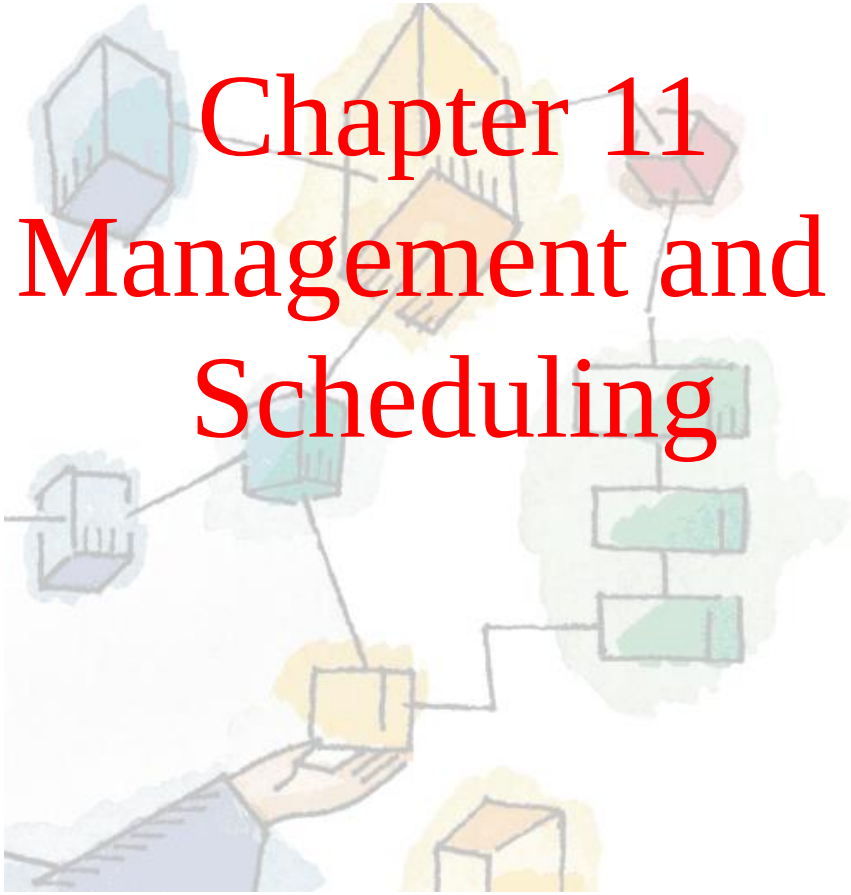
*Operating Systems:*
*Internals and Design Principles, 6/E*
William Stallings

# Chapter 11
# I/O Management and Disk Scheduling



Patricia Roy
Manatee Community College, Venice, FL

# Differences in I/O Devices

- Data rate
  - May be differences of several orders of magnitude between the data transfer rates
- Unit of transfer – Character and Block devices
  - Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk
- Data representation
  - Encoding and error-correction schemes
- Error conditions
  - Different types of errors

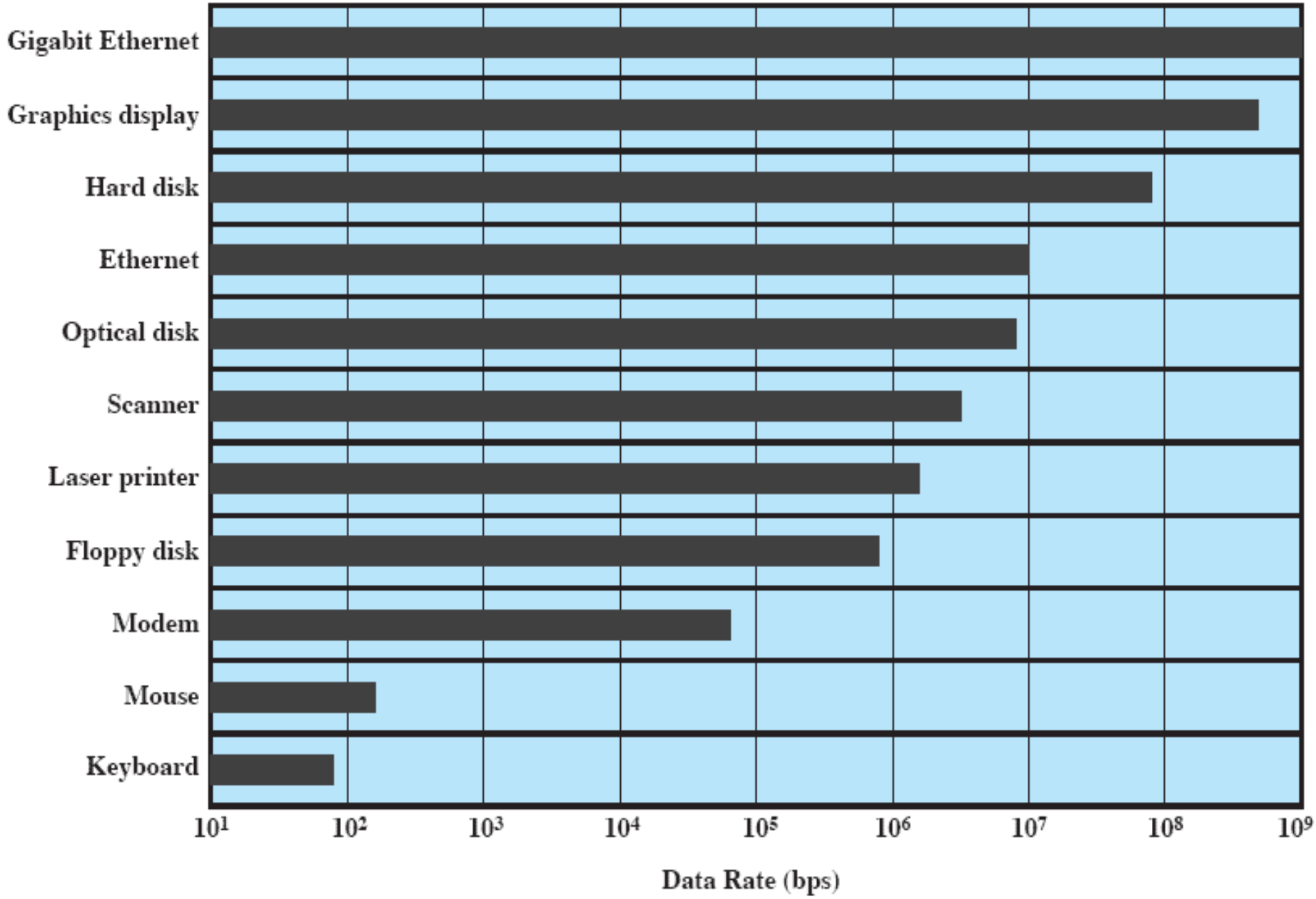



**Figure 11.1 Typical I/O Device Data Rates**

# Device Controllers

- I/O devices have components:
  - mechanical component
  - electronic component
- The electronic component is the device controller
  - may be able to handle multiple devices
- Controller's tasks
  - convert serial bit stream to block of bytes
  - perform error correction as necessary
  - communicate with CPU

# Performing I/O

- Programmed I/O

Process(or) is busy-waiting for the operation to complete

- Interrupt-driven I/O

I/O command is issued

Processor continues executing other instructions

Interrupt is sent when device is ready again

- Direct Memory Access

DMA module controls exchange of data between main memory and the I/O device

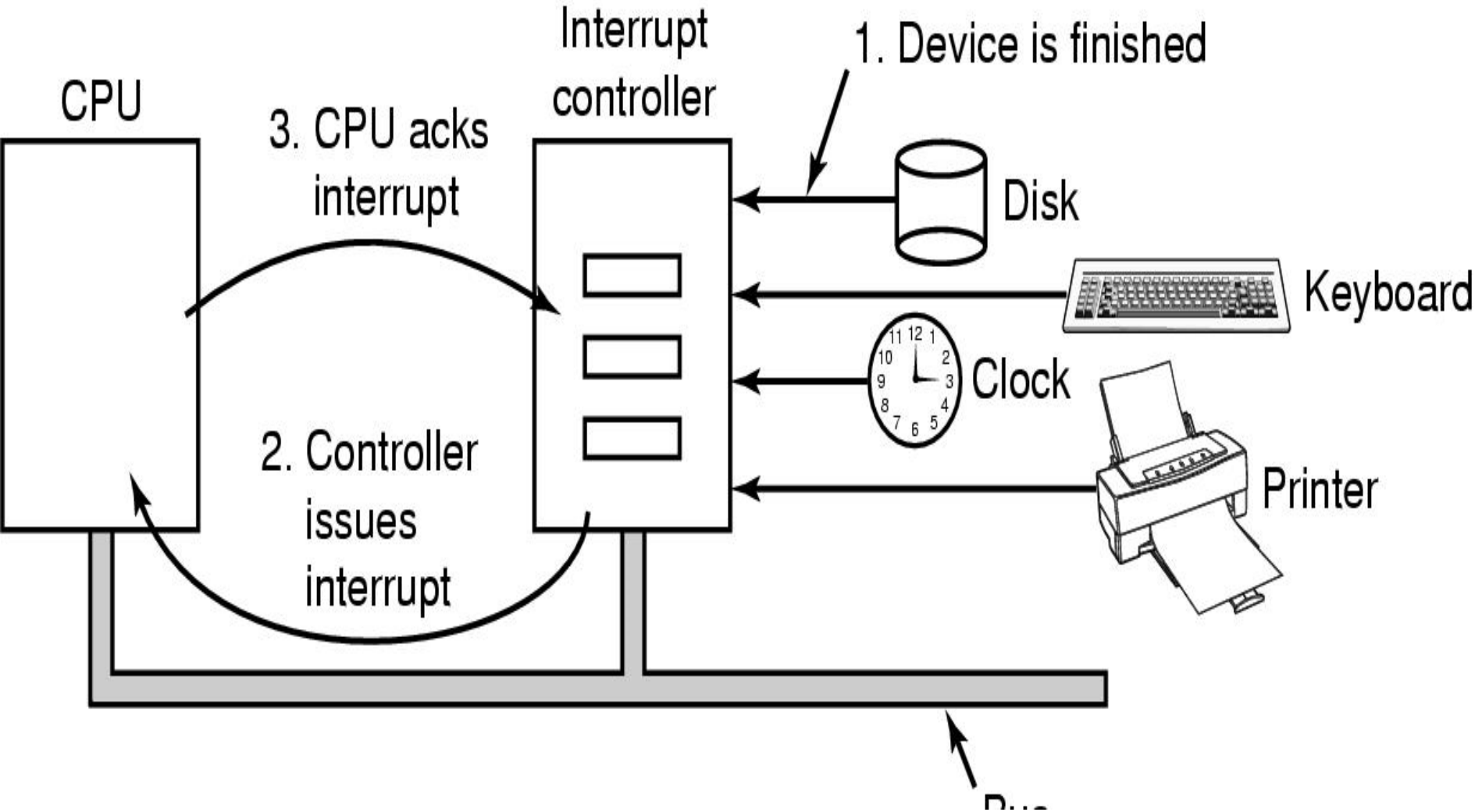
Processor interrupted only after entire I/O finished

# Programmed I/O

```
copy_from_user(buffer, p, count);          /* p is the kernel bufer */
for (i = 0; i < count; i++) {              /* loop on every character */
    while (*printer_status_reg != READY) ; /* loop until ready */
    *printer_data_register = p[i];         /* output one character */
}
return_to_user( );
```

Writing a string to the printer using programmed I/O --- busy waiting

# Interrupts Revisited



# Interrupt-Driven I/O

```
copy_from_user(buffer, p, count);
enable_interrupts( );
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler( );
```

