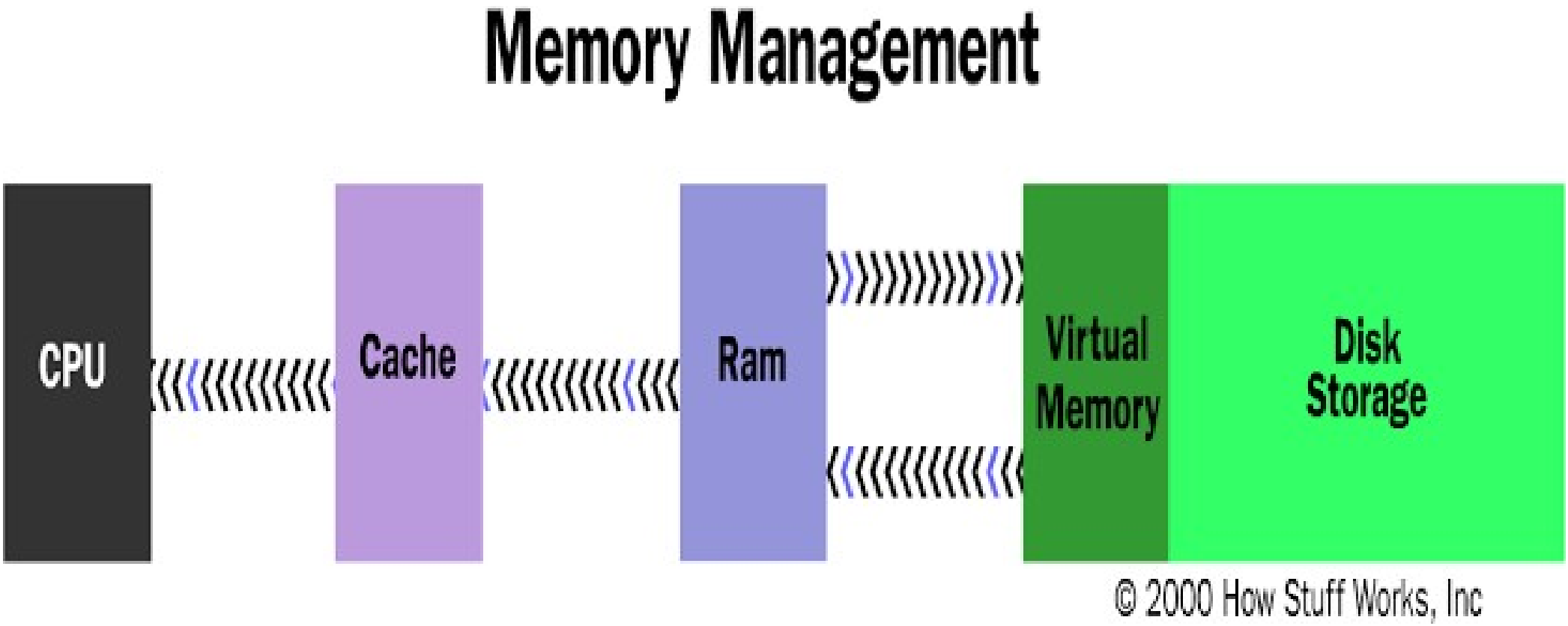


- an associative memory consists of a memory array and logic for 'm' words with 'n' bits per word.
- The functional registers like the argument register A (contains content) and key register K (for comparison) each have n bits, one for each bit of a word.
- The match register M consists of m bits, one for each memory word-contains 1/0

- The words which are kept in the memory are compared in parallel with the content of the argument register

- key register (K) provides a mask for choosing a particular field or key in the argument word
  - If the **key register contains a binary value of all 1's**, then the entire argument is compared with content of associative memory
  - When content of argument register is compared with content of associative memory, match is occurred (corresponding word in match register ,m value is 1)
  - If **key register value is 0, no comparison**, match is not performed (m value is 0)
- **Reference clues** are "associated" with actual memory contents until a desirable match (or set of matches) is found.
- Humans retrieve information best when it can be linked to other related information.

Virtual Memory

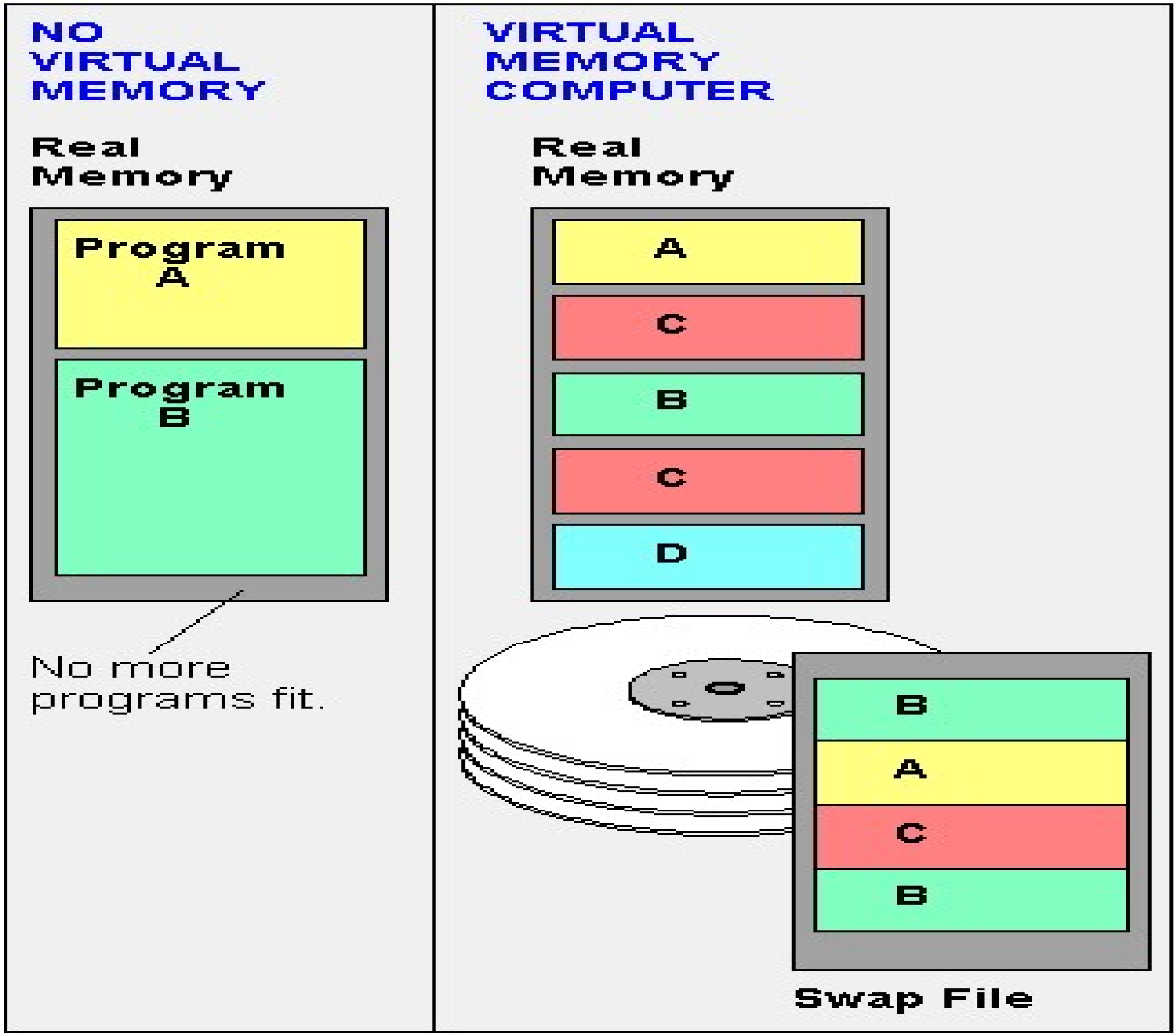


•Memory management is a method in the operating system to **manage operations between main memory and disk during process execution.**

•The main aim of memory management is to achieve **efficient utilization of memory.**

- All modern CPUs have memory management units (MMUs) that support virtual memory.
- Virtual memory is a method that computers use to **manage storage space to keep systems running quickly and efficiently.**
- Virtual memory uses both the computer's software and hardware to work.
- It transfers processes between the computer's RAM and hard disk by copying any files from the computer's RAM that aren't currently in use and moving them to the hard disk.
- By **moving unused files to the hard disk**, a computer frees up space in its RAM to perform current tasks

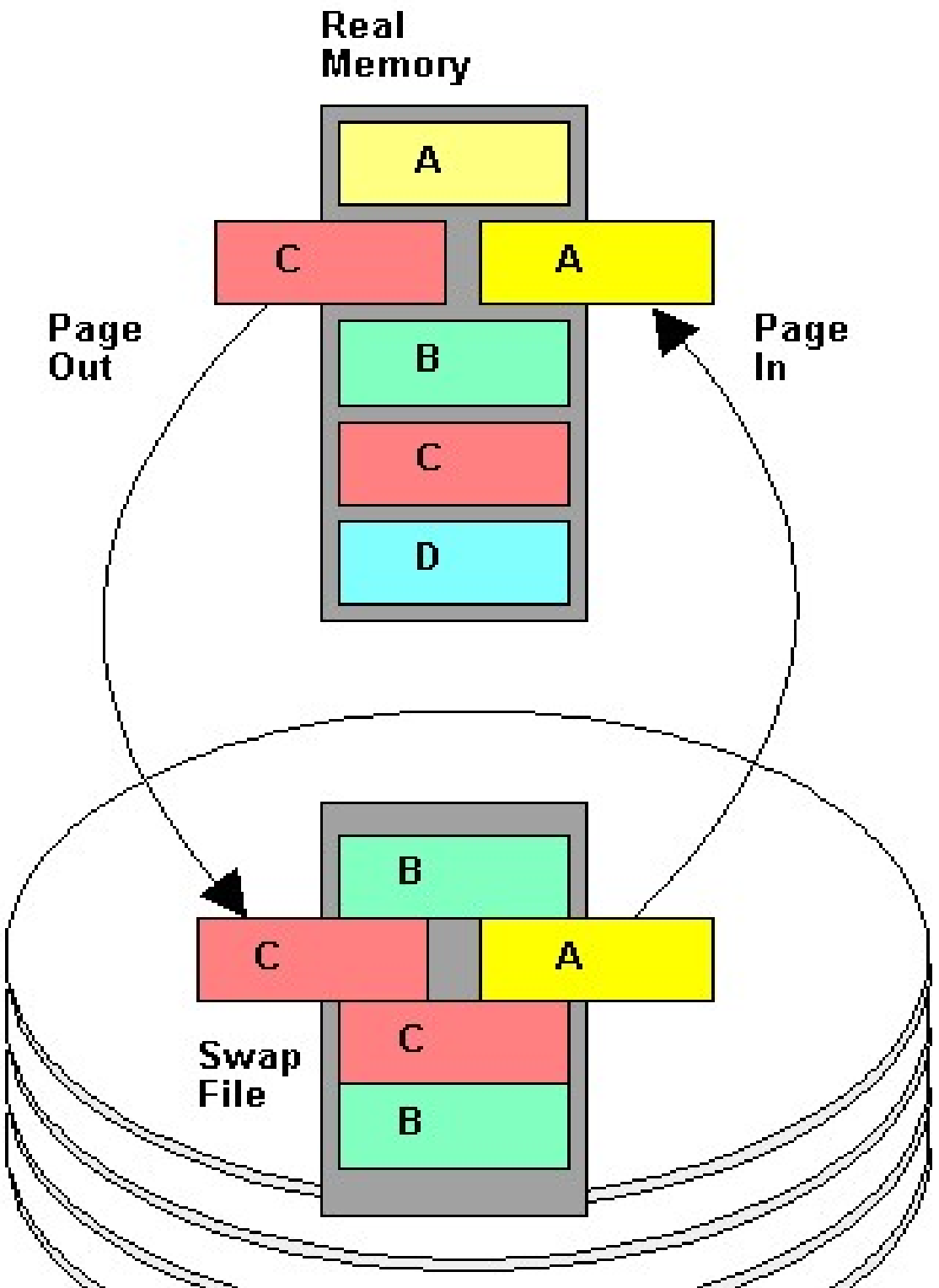
- virtual memory increases the computer's capacity to do work.
- Installing more RAM chips can be expensive, so virtual memory allows the computer to move files between systems as needed to optimize its use of the available RAM.
- They provide "page tables" that are used to translate between the program's "virtual" addresses and the "real" addresses in RAM and storage


