An Introduction to Computer Science

Jingde Cheng

Southern University of Science and Technology

An Introduction to Computer Science

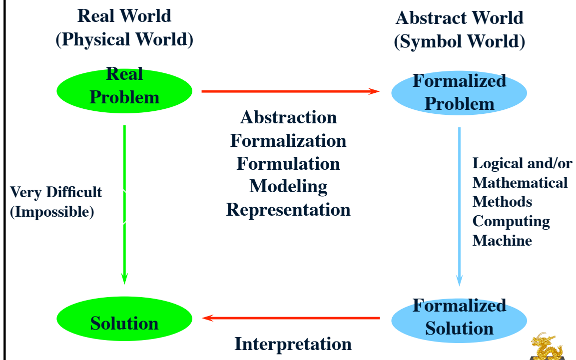
- Computer Science: What Is It and Why Study It?
- Computation: What Is It and Why Study It?
- Computability
- Computational Complexity (CS101A class only)
- Algorithms
- Data, Information, and Knowledge, and Their Representations
- Data Storage
- Computer Architecture
- Data Manipulation in Computer Systems
- Programming Languages and Compilers
- Operating Systems
- System Software and Application Software
- Software Engineering (CS101A class only)
- Knowledge Engineering and Artificial Intelligence (CS101A class only)
- Information Security Engineering (CS101A class only)

The Fundamental Question: How to Automate Computing?

How to automate computing ?

(in the electronic digital way)

Real (Physical) World and Abstract (Symbol) World



The Question: How to Represent Various Things ?

How to represent various things ?

(in the electronic digital way)

Data: What Is It?

- Datum, Data [The Oxford English Dictionary, 2nd Edition, 1989]
  - 1.a. “A thing given or granted; something known or assumed as fact, and made the basis of reasoning or calculation; an assumption or premiss from which inferences are drawn.” 1.d. “pl. The quantities, characters, or symbols on which operations are performed by computers and other automatic equipment, and which may be stored or transmitted in the form of electrical signals, records on magnetic tape or punched cards, etc.”
  - 2. “pl. Facts, esp. numerical facts, collected together for reference or information. ”

Data: What Is It?

- Data [A Dictionary of Computer Science, 7th Edition, OUP, 2016]
  - "Information, in any form, on which computer programs operate."
  - "In a more limited sense, data is distinguished from other contrasting forms of information on which computer operate, such as text, graphics, speech, and image. The distinguishing characteristic is that it is organized in a structured, repetitive, and often compressed way."

Data: What Is It?

- Data [IEEE Standard Computer Dictionary]
  - "A representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means."
  - "Anything observed in the documentation or operation of software that deviates from expectations based on previously verified software products or reference documents."
  - "Sometimes used as a synonym for documentation."

Data: What Is It?

- The concept/notion of "data"
  - Data is a collection of symbols recorded things, it represent some meaning (information) ONLY IF it is interpreted.
- Examples
  - 20080808: Password? Student No.? Account No.?
    Who's birthday? A special day to some one?
    (Consider: if Zhang Yimou see this number?)
  - 518055: What is this data?
    (Do you remember Zip code of your university?)
  - ytisrevinu nrehtuos: What is this data?
    (For your classroom exercise)

Information: What Is It?

- Information [The Oxford English Dictionary, 2nd Edition, 1989]
  - 1.a. "The action of informing (in sense 4 of the verb); formation or moulding of the mind or character, training, instruction, teaching; communication of instructive knowledge."
    1.b. "with an and pl. An item of training; an instruction."
    1.c. "Divine instruction, inspiration."
    1.d. "Capacity of informing; instructiveness."
  - 2. "The action of informing (in sense 5 of the verb); communication of the knowledge or 'news' of some fact or occurrence; the action of telling or fact of being told of something."
  - 3.a. "Knowledge communicated concerning some particular fact, subject, or event; that of which one is apprised or told; intelligence, news. spec. contrasted with data."
    3.b "with an and pl. An item of information or intelligence; a fact or circumstance of which one is told. In earlier use, An account, relation, narrative (of something)."

Information: What Is It?

- Information [The Oxford English Dictionary, 2nd Edition, 1989]
  - 3.c. "Separated from, or without the implication of, reference to a person informed: that which inheres in one of two or more alternative sequences, arrangements, etc., that produce different responses in something, and which is capable of being stored in, transferred by, and communicated to inanimate things."
    3.d. "As a mathematically defined quantity (see quots.); now esp. one which represents the degree of choice exercised in the selection or formation of one particular symbol, sequence, message, etc., out of a number of possible ones, and which is defined logarithmically in terms of the statistical probabilities of occurrence of the symbol or the elements of the message."
  - 4. "The action of informing against, charging, or accusing (a person)."