(a)

```
if (count == 0) {
    unblock_user( );
} else {
    *printer_data_register = p[i];
    count = count – 1;
    i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

(b)

- Writing a string to the printer using interrupt-driven I/O
  - Code executed when print system call is made
  - Interrupt service procedure

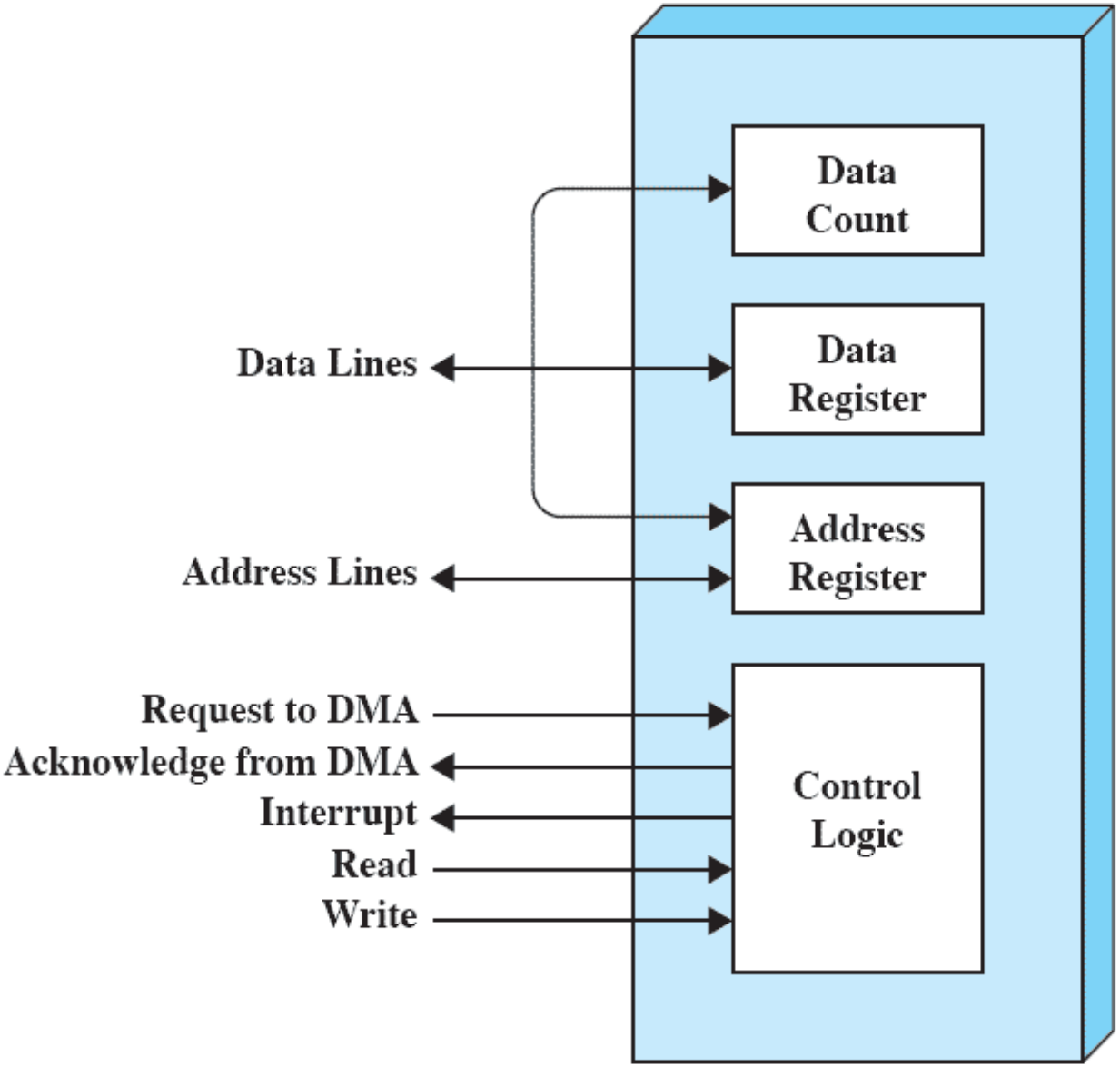
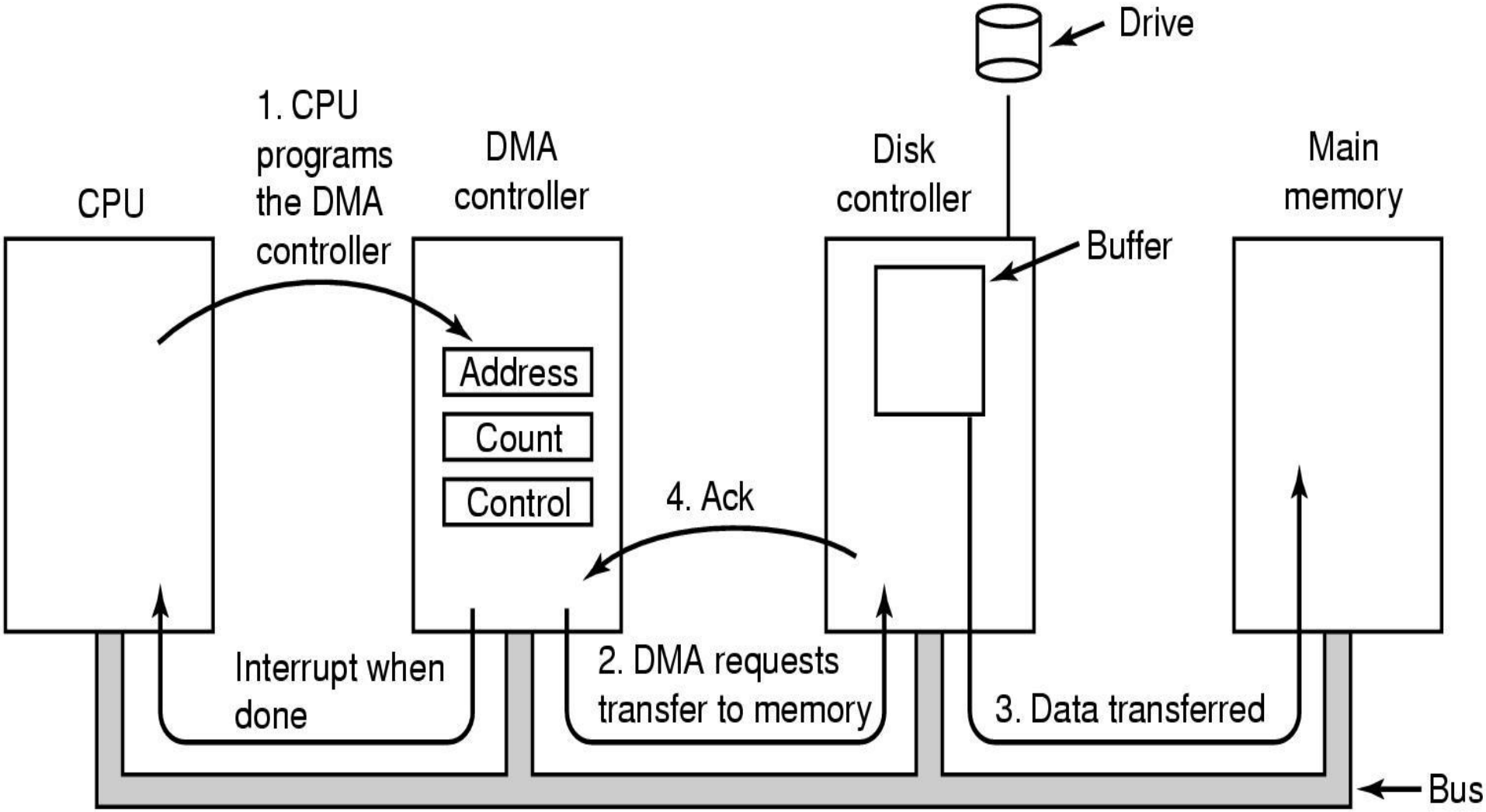
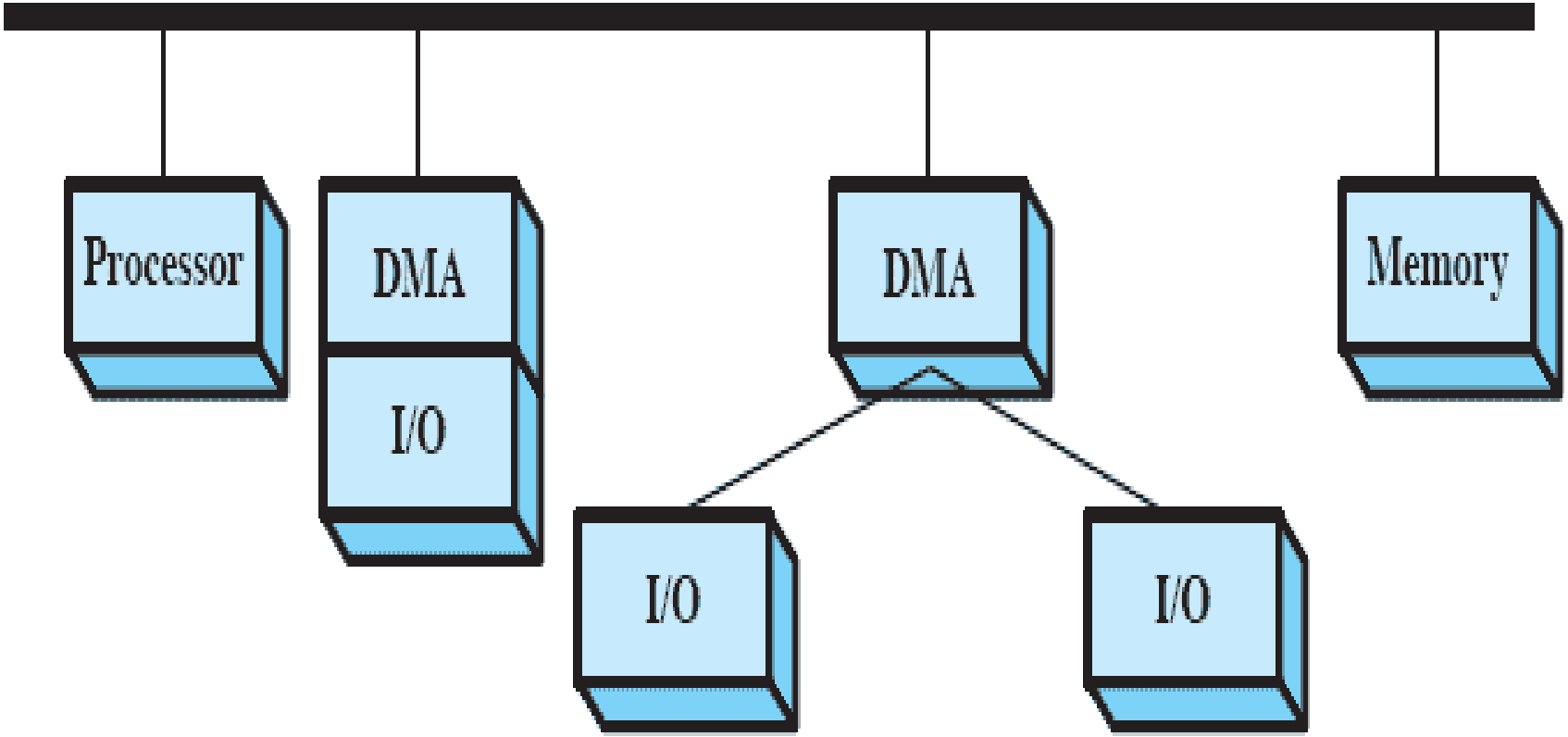



**Figure 11.2 Typical DMA Block Diagram**

# Direct Memory Access (DMA)

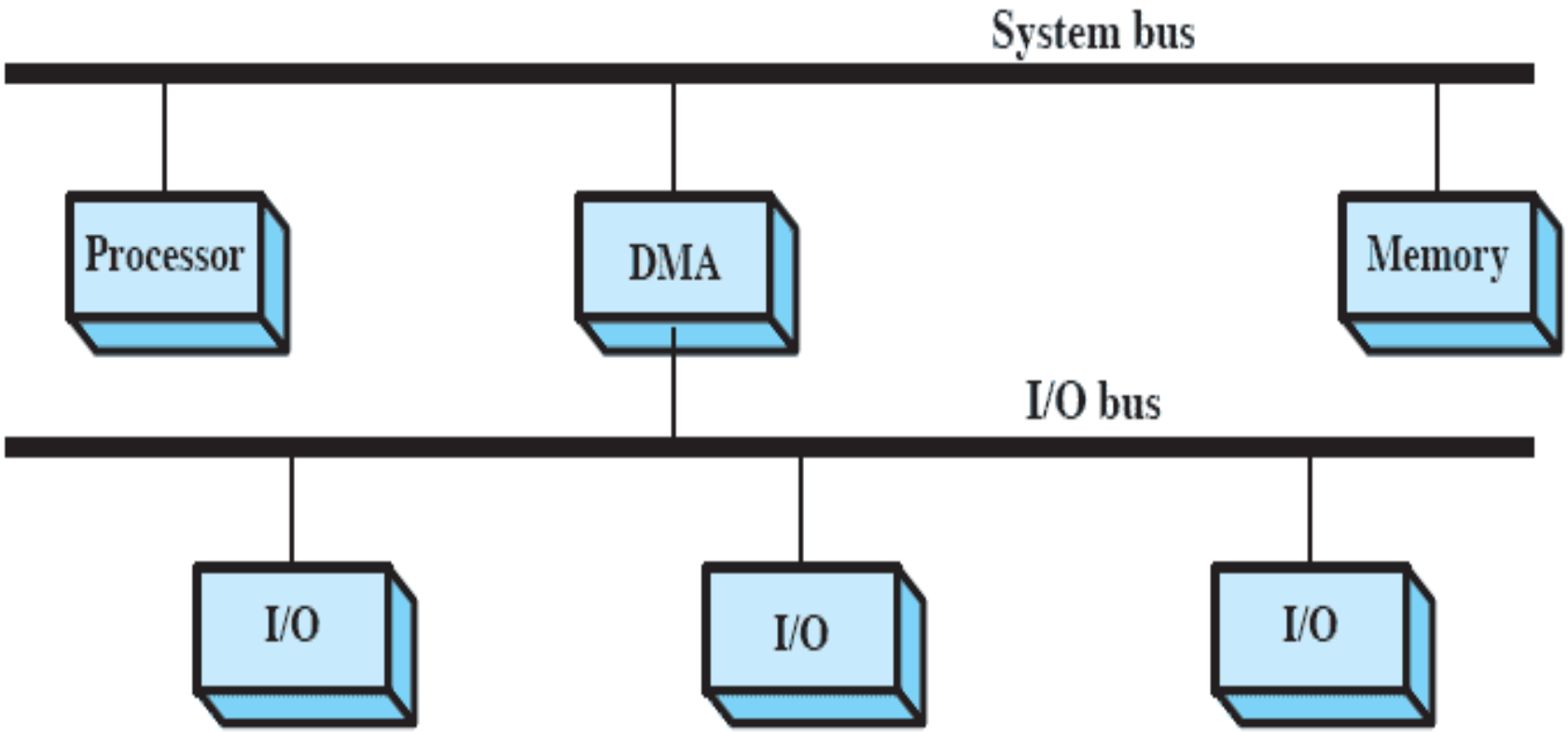


# DMA Configurations (1)




(b) Single-bus, Integrated DMA-I/O

# DMA Configurations (2)




(c) I/O bus

# I/O Using DMA