NO VIRTUAL MEMORY
Real Memory
Program A
Program B
No more programs fit.
VIRTUAL MEMORY COMPUTER
Real Memory
A
C
B
C
D
B
A
C
B
Swap File

- **Memory Is Extended to Storage**
- The computer's RAM is divided into "pages," typically 4KB in size.
- When RAM fills up, pages not currently in use by open programs are written to storage in a "swap file."
- When instructions in a swapped out page are required again, some other page in RAM is written to storage to make room

- Virtual memory

Allows more programs to be opened simultaneously by using the hard disk as temporary storage of memory pages.



- **Page Out, Page In**
- When real memory (RAM) is full and the current program needs instructions that are not in RAM, pages are swapped.

- In this example, program A needs a page from the disk, and a page from program C is swapped out to make room.

VIRTUAL MEMORY

- 32 or 64MB of RAM available for CPU usage.
- Users expect all their programs to run at once.
- Ex Email program,a Web browser and word processor(all in RAM simultaneously)
- Find RAM for areas that have not been used recently and copy them onto the hard disk
- Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

VIRTUAL MEMORY

- Frees up space in RAM to load the new application.
- **Copying** happens automatically(feels like unlimited RAM space)
- Hard disk space is much cheaper than RAM chips, thus has a economic benefit.
- Read/write speed of a hard drive & technology is not geared toward accessing small pieces of data at a time.

VIRTUAL MEMORY

- **Operating system** has to constantly swap information back and forth between RAM and the hard disk.
- **Thrashing-** computer feels incredibly slow.

PAGING

- Unequal fixed size /Variable Size partitions(Inefficient)
- Primary memory is divided into **small equal fixed sized partitions** (256, 512, 1K) called **page frames**.
- Process are divided into **same sized blocks(pages)** called **paging.**
- Recently referenced pages in the memory.
- Need a page table to this management.

•Paging is a virtual memory technique **that separates memory into sections called paging files.**

•When a computer reaches its RAM limits, it transfers any **currently unused pages into the part of its hard drive used for virtual memory**

•The computer performs this process using a **swap file,** a designated space within its hard drive for extending the virtual memory of the computer's RAM-frees up space

•As part of this process, the computer uses **page tables**, which translate **virtual addresses into the physical addresses** that the computer's memory management unit (MMU) uses to process instructions.
•The MMU communicates between the computer's OS and its page tables.
•When the user performs a task, the OS searches its RAM for the processes to conduct the task.
• If it can't find the processes to complete the task in RAM, the MMU prompts the OS to move the required pages into RAM and uses a page table to note the new storage location of the pages.

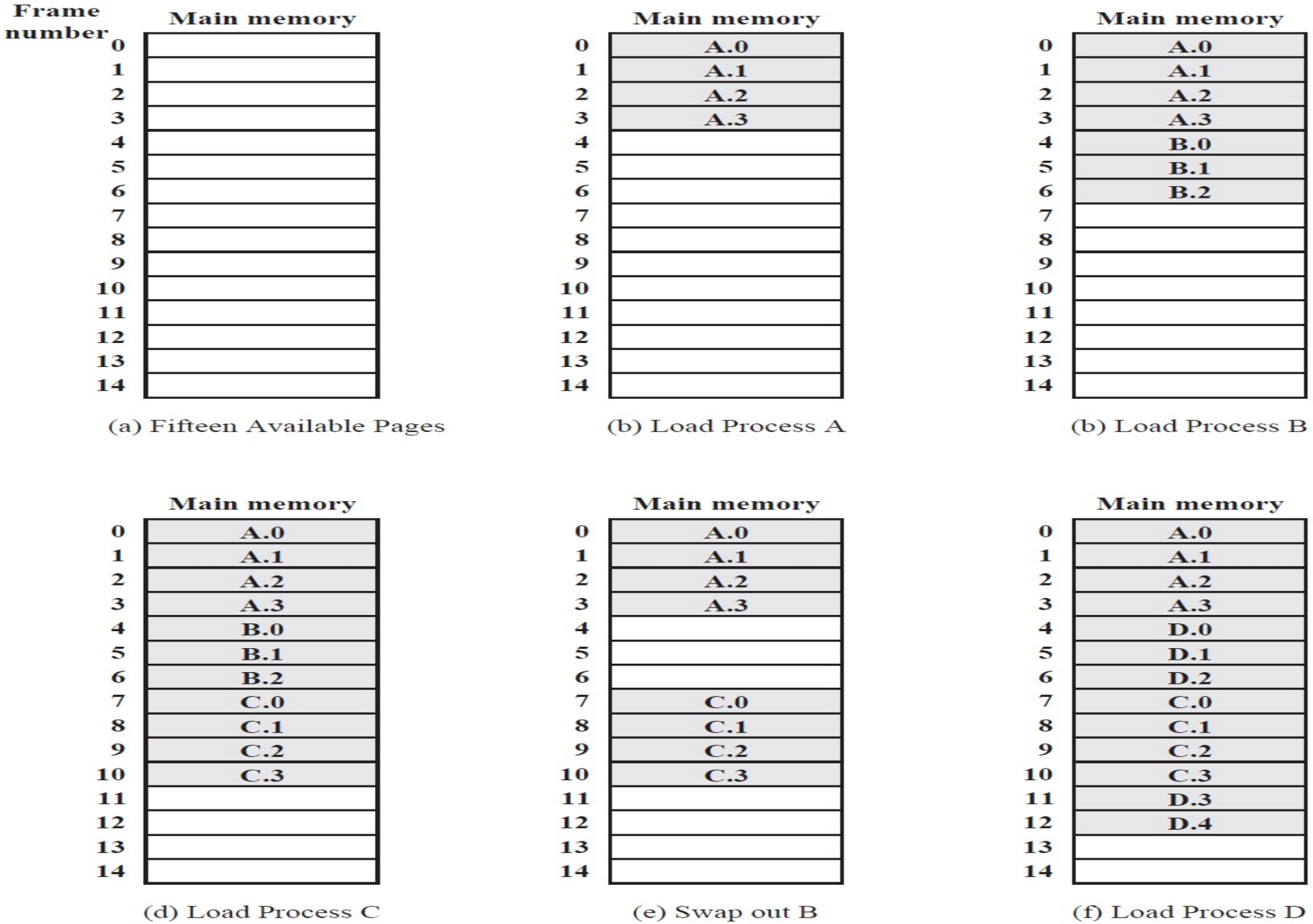
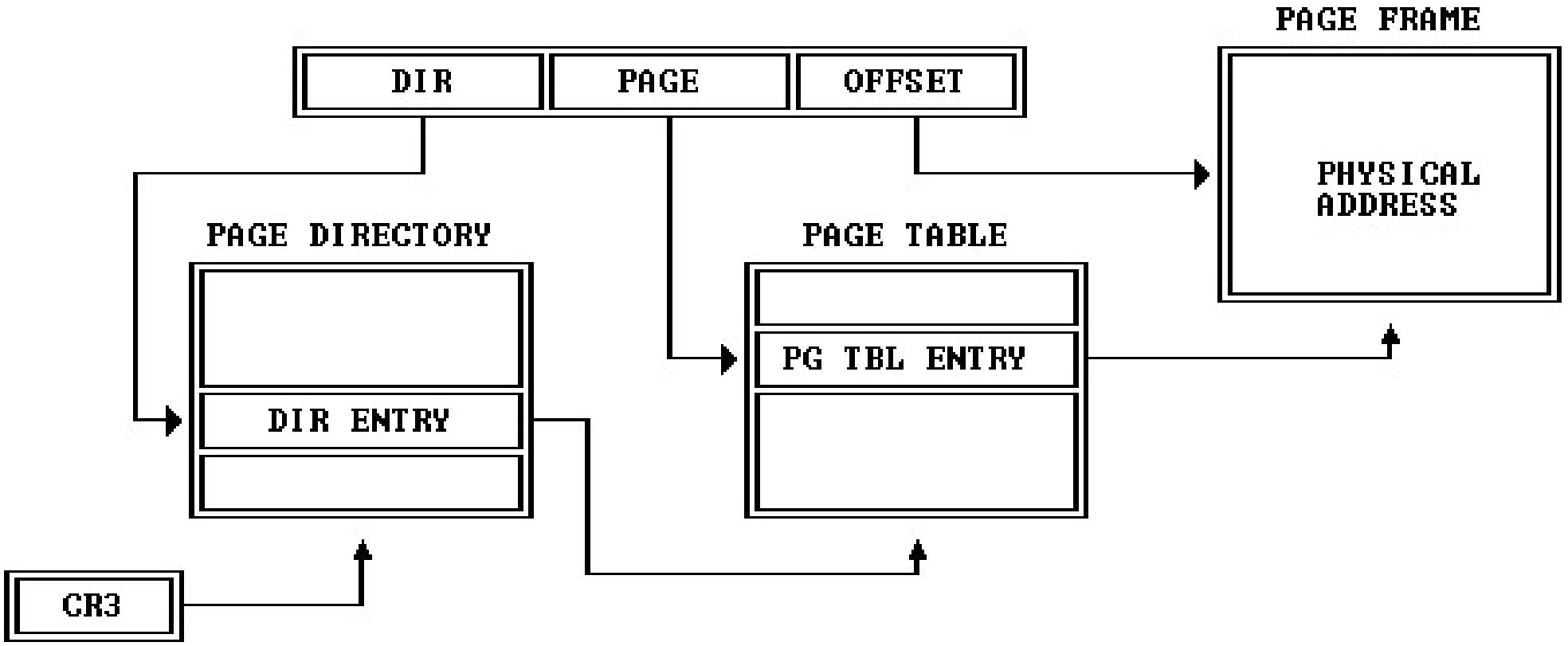


Figure 7.9 Assignment of Process Pages to Free Frames

Page Table Sample

Figure 5-8. Format of a Linear Address

Figure 5-9. Page Translation



•A linear address refers indirectly to a physical address by specifying a page table, a page within that table, and an offset within that page

•how the processor converts the DIR, PAGE, and OFFSET fields of a linear address into the physical address by consulting two levels of page tables.
•The addressing mechanism uses the DIR field as an index into a page directory, uses the PAGE field as an index into the page table determined by the page directory, and uses the OFFSET field to address a byte within the page determined by the page table.
•A page table is simply an array of 32-bit page specifiers. A page table is itself a page, and therefore contains 4 Kilobytes of memory or at most 1K 32-bit entries.

