Chapter 1:
Data Storage

Computer Science: An Overview
Tenth Edition

by
J. Glenn Brookshear

Presentation files modified by Farn Wang

Chapter 1: Data Storage

- 1.1 Bits and Their Storage
- 1.2 Main Memory
- 1.3 Mass Storage
- 1.4 Representing Information as Bit Patterns
- 1.5 The Binary System
- 1.6 Storing Integers
- 1.7 Storing Fractions
- 1.8 Data Compression
- 1.9 Communications Errors

Bits and Bit Patterns

- **Bit:** Binary Digit (0 or 1)
- Bit Patterns are used to represent information.
  - Numbers
  - Text characters
  - Images
  - Sound
  - And others

Boolean Operations

- **Boolean Operation:** An operation that manipulates one or more true/false values
- Specific operations
  - AND
  - OR
  - XOR (exclusive or)
  - NOT

The Boolean operations AND, OR, and XOR (exclusive or)

The AND operation

$$
\begin{array}{r}
0 \\
\text{AND}\quad 0 \\
\hline
0
\end{array}
\qquad
\begin{array}{r}
0 \\
\text{AND}\quad 1 \\
\hline
0
\end{array}
\qquad
\begin{array}{r}
1 \\
\text{AND}\quad 0 \\
\hline
0
\end{array}
\qquad
\begin{array}{r}
1 \\
\text{AND}\quad 1 \\
\hline
1
\end{array}
$$

The OR operation

$$
\begin{array}{r}
0 \\
\text{OR}\quad 0 \\
\hline
0
\end{array}
\qquad
\begin{array}{r}
0 \\
\text{OR}\quad 1 \\
\hline
1
\end{array}
\qquad
\begin{array}{r}
1 \\
\text{OR}\quad 0 \\
\hline
1
\end{array}
\qquad
\begin{array}{r}
1 \\
\text{OR}\quad 1 \\
\hline
1
\end{array}
$$

The XOR operation

$$
\begin{array}{r}
0 \\
\text{XOR}\quad 0 \\
\hline
0
\end{array}
\qquad
\begin{array}{r}
0 \\
\text{XOR}\quad 1 \\
\hline
1
\end{array}
\qquad
\begin{array}{r}
1 \\
\text{XOR}\quad 0 \\
\hline
1
\end{array}
\qquad
\begin{array}{r}
1 \\
\text{XOR}\quad 1 \\
\hline
0
\end{array}
$$

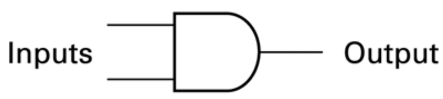
Gates

A device that computes a Boolean operation

- Often implemented as (small) electronic circuits
- Provide the building blocks from which computers are constructed
- VLSI (Very Large Scale Integration)

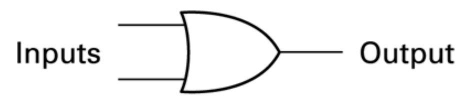
A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values

AND



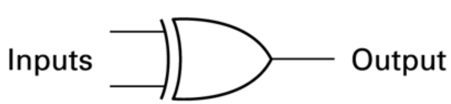
Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

OR



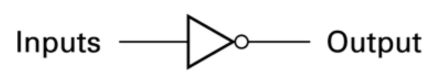
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR



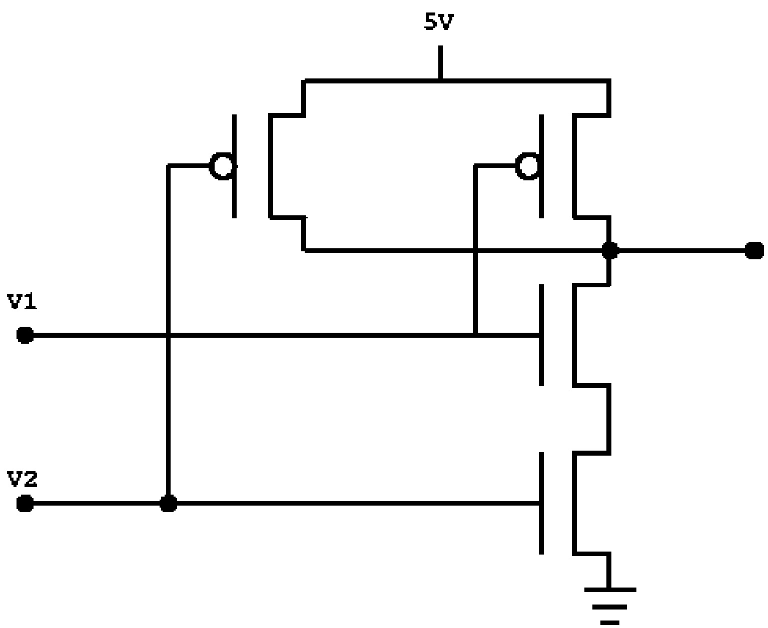
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

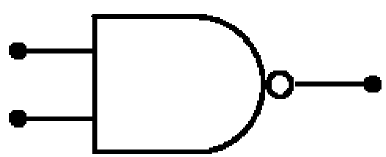
NOT

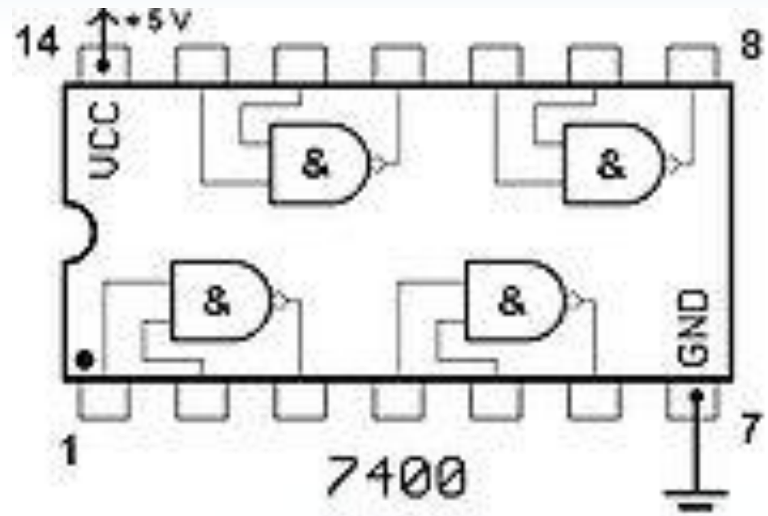


Inputs	Output
0	1
1	0

A CMOS NAND and a TTL NAND IC





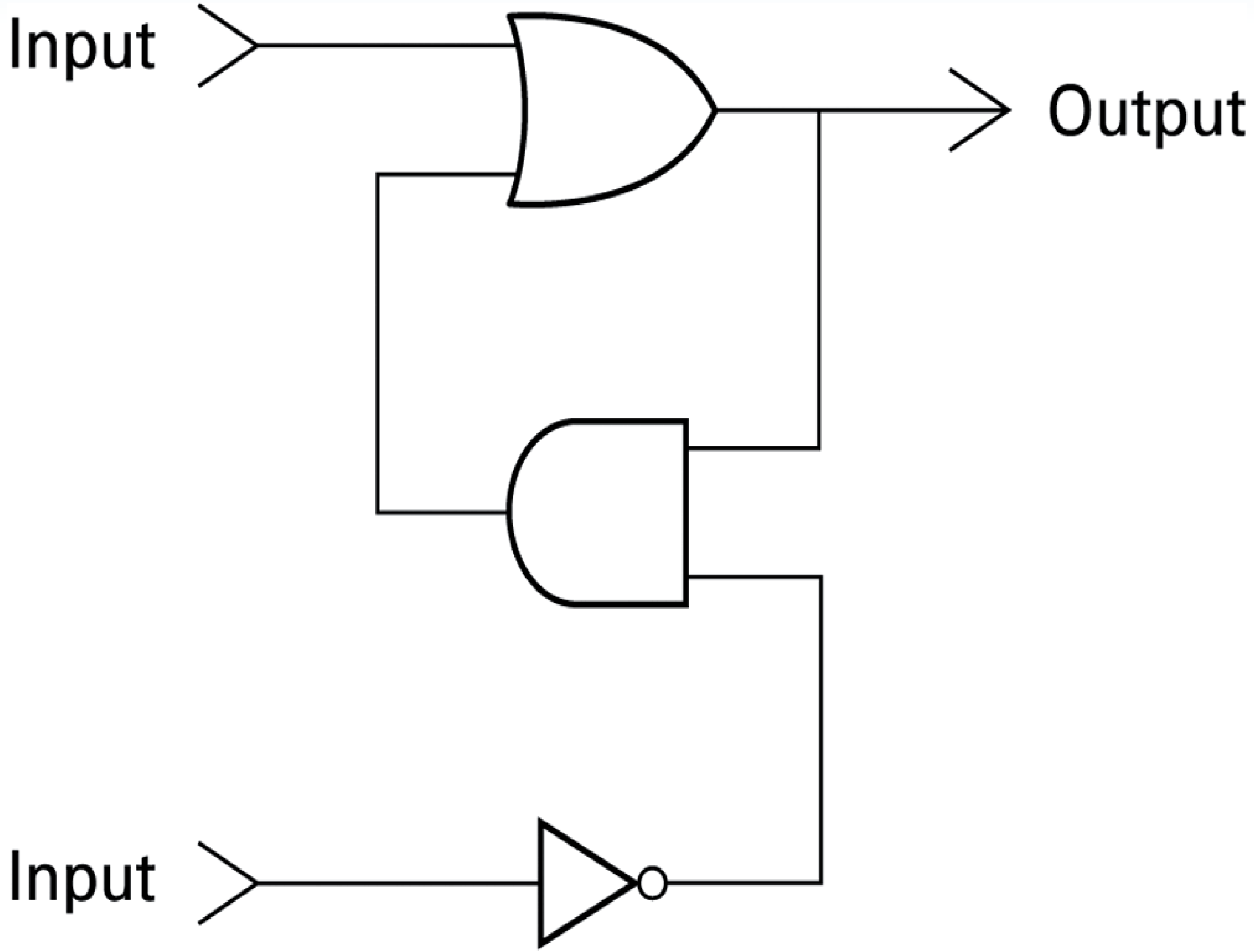


Flip-flops

A circuit built from gates that can store one bit.