```
copy_from_user(buffer, p, count);
set_up_DMA_controller( );
scheduler( );
```

(a)

```
acknowledge_interrupt( );
unblock_user( );
return_from_interrupt( );
```

(b)

- Printing a string using DMA
  - –code executed when the print system call is made

## Table 11.1 I/O Techniques

| | No Interrupts | Use of Interrupts |
|---|---|---|
| **I/O-to-memory transfer through processor** | Programmed I/O | Interrupt-driven I/O |
| **Direct I/O-to-memory transfer** | | Direct memory access (DMA) |

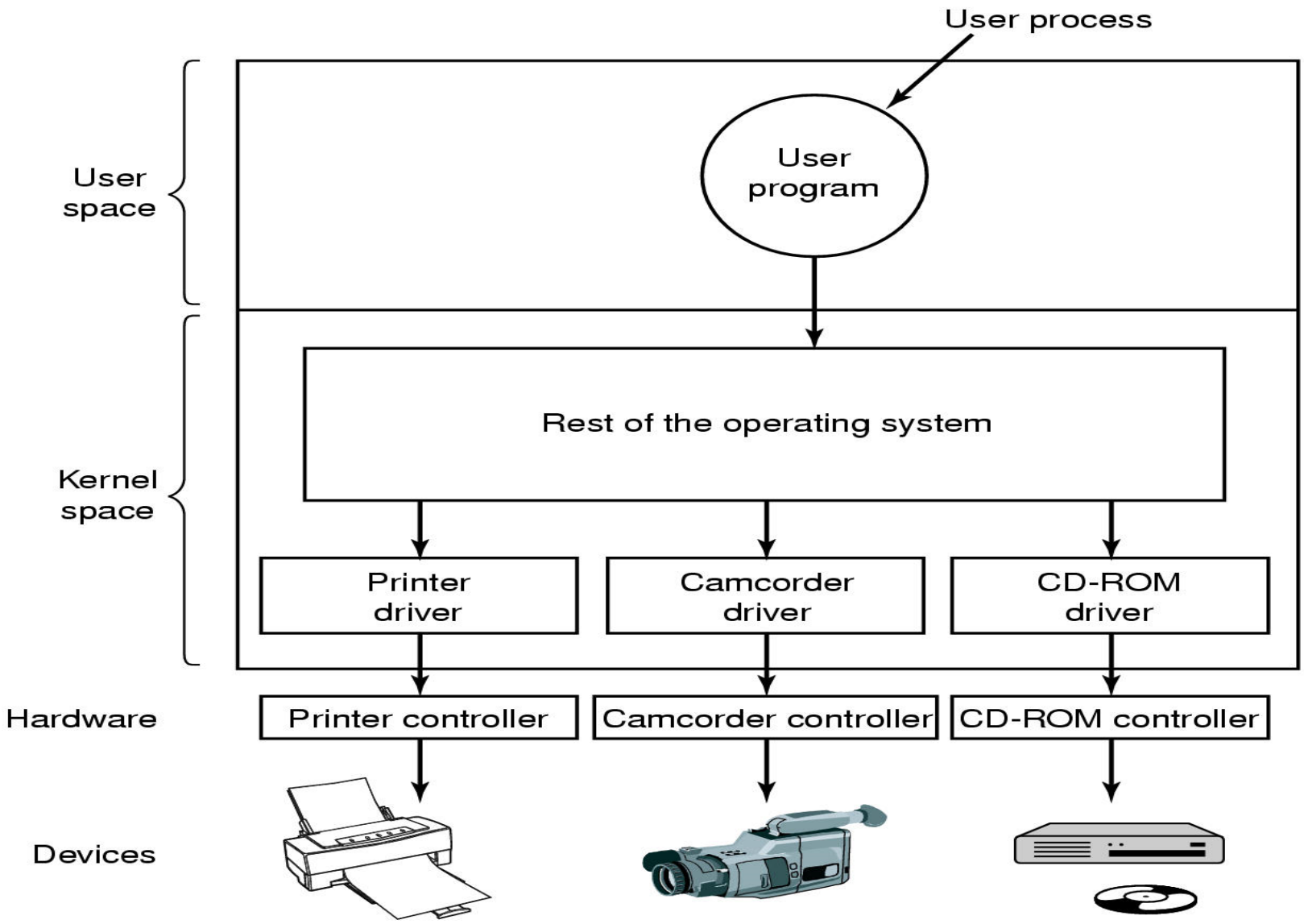
# Operating System Design Issues (1)

- Efficiency
  - Most I/O devices extremely slow compared to main memory
  - Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
  - I/O cannot keep up with processor speed
  - Swapping is used to bring in additional Ready processes which is an I/O operation

# Operating System Design Issues (2)

- Generality
  - Desirable to handle all I/O devices in a uniform manner
  - Hide most of the details of device I/O in lower-level routines

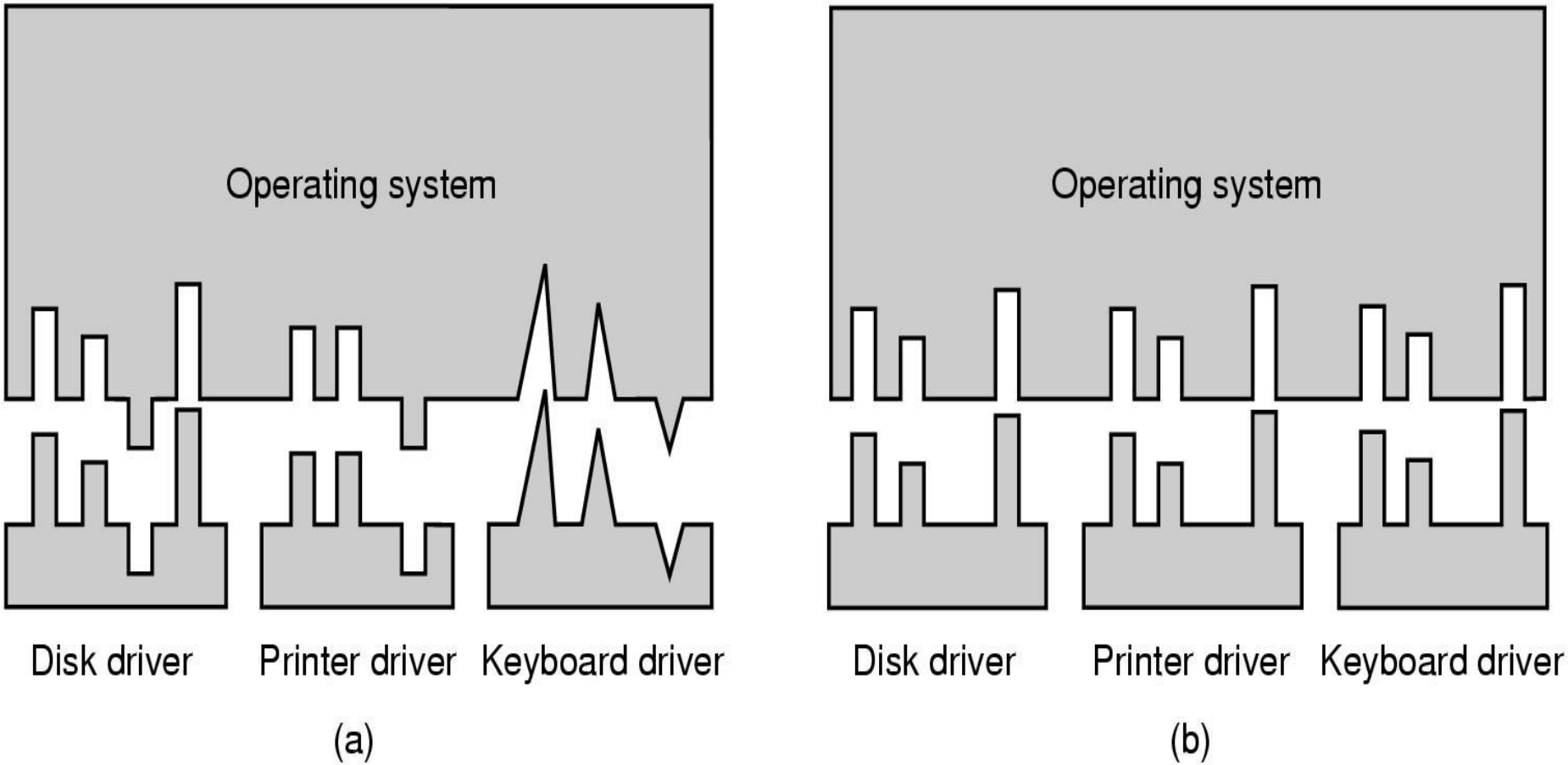
# Device Drivers



# Tasks of Device Drivers

- Accept abstract requests
- Check input parameters
- Translate from abstract to concrete
- Check if device is in use
- Issue commands to controller
- (Block)