SEGMENTATION

- Paging → **internal fragmentation**.
- Segmentation maps segments representing data structures, modules, etc. into **variable partitions.**
- Nor contiguous memory blocks neither all segments of a process are loaded at a time.
- We need a segment table very much like a page table.

- As processes are loaded and removed from memory, the free memory space is broken into little pieces. It may so happen after sometime that processes cannot be allocated to memory blocks considering their small size and memory blocks remain unused. This problem is known as **Fragmentation**

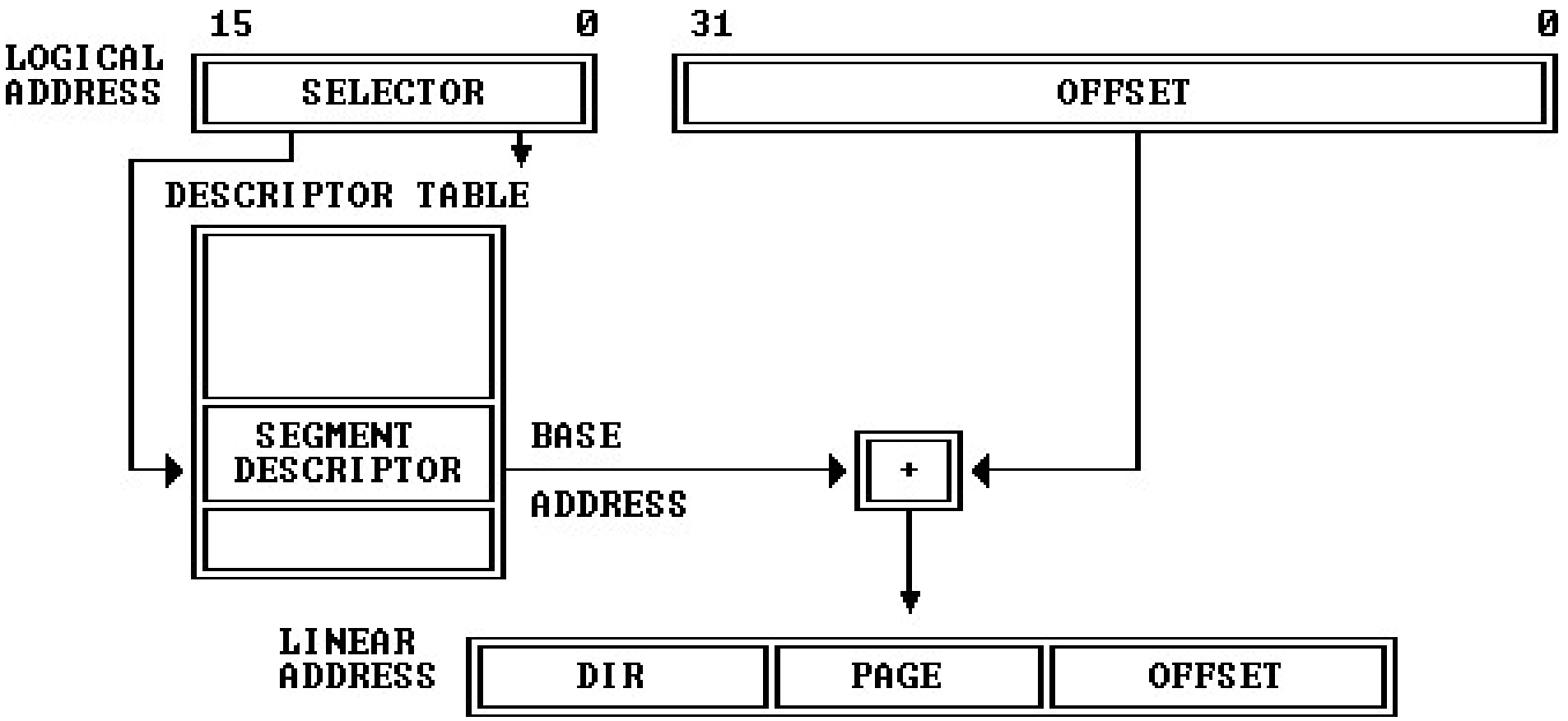
- **Internal fragmentation** is defined as the difference between memory allocated and the memory space required by a process
- **External fragmentation-**When there are small and non-contiguous memory blocks which cannot be assigned to any process, the problem is termed as External Fragmentation.

- Segmentation-divides virtual memory into **varying lengths** and moves any segments that aren't in use from the computer's virtual memory space to its hard drive.
- Like page tables, segment tables track whether the computer stores the segment in memory or a physical address

•Segmentation is often slower than paging, but it offers the user more control over how to divide memory and may make it easier to share data between processes.

•You can customize the segments based on the machine's purpose and usage

Figure 5-2. Segment Translation

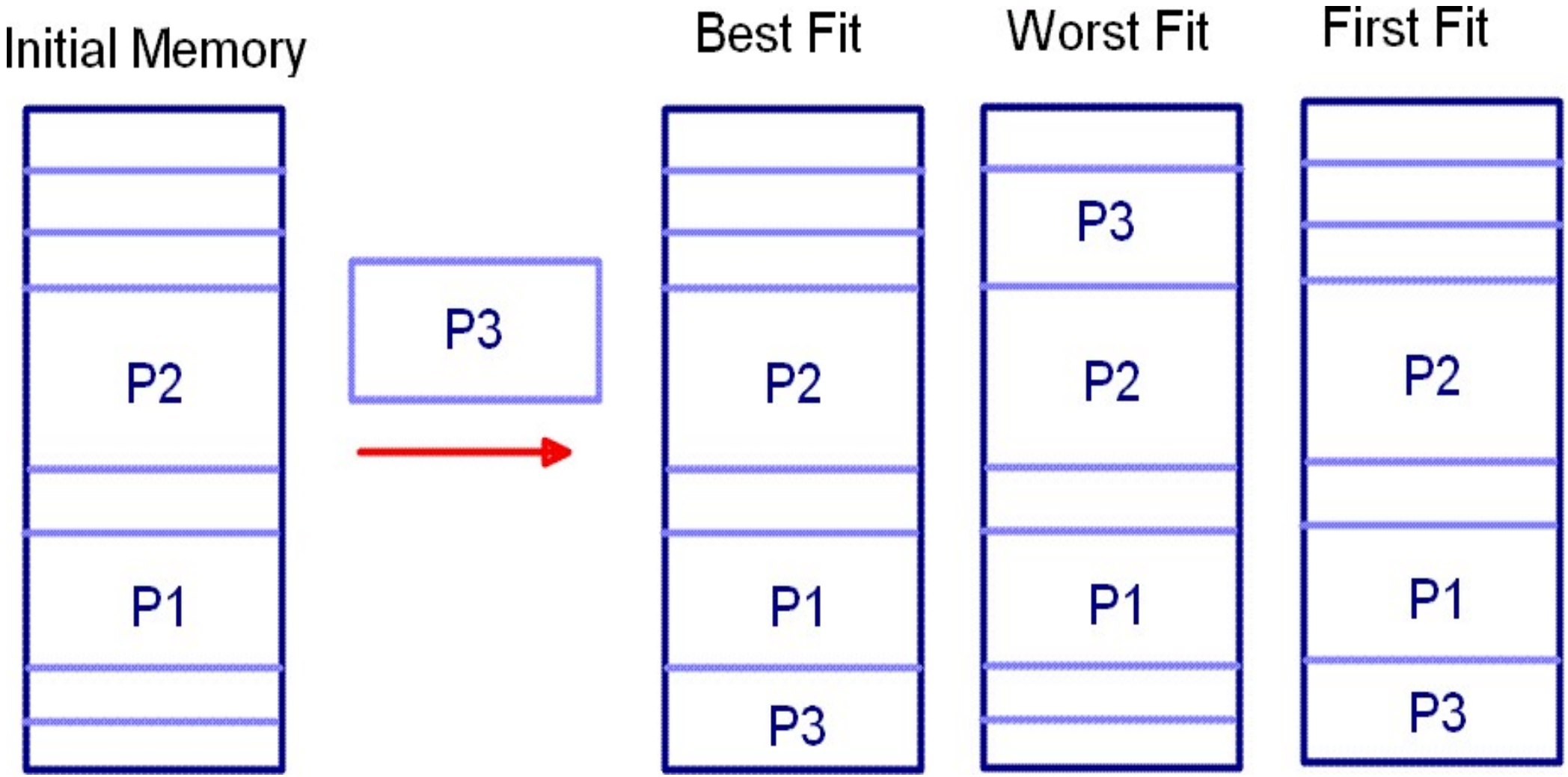


Main Memory Allocation

- Memory is divided into set of contiguous locations called regions/segments/pages
- Store blocks of data
- Placement of blocks of information in memory is called **Memory Allocation**
- **Memory Management Systems** keeps information in a table containing available and free slots

Allocation is done only as per needs

- First Fit
- Best Fit



- The best-fit algorithm searches for the **smallest free partition** that is large enough to accommodate the process.
- The operating system searches the entire memory and selects the free partition that is closest in size to the process.

- The worst-fit algorithm searches for **the largest free partition and allocates the process to it.**
- This algorithm is designed to leave the largest possible free partition for future use.

- The first-fit algorithm searches for the **first free partition that is large enough to accommodate the process.** The operating system starts searching from the beginning of the memory and allocates the first free partition that is large enough to fit the process.

Refer pros & cons of all
Compare segmentation and paging

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, such an algorithm must be implemented in hardware

Replacement Algorithms

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
  - —Pick the slot that hasn't been used in the longest time.