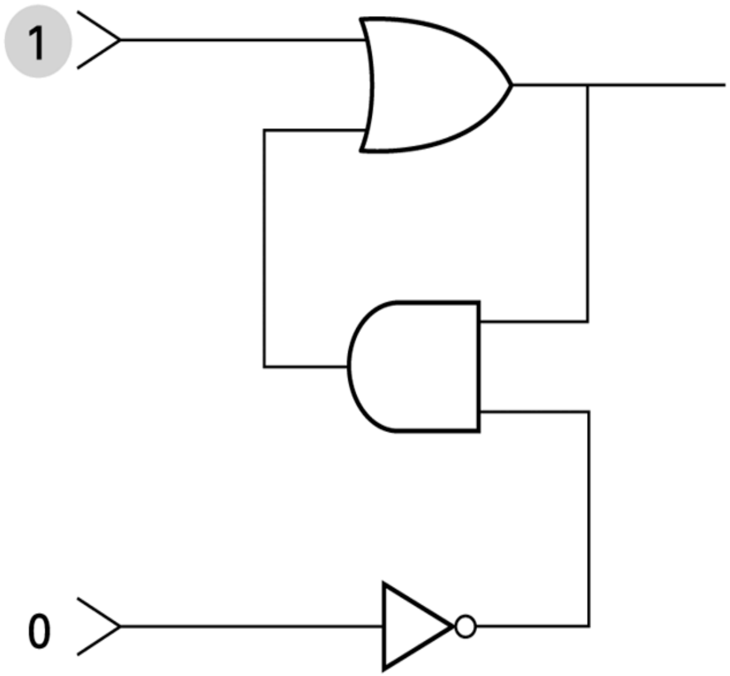
- One input line is used to set its stored value to 1
- One input line is used to set its stored value to 0
- While both input lines are 0, the most recently stored value is preserved

A simple flip-flop circuit



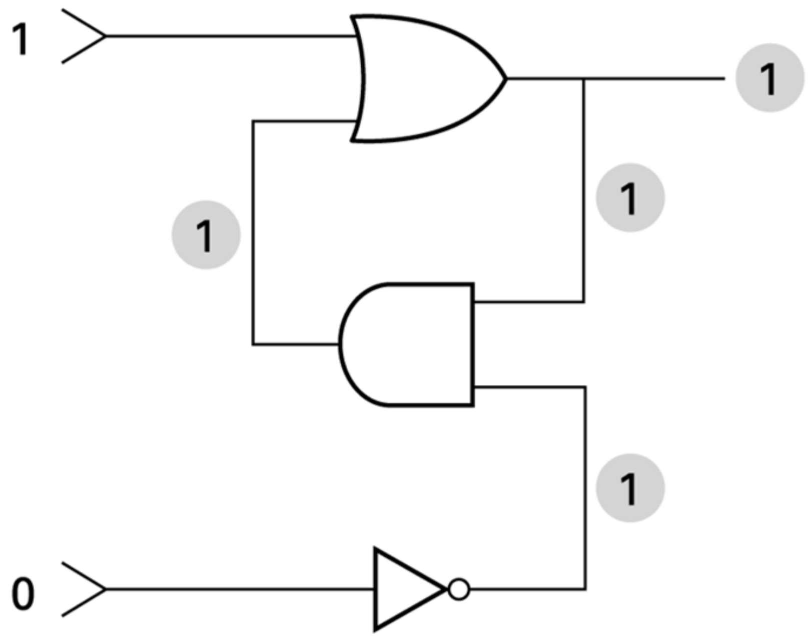
Setting the output of a flip-flop to 1

a. 1 is placed on the upper input.



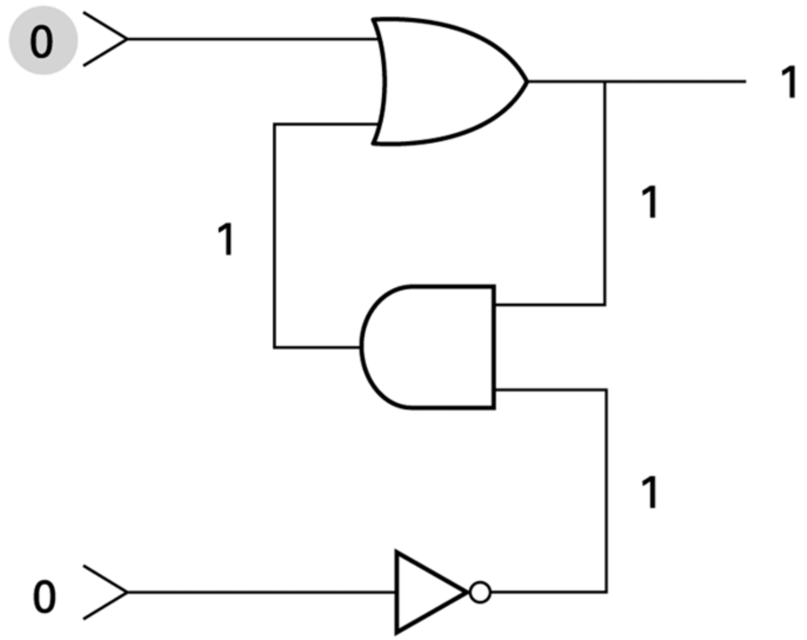
Setting the output of a flip-flop to 1 (continued)

b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.

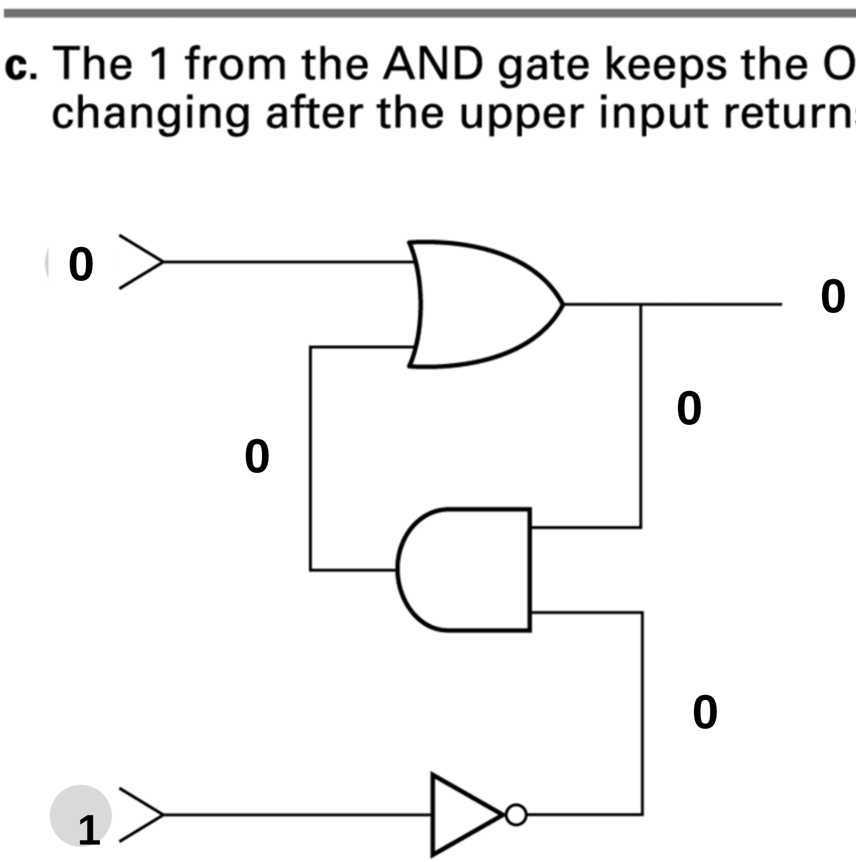


Setting the output of a flip-flop to 1 (continued)

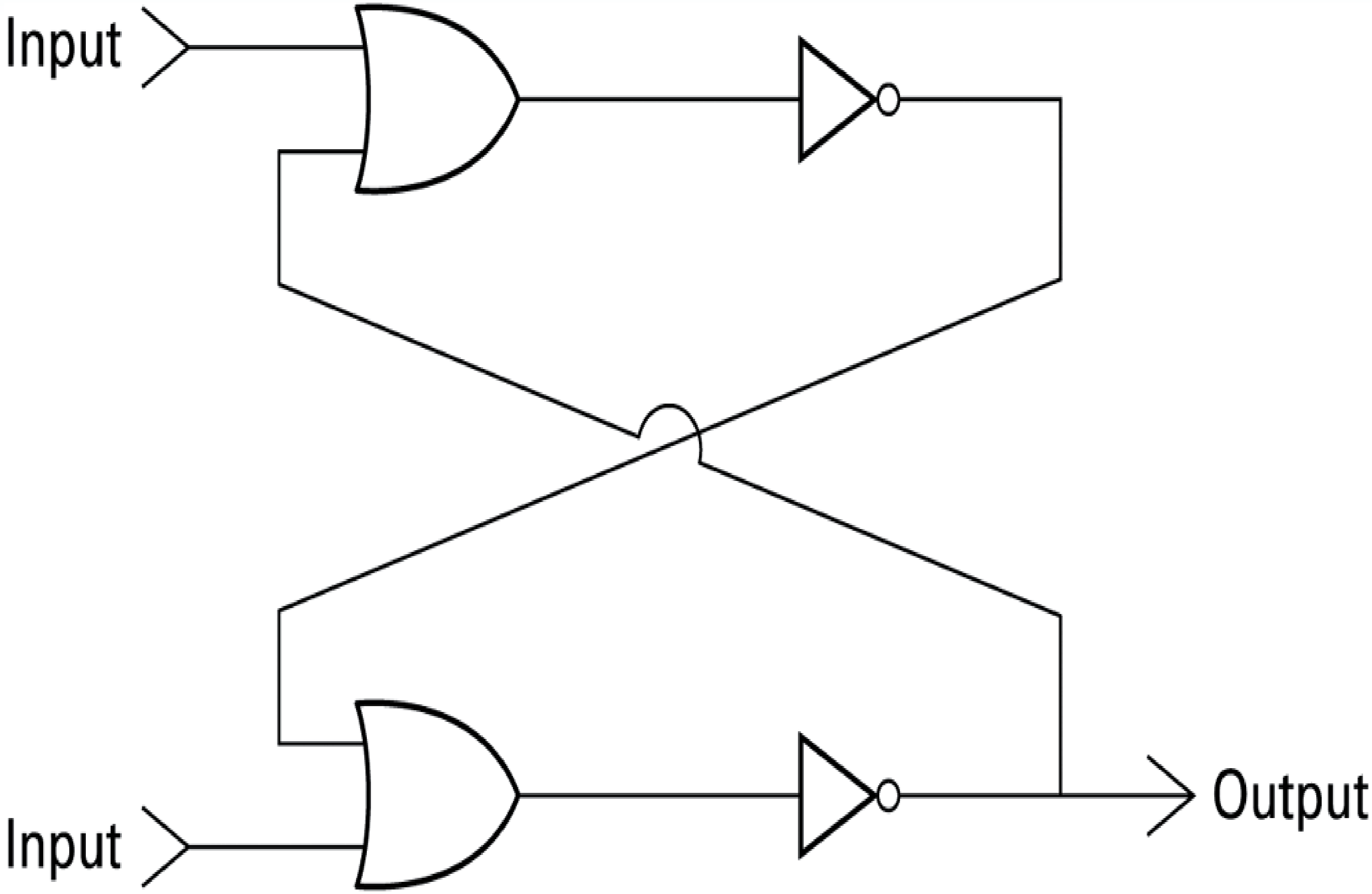
c. The 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.