Information: What Is It?

- Information [A Dictionary of Computer Science, 7th Edition, OUP, 2016]
  - "Generally, information is whatever is capable of causing a human mind to change its opinion about the current state of the real world. Formally, and especially in science and engineering, information is whatever contributes to a reduction in the uncertainty of the state of a system; in the case, uncertainty is usually expressed in an objectively measurable form."
- Information [IEEE Standard Computer Dictionary]
  - "The meaning that humans assign to data by means of known conventions that are applied to the data."

Information: What Is It?

- Factors of "information"
  - "meaning", "humans", "assign to", "data"
- The concept/notion of "information"
  - Information is a concept/notion that holds ONLY IF there is a recipient, i.e., taking the existence of the recipient as a premise.
  - Information has some certain meaning for the recipient, therefore, it is a concept/notion that is strongly linked to and is dependent on the values of the recipient.
- Oldest example about "information": Beacon tower
  - (ancient China, Zhou Dynasty, about B.C. 1000) a fire or light set up in a high or prominent position as a warning or signal to inform (by ignition and by the speed of light) the next beacon tower that the enemy is coming quickly.

Beacon Tower (烽火台)

Data vs. Information [DL-CS-16]

- Data
  - Data is basic values or facts.
  - Data can be unstructured and lack context.
- Information
  - Information is data that has been organized and/or processed in a way that is useful in solving some kind of problem.
  - Information helps us answer questions (it "informs").

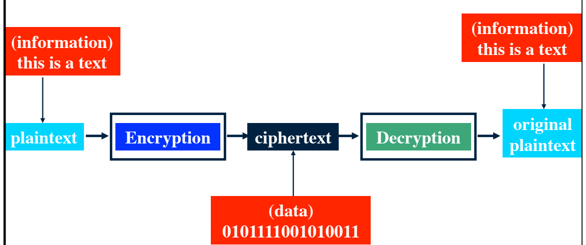
Data Basic values or facts

Information Data that has been organized or processed in a useful manner

Information vs Data: What is the Difference?

- The existence of the recipient
  - Information: Require the existence of the recipient, and is strongly linked to and is dependent on the values of the recipient.
  - Data: Do not require the existence of the recipient.
- Interpretation
  - Information: Its meaning is dependent on the values of the recipient, and has no meaning if the recipient cannot understand it.
  - Data: Has no meaning if it is not interpreted, and may have different meaning by different interpretation.

Information and Data in Encryption and Decryption



Data (NOT information!) Compression [DL-CS-16]

Data compression Reducing the amount of space needed to store a piece of data

Bandwidth The number of bits or bytes that can be transmitted from one place to another in a fixed amount of time

Compression ratio The size of the compressed data divided by the size of the uncompressed data

Lossless compression A data compression technique in which there is no loss of information

Lossy compression A data compression technique in which there is loss of information

Analog and Digital Data [DL-CS-16]

- Data representation: Analog data and digital data
  - Data can be represented in one of two ways: analog or digital.
- Analog data
  - Analog data is a continuous representation, analogous to the actual data it represents.
- Digital data
  - Digital data is a discrete representation, breaking the actual data up into separate elements.

Analog data A continuous representation of data

Digital data A discrete representation of data

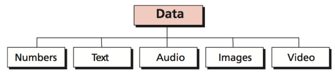
Various Types of Data [DL-CS-16] [F-CS-18]

Multimedia Several different media types

- Numbers
- Text
- Audio
- Images and graphics
- Video

Ultimately, all of this data is stored as binary digits. Each document, picture, and sound bite is somehow represented as strings of 1s and 0s. This chapter

Figure 3.1 *Different types of data*



Knowledge: What Is It?

- Knowledge [The Oxford English Dictionary, 2nd Edition, 1989]
  - "The fact of recognizing as something known, or known about, before; recognition."
  - "The fact of knowing a thing, state, etc., or (in general sense) a person; acquaintance; familiarity gained by experience."
  - "Acquaintance with a fact; perception, or certain information of, a fact or matter; state of being aware or informed; consciousness (of anything). The object is usually a proposition expressed or implied: e.g. the knowledge that a person is poor, knowledge of his poverty."
  - "Acquaintance with facts, range of information."
  - "knowledge of a person, thing, or perception gained through information or facts about it rather than by direct experience."

