



Introduction to Computer Science

Polly Huang
NTU EE
<http://homepage.ntu.edu.tw/~pollyhuang>
pollyhuang@ntu.edu.tw



Chapter 1

Data Storage

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 Communication Errors

Bits

- ☀ **Bit**
 - Binary Digit
- ☀ A symbol whose meaning depends on the application at hand.
 - Numeric value (1 or 0)
 - Boolean value (true or false)
 - Voltage (high or low)

Bit Patterns

- ☀ All data stored in a computer are represented by patterns of bits:
 - Numbers
 - Text characters
 - Images
 - Sound
 - Anything else...

Boolean Operations

- ☀ **Boolean operation**
 - any operation that manipulates one or more true/false values
 - Can be used to operate on bits
- ☀ **Specific operations**
 - AND
 - OR
 - XOR
 - NOT

AND, OR, XOR

The AND operation

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

The OR operation

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

The XOR operation

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


Gates

☀ Gates

- devices that produce the outputs of Boolean operations when given the operations' input values
- ☀ Often implemented as **electronic circuits**
- ☀ Provide the **building blocks** from which computers are constructed

AND, OR, XOR, NOT Gates


AND



Inputs — Output

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


OR



Inputs — Output

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


XOR



Inputs — Output

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

NOT



Inputs — Output

Inputs	Output
0	1
1	0

Polly Huang

9

Flip-Flops

☀ Flip-Flop

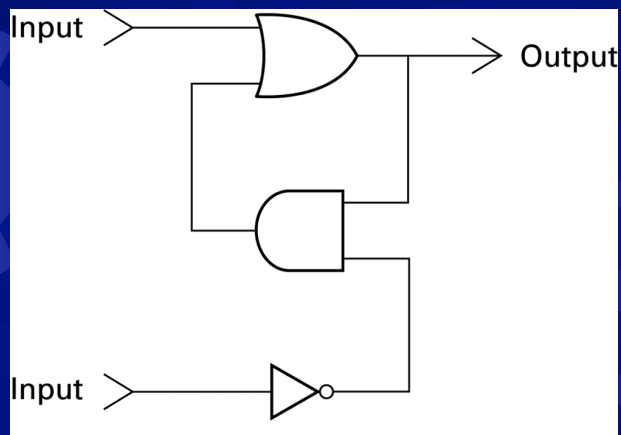
- a circuit built from gates that can store one bit of data
- ☀ Two input lines
 - Both lines are 0 at the beginning
 - One input line sets its stored value to 1
 - Another input line sets its stored value to 0
 - When both input lines are 0, the most recently stored value is preserved

Flip-Flops Illustrated



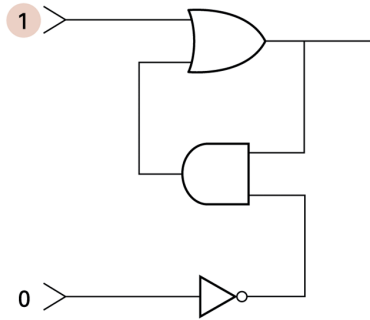
X	Y	Z
0	0	--
0	1	0
1	0	1
1	1	*

A Simple Flip-Flop Circuit



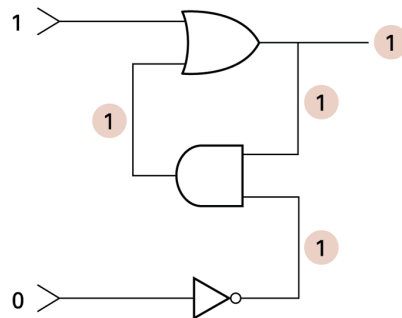
Setting to 1 – Step a.

a. 1 is placed on the upper input.



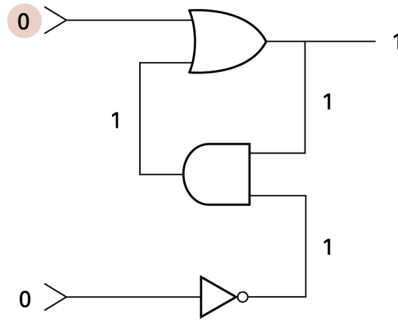
Setting to 1 – Step b.

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 to 1 – Step c.

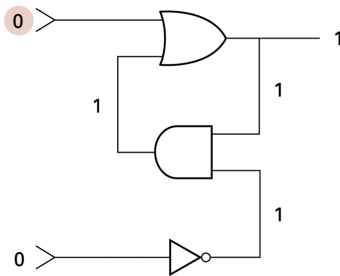
c. The 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.