Setting the output of a flip-flop to 1 (continued)



Another way of constructing a flip-flop



2015/03/04 stopped here

Hexadecimal Notation

- **Hexadecimal notation:** A shorthand notation for long bit patterns
  - Divides a pattern into groups of four bits each
  - Represents each group by a single symbol
- Example: 10100011 becomes A3

The hexadecimal coding system

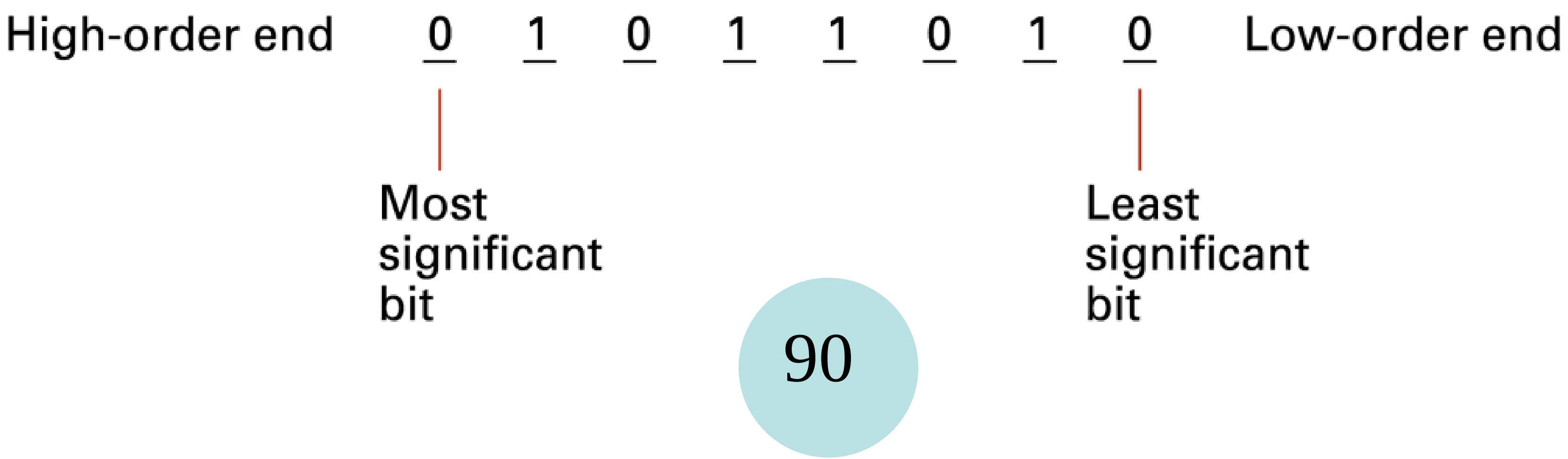
Bit pattern	Hexadecimal representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Main Memory Cells

Cell: A unit of main memory (typically 8 bits which is one **byte**)

- **Most significant bit:** the bit at the left (high-order) end of the conceptual row of bits in a memory cell
- **Least significant bit:** the bit at the right (low-order) end of the conceptual row of bits in a memory cell

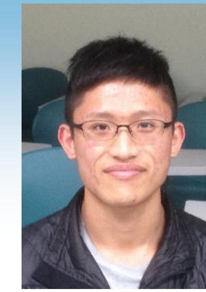
The organization of a byte-size memory cell



Measuring Memory Capacity

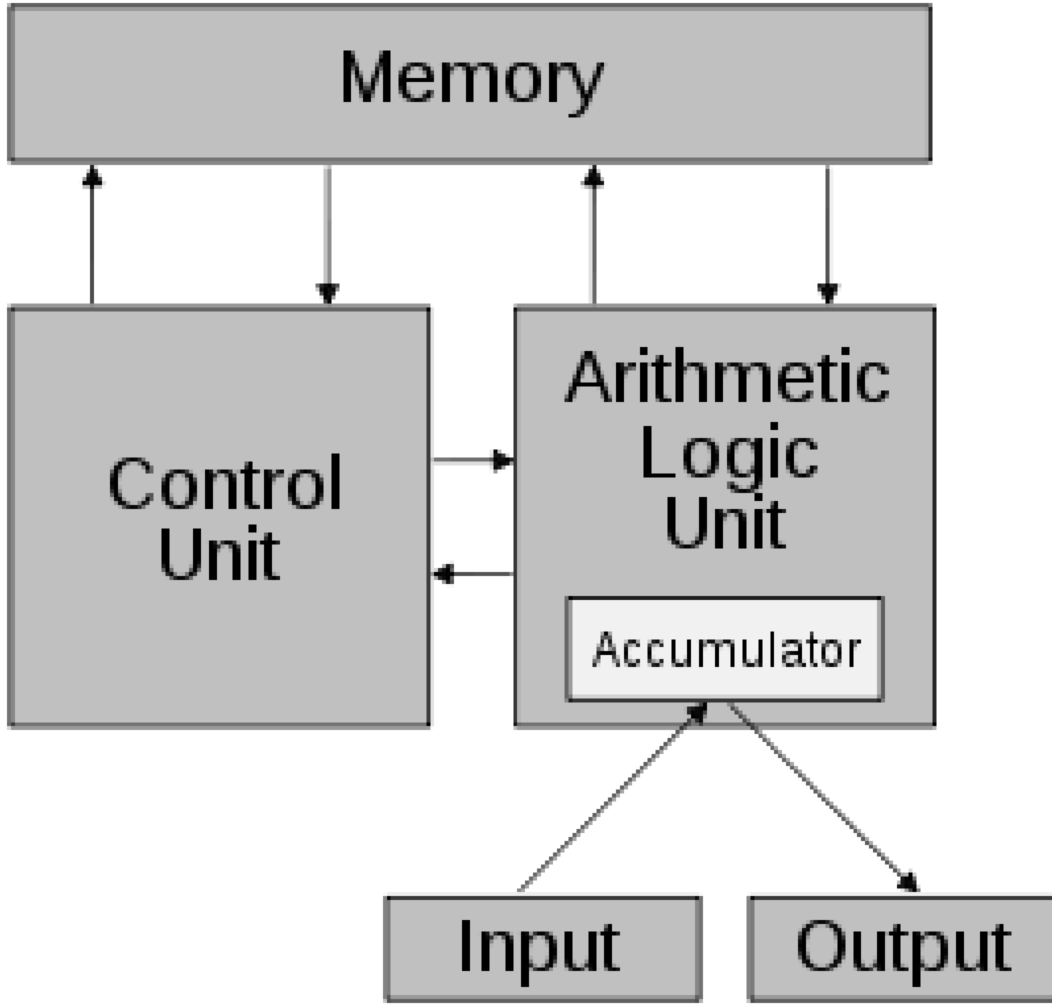
- **Kilobyte:** 1K = 2^{10} bytes = 1024 bytes
  - Example: 3 KB = 3 times1024 bytes
  - Sometimes “kibi” rather than “kilo”
- **Megabyte:** 1M = 2^{20} bytes = 1,048,576 bytes
  - Example: 3 MB = 3 times 1,048,576 bytes
  - Sometimes “megi” rather than “mega”
- **Gigabyte:** 1G = 2^{30} bytes = 1,073,741,824 bytes
  - Example: 3 GB = 3 times 1,073,741,824 bytes
  - Sometimes “gigi” rather than “giga”

Memory Management (I)（楊其昇）

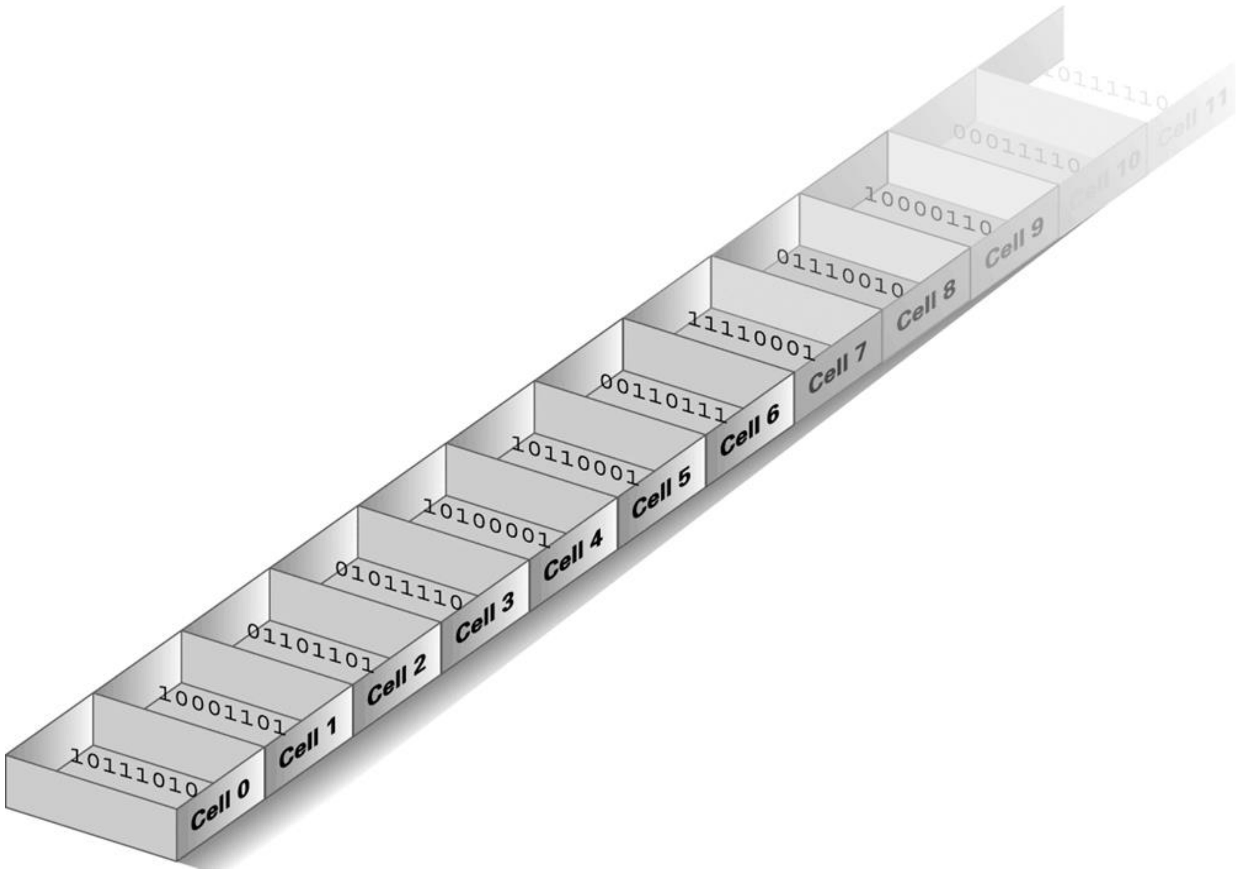


Von Neumann's model

John von Neumann
1903-1957



Memory cells arranged by address - Conceptually （馬詠治）

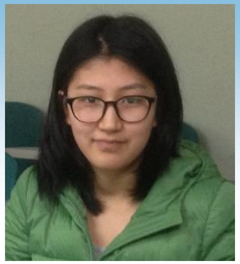


Ideal storage devices （陳楷訓）

- stable – contents do not corrupt
- fast – to match the speed of CPU
- large in capacity – 1000 G ?
- cheap in per unit capacity
- small in size – portable ?
- low-power

Unfortunately, there is no single technology that fulfills all the requirements.