  -easy to implement for a fully associative cache

  -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.
- First in first out (FIFO)
  - —replace block that has come into cache first

- Random
- OPT-Optimal(Future)

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

FIFO,LRU,OPT

1) 1 , 6 ,4 , 5 , 1 ,4, 3, 2, 1, 2, 1, 4,6,7,4

FIFO→ 7 – 4 – 6

LRU→ 4 - 6 - 7

OPT→ 7 – 6 -4 (Conflict resolved using LRU)

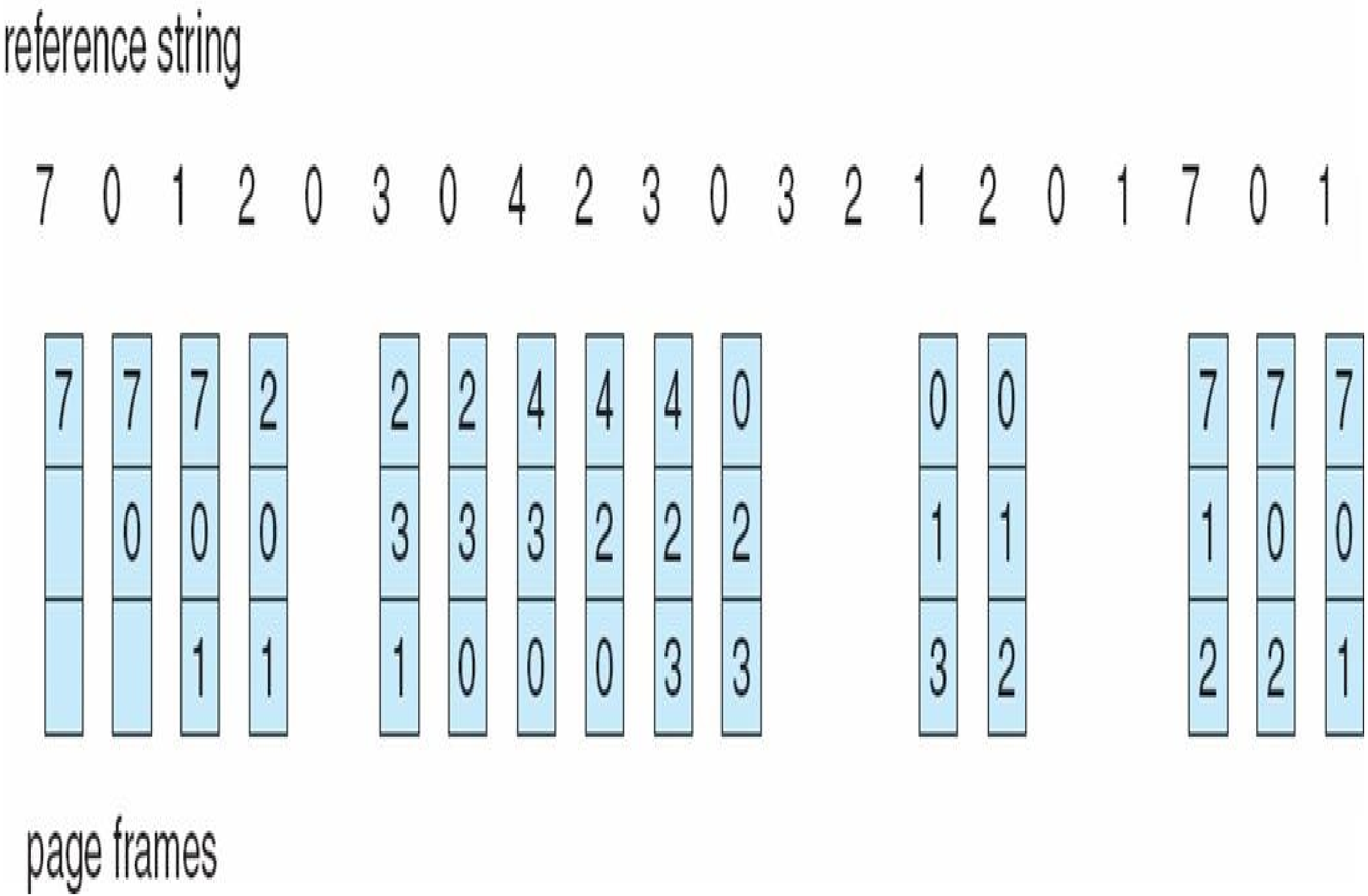
2) 2 , 3 , 2 , 1 , 5 , 2 , 4 , 5 , 3 , 2 , 5 , 2

FIFO→ 3 - 2 - 5

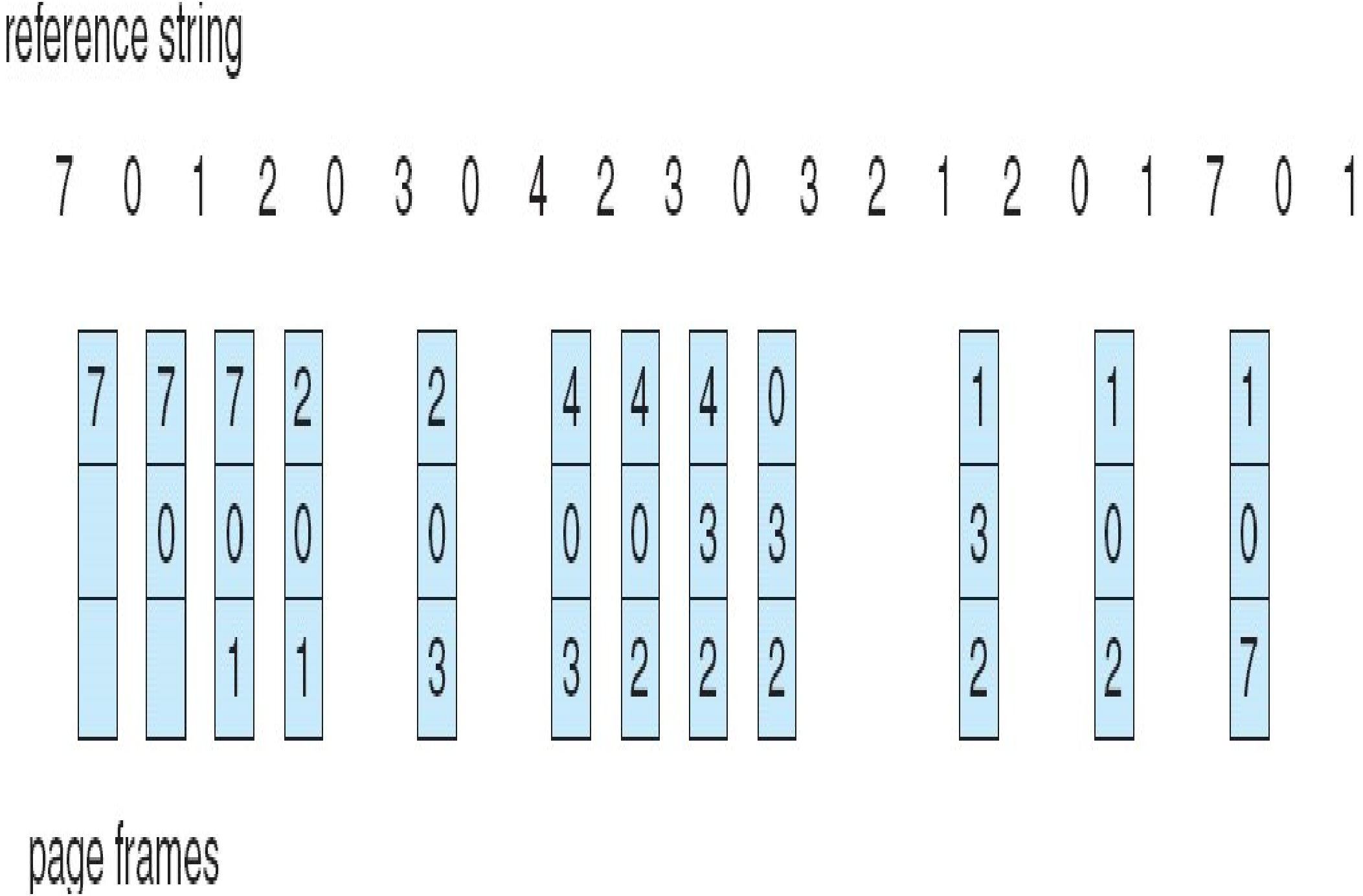
LRU→ 3 – 5 - 2

OPT→ 2 - 3 - 5

FIFO Page Replacement

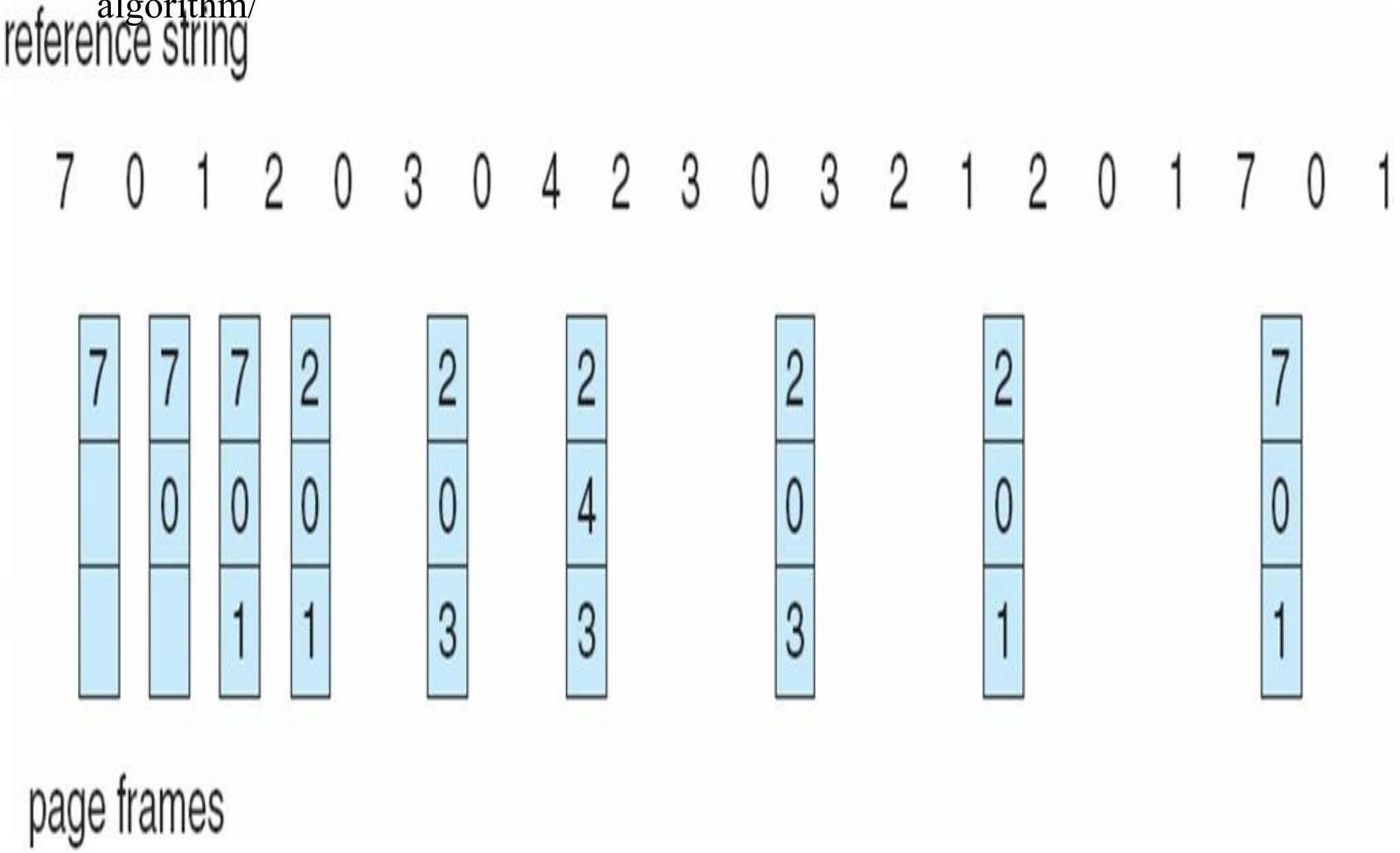


LRU Page Replacement



Optimal Page Replacement

https://www.scaler.com/topics/optimal-page-replacement-algorithm/



The page replacement algorithm aims to reduce page faults as much as possible.
To do this, this algorithm searches for the page from reference, which is not going to be used earliest
And that page is replaced by the new page in the frame.

The 'optimal page replacement algorithm' is not practical because it cannot predict whether the page it is going to remove will not be used in the future.