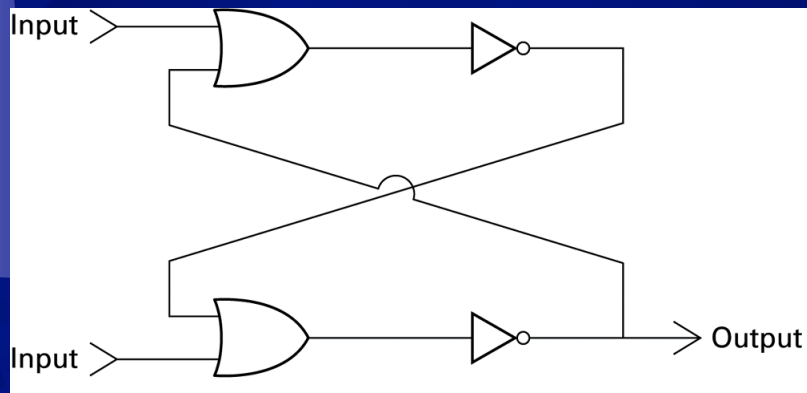
Try This

☀ Show how the output can be turned to 0

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



Another Flip-Flop Circuit



Flip-Flops

- A kind of storage
- Holds its value until the power is turned off

Types of Memory

- ☀ Volatile memory
 - A.k.a. dynamic memory
 - Holds its value until the power is turned off
- ☀ Non-volatile memory
 - Holds its value even after the power is off

Capacitors

- ☀ Two metallic plates positioned in parallel
- ☀ Principle
 - Tiny space between two plates
 - Voltage thru, charge the plats
 - Voltage off, charge remains on the plats
 - Connect two plates, charge off the plates
- ☀ Many of these capacitors with their circuits on a wafer (chip)
- ☀ Even more volatile than flip-flops
 - Must be replenished periodically

Flash Memory

- ☀ Trapping electrons in silicon dioxide SiO_2 chambers
- ☀ Chamber very small
 - 90 angstroms
 - 1 angstrom is 1/10,000,000,000 of a meter
- ☀ Non-volatile
 - Holds its value after the power is off

Types of Memory

- ☀ Volatile memory
 - A.k.a. dynamic memory
 - Holds its value until the power is turned off
 - Examples, flip-flops, capacitors
- ☀ Non-volatile memory
 - Holds its value even after the power is off
 - Examples, flash memory, magnetic storage, compact disk, etc

Hexadecimal Notation

Why?

- Things are in bit streams
- Stream = a long string of bits.
- Long bit streams
- E.g., $100 = 1100100_2$

Hexadecimal notation

- A shorthand notation for streams of bits.
- More compact.

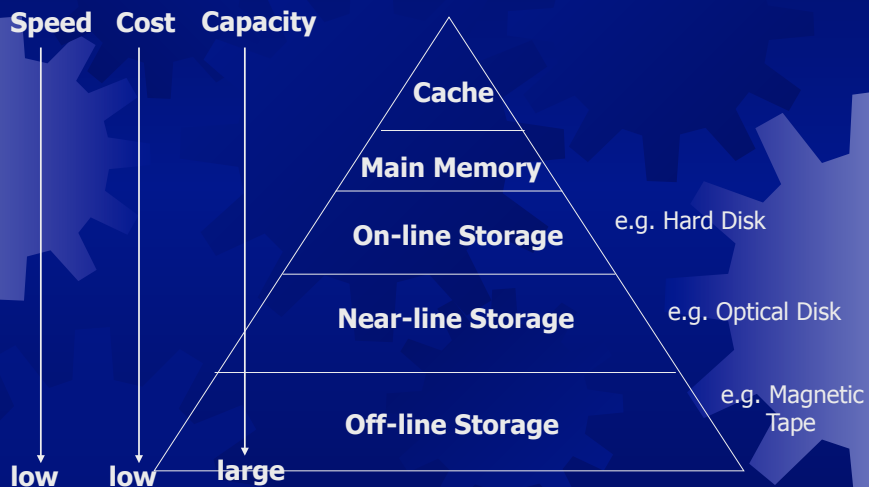
The Hexadecimal System

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

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

Storage Hierarchy



Main Memory: Addresses

- ☀ **Address**

- A “name” to uniquely identify one cell

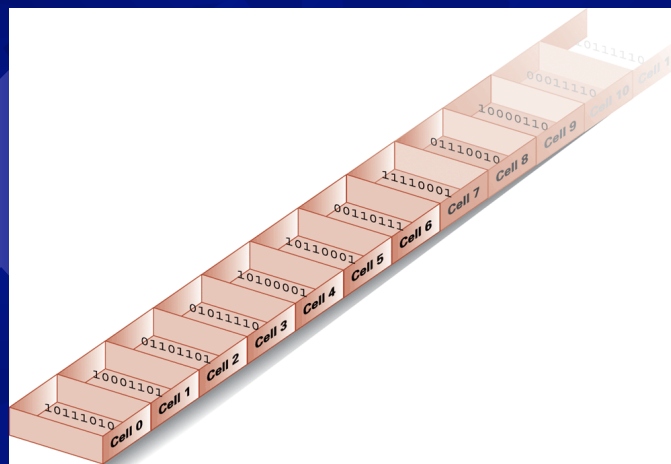
- ☀ **Consecutive numbers**

- Usually starting at zero
- I.e., having an order
- “previous cell” and “next cell” have reasonable meanings