- Check errors
- Return (error) to caller

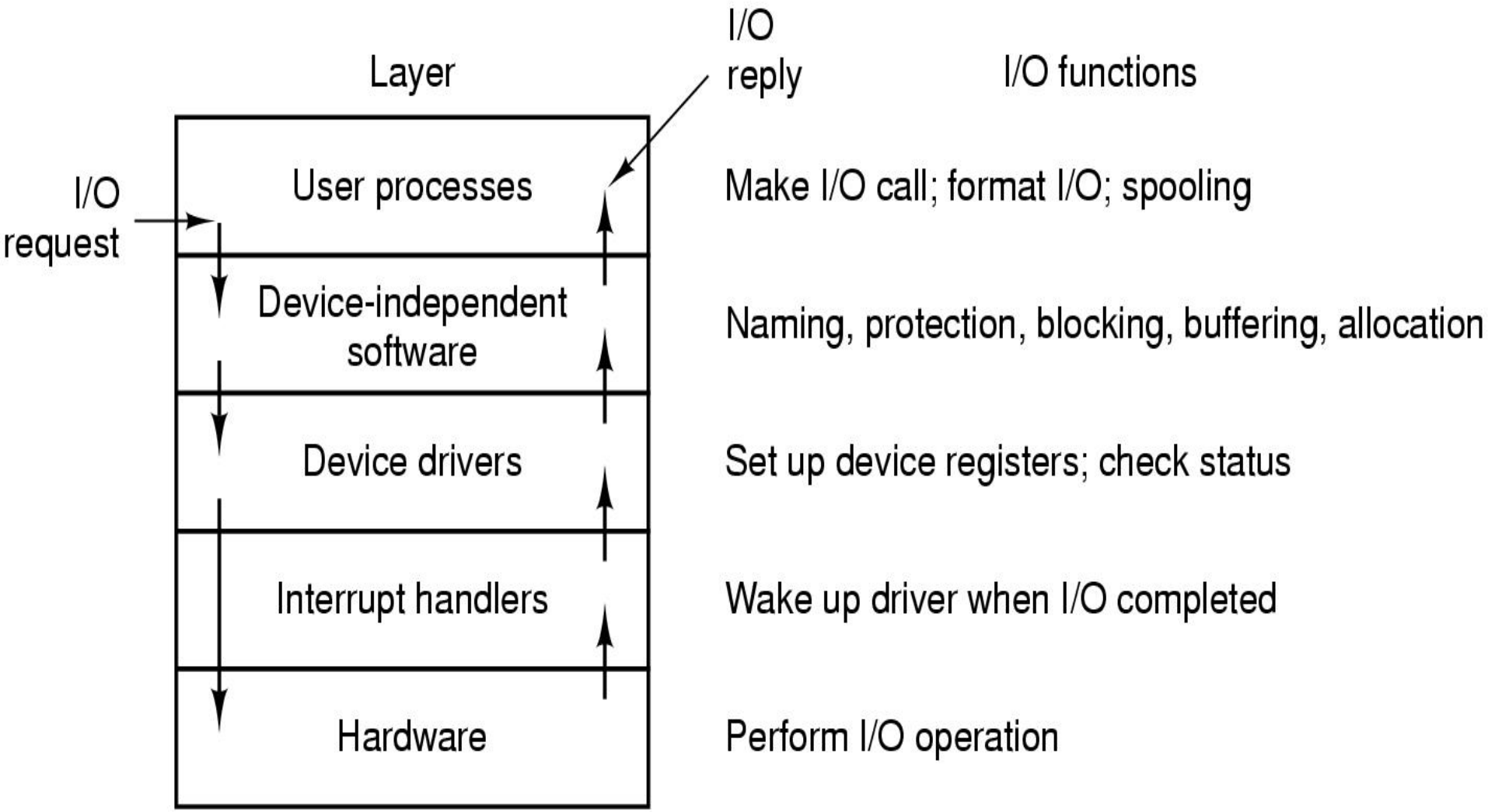
# Device-Independent I/O Software



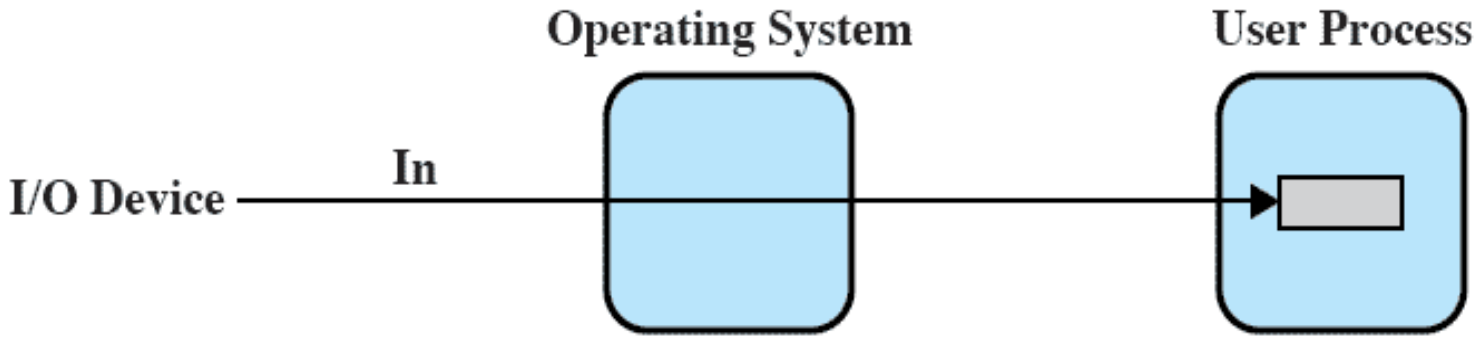

(a) Without a standard driver interface

(b) With a standard driver interface

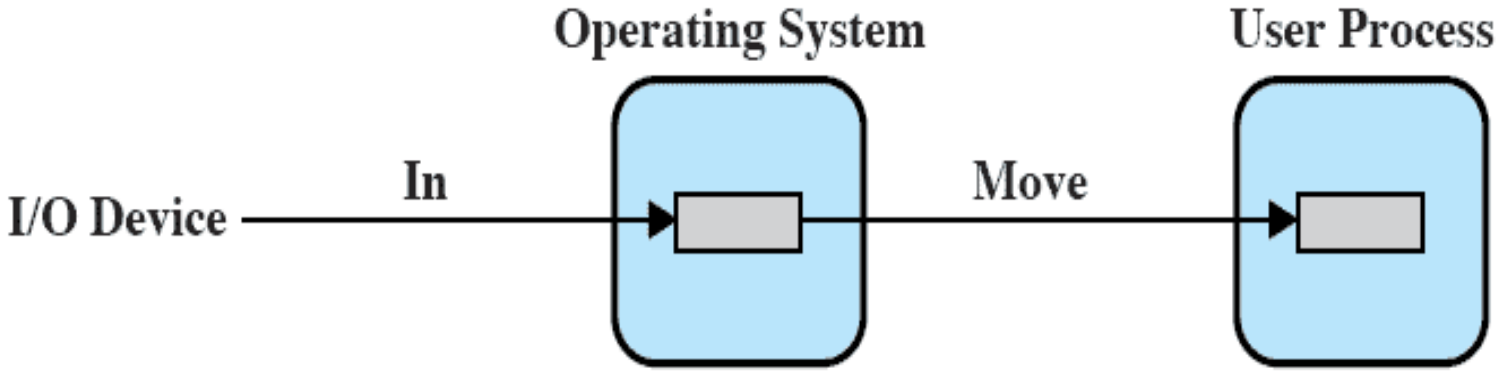
# I/O Software Layers



# With or without Buffering




(a) No buffering




(b) Single buffering

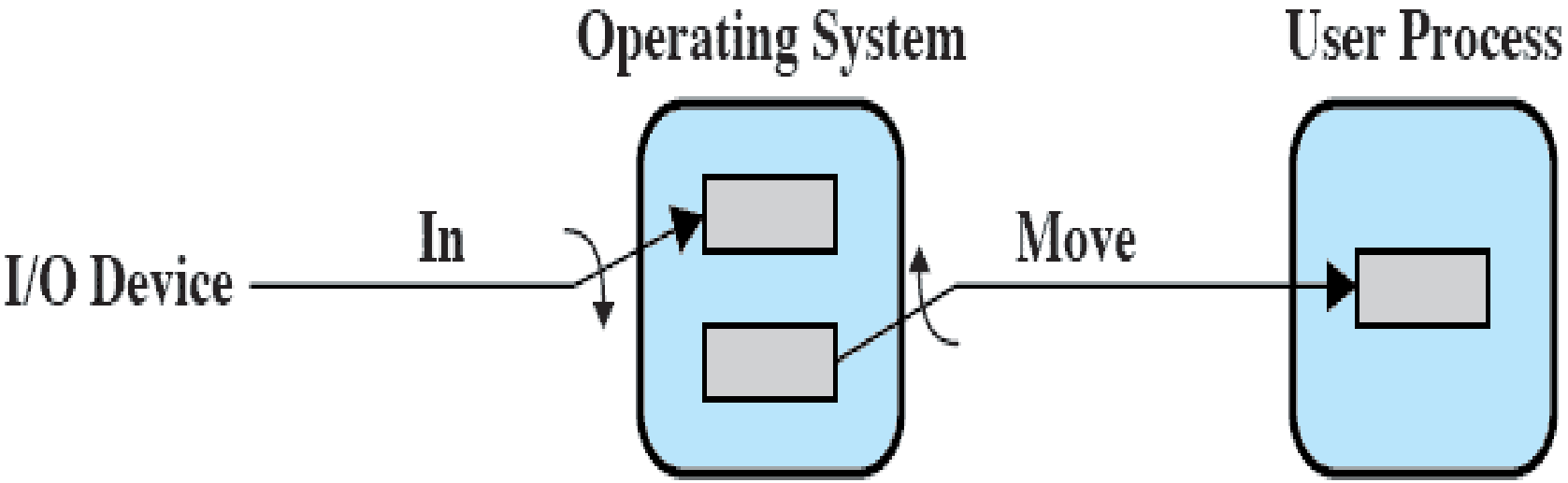
# Single Buffer

- Operating system assigns a buffer in main memory for an I/O request
  - Input transfers made to buffer
  - Data moved to user space when needed
  - Extra data is moved into the buffer
  - User process can process data while additional data is read in
  - Swapping can occur since input is taking place in system memory, not user memory
  - Operating system keeps track of assignment of system buffers to user processes

# Double Buffer

- Use two system buffers instead of one

A process can transfer data to or from one buffer while the operating system empties or fills the other buffer

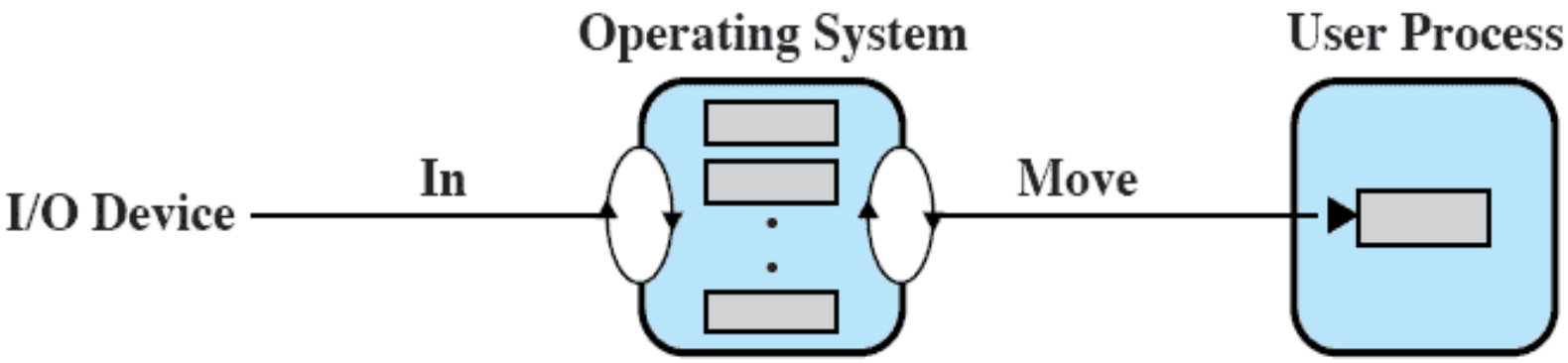



(c) Double buffering

# Circular Buffer

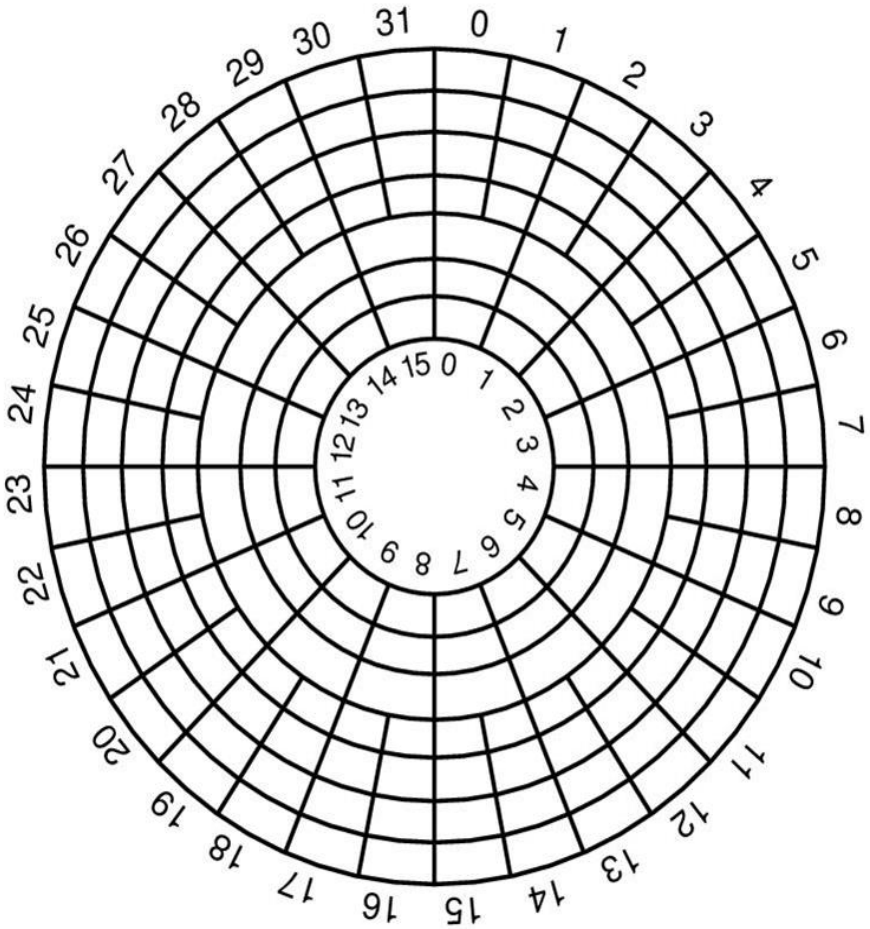
- More than two buffers are used
- Each individual buffer is one unit in a circular buffer

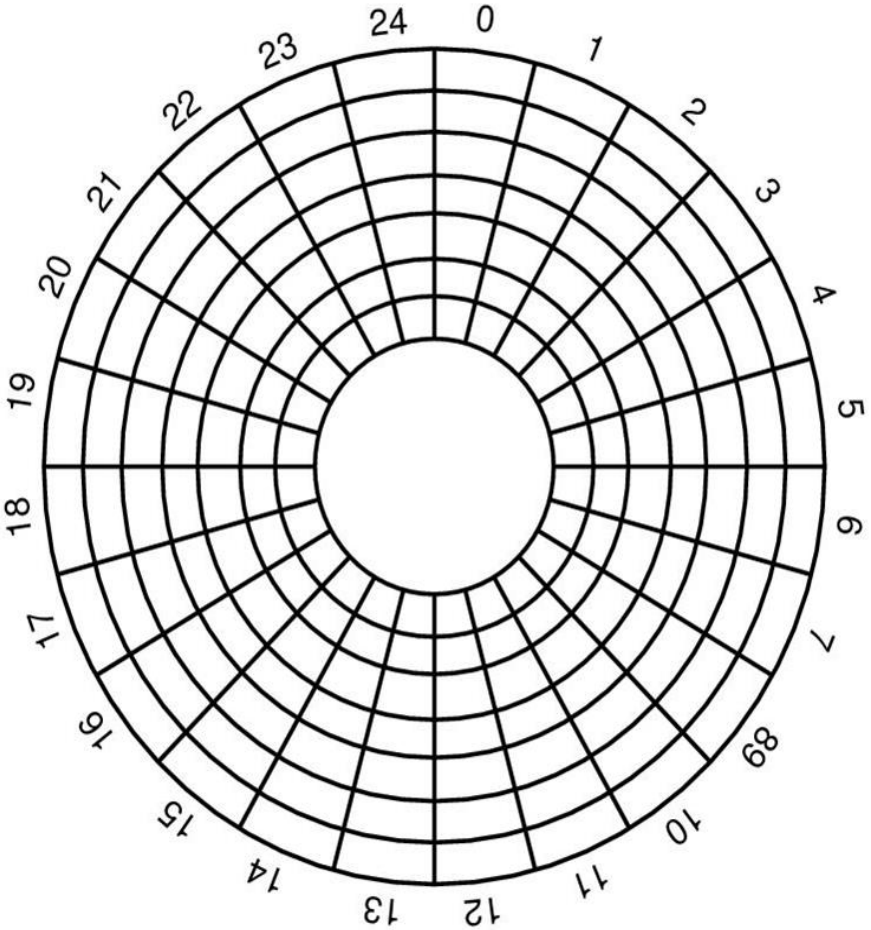