Knowledge: What Is It?

- Knowledge [The Oxford English Dictionary, 2nd Edition, 1989]
  - "Intellectual acquaintance with, or perception of, fact or truth; clear and certain mental apprehension; the fact, state, or condition of understanding. Formerly, also, the faculty of understanding, intelligence, intellect."
  - "Acquaintance with a branch of learning, a language, or the like; theoretical or practical understanding of an art, science, industry, etc."
  - "The fact or condition of being instructed, or of having information acquired by study or research; acquaintance with ascertained truths, facts, or principles; information acquired by study; learning; erudition."
  - "The sum of what is known."
  - "A branch of learning; a science; an art."

Knowledge: What Is It?

- Knowledge [A Dictionary of Computer Science, 7th Edition, OUP, 2016]
  - "Information that can be expressed as a set of facts and is known to an agent or program. Knowledge can be distinguished from information or data by its embodiment in an agent; for example, an agent might receive information that increases its knowledge."

Knowledge: What Is It?

- Knowledge [Wikipedia]
  - "Knowledge is a familiarity, awareness, or understanding of someone or something, such as facts, information, descriptions, or skills, which is acquired through experience or education by perceiving, discovering, or learning."
  - "Knowledge can refer to a theoretical or practical understanding of a subject. It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject); it can be more or less formal or systematic."
  - "In philosophy, the study of knowledge is called epistemology."

The Question: How to Represent Various Data ?

How to represent various data ?

(in the electronic digital way)

Examples of Numeral Systems

- **Some questions**
  - 110: what is the natural number represented by this number?
  - ABCDEF: what is the natural number represented by this "number"?
- **More questions about the questions**
  - $1 \times 10^2 + 1 \times 10^1 + 0 \times 10^0 = ?$ (denary, the decimal system)
  - $1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = ?$ (binary, the binary system)
  - $1 \times 16^2 + 1 \times 16^1 + 0 \times 16^0 = ?$ (hex, the hexadecimal system)
- **Last question**
  - What can you think of from these examples?

Numbers and Computing

- **Numbers are crucial to computing**
  - In addition to using a computer to execute numeric computations, all types of data that we store and manage using a computer are ultimately stored as numbers.
  - At the lowest level, computers store all data using just the digits **0** and **1**.
- **Numbers in Math and numbers in CS**
  - Numbers can be classified into all sorts of categories: natural numbers, negative numbers, rational numbers, irrational numbers, and many others that are important in mathematics but not to the understanding of computing.

Number: What Is It? [DL-CS-16]

- **Definition of number**
  - A number is a unit belonging to an abstract mathematical system and is subject to specified laws of succession, addition, and multiplication.
  - A number is a representation of a value, and certain arithmetic operations can be consistently applied to such values.

Number A unit of an abstract mathematical system subject to the laws of arithmetic

Positional Notation [DL-CS-16]

- **Base**
  - The base of a number system specifies the number of digits used in the system. The digits always begin with 0 and continue through one less than the base.
  - The base also determines what the positions of digits mean.

Base The foundational value of a number system, which dictates the number of digits and the value of digit positions

- **Examples**
  - There are 2 digits in base 2: 0 and 1.
  - There are 8 digits in base 8: 0 through 7.
  - There are 10 digits in base 10: 0 through 9.

Positional Notation

- **Positional notation**
  - The rightmost digit represents its value multiplied by the base to the zeroth power. The digit to the left of that one represents its value multiplied by the base to the first power. The next digit represents its value multiplied by the base to the second power, and so on.
  - If a number in the base-R number system has n digits, it is represented as follows, where d_i represents the digit in the ith position in the number:

$$d_n * R^{n-1} + d_{n-1} * R^{n-2} + \dots + d_2 * R^1 + d_1 * R^0$$

Positional notation A system of expressing numbers in which the digits are arranged in succession, the position of each digit has a place value, and the number is equal to the sum of the products of each digit by its place value[1]

The Importance of Zero [DL-CS-16]

The importance of zero

Positional notation is possible only because of the concept of zero. Zero was the fundamental concept at the intersection of all branches of modern mathematics. As Georges Ifrah noted in his book *The Universal History of Computing:* "To sum up, the vital discovery of zero gave the human mind an extraordinarily powerful potential. No other human creation has exercised such an influence on the development of mankind's intelligence."[2]

Positional Number Systems [F-CS-18]