- ☀ **Random Access Memory**

- Memory where any cell can be accessed independently

Memory Cells



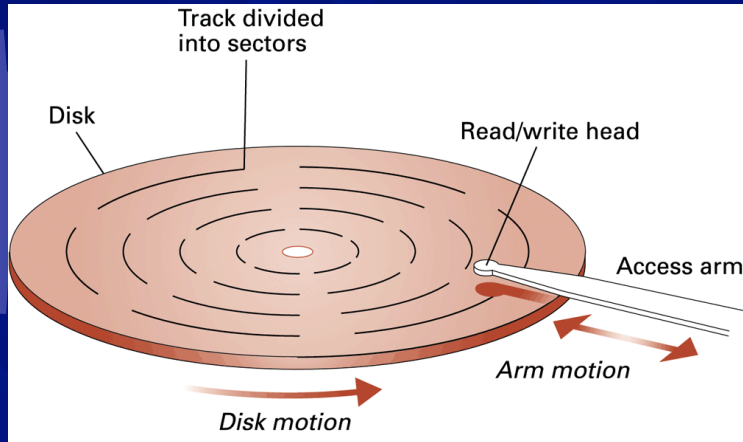
Not Quite the Metric System

- ☀ K
 - “Kilo-” normally means 1,000
 - Kilobyte = $2^{10} = 1024$
- ☀ M
 - “Mega-” normally means 1,000,000
 - Megabyte = $2^{20} = 1,048,576$
- ☀ G
 - “Giga-” normally means 1,000,000,000
 - Gigaabyte = $2^{30} = 1,073,741,824$

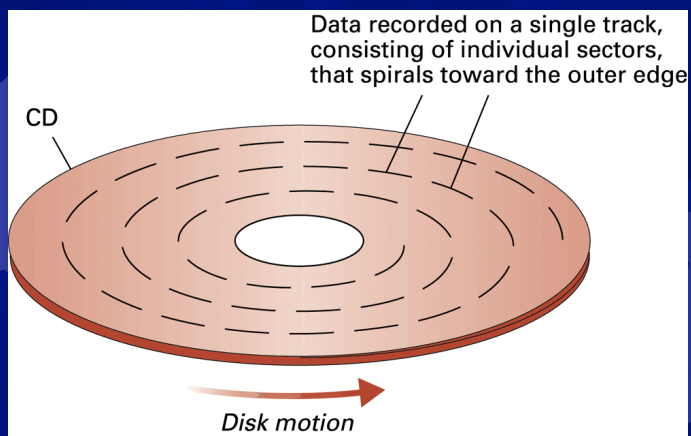
Mass Storage Systems

- ☀ Non-Volatile
 - Data remains when computer is off
- ☀ Usually much bigger than main memory
- ☀ Usually rotating disks
 - Hard disk, floppy disk, CD-ROM
 - Much slower than main memory
 - Data access must wait for **seek time** (head positioning)
 - Data access must wait for **rotational latency**

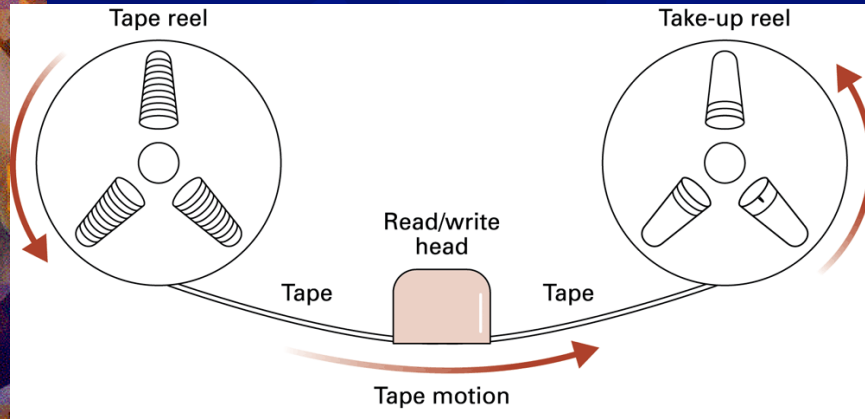
Hard Disk



Compact Disk



Magnetic Tape



Files

☀ File

- The unit of data stored on a mass storage system.

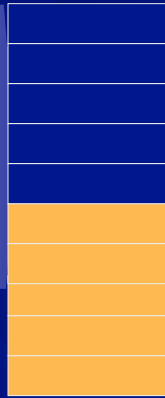
☀ Logical record and Field

- Natural groups of data within a file

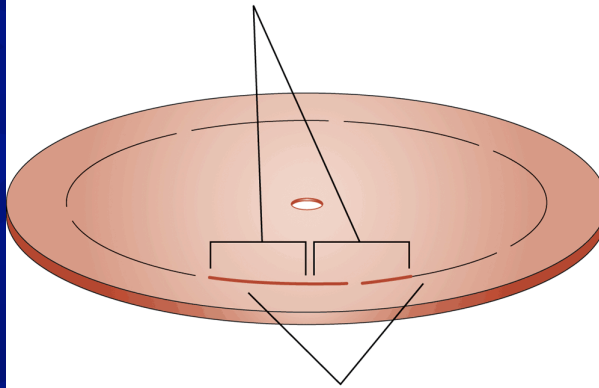
☀ Physical record

- A block of data conforming to the physical characteristics of the storage device.