Storage Structure (I) （古行涵）



- Main memory
  - *only large storage media that the CPU can access directly.*
- Secondary storage
  - *extension of main memory that provides large nonvolatile storage capacity.*

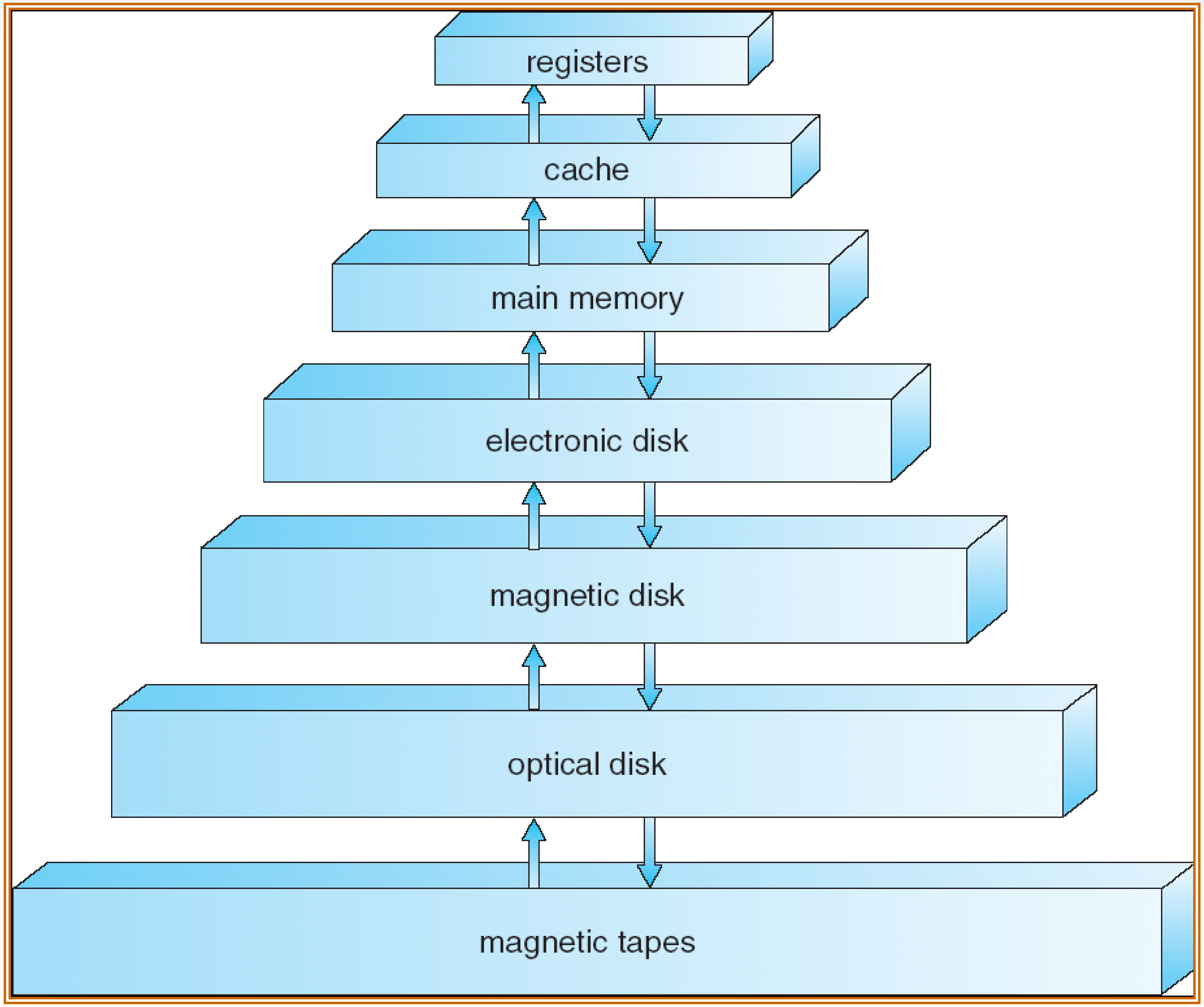
Storage Structure (II)（吴宇）

- Secondary storage
  - *extension of main memory that provides large nonvolatile storage capacity.*
  - *Magnetic disks*
    - rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
    - The *disk controller* determines the logical interaction between the device and the computer.

Storage Hierarchy （張瑞）

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility
- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

Storage-Device Hierarchy



Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Caching (I)

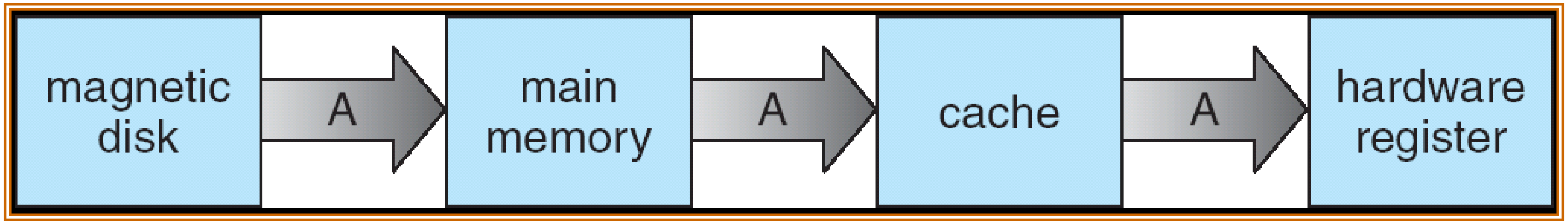
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily

Caching (II)

- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - *Cache management*: important design problem
  - Cache size and replacement policy

Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy
- Multiprocessor environment
  – cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment
  – Several copies of a datum can exist.



Main Memory Addresses

- **Address:** A “name” that uniquely identifies one cell in the computer’s main memory
  – The names are actually numbers.
  – These numbers are assigned consecutively starting at zero.
  – Numbering the cells in this manner associates an order with the memory cells.

Memory Terminology

- **Random Access Memory (RAM):**
  - Memory in which individual cells can be easily accessed in any order
  - RAM composed of volatile memory
  - **Dynamic RAM**
    - charges in capacitors, refresh needed
    - fast, high-density, low-cost
  - **Static RAM**
    - flip-flops, no refresh
    - slower, lower-density, high-cost

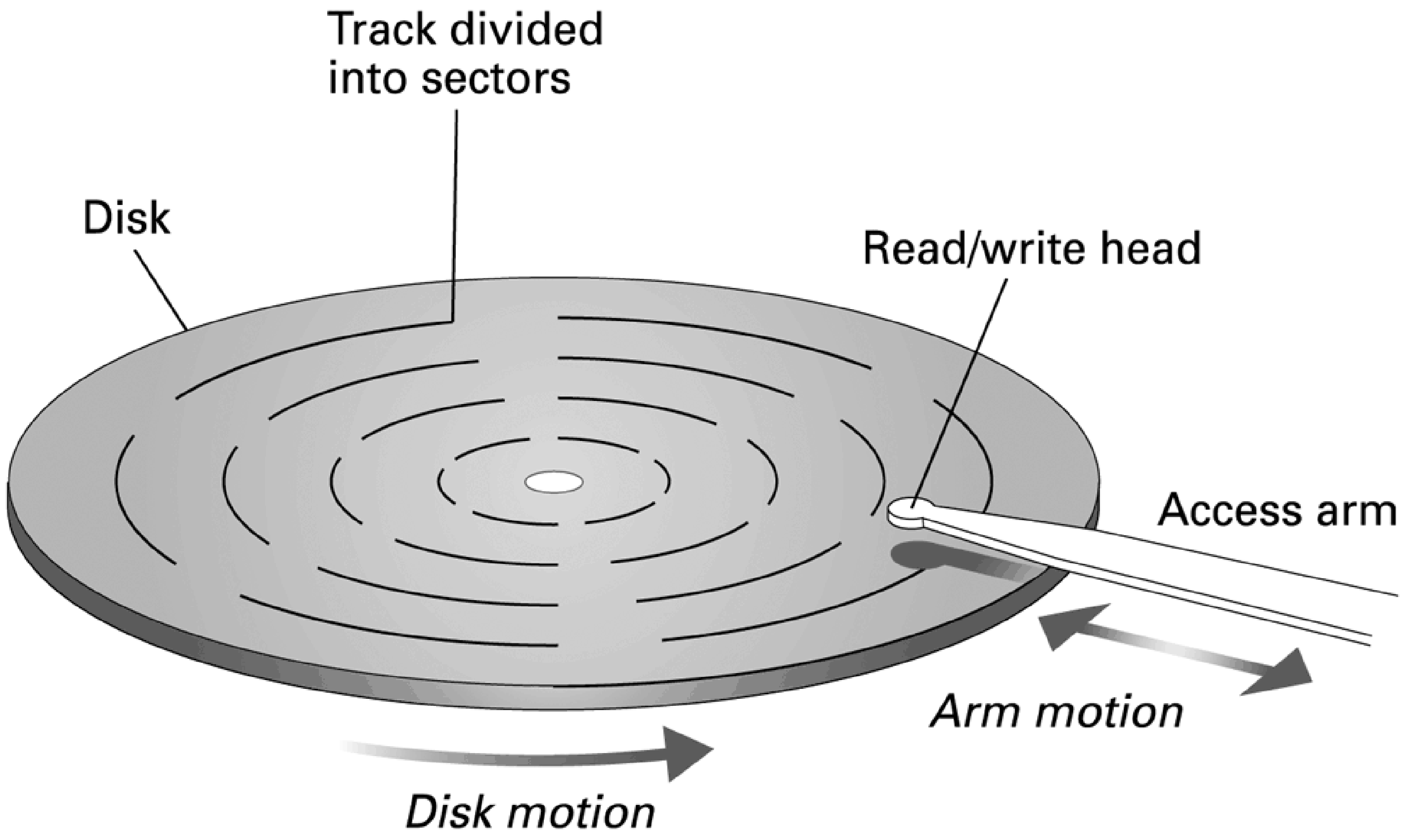
Mass Storage

- On-line versus off-line
- Typically larger than main memory
- Typically less volatile than main memory
- Typically slower than main memory