2.2 POSITIONAL NUMBER SYSTEMS

In a **positional number system**, the position a symbol occupies in the number determines the value it represents. In this system, a number represented as:

$$\pm\,(S_{K-1} \ldots S_2 S_1 S_0 . S_{-1} S_{-2} \ldots S_{-L})_b$$

has the value of:

$$n = \pm \; S_{K-1} \times b^{K-1} + \ldots + S_1 \times b^1 + S_0 \times b^0$$
$$+ \; S_{-1} \times b^{-1} + S_{-2} \times b^{-2} + \ldots + S_{-L} \times b^{-L}$$

in which S is the set of symbols, b is the **base** (or **radix**), which is equal to the total number of symbols in the set S, and S_K and S_L are symbols in the whole and fraction parts of the number. Note that we have used an expression that can be extended from the right or from the left. In other words, the power of b can be 0 to $K - 1$ in one direction and -1 to $-L$ in the other direction. The terms with non negative powers of b are related to the integral part of the number, while the terms with negative power of b are related to the fractional part of the number. The ± sign shows that the number can be either positive or negative. We will study several positional number systems in this chapter.

Decimal Number Systems [F-CS-18]

In the decimal system, a number is written as:

$$\pm\,(S_{K-1} \ldots S_2 S_1 S_0 . S_{-1} S_{-2} \ldots S_{-L})_{10}$$

but for simplicity, we often drop the parentheses, the base, and the plus sign (if the number is positive). For example, we write the number $+(552.23)_{10}$ as 552.23—the base and plus signs are implicit.

Decimal Number Systems: Integers [F-CS-18]

Integers

An **integer** (an integral number with no fractional part) in the decimal system is familiar to all of us—we use integers in our daily life. In fact, we have used them so much that they are intuitive. We represent an integer as $\pm S_{K-1} \ldots S_1 S_0$. The value is calculated as:

$$N = \pm \; S_{K-1} \times 10^{K-1} + S_{K-2} \times 10^{K-2} + \ldots + S_2 \times 10^2 + S_1 \times 10^1 + S_0 \times 10^0$$

in which S_i is a digit, $b = 10$ is the base, and K is the number of digits.

Another way to show an integer in a number system is to use **place values**, which are powers of 10 ($10^0, 10^1, \ldots, 10^{K-1}$) for decimal numbers. Figure 2.1 shows an integer in the decimal system using place values.

Figure 2.1 *Place value for an integer in decimal system*

	10^{K-1}	10^{K-2}	...	10^2	10^1	10^0	Place values
±	S_{K-1}	S_{K-2}	...	S_2	S_1	S_0	Number
$N =$	$\pm S_{K-1} \times 10^{K-1}$ +	$S_{K-2} \times 10^{K-2}$ +	... +	$S_2 \times 10^2$ +	$S_1 \times 10^1$ +	$S_0 \times 10^0$	Values

Decimal Number Systems: Integers [F-CS-18]

Figure 2.1 *Place value for an integer in decimal system*

	10^{K-1}	10^{K-2}	...	10^2	10^1	10^0	Place values
±	S_{K-1}	S_{K-2}	...	S_2	S_1	S_0	Number
$N =$	$\pm S_{K-1} \times 10^{K-1}$ +	$S_{K-2} \times 10^{K-2}$ +	... +	$S_2 \times 10^2$ +	$S_1 \times 10^1$ +	$S_0 \times 10^0$	Values

Example 2.1

The following shows the place values for the integer +224 in the decimal system:

		10^2		10^1		10^0	Place values
		2		2		4	Number
N =	+	2×10^2	+	2×10^1	+	4×10^0	Values

Note that the digit in position 1 has the value 20, but the same digit in position 2 has the value 200. Also note that we normally drop the plus sign, but it is implicit.

Decimal Number Systems: Reals [F-CS-18]

Reals

A **real** (a number with a fractional part) in the decimal system is also familiar. For example, we use this system to show dollars and cents ($23.40). We can represent a real as $\pm S_{K-1} \ldots S_1 S_0 \bullet S_{-1} \ldots S_{-L}$. The value is calculated as:

$$R = \pm \; \underbrace{S_{K-1} \times 10^{K-1} + \ldots + S_1 \times 10^1 + S_0 \times 10^0}_{\text{Integral part}} + \underbrace{S_{-1} \times 10^{-1} + \ldots + S_{-L} \times 10^{-L}}_{\text{Fractional part}}$$

in which S_i is a digit, $b = 10$ is the base, K is the number of digits in the integral part, and L is the number of digits in the fractional part. The decimal point we use in our representation separates the fractional part from the integral part.