Secondary Storage

- Magnetic disks
- Floppy disks
- Magnetic Tape
- RAID
- Optical Memory
- CD-ROM
- DVD

RAID Levels 0 - 6

REDUNDANT ARRAY OF INDEPENDENT DISKS

- Storage is an important consideration when setting up a server.
- Almost all of the important information that you and your users care about will at one point be written to a storage device to save for later retrieval.
- Single disks can serve you well if your needs are straight forward.
- However, if you have more complex redundancy or performance requirements, solutions like RAID can be helpful.

- RAID is a technique that makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy, or both.
- The RAID scheme consists of seven levels (0 to 6)
- 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.

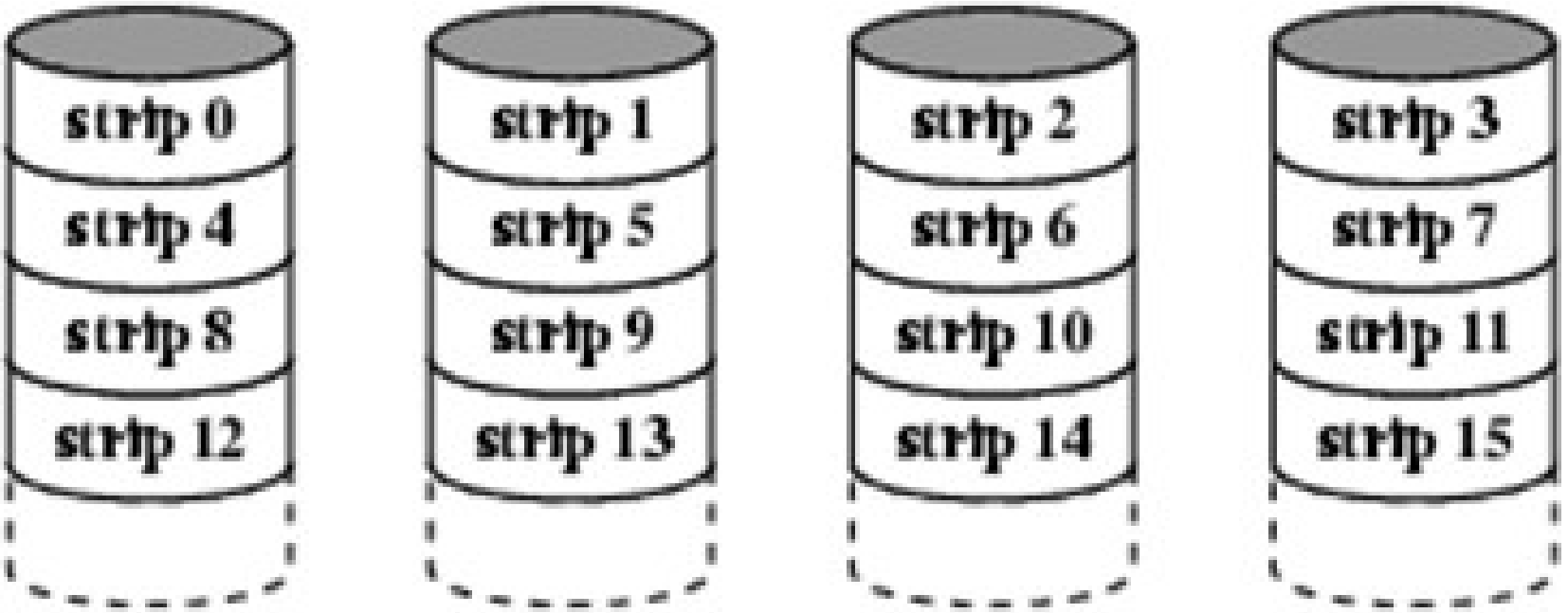
Table 6.3 RAID Levels

Category	Level	Description	Disks Required	Data Availability	Large I/O Data Transfer Capacity	Small I/O Request Rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to log N

It employs multiple disk drives and distributes data in such a way as to enable simultaneous access to data from multiple drives, thereby improving I/O performance and allowing easier incremental increases in capacity

RAID Level 0- Non Redundant



(a) RAID 0 (non-redundant)

- not a true member of the RAID family because it does not include redundancy to improve performance
- For RAID 0, the user and system data are distributed across all of the disks in the array.
- the two requests can be issued in parallel, reducing the I/O queuing time
- The logical disk is divided into strips; these strips may be physical blocks, sectors, or some other unit.
- The strips are mapped round robin to consecutive physical disks in the RAID
- A set of logically consecutive strips that maps exactly one strip to each array member is referred to as a stripe

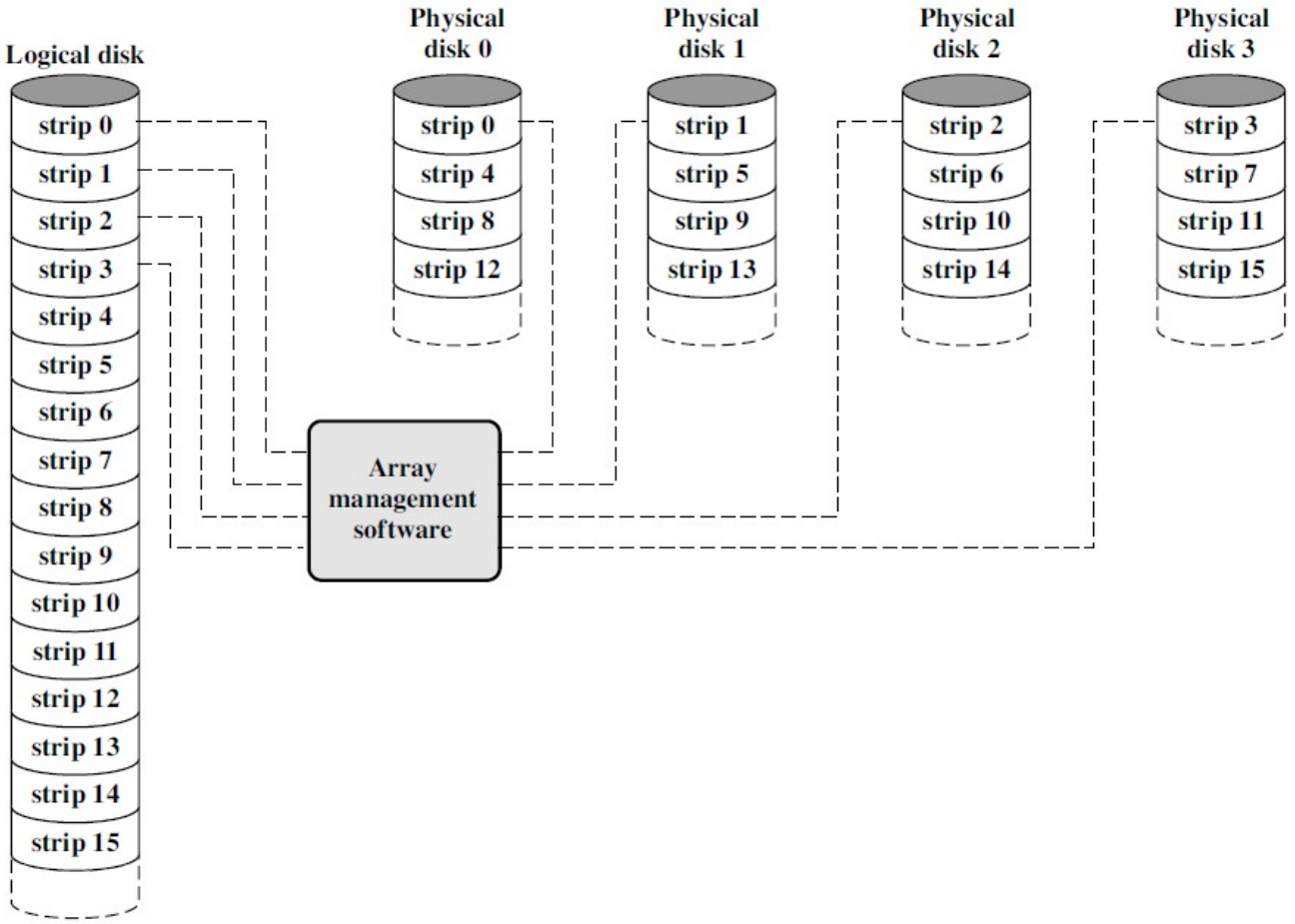
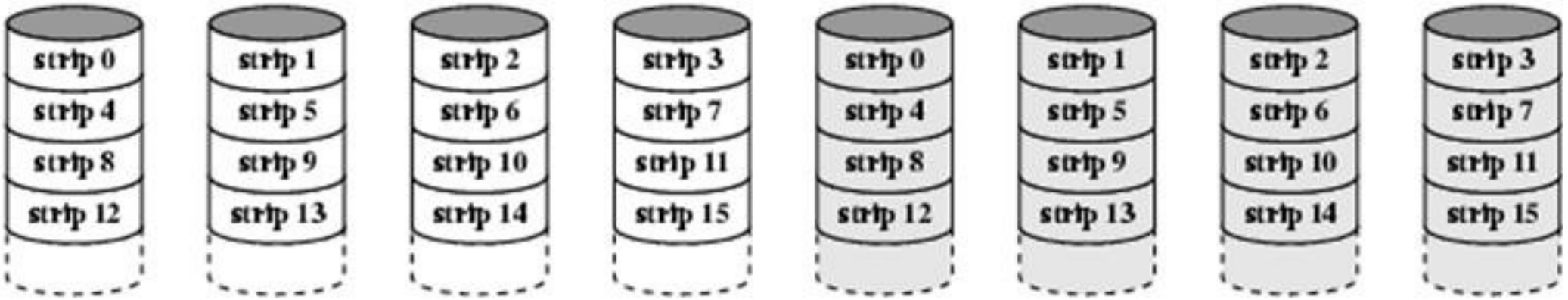


Figure 6.9 Data Mapping for a RAID Level 0 Array

- use of array management software to map between logical and physical disk space
- Logical disk divided to 4 physical disks in round robin
- 1st strip (0) to disk 0, 2nd to 1, etc until all are mapped

RAID Level -1 Mirrored



(b) RAID 1 (mirrored)

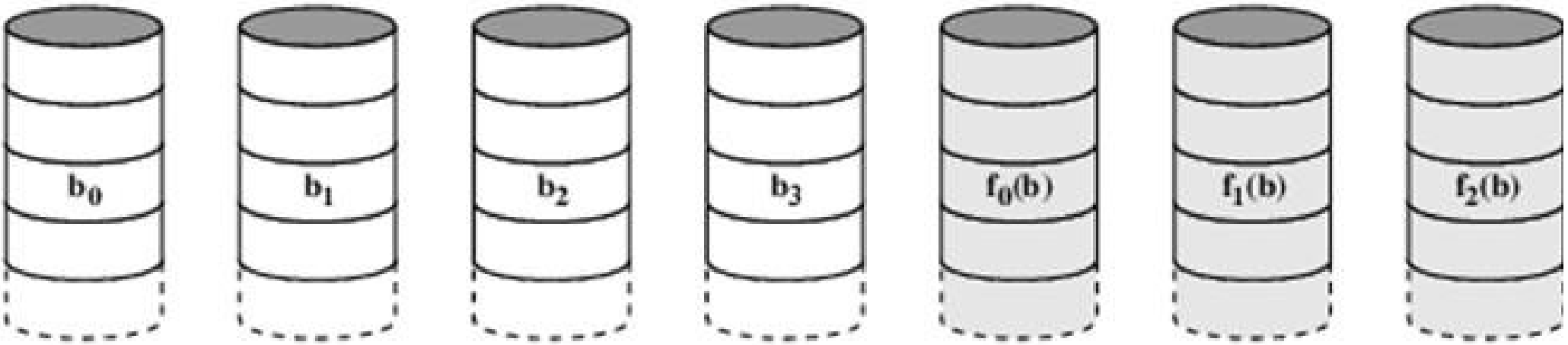
- redundancy is achieved by the simple expedient of duplicating all the data
- data striping is used,
- each logical strip is mapped to two separate physical disks so that every disk in the array has a mirror disk that contains the same data

When data is read, it can be retrieved from the disk with the shorter queuing, seek and rotational delays.