Logical vs. Physical Records



Logical records correspond to natural divisions within the data



Physical records correspond to the size of a sector

Buffers

- ☀ Storage area used to hold data on a temporary basis
 - Usually during the process of being transferred from one device to another
- ☀ Example: Printers with memory
 - Holding portions of a document
 - that have transferred to the printer
 - but not yet printed

Quiz Time!

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

Bit Patterns

- ☀ All data stored in a computer are represented by patterns of bits:
 - Text characters
 - Numbers
 - Images
 - Sound

ASCII - "Hello."

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

Representing Text

- ☀ Printable character
 - Letter, punctuation, etc.
 - Assigned a unique bit pattern.
- ☀ ASCII
 - 7-bit values
 - For most symbols used in written English text
- ☀ Unicode
 - 16-bit values
 - For most symbols used in most world languages
- ☀ ISO
 - 32-bit values

Representing Number

- ☀ Binary notation
 - Uses bits to represent a number in base two
 - We'll talk more in the next 3 subsections

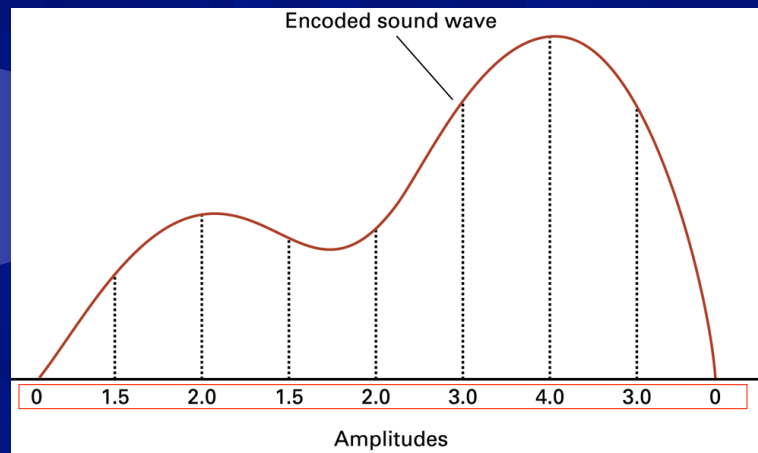
Limitations

- ☀ Overflow
 - Happens when a number is too big to be represented
- ☀ Truncation
 - Happens when a number is between two representable numbers
 - Think the real numbers..
- ☀ We will hear more in a little bit

Representing Images

- ☀ Bitmap techniques
 - Points, pixels
 - e.g. 小畫家程式
 - Representing colors
 - BMP, DIB, GIF and JPEG formats
 - Scanners, digital cameras, computer displays
- ☀ Vector techniques
 - Lines, scalable fonts, e.g., TrueType
 - e.g. Acrobat Reader
 - PostScript
 - CAD systems, printers

Representing Sound

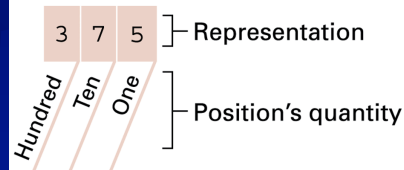


Chapter 1: Data Storage

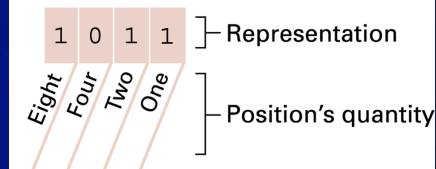
- 1.1 Bits and Their Storage
- 1.2 Main Memory
- 1.3 Mass Storage
- 1.4 Representing Information as Bit Patterns
- **1.5 The Binary System**
- 1.6 Storing Integers
- 1.7 Storing Fractions
- 1.8 Data Compression
- 1.9 Communications Errors