Used when I/O operation must keep up with process




(d) Circular buffering

# Disk Hardware




- Physical geometry of a disk with two zones
- A possible virtual geometry for this disk

# Disk Performance Parameters (1)

- To read or write, the disk head must be positioned at the desired track and at the beginning of the desired sector
- Seek time
  - Time it takes to position the head at the desired track
- Rotational delay or rotational latency
  - Time it takes for the beginning of the sector to reach the head

# Disk Performance Parameters (2)

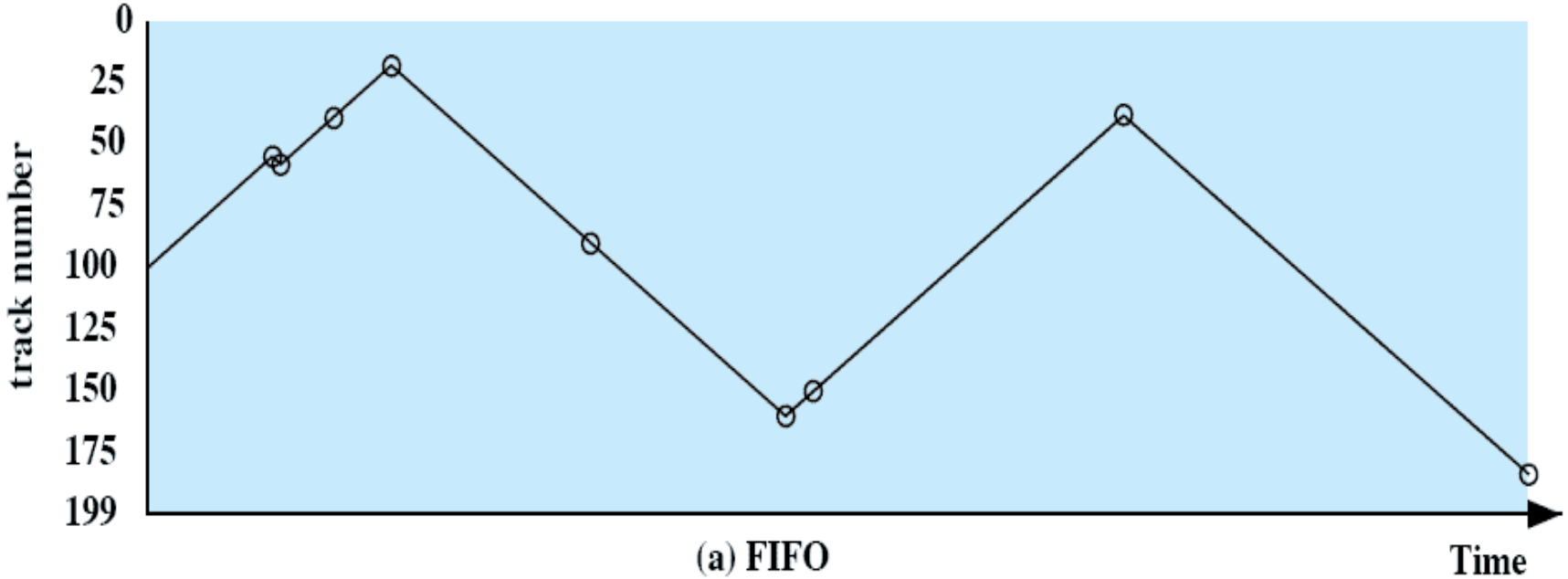
- Access time
  - Sum of seek time and rotational delay
  - The time it takes to get in position to read or write
- Data transfer occurs as the sector moves under the head

# Disk Scheduling Policies

- Seek time is the reason for differences in performance
- For a single disk there will be a number of I/O requests
- If requests are selected randomly, get poor performance

# Disk Scheduling Policies - FIFO

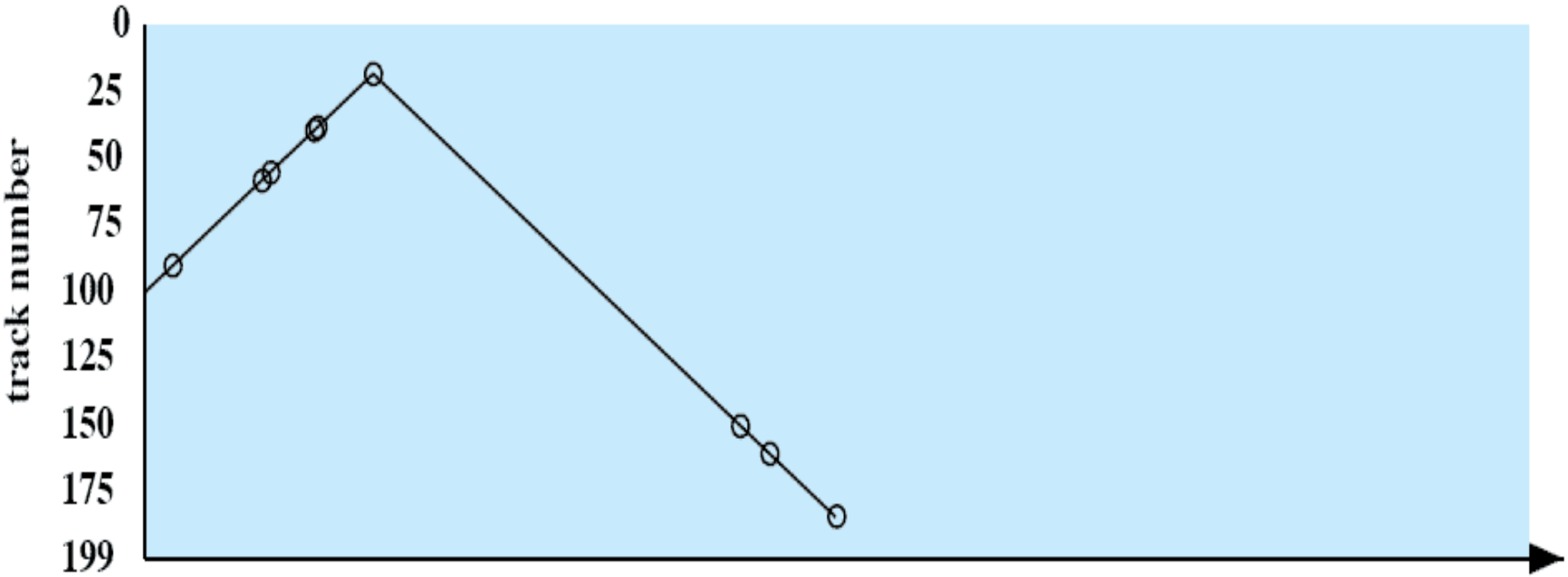
- First-in, first-out (FIFO)
  - Process request sequentially
  - Fair to all processes
- Approaches random scheduling in performance if there are many processes



(a) FIFO

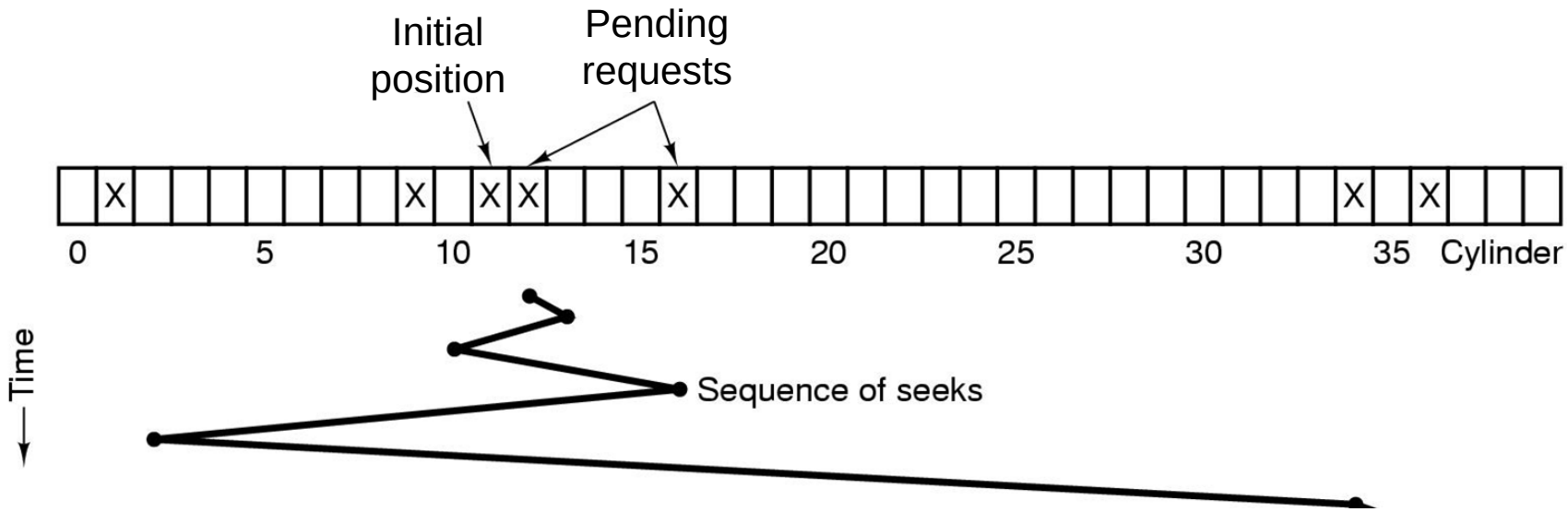