Mirroring is frequently used in database applications where availability and transaction time are more important than storage efficiency.

- Redundancy achieved through duplicating all data.
- Data stripping is similar to RAID level 0.
- Each logical strip is mapped to two physical disks.
  - Read request can be serviced from either of available 2 disks
  - Write request requires both disks to be updated – but this can be done in parallel. (Slower write dictates overall speed).
  - Recover from failure is simple! (data may still be accessed from the second drive
- Disadvantage:
  - Cost
  - requires twice the disk space
  - used only for system software and other highly critical files.
- Improvement occurs if the application can split each read request so that both disk members participate

RAID Level 2- Hamming Code

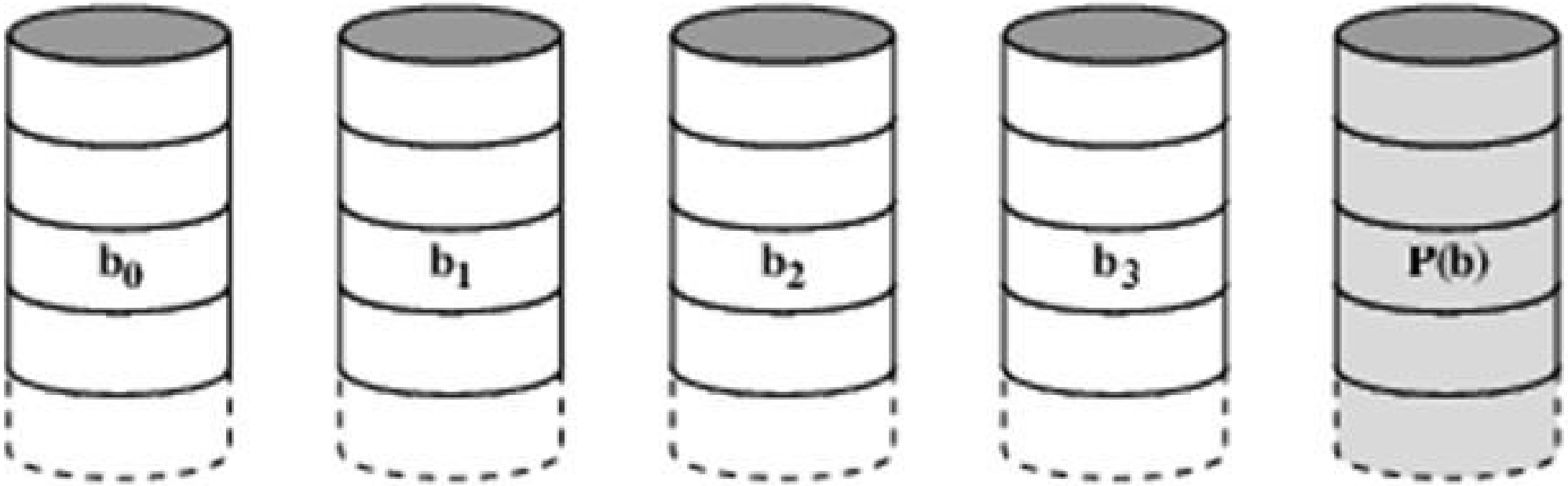


(c) RAID 2 (redundancy through Hamming code)

- make use of a parallel access technique
- all member disks participate in the execution of every I/O request
- the strips are very small
- With RAID 2, an error-correcting code (Hamming code) is calculated across corresponding bits on each data disk, and the bits of the code are stored in the corresponding bit positions on multiple parity disks.
- B0, b1, b2 & b3 are data disk, f0(b), f1(b), f2(b) are redundant disk
- Whatever be the value of bit b0b1b2& b3 for the value I, hamming code will be found and placed in f0(b), f1(b), f2(b)
- Bit level stripping- For 4 bit data, First bit of data goes to b0, 2nd bit to b1, 3rd bit to b2 and 4th bit to b3.
- Easily rectify data if a particular data is failed
- Costly

- Utilizes parallel access techniques - All disks participate in the execution of every I/O request.
- Data striping – bit level stripping
- Cost is reduced by using hamming code rather than mirroring
- Multiple redundant disks are needed to identify the failed disk, but only one is needed to recover the lost information.
- Error correcting code is calculated across corresponding bits on each disk, and the code bits are stored in corresponding bit positions on multiple parity disks.

RAID Level 3 –Bit Interleaved Parity

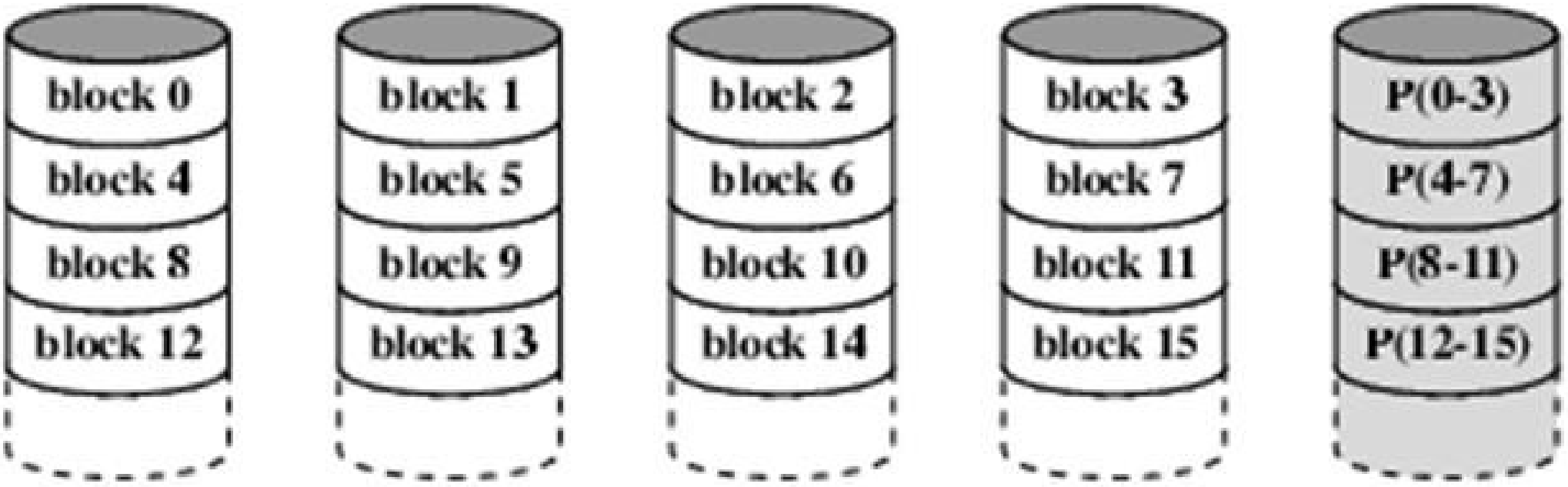


(d) RAID 3 (bit-interleaved parity)

- requires only a single redundant disk
- employs parallel access, with data distributed in small strips.
- Instead of an error-correcting code, a simple parity bit is computed for the set of individual bits in the same position on all of the data disks.
- Bit level stripping
- No of disks required to store parity data is low-Only one extra disk

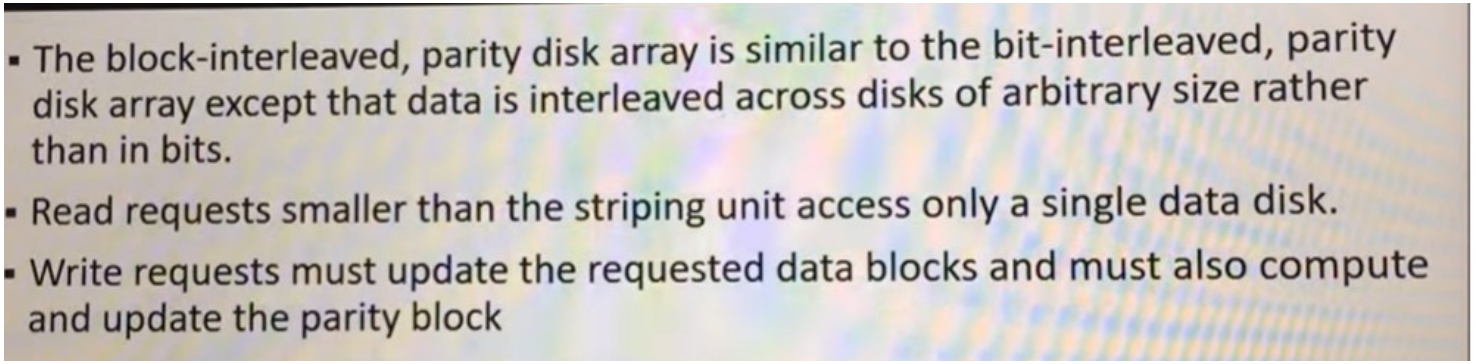
- Each read request accesses all data disks and each write request accesses all data disks and the parity disk.
- Thus, only one request can be serviced at a time.
- Bit-interleaved, parity disk arrays are frequently used in applications that require high bandwidth but not high I/O rates.