Decimal vs. Binary Systems

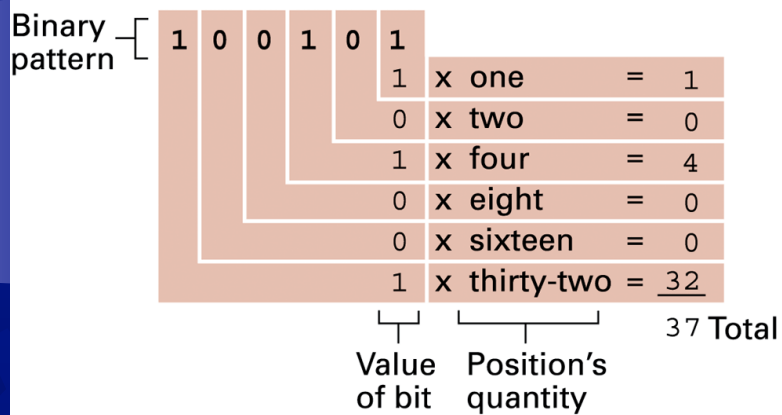
a. Base ten system



b. Base two system



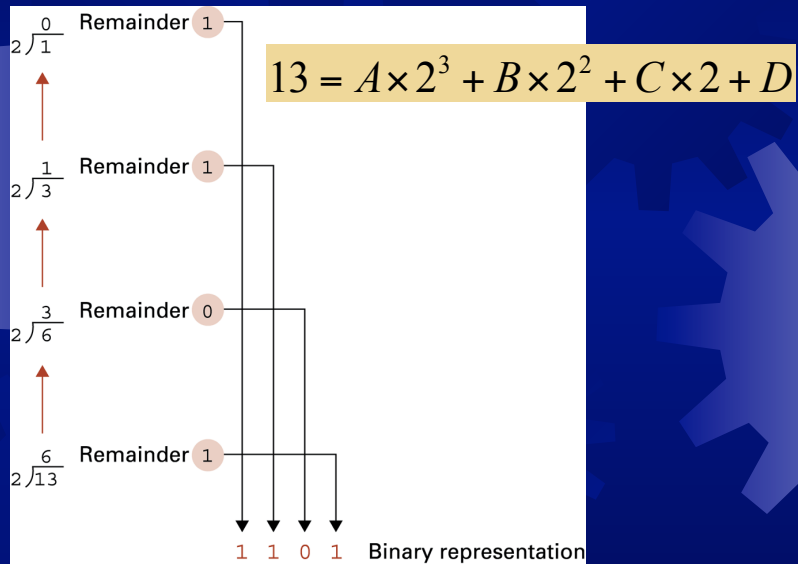
From Binary to Decimal



From Decimal to Binary

- 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.

An Example



Adding Binary Numbers

0	1	0	1
+ 0	+ 0	+ 1	+ 1
0	1	1	10

$$\begin{array}{r}
 00111010 \\
 + 00011011 \\
 \hline
 01010101
 \end{array}$$

Fraction in Binary System

Binary pattern	1	0	1	.	1	0	1	
							1	x one-eighth = $\frac{1}{8}$
							0	x one-fourth = 0
							1	x one-half = $\frac{1}{2}$
							1	x one = 1
							0	x two = 0
							1	x four = <u>4</u>
								$5\frac{5}{8}$ Total
	Value of bit			Position's quantity				

Adding Fractions

$$\begin{array}{r} 00010.011 \\ + 00100.110 \\ \hline 00111.001 \end{array}$$

Quiz Time!

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

Two Kinds of Integers

- Unsigned integers
 - Numbers that are always positive
- Signed integers
 - Numbers that can be positive or negative

Representing Signed Integers

- ☀ Two's complement notation
 - The most popular representation
- ☀ Excess notation
 - A less popular representation

Two's Complement Notation

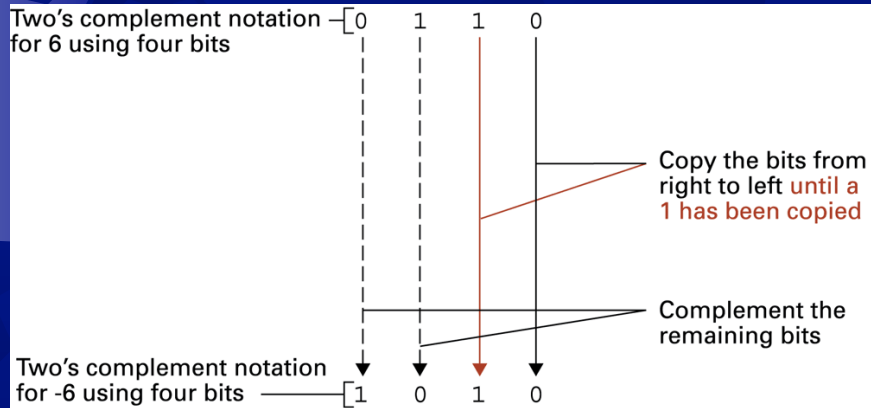
a. Using patterns of length three

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 -6 using Four Bits



Adding Numbers

Problem in base ten		Problem in two's complement		Answer in base ten
$\begin{array}{r} 3 \\ + 2 \\ \hline \end{array}$	→	$\begin{array}{r} 0011 \\ + 0010 \\ \hline 0101 \end{array}$	→	5
$\begin{array}{r} -3 \\ + -2 \\ \hline \end{array}$	→	$\begin{array}{r} 1101 \\ + 1110 \\ \hline 1011 \end{array}$	→	-5
$\begin{array}{r} 7 \\ + -5 \\ \hline \end{array}$	→	$\begin{array}{r} 0111 \\ + 1011 \\ \hline 0010 \end{array}$	→	2

Try This

$$\begin{array}{r} 5 \\ + 4 \\ \hline ? \end{array} \xrightarrow{\text{Two's Complement}} \begin{array}{r} 0101 \\ + \quad ? \\ \hline ? \end{array}$$