# Disk Scheduling Policies - SSTF

- Shortest Service/Seek Time First
  - Select the disk I/O request that requires the least movement of the disk arm from its current position
  - Always choose the minimum seek time




(b) SSTF

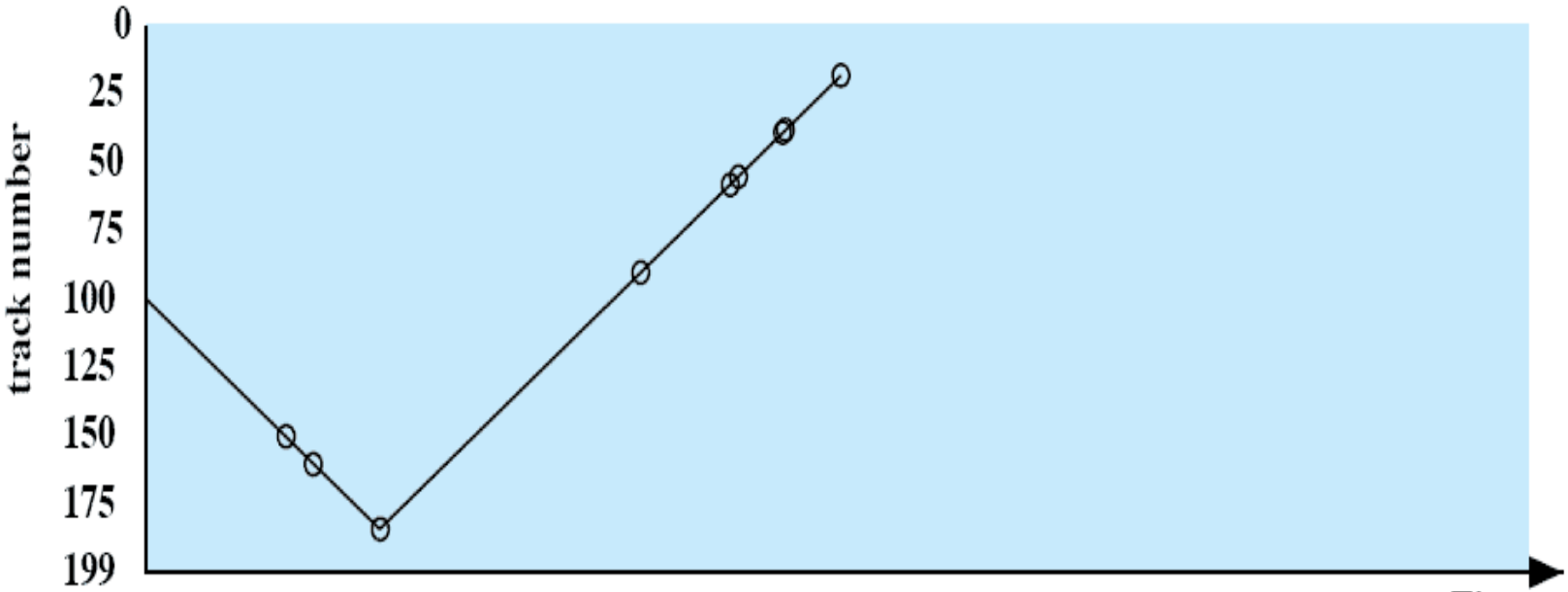
# SSF (2)




Shortest Seek First (SSF) disk scheduling algorithm

# Disk Scheduling Policies - SCAN

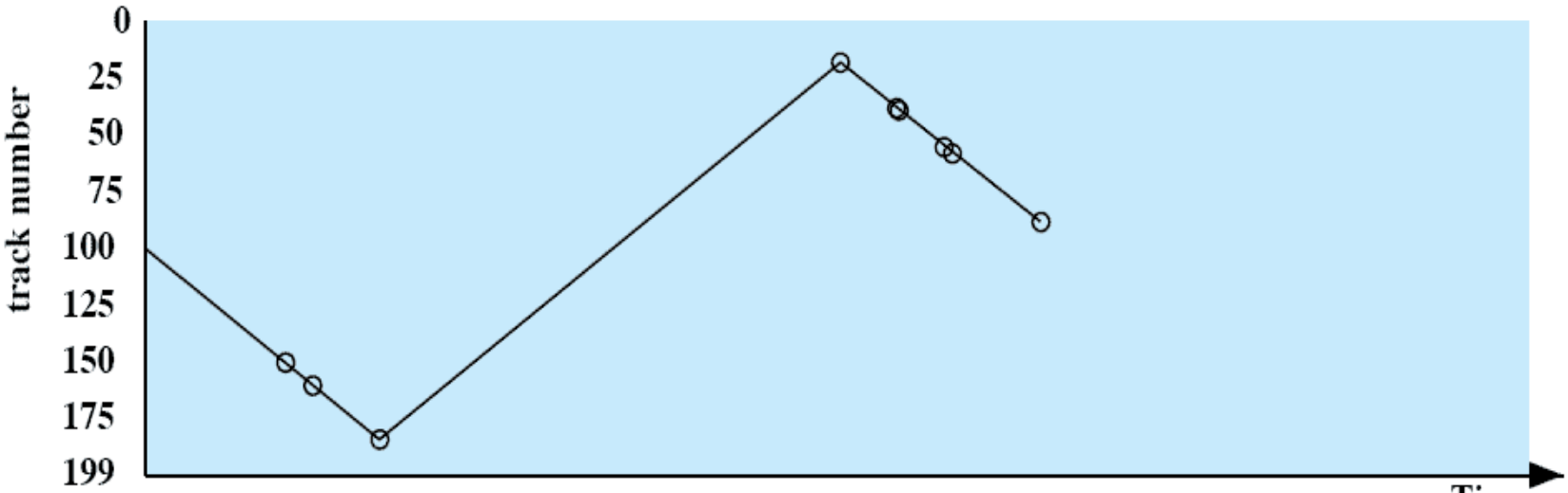
- SCAN or Elevator
  - Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction
- Direction is reversed



(c) SCAN

# Disk Scheduling Policies – C-SCAN

- C-SCAN
  - Restricts scanning to one direction only
  - When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again



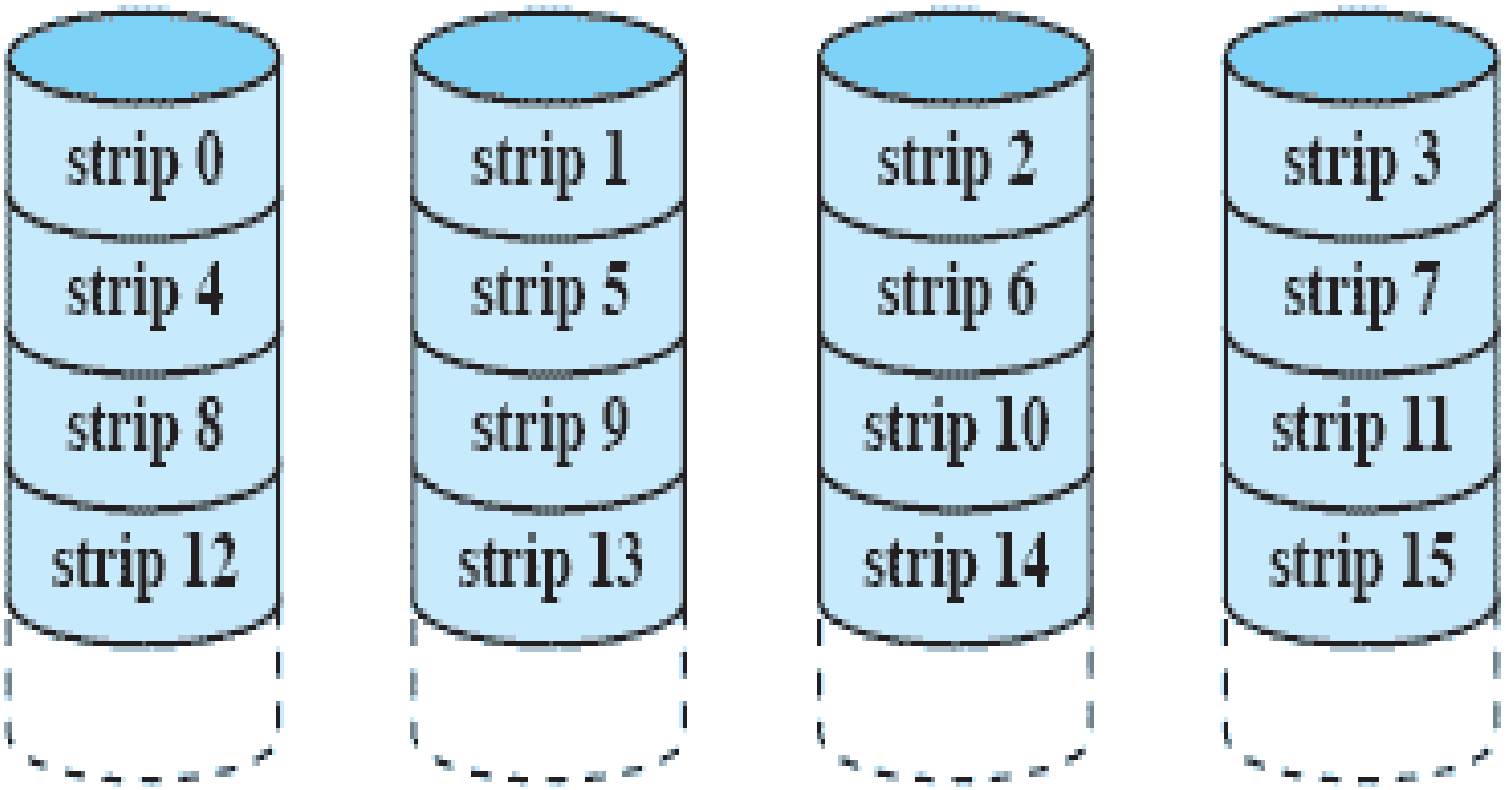

(d) C-SCAN

| (a) FIFO (starting at track 100) | | (b) SSTF (starting at track 100) | | (c) SCAN (starting at track 100, in the direction of increasing track number) | | (d) C-SCAN (starting at track 100, in the direction of increasing track number) | |
|---|---|---|---|---|---|---|---|
| **Next track accessed** | **Number of tracks traversed** | **Next track accessed** | **Number of tracks traversed** | **Next track accessed** | **Number of tracks traversed** | **Next track accessed** | **Number of tracks traversed** |