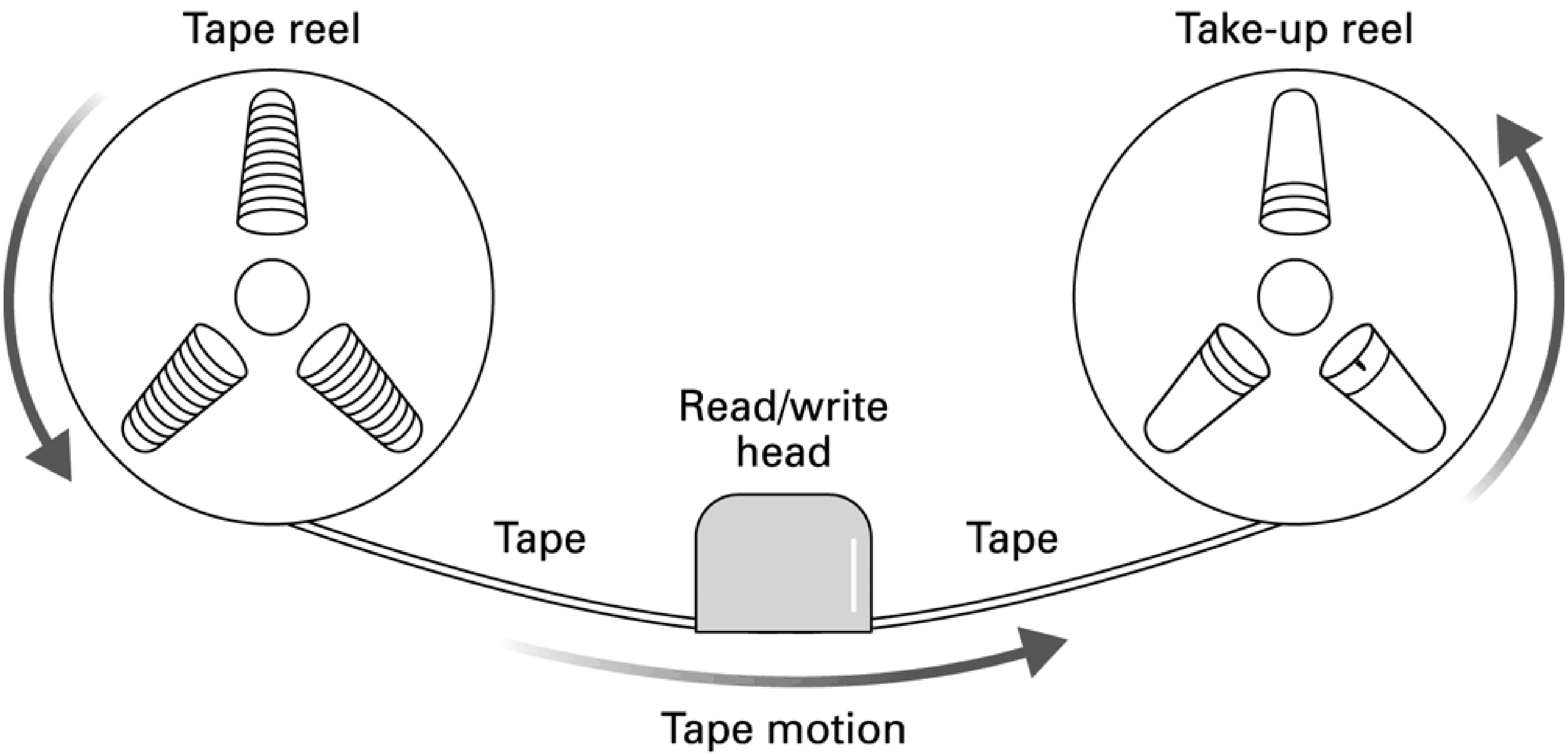
Mass Storage Systems

- Magnetic Systems
  - Disk
  - Tape
- Optical Systems
  - CD
  - DVD
- Flash Drives

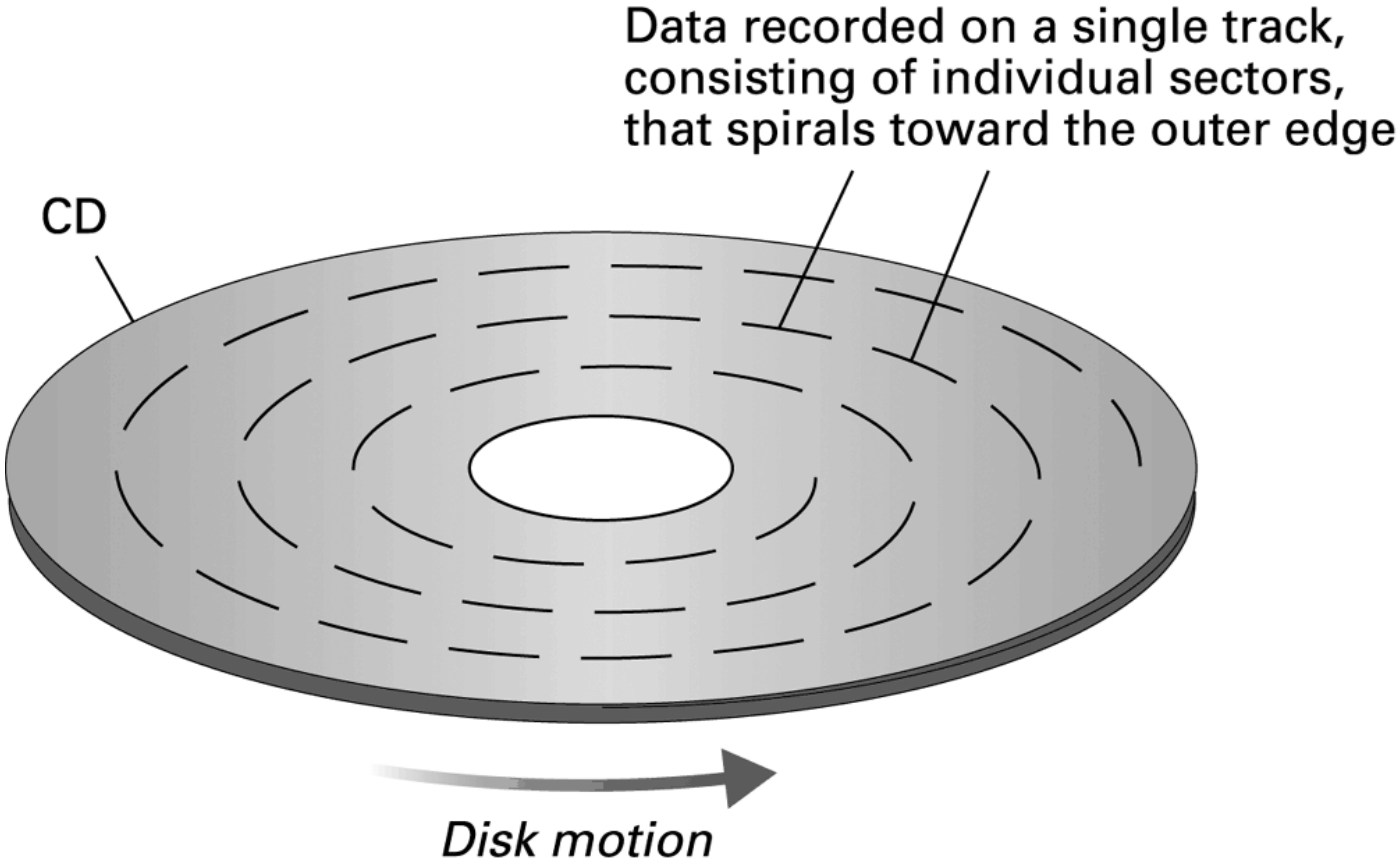
A magnetic disk storage system



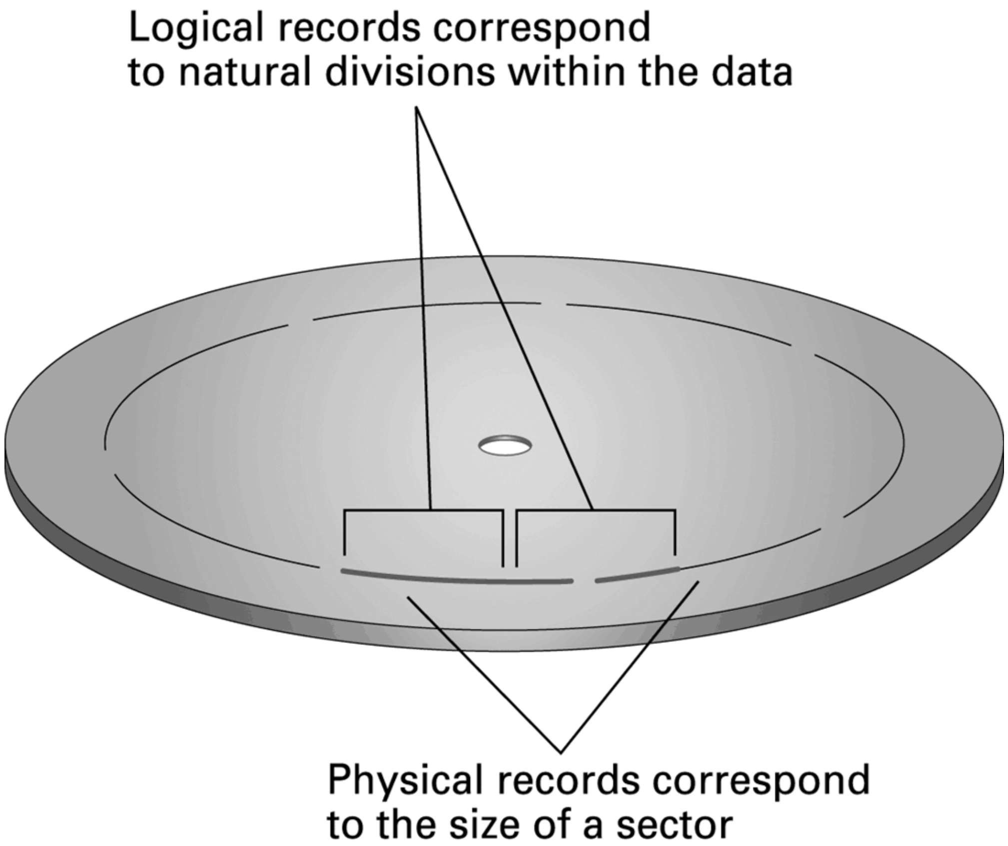
Magnetic tape storage



CD storage


Data recorded on a single track, consisting of individual sectors, that spirals toward the outer edge
CD
Disk motion

Logical records versus physical records on a disk



2014/03/04 stopped here.

Representing Numeric Values

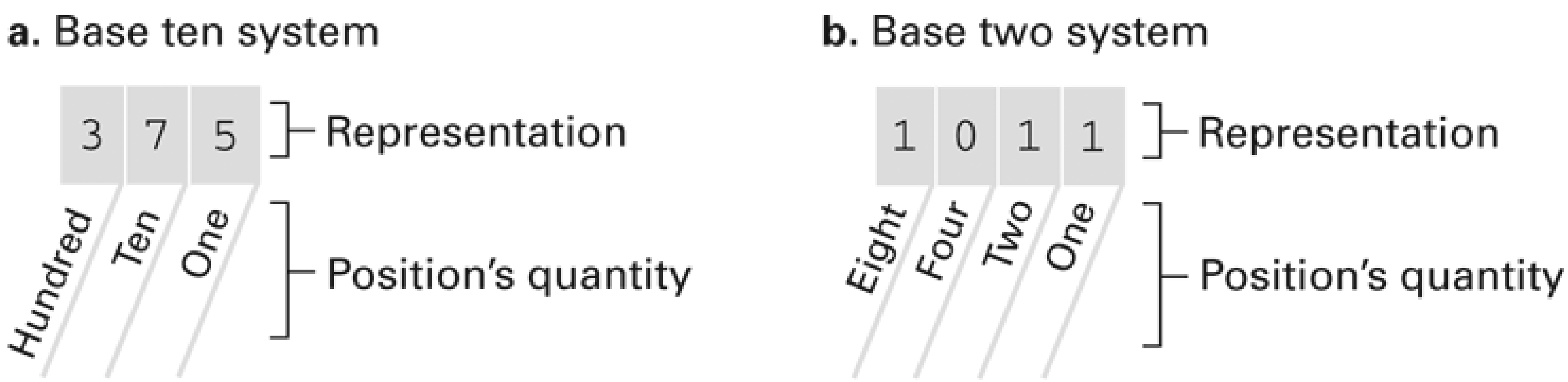
- Binary notation: Uses bits to represent a number in base two
- Limitations of computer representations of numeric values
  - Overflow – occurs when a value is too big to be represented
  - Truncation – occurs when a value cannot be represented accurately with deletion of the LSBs.