RAID Level 4- Block level parity



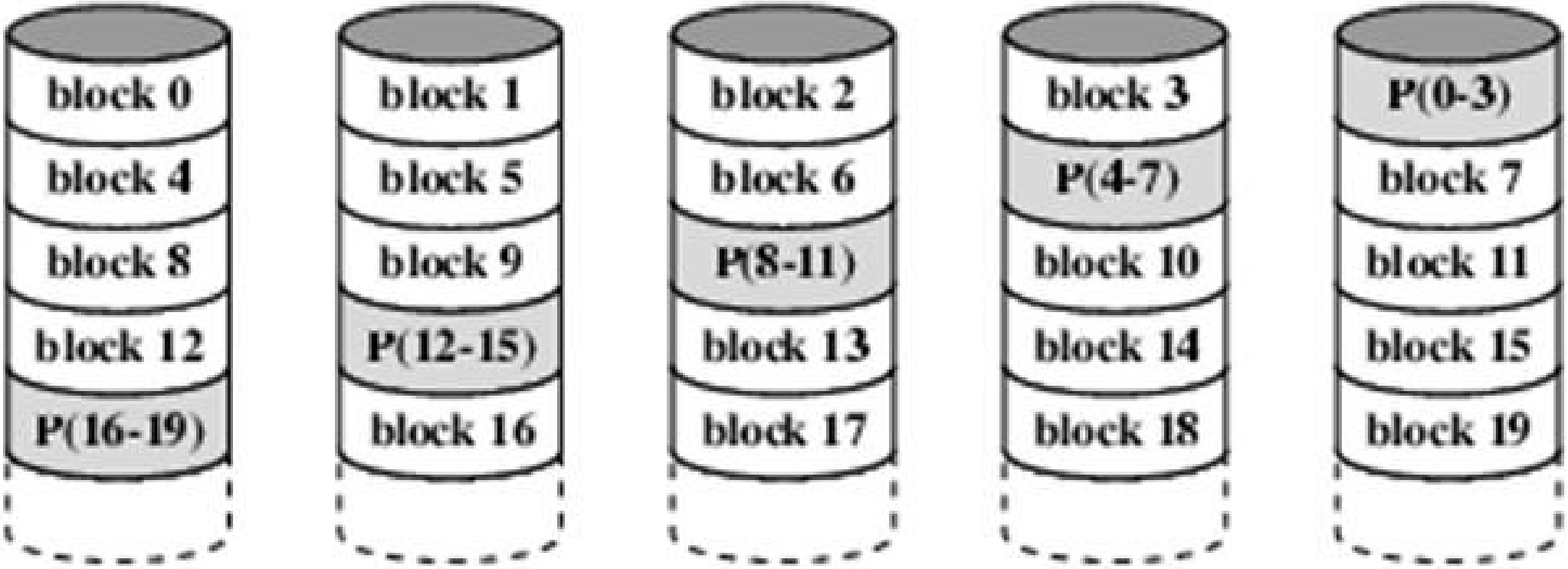
(e) RAID 4 (block-level parity)

- In the case of RAID 4 through 6, the strips are relatively large.
- With RAID 4, a bit-by-bit parity strip is calculated across corresponding strips on each data disk, and the parity bits are stored in the corresponding strip on the parity disk.
- Block level stripping
- Block is composed of 1/more words
- If one word, One complete word will come to one block and parity will be calculated based on blocks 0 to 3
- One extra disk is required to maintain redundant data



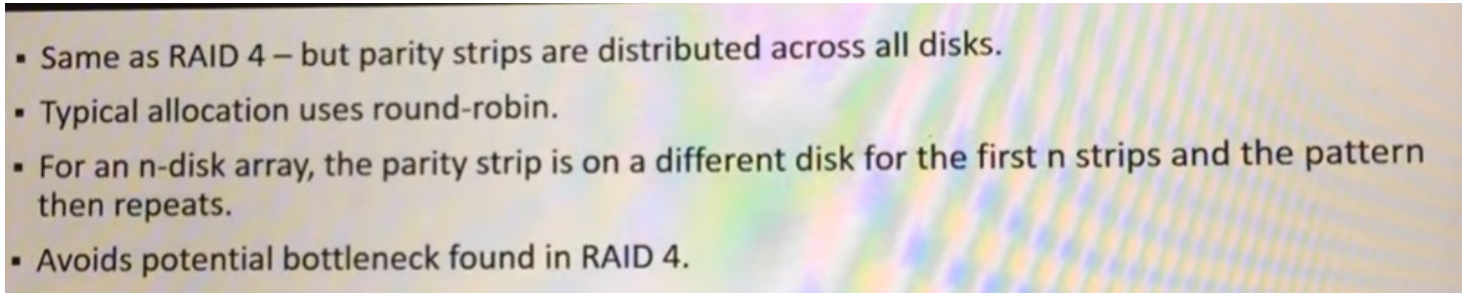
- For large writes that touch blocks on all disks, parity is easily computed by exclusive-or'ing the new data for each disk.
- For small write requests that update only one data disk, parity is computed by noting how the new data differs from the old data and applying those differences to the parity block.
- Small write requests thus require four disk I/Os:
  - one to write the new data
  - two to read the old data and old parity for computing the new parity
  - one to write the new parity.
  - This is referred to as a read-modify-write procedure.
- Because a block-interleaved, parity disk array has only one parity disk, which must be updated on all write operations, the parity disk can easily become a bottleneck

RAID Level 5- Block level Distributed Parity

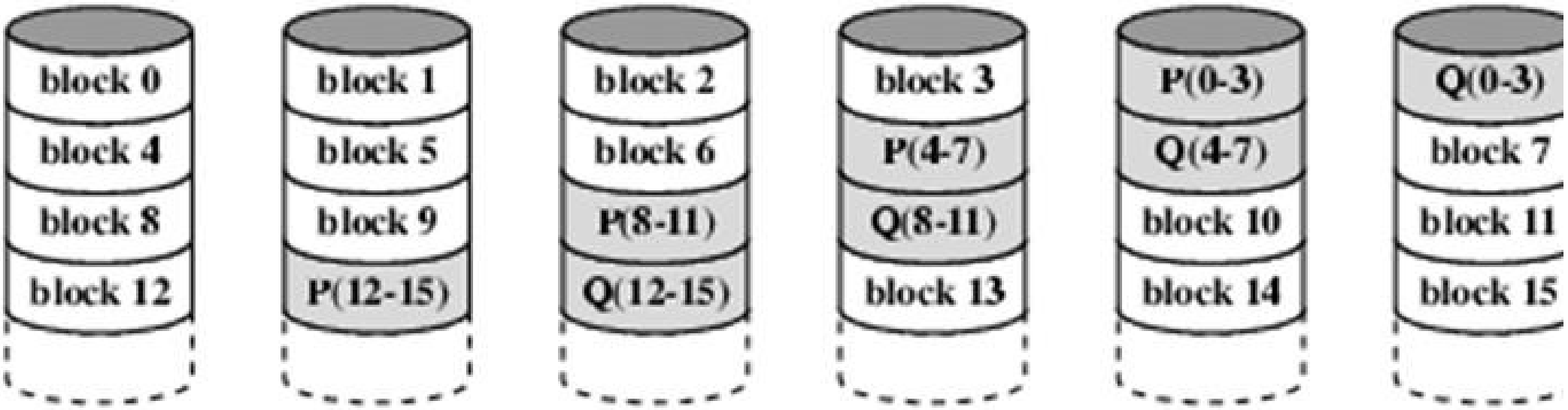


(f) RAID 5 (block-level distributed parity)

- For an n-disk array, the parity strip is on a different disk for the first n stripes, and the pattern then repeats.
- The distribution of parity strips across all drives avoids the potential I/O bottleneck found in RAID 4.

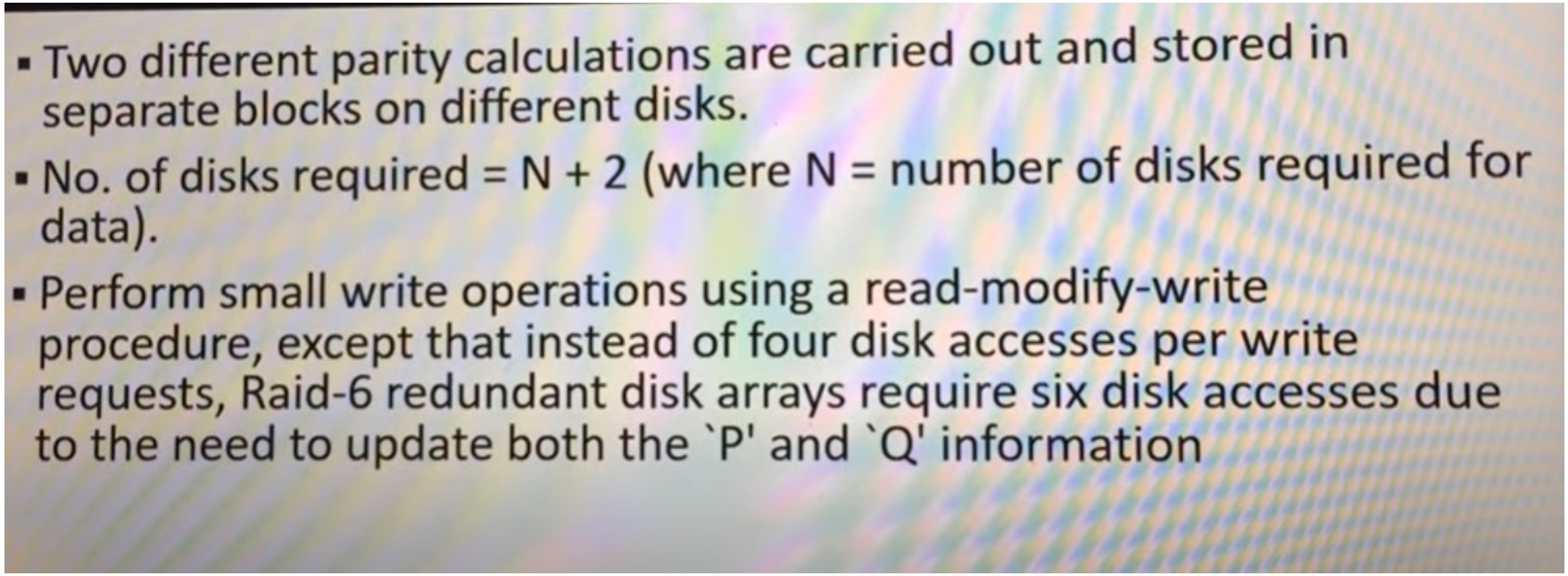


RAID Level 6- Dual Redundancy

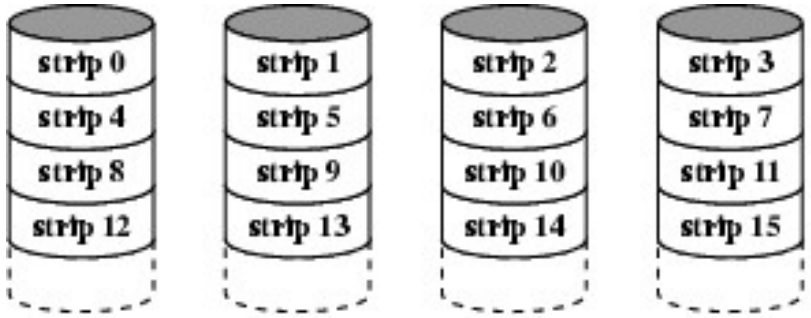


(g) RAID 6 (dual redundancy)

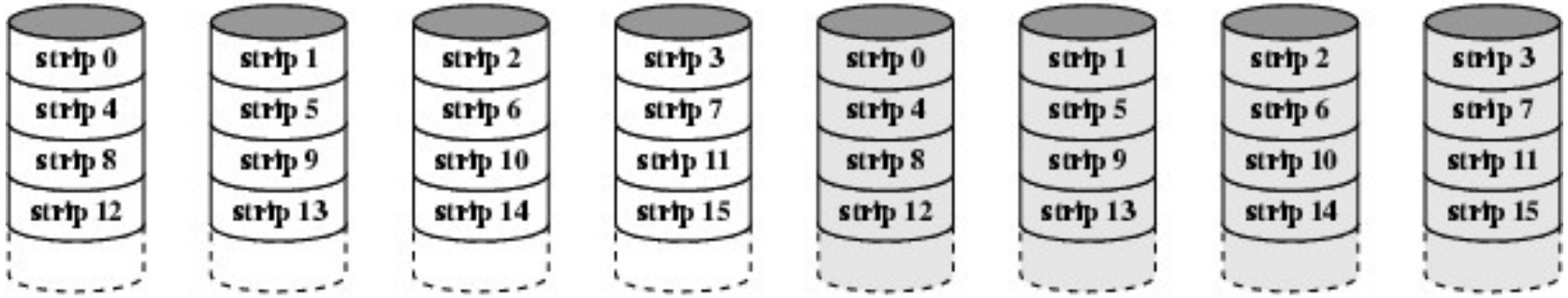
- Increasing the no: of redundant disk arrays (2)
- P and Q are two different data check algorithms.
- This makes it possible to regenerate data even if two disks containing user data fail.
- it provides extremely high data availability