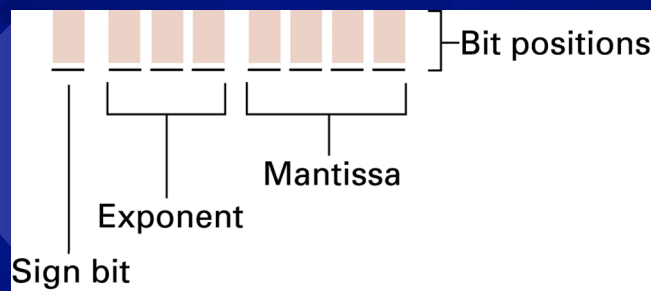
Overflow

$$\begin{array}{r} 5 \\ + 4 \\ \hline -7 \end{array} \xrightarrow{\text{Two's Complement}} \begin{array}{r} 0101 \\ + 0100 \\ \hline 1001 \end{array} \xleftarrow{\text{Decimal}}$$

Excess Notation

Bit pattern	Value represented	Bit pattern	Value represented
111	3	1111	7
110	2	1110	6
101	1	1101	5
100	0	1100	4
011	-1	1011	3
010	-2	1010	2
001	-3	1001	1
000	-4	1000	0
		0111	-1
		0110	-2
		0101	-3
		0100	-4
		0011	-5
		0010	-6
		0001	-7
		0000	-8

Floating-Point notation (8 Bits)



$$01101011 \rightarrow .1011 \times 2^{110} \rightarrow 10.11$$

$$\Rightarrow 2 \frac{3}{4}$$

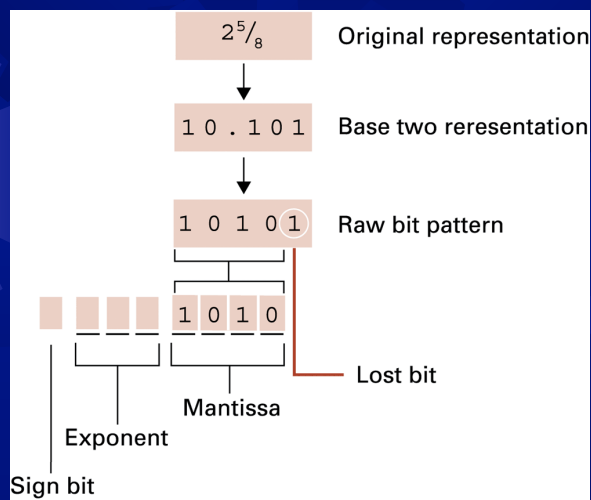
Fraction to Floating Point

$$3/8 \Rightarrow .0110 \times 2^{100} \rightarrow 01000110$$

$$3/8 \Rightarrow .1100 \times 2^{011} \rightarrow 00111100$$

$$0 \Rightarrow .0000 \times 2^{000} \rightarrow 00000000$$

Coding 2 and 5/8



Loss of Digits

$$\begin{aligned}2 \frac{1}{2} + \frac{1}{8} + \frac{1}{8} &\Rightarrow 10.1 + 0.001 + 0.001 \\ &\Rightarrow 01101010 + 00011000 + 00011000 \\ &\Rightarrow 01101010 + 01100000 + 01100000 \\ &\rightarrow 01101010 \Rightarrow 2 \frac{1}{2}\end{aligned}$$

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

Generic Data Compression

- ☀ Run-length encoding
- ☀ Relative encoding
- ☀ Frequency-dependent encoding
- ☀ Adaptive dictionary encoding

Run-Length Encoding

- ☀ Replace a sequence with
 - The repeating value
 - The number of times it occurs in the sequence

- ☀ Example:

- 111111111110000000001111111100
- (1, 11), (0, 10), (1, 8), (0, 2)

Relative Encoding

- ☀ Record the differences between consecutive data blocks
- ☀ Each block is encoded in terms of its relationship to the previous block
- ☀ Example
 - Consecutive frames of a motion picture

Frequency-Dependent Encoding

- ☀ Variable-length codes
 - “a” represented by 01
 - “q” represented by 10101
- ☀ The length of the bit pattern used to represent a data item is inversely related to the frequency of the item’s use
 - x : data item
 - $F(x)$: frequency of x appearing
 - $C(x)$: code to represent the data item
 - $|C(x)| \propto 1/F(x)$



Let's Play Game

Hangman



The Idea

- Letters e, t, a, i are used more frequently than letters z, q, x
- Use short bit patterns for letters e, t, a, i, and longer patterns for z, q, x

A Simple Example

- A 4-symbol case

- Symbol A B C D

- Without special code

- Symbol A B C D
- Code 00 01 10 11

Huffman Coding

- A 4-symbol case

- Symbol A B C D
- Probability 0.75 0.1 0.075 0.075

- Results

- Symbol A B C D
- Code 0 10 110 111

$$\langle R \rangle = 0.75 * 1 + 0.1 * 2 + 0.15 * 3 = 1.4 \text{ bits}$$

Saving with Huffman Coding

- ☀ Normally
 - 2 bits/sample for 4 symbols
- ☀ Huffman coding
 - 1.4 bits/sample on average

Adaptive Dictionary Encoding

- ☀ Dictionary
 - Collection of **building blocks** from which the message being compressed is constructed
 - **Frequent patterns**
- ☀ Dictionary is allowed to change during the encoding process
- ☀ Future occurrence as a single reference to the dictionary

Lempel-Ziv (LZ77) Decoding: xyxxyzy (5, 4, x)

x y x x y z y

a. Count backward 5 symbols.

x y x x y z y

b. Identify the four-bit segment to be appended to the end of the string.

x y x x y z y x x y z

c. Copy the four-bit segment onto the end of the message.

x y x x y z y x x y z x

d. Add the symbol identified in the triple to the end of the message.