The Binary System

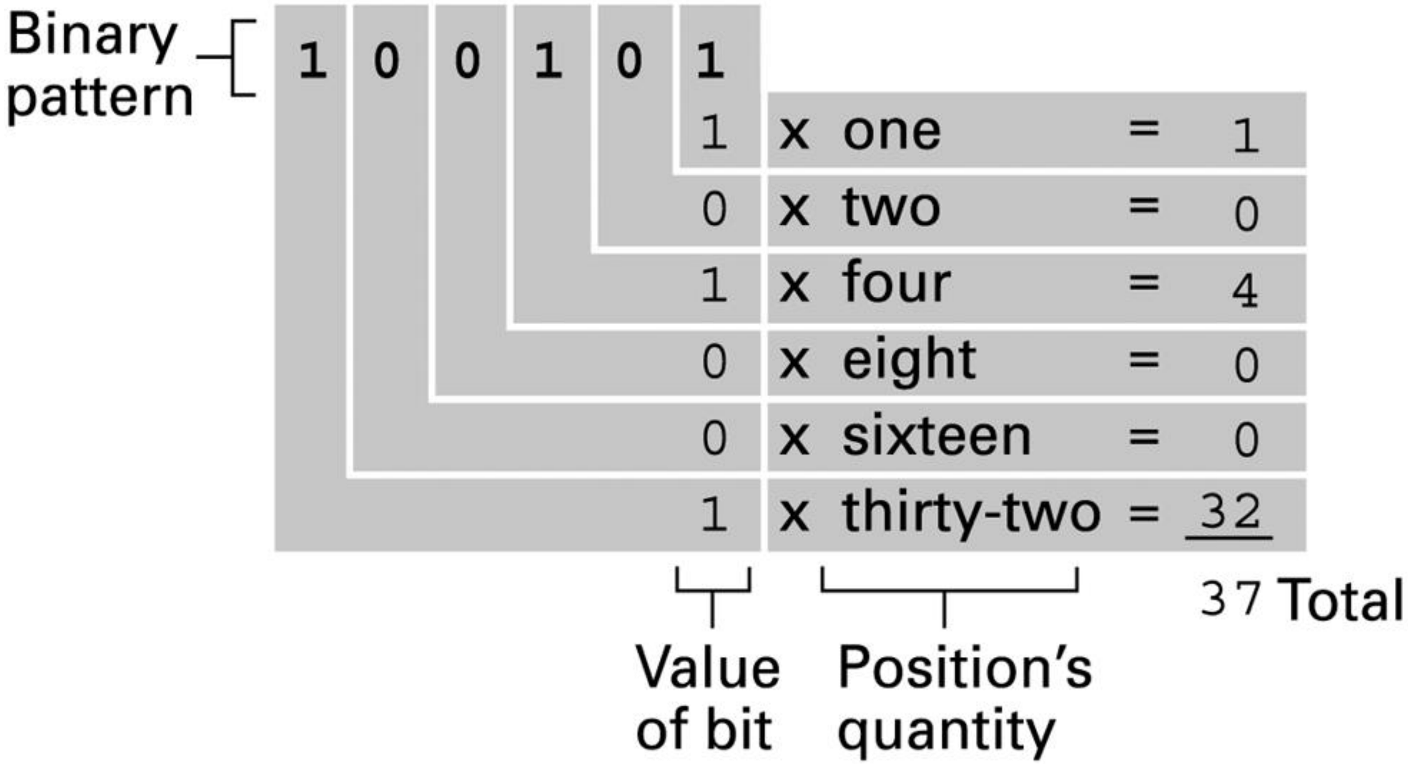
The traditional decimal system is based on powers of ten.

The Binary system is based on powers of two.

The base ten and binary systems



Decoding the binary representation 100101



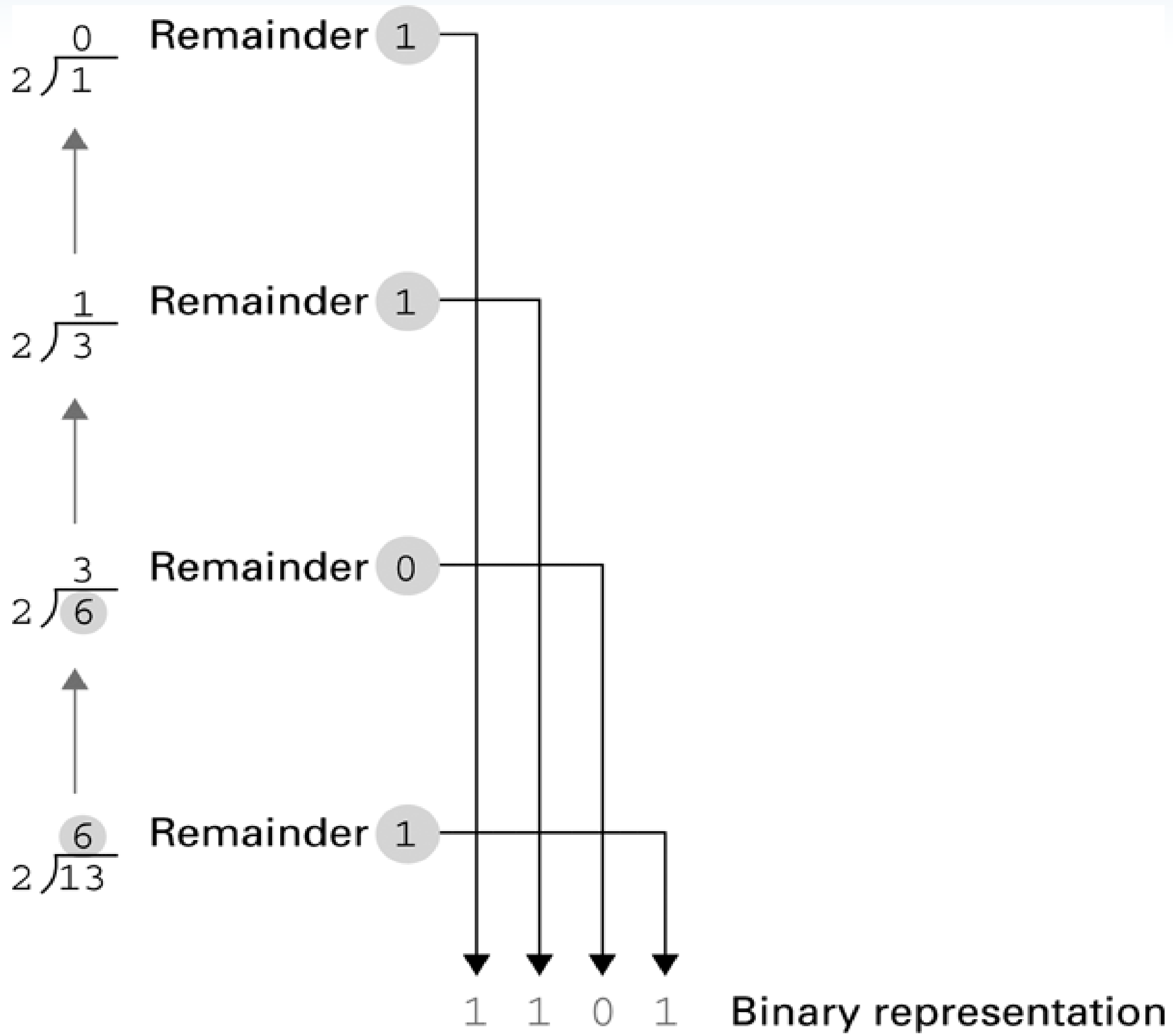
An algorithm for finding the binary representation of a positive integer

Step 1. Divide the value by two and record the remainder.

Step 2. As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

Step 3. Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.

Obtaining the binary representation of thirteen



The binary addition facts

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}$$

Storing Integers

- **Two's complement notation:** The most popular means of representing integer values
- **Excess notation:** Another means of representing integer values
- Both can suffer from overflow errors.

Two's complement notation systems

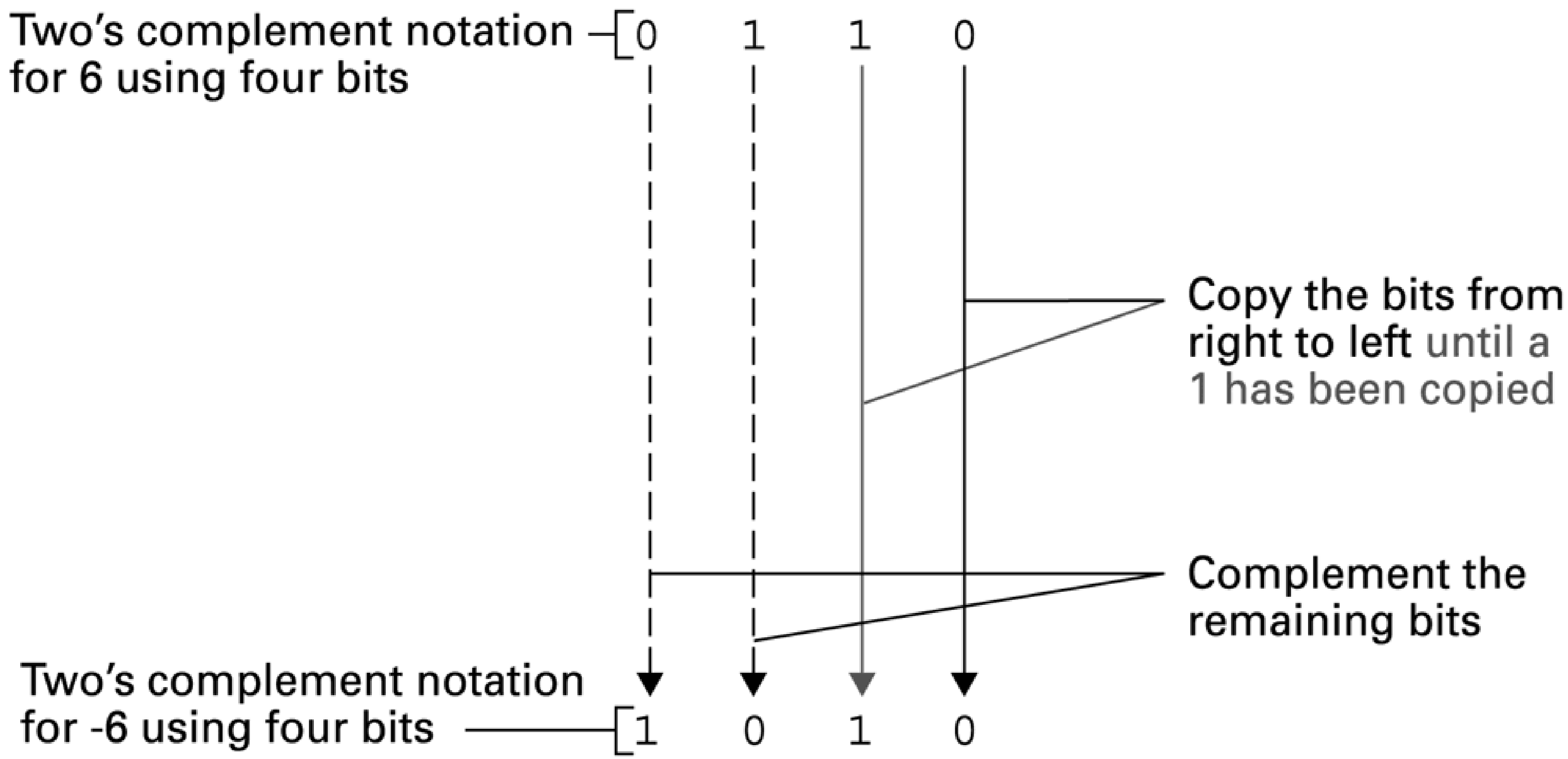
a. Using patterns of length three

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	−1
110	−2
101	−3
100	−4

b. Using patterns of length four

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Coding the value -6 in two's complement notation using four bits