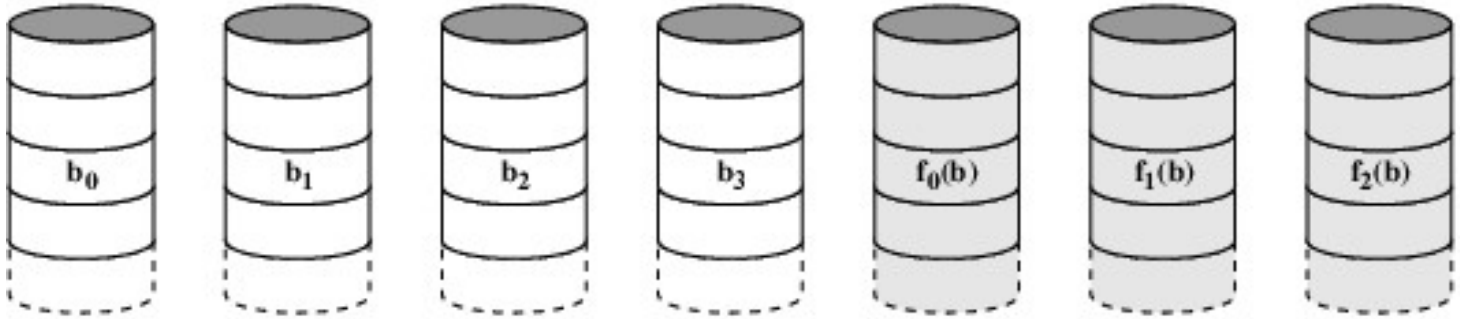
RAID 0, 1, 2 – Redundant Array of Independent Disks



(a) RAID 0 (non-redundant)

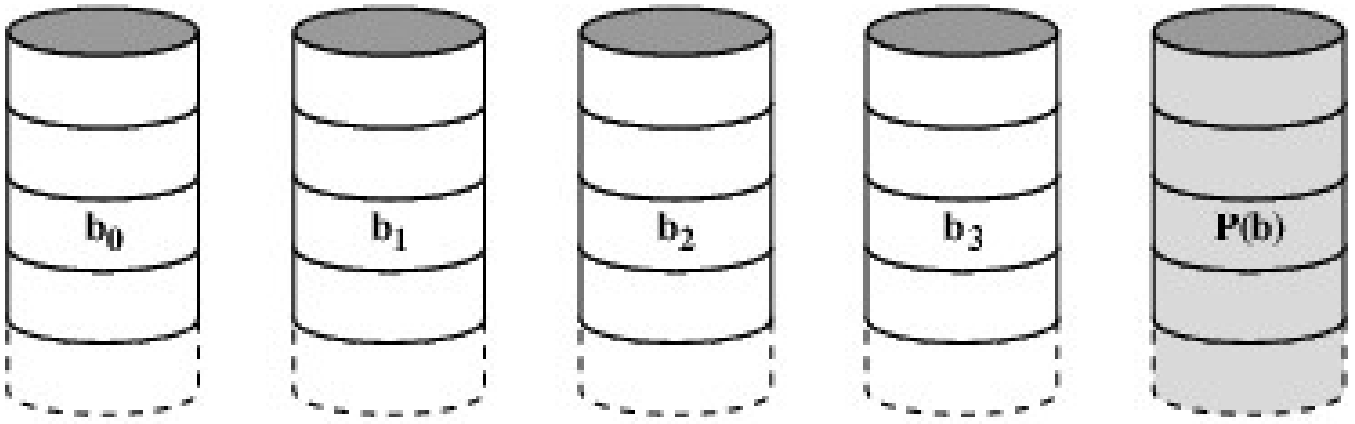


(b) RAID 1 (mirrored)

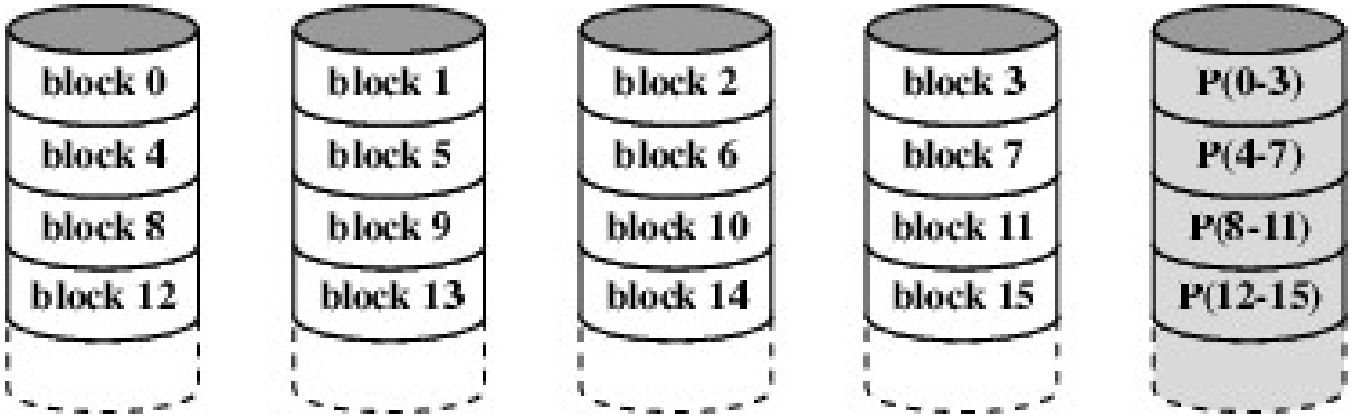


(c) RAID 2 (redundancy through Hamming code)

RAID 3 & 4

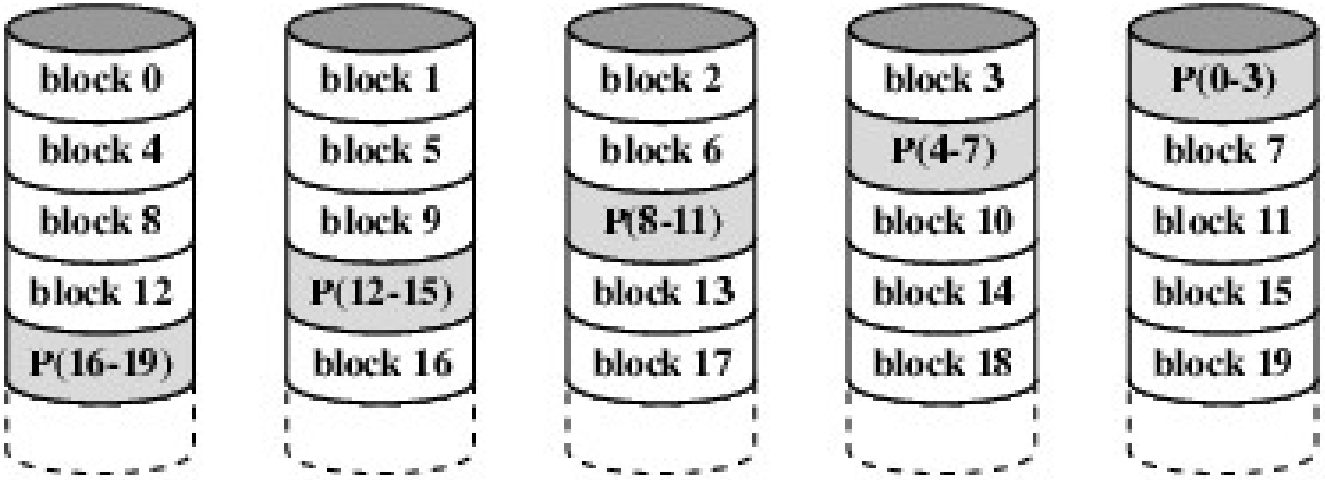


(d) RAID 3 (bit-interleaved parity)

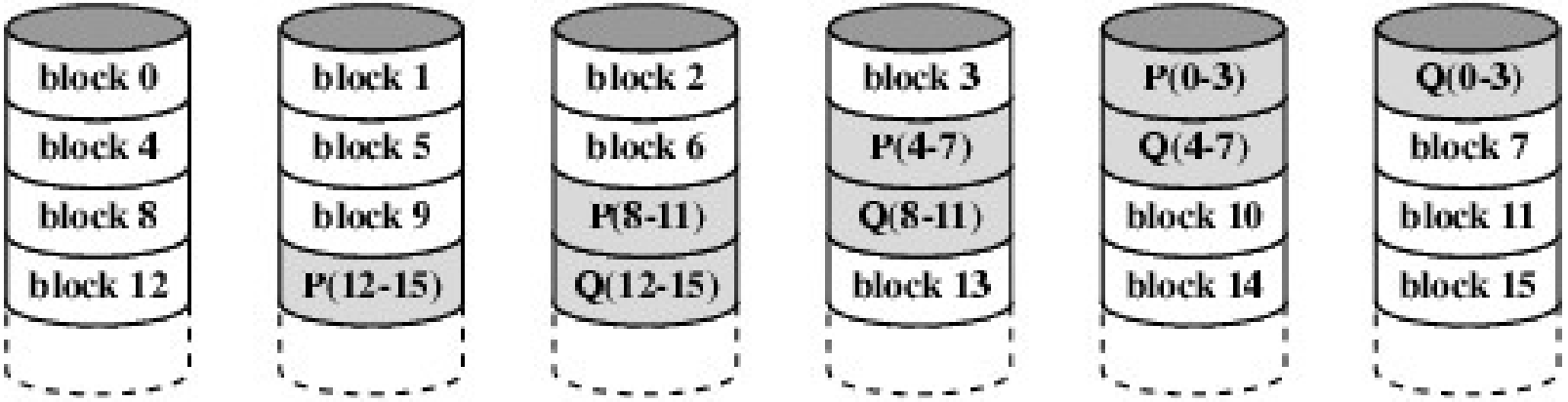


(e) RAID 4 (block-level parity)

RAID 5 & 6



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



(g) RAID 6 (dual redundancy)

Solve using FIFO , LRU and OPT page replacement algorithms

- Given page reference string:
  1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

Virtual memory

https://pdos.csail.mit.edu/6.828/2011/readings/i386/s05_02.htm
https://www.slideserve.com/ayasha/memory-management
https://tildesites.bowdoin.edu/~allen/courses/cs220/lab7/notes.html#:~:text=Direct%20Mapping&text=The%20%22Tag%22%20field%20of%20the,with%20the%20desired%20memory%20line.

5.0	**I/O Organization**		03	CO4
	5.1	External Devices, I/ O Modules		
	5.2	Programmed I/O, Interrupt driven I/O, DMA		
6.0	**Multiprocessor Configurations**		06	CO4
	6.1	Flynn's classification, Parallel processing systems and concepts		
	6.2	Introduction to pipeline processing and pipeline hazards		
	6.3	Design issues of pipeline architecture, Instruction pipelining: Six Stage instruction pipeline.		
	6.4	8086 Instruction (Arithmetic Instructions, Logical Instructions, Data transfer instructions)		
	Self-Learning; Pin Diagram of 8086, Minimum Mode and Maximum mode with timing diagram			
		Total	**45**	