Another Example

- Decoding xyxxyzy (5,4,x)(0,0,w)(8,6,y)
 - xyxxyzyxxyzx(0,0,w)(8,6,y)
 - xyxxyzyxxyzxw(8,6,y)
 - xyxxyzyxxyzxwzyxxyzy

Lempel-Ziv (LZ77) Encoding

- For a message, divide it into
 - A text window (~Kbytes) followed by
 - A look-ahead buffer (10~1000 Bytes)
- Then find the longest segment
 - In text window that agrees with a pattern
 - In the look-ahead buffer
- Use a triplet, (,,) to encode the segment in the look-ahead buffer
- Slide the window and buffer and repeat similar process

Lempel-Ziv Encoding: LZ77

- Look-ahead buffer: begin with 7
- Text window: 8
- Encoding `xyxxzyxyxzxwzyxyzy`
 - `xyxxzyxyxzxwzyxyzy`
`xyxxzy(5, 4,)`
 - `xyxxzyxyxzxwzyxyzy`
`xyxxzy(5, 4, x)(0, 0, w)`
 - `xyxxzyxyxzxwzyxyzy`
`xyxxzy(5, 4, x)(0, 0, w)(8, 6,)`
 - `xyxxzyxyxzxwzyxyzy`
`xyxxzy(5, 4, x)(0, 0, w)(8, 6,y)`

Compressing Images

- ☀ Images

- 3-byte-per-pixel
- Large, unmanageable maps

- ☀ Compression techniques

- GIF (Graphic Interchange Format)
- JPEG (Joint Photographic Experts Group)

GIF

- ☀ Only 256 colors

- ☀ Each pixel can be represented by a byte instead of 3

- ☀ Palette

- A table of 256 RGB combinations

- ☀ Better for these images

- Blocks of uniform colors
- Sharp edges
- Such as color cartoons

JPEG's Lossless Mode

- No information lost in encoding the picture
- Code the difference between adjacent pixels (relative encoding)
- The differences are encoded using a variable-length code (frequency-dependent encoding)
- Leads to storage size not manageable
- Seldom used

JPEG's Baseline Standard

- Human eye is more sensitive to changes in **brightness** than to changes in color
- Encoding each brightness component but dividing the image into four-pixel blocks and recording only the average color of each block



JPEG's Baseline Standard

- Save additional space by recording brightness and color **change** between components
- The final bit pattern is further compressed by applying **variable-length** encoding system
- Compression ratio about 20 to 1

Compressing Sound

- MP3
 - MPEG-1 Audio Layer 3
 - Sound are composed of sine waves of different frequencies
 - Allocate number of bits according to the responses of human ear to sound in different frequencies
 - Also uses Huffman coding for further compression
 - Compression ratio about 12 to 1

Chapter 1: Data Storage

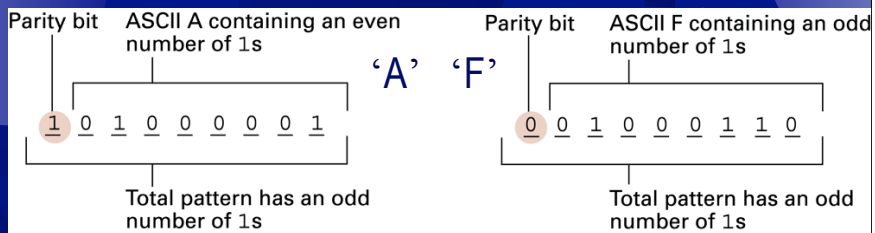
- ☀ 1.1 Bits and Their Storage
- ☀ 1.2 Main Memory
- ☀ 1.3 Mass Storage
- ☀ 1.4 Representing Information as Bit Patterns
- ☀ 1.5 The Binary System
- ☀ 1.6 Storing Integers
- ☀ 1.7 Storing Fractions
- ☀ 1.8 Data Compression
- ☀ 1.9 Communications Errors

Communication Errors

- ☀ Communication
 - Transmission of data
- ☀ Data
 - bits
- ☀ Error
 - bit pattern transferred \neq received
- ☀ Error encoding techniques
 - Allow detection
 - Allow correction of errors

Adjusted for Odd Parity

- ☀ Making sure there's an odd number of '1's



Error Correction Codes

- ☀ So that errors can be
 - Not only detected
 - But also corrected
- ☀ Hamming distance
 - The number of bits in which two bit patterns differ