Addition problems converted to two's complement notation

Problem in base ten	Problem in two's complement	Answer in base ten
3 + 2	0011 + 0010 0101	5
−3 + −2	1101 + 1110 1011	−5
7 + −5	0111 + 1011 0010	2

An excess eight conversion table

Bit pattern	Value represented
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	−1
0110	−2
0101	−3
0100	−4
0011	−5
0010	−6
0001	−7
0000	−8

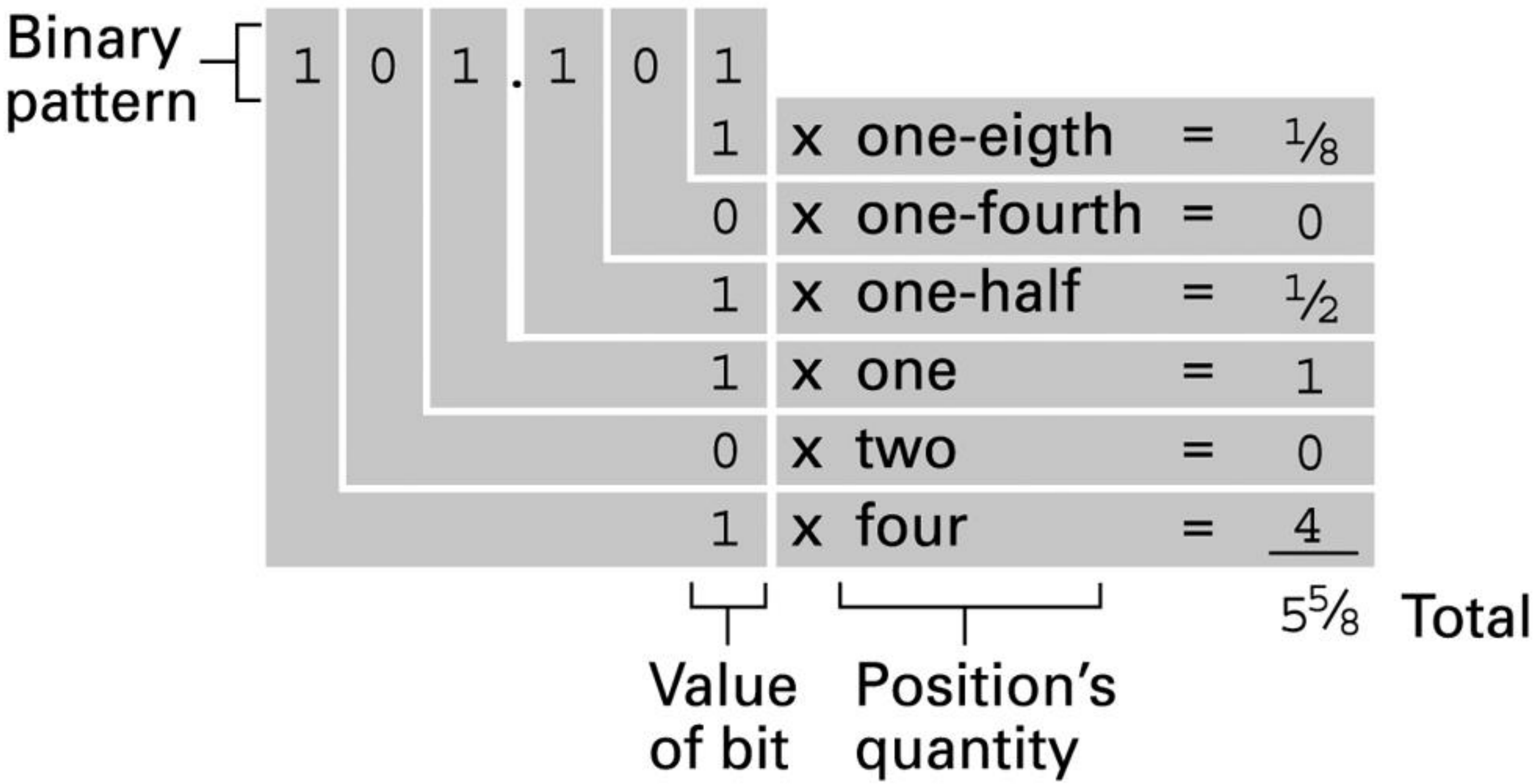
An excess notation system using bit patterns of length three

Bit pattern	Value represented
111	3
110	2
101	1
100	0
011	−1
010	−2
001	−3
000	−4

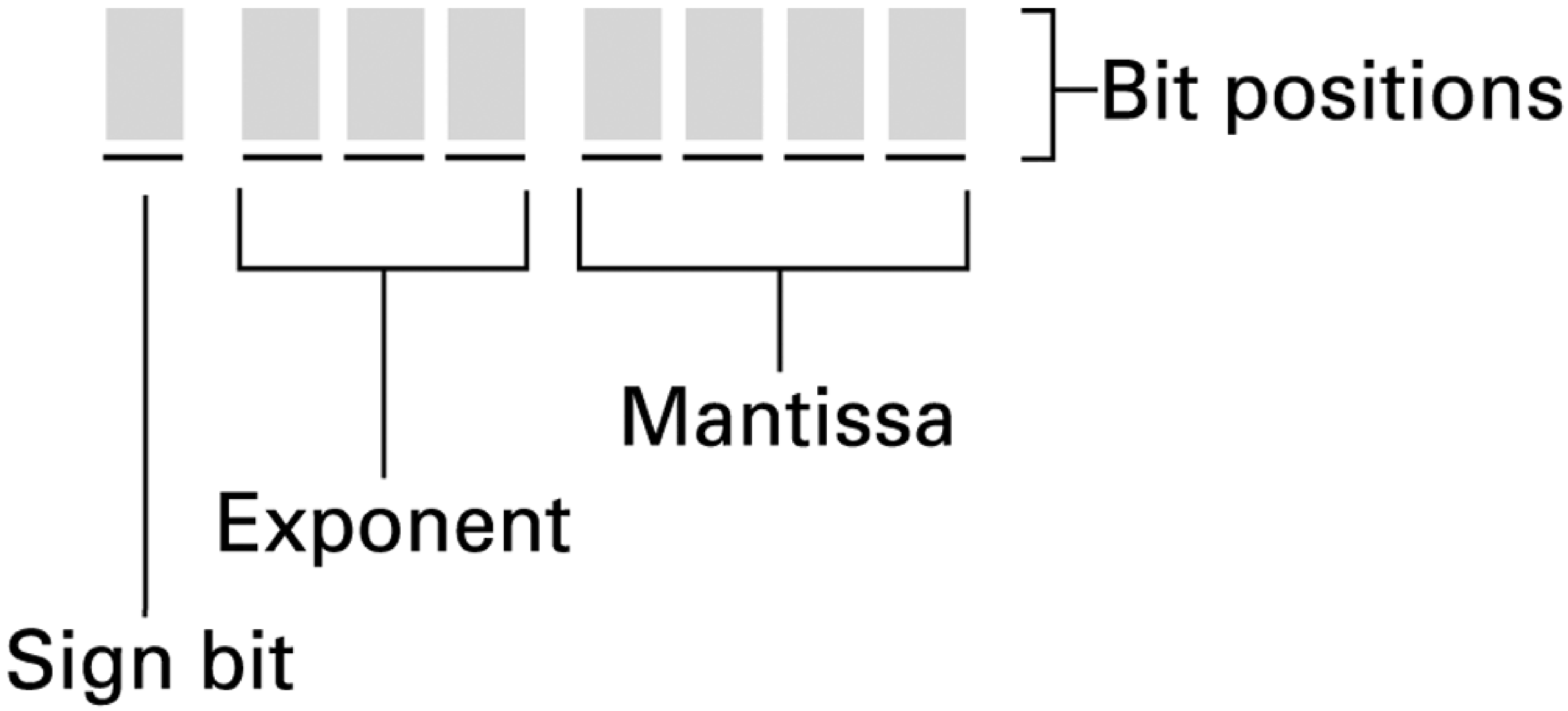
Storing Fractions

- **Floating-point Notation:** Consists of a sign bit, a mantissa field, and an exponent field.
- Related topics include
  - Normalized form
  - Truncation errors

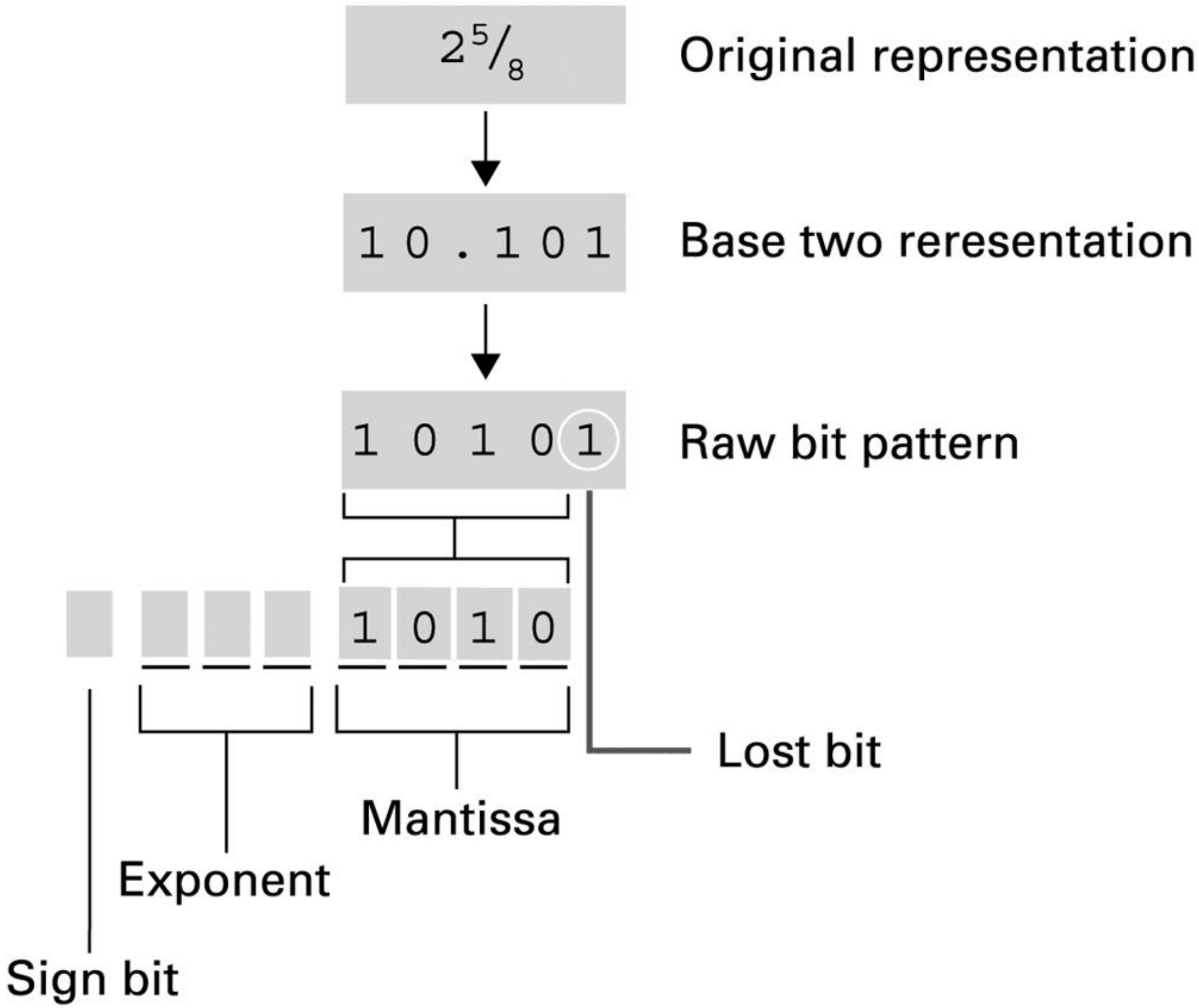
Decoding the binary representation 101.101