| 55 | 45 | 90 | 10 | 150 | 50 | 150 | 50 |
| 58 | 3 | 58 | 32 | 160 | 10 | 160 | 10 |
| 39 | 19 | 55 | 3 | 184 | 24 | 184 | 24 |
| 18 | 21 | 39 | 16 | 90 | 94 | 18 | 166 |
| 90 | 72 | 38 | 1 | 58 | 32 | 38 | 20 |
| 160 | 70 | 18 | 20 | 55 | 3 | 39 | 1 |
| 150 | 10 | 150 | 132 | 39 | 16 | 55 | 16 |
| 38 | 112 | 160 | 10 | 38 | 1 | 58 | 3 |
| 184 | 146 | 184 | 24 | 18 | 20 | 90 | 32 |
| **Average seek length** | 55.3 | **Average seek length** | 27.5 | **Average seek length** | 27.8 | **Average seek length** | 35.8 |

# RAID

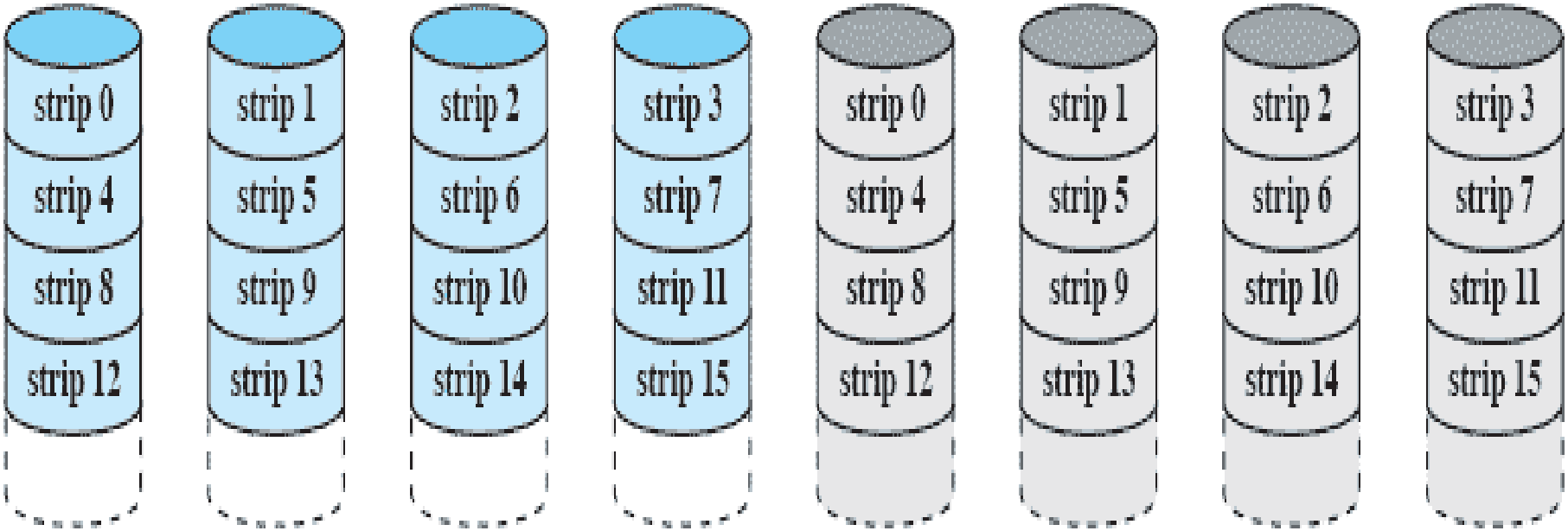
- Redundant Array of Independent Disks
- Set of physical disk drives viewed by the operating system as a single logical drive
- Data are distributed across the physical drives of an array
- Redundant disk capacity is used to store parity information

# RAID 0 (non-redundant)



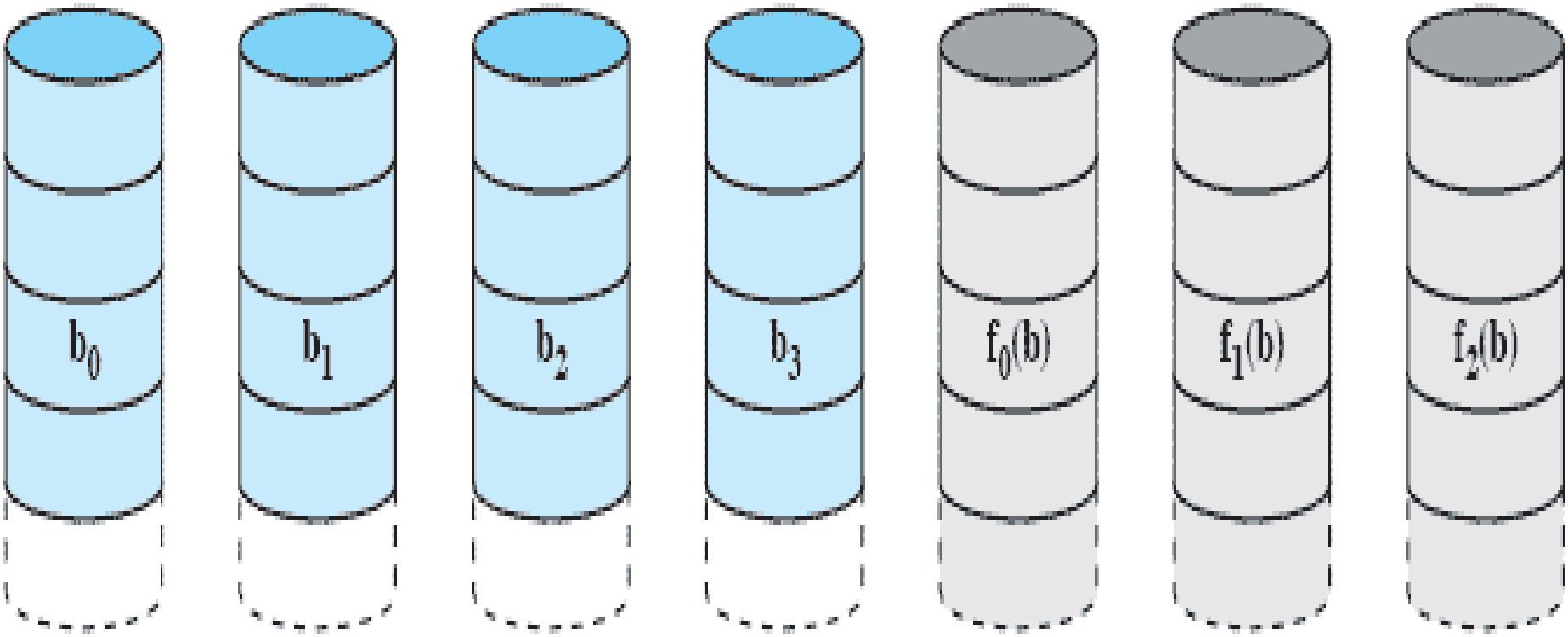

(a) RAID 0 (non-redundant)

# RAID 1 (mirrored)



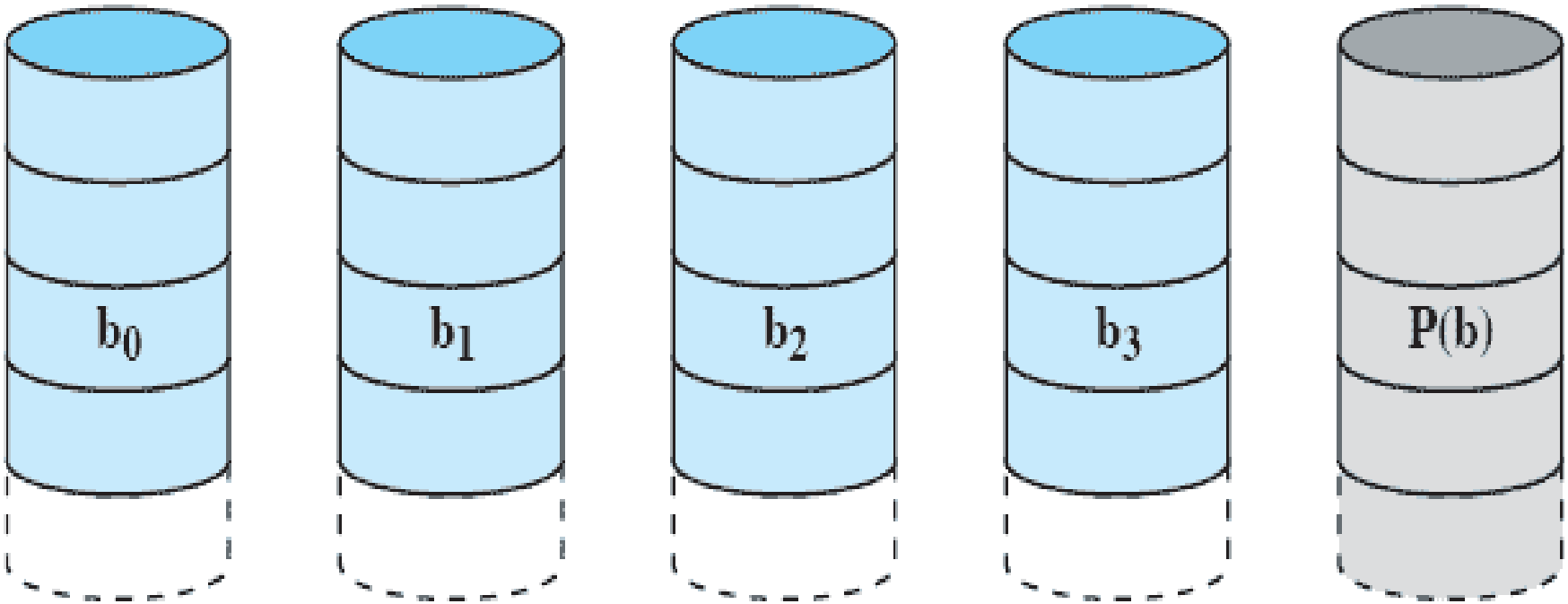

(b) RAID 1 (mirrored)

# RAID 2 (Hamming code)



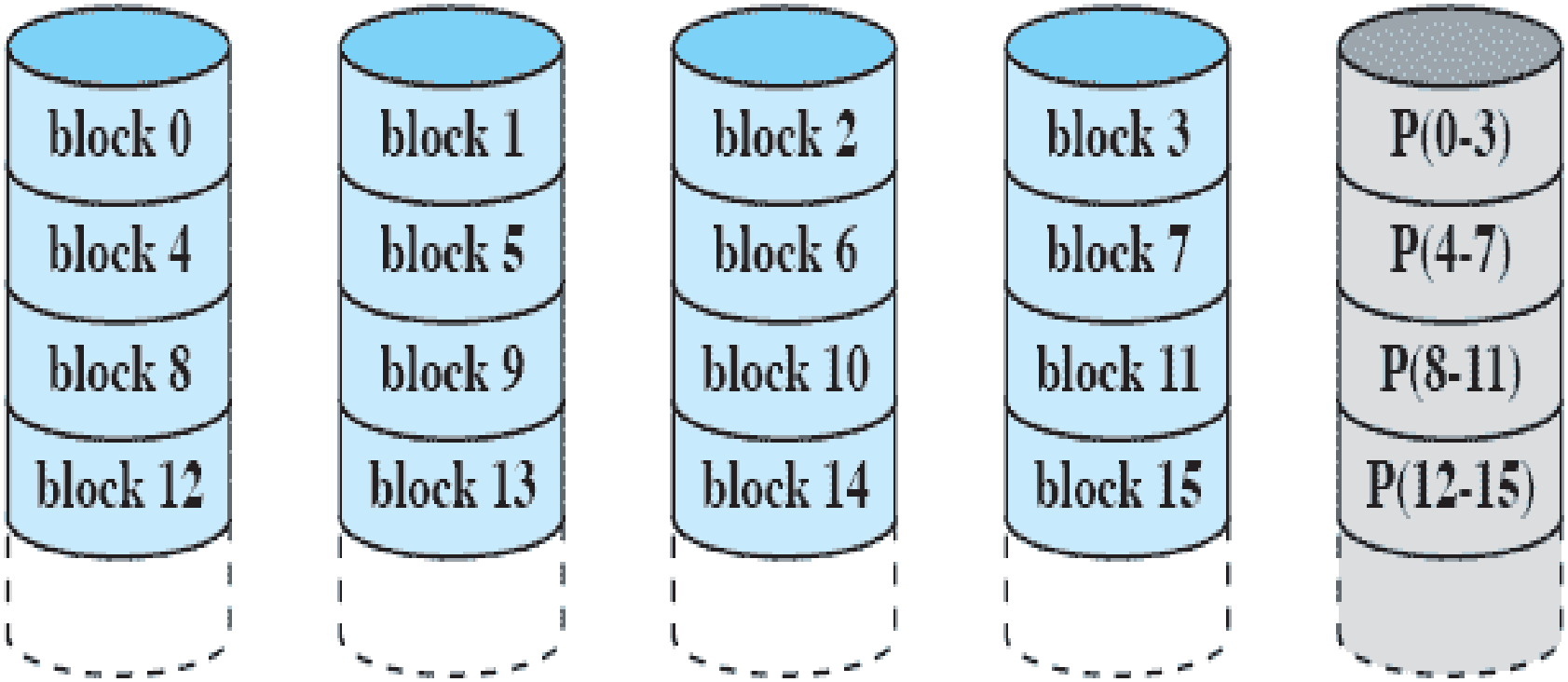

(c) RAID 2 (redundancy through Hamming code)

# RAID 3 (bit-interleaved parity)



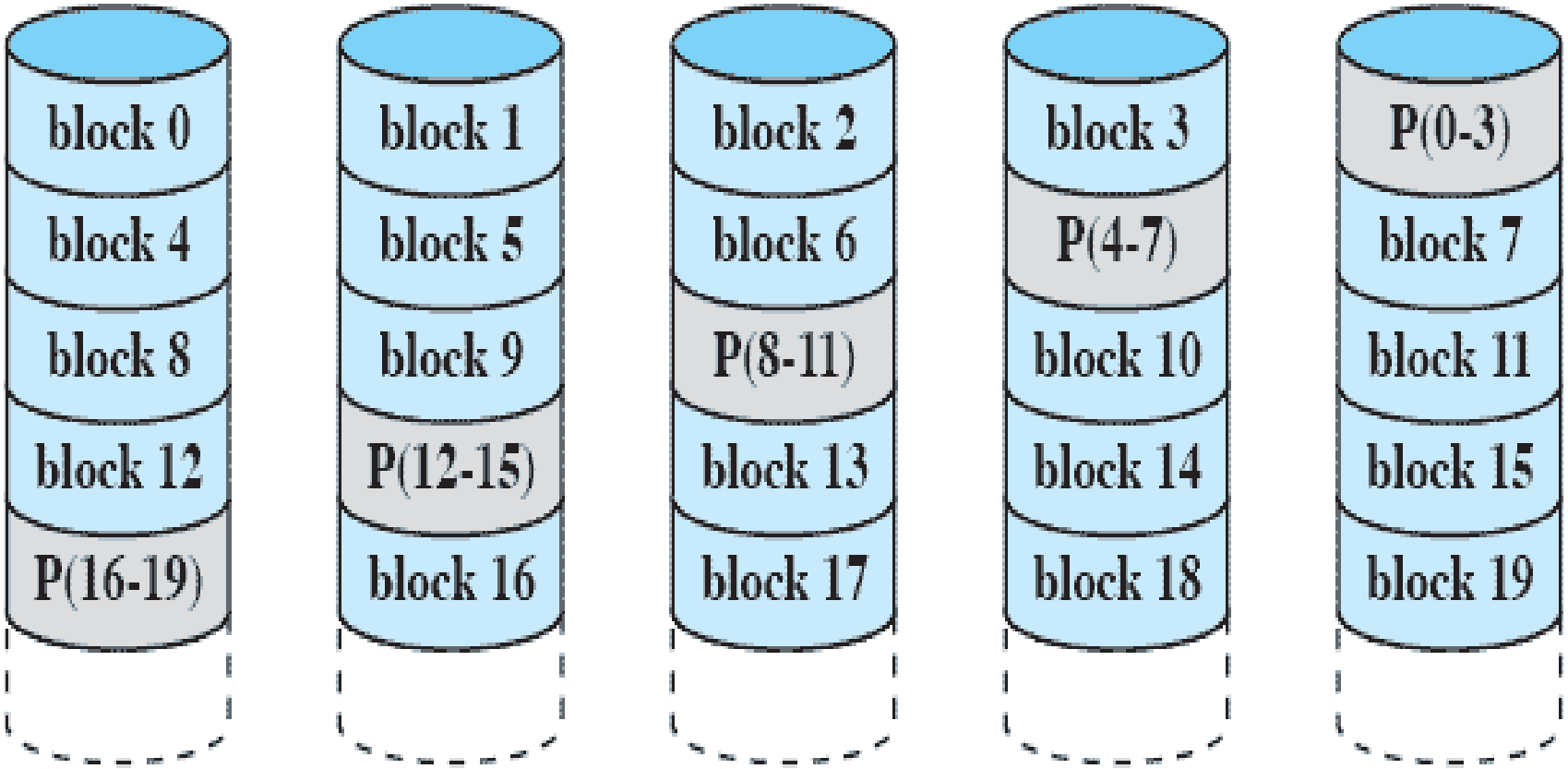

(d) RAID 3 (bit-interleaved parity)

# RAID 4 (block-level parity)



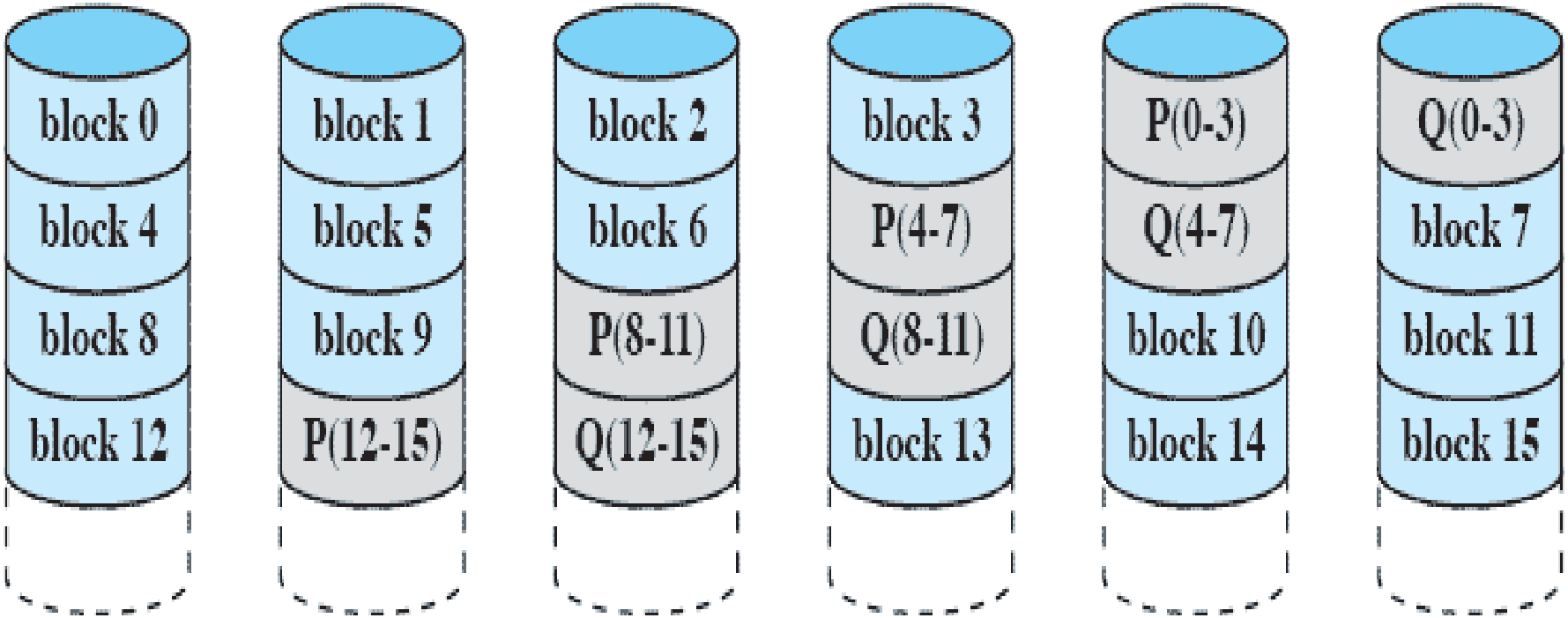