Error-Correcting Code

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

- Hamming distance between A and B is 4

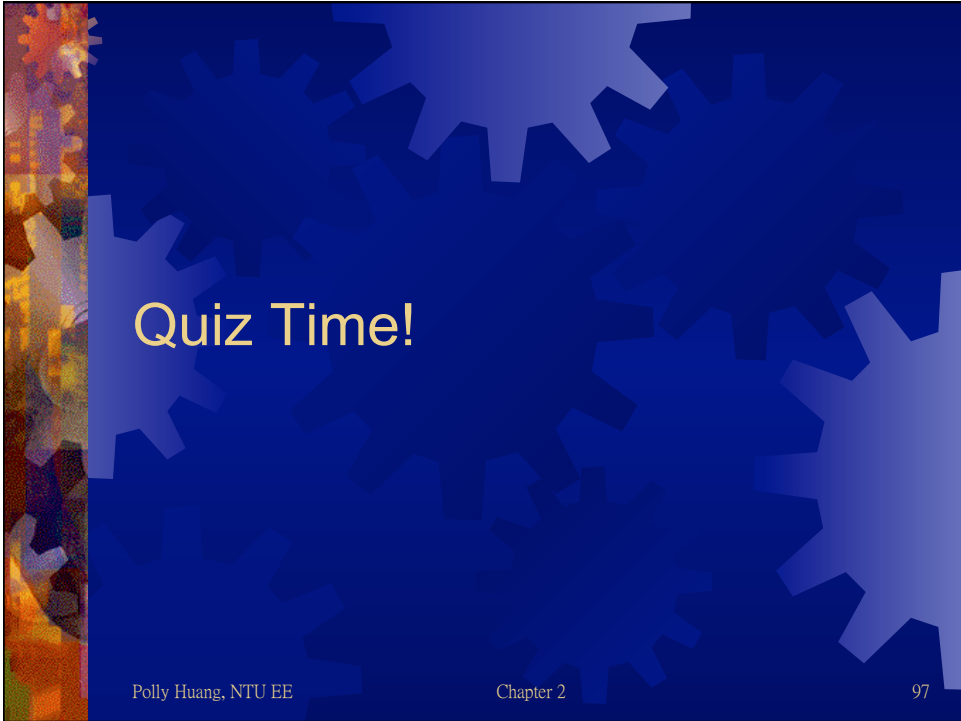
- Any two patterns are separated by a Hamming distance of at least 3

- An error bit will result in an illegal pattern

Decoding 010100

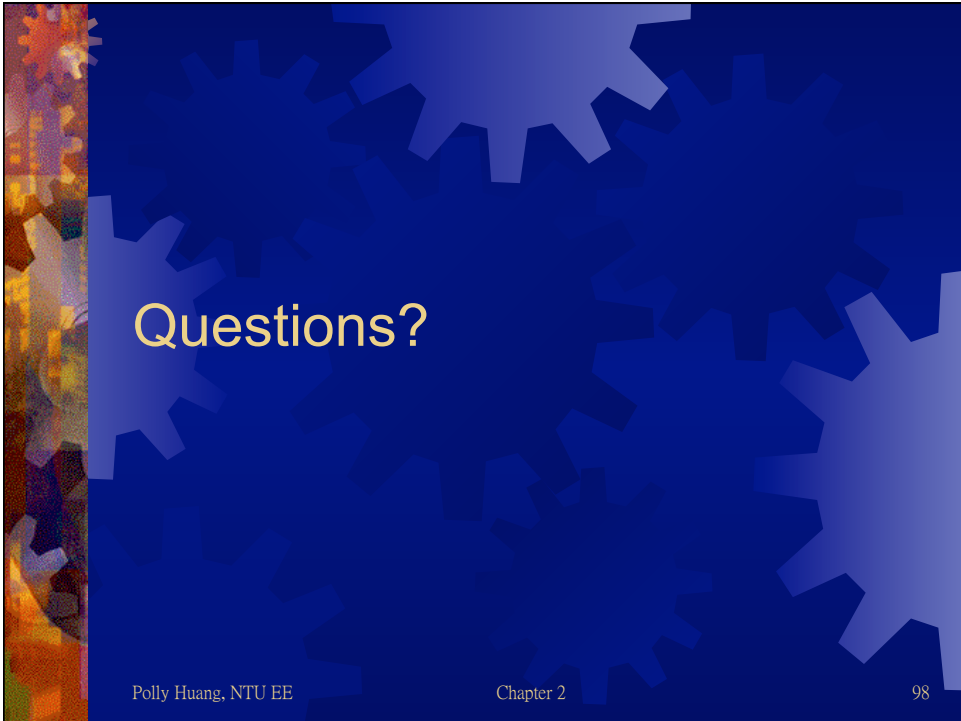
Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

Character	Distance between the received pattern and the character being considered
A	2
B	4
C	3
D	1 <i>Smallest distance</i>
E	3
F	5
G	2
H	4

A presentation slide with a dark blue background featuring a pattern of interlocking gears. The text "Quiz Time!" is centered in a light yellow font. At the bottom, there is a footer with the author's name, chapter number, and page number.

Quiz Time!

Polly Huang, NTU EE Chapter 2 97

A presentation slide with a dark blue background featuring a pattern of interlocking gears. The text "Questions?" is centered in a light yellow font. At the bottom, there is a footer with the author's name, chapter number, and page number.

Questions?

Polly Huang, NTU EE Chapter 2 98