Floating-point notation components



Encoding the value 2 5⁄8



Representing Text

- **Each character (letter, punctuation, etc.) is assigned a unique bit pattern.**
  - ASCII: Uses patterns of 7-bits to represent most symbols used in written English text
  - Unicode: Uses patterns of 16-bits to represent the major symbols used in languages world wide
  - ISO standard: Uses patterns of 32-bits to represent most symbols used in languages world wide

The message “Hello.” in ASCII

01001000	01100101	01101100	01101100	01101111	00101110
H	**e**	**l**	**l**	**o**	**.**

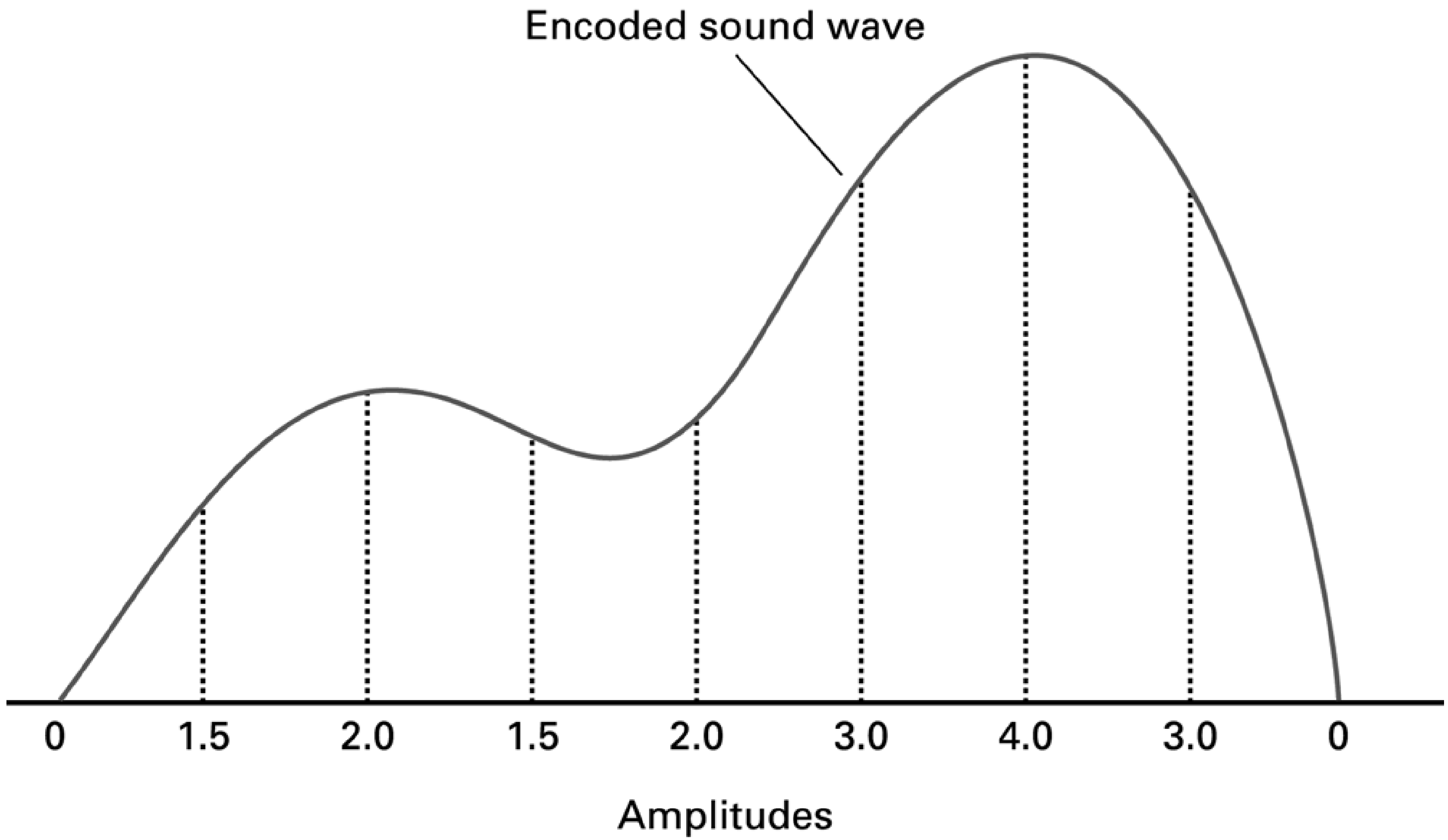
Representing Images

- Bit map techniques
  - Pixel: short for “picture element”
  - RGB
  - Luminance and chrominance
- Vector techniques
  - Scalable
  - TrueType and PostScript

Representing Sound

- Sampling techniques
  - Used for high quality recordings
  - Records actual audio
- MIDI (**M**usical **I**nstrument **D**igital **I**nterface)
  - Used in music synthesizers
  - Records “musical score”

The sound wave represented by the sequence 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0



Data Compression

- Lossy versus lossless
- Run-length encoding
  – *replacing a long sequence of same value pattern by a count and the value pattern.*
- Frequency-dependent encoding

  (Huffman codes)
- Relative encoding
- Dictionary encoding (Includes adaptive dictionary encoding such as LZW encoding.)

Huffman codes (optimal)

Given ‘xyx xyz xyx xyw xyx xyw‘ with table

23x8bits =184bits

01 10 01 00 01 10 111 00 01 10 01 00
01 10 110 01 10 01 00 01 10 110

$8+12+5\times8+47=107$bits

code	char
00	space
01	x
10	y
110	w
111	z

LZW (Lempel-Ziv-Welsh) encoding

Given ‘xyx xyx xyx xyw xyx xyw‘ with table

- ‘x’ → 1
- ‘xy’ → 12
- ‘xyx’ →121
- ‘xyx ‘ → 1210
- ‘xyx xyx’ → 12104
- ‘xyx xyx xyx’ → 1210404
- … …
- ‘xyx xyx xyx xyw xyx xyw’ → 121040401230405

23x8bits =184bits

8+(10+5)x8+15x3=173bits

#	words
0	space
1	x
2	y
3	w
4	xyx
5	xyw

Compressing Images

- GIF (Graphic Interchange Format)
  - a palette of 256 colors for each file
  - 256x256x256 choices of colors
  - Good for cartoons
- JPEG (Joint Photographic Experts Group): Good for photographs
- TIFF (Tagged Image File Format): Good for image archiving

Compressing Audio and Video

- MPEG (Motion Picture Experts Group)
  - ISO
  - Relative techniques
  - High definition television broadcast
  - Video conferencing

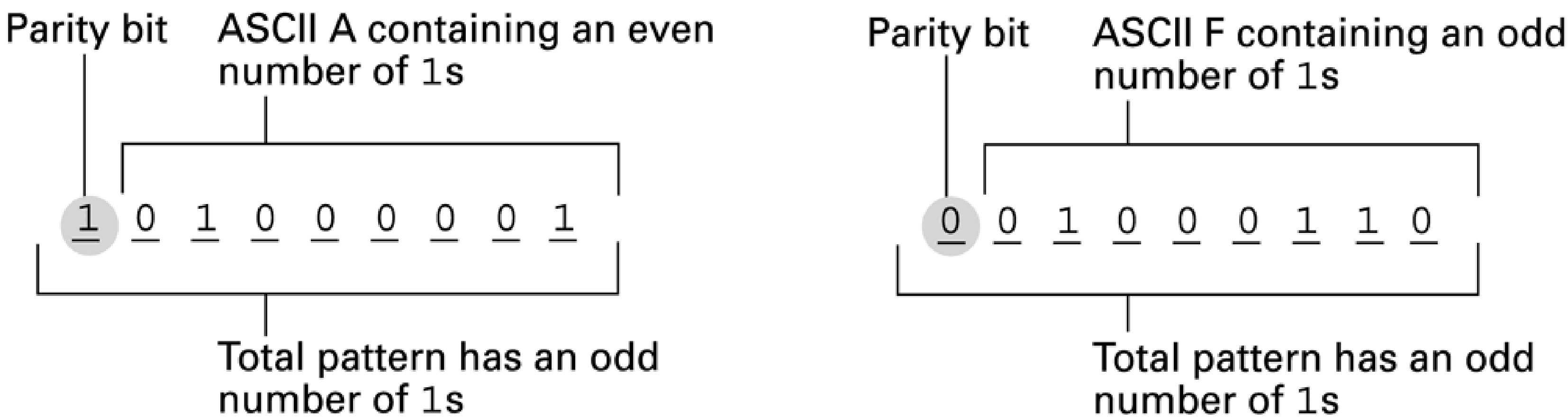
Compressing Audio and Video

- MP3 (MPEG Layer 3)
  - use human ear properties for compression.
  - Temporal masking
    - loud sound masks following softer sound
  - Frequency masking:
    - One frequency masks softer sounds at nearby frequency.
  - near CD quality

Communication Errors

- Parity bits (even versus odd)
- Checkbytes
- Error correcting codes

The ASCII codes for the letters A and F adjusted for odd parity



An error-correcting code

Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

Decoding the pattern 010100 using the code in Figure 1.30

Character	Code	Pattern received	Distance between received pattern and code	
A	0 0 0 0 0 0	0 **1** 0 **1** 0 0	2	
B	0 0 1 1 1 1	0 **1** **0** 1 **0** **0**	4	
C	0 1 0 0 1 1	0 1 0 **1** **0** **0**	3	
D	0 1 1 1 0 0	0 1 **0** 1 0 0	**1**	Smallest distance
E	1 0 0 1 1 0	**0** **1** 0 1 **0** 0	3	
F	1 0 1 0 0 1	**0** **1** **0** **1** 0 **0**	5	
G	1 1 0 1 0 1	**0** 1 0 1 0 **0**	2	
H	1 1 1 0 1 0	**0** 1 **0** **1** **0** 0	4	