(e) RAID 4 (block-level parity)

# RAID 5 (block-level distributed parity)




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

# RAID 6 (dual redundancy)




(g) RAID 6 (dual redundancy)

# Disk Cache

- Buffer in main memory for disk sectors
- Contains a copy of some of the sectors on the disk

# Least Recently Used (1)

- The block that has been in the cache the longest with no reference to it is replaced
- The cache consists of a stack of blocks
- Most recently referenced block is on the top of the stack
- When a block is referenced or brought into the cache, it is placed on the top of the stack

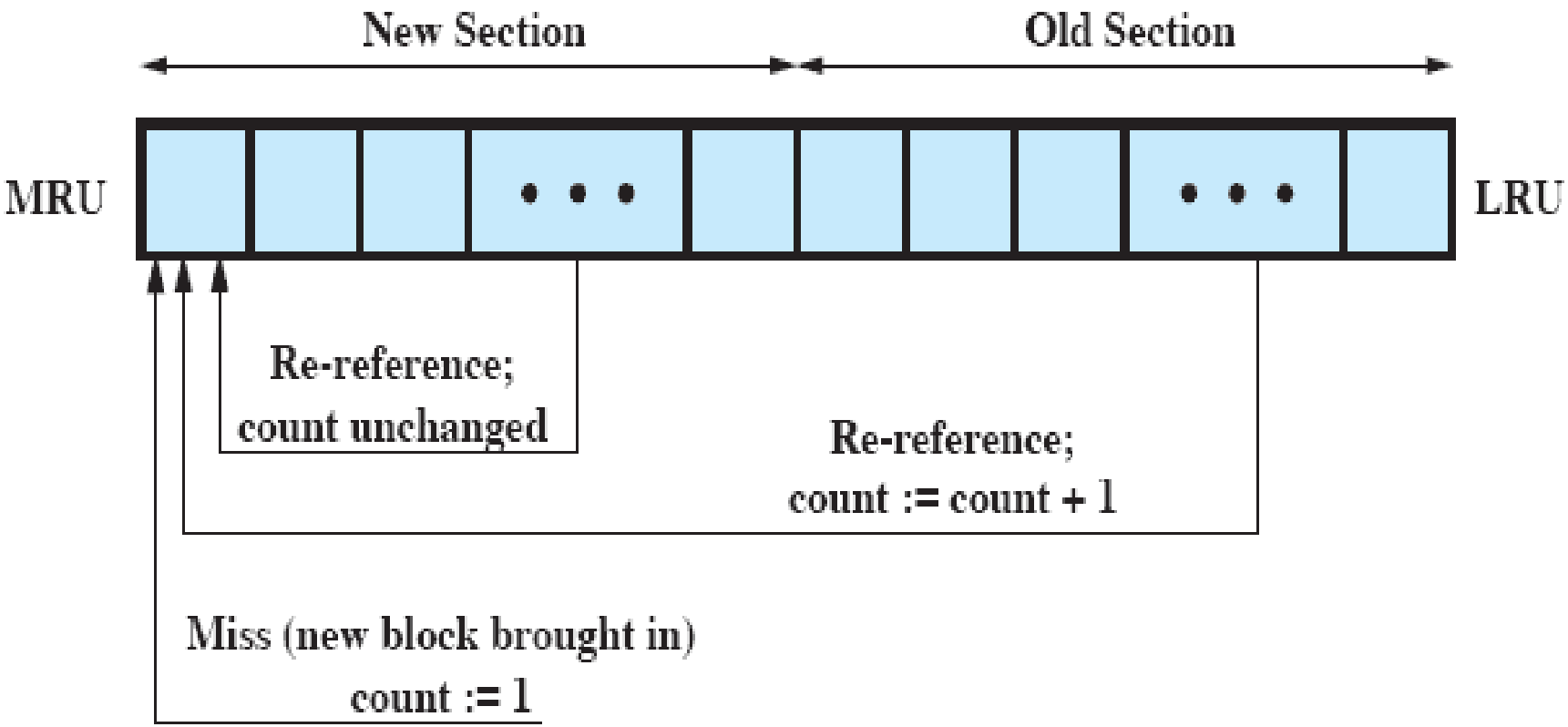
# Least Recently Used (2)

- The block on the bottom of the stack is removed when a new block is brought in
- Blocks don’t actually move around in main memory
- A stack of pointers is used

# Least Frequently Used

- The block that has experienced the fewest references is replaced
- A counter is associated with each block
- Counter is incremented each time block accessed
- Block with smallest count is selected for replacement
- Some blocks may be referenced many times in a short period of time and the reference count is misleading

# Frequency-Based Replacement




(a) FIFO