Example 2.3

The following shows the place values for the real number +24.13:

	10^1		10^0		10^{-1}		10^{-2}	Place values
	2		4	•	1		3	Number
$R = +$	2×10	+	4×1	+	1×0.1	+	3×0.01	Values

Binary, Octal, and Hexadecimal Number Systems

Binary number system

- **The *base-2 (binary) number system* (has two digits 0 and 1) is particularly important in computing.**
- **It is also helpful to be familiar with number systems that are powers of 2, such as base 8 (octal) and base 16 (hexadecimal).**

Octal number system

- **The *base-8 (octal) number system* has eight digits (0–7).**

Hexadecimal number system

- **The *base-16 (hexadecimal) number system* has sixteen digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.**

The Binary System

Figure 1.15 The base ten and binary systems

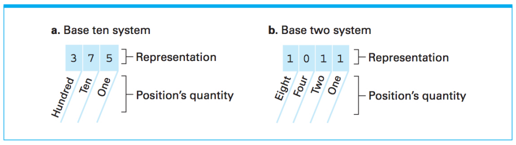
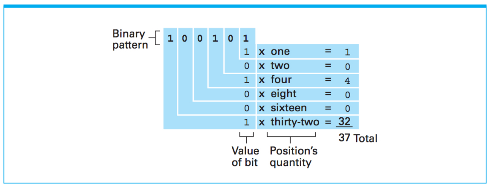


Figure 1.16 Decoding the binary representation 100101



The Binary System

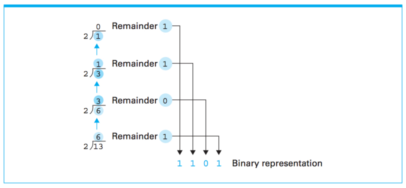
Figure 1.17 An algorithm for finding the binary representation of a positive integer

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.

Figure 1.18 Applying the algorithm in Figure 1.17 to obtain the binary representation of thirteen



Binary Values and Computers

Binary representation

- **Although some of the early computers were decimal machines, modern computers are binary machines, i.e., data and instructions within the computer are represented in binary form.**
- **The reason is that each storage location within a computer contains either a low-voltage signal or a high-voltage signal. Because each location can have only one of two states, it is logical to equate those states to 0 and 1.**

Bit, byte, and word [DL-CS-16]

Binary digit A digit in the binary number system; a 0 or a 1

Bit Binary digit

Byte Eight binary digits

Word A group of one or more bytes; the number of bits in a word is the word length of the computer

Binary Representation of Data

The inherent nature of binary representations

- **One bit can represent only two things.**
- **Two bits can represent four things because four combinations of 0 and 1 can be made from two bits: 00, 01, 10, and 11.**
- **In general, n bits can represent 2^n things because 2^n combinations of 0 and 1 can be made from n bits.**
- **Every time we increase the number of available bits by 1, we double the number of things we can represent.**

The minimum amount of storage

- **There is a minimum number of bits that a computer architecture can address and move around at one time, and it is usually a power of 2.**

Bit Combinations [DL-CS-16]

1 Bit	2 Bits	3 Bits	4 Bits	5 Bits
0	00	000	0000	00000
1	01	001	0001	00001
	10	010	0010	00010
	11	011	0011	00011
		100	0100	00100
		101	0101	00101
		110	0110	00110
		111	0111	00111
			1000	01000
			1001	01001
			1010	01010
			1011	01011
			1100	01100
			1101	01101
			1110	01110
			1111	01111
				10000
				10001
				10010
				10011
				10100
				10101
				10110
				10111
				11000
				11001
				11010
				11011
				11100
				11101
				11110
				11111

FIGURE 3.4 Bit combinations

Binary Number Systems: Integers [F-CS-18]

Integers

We can represent an integer as $\pm\ (S_{K-1} \ldots S_1\ S_0)_2$. The value is calculated as:

$$N = \pm\ S_{K-1} \times 2^{K-1} + S_{K-2} \times 2^{K-2} + \ldots + S_2 \times 2^2 + S_1 \times 2^1 + S_0 \times 2^0$$

in which S_i is a digit, $b = 2$ is the base, and K is the number of bits. Another way to show a binary number is to use place values $(2^0, 2^1, \ldots 2^{K-1})$. Figure 2.2 shows a number in the binary number system using place values:

Figure 2.2 Place values in an integer in the binary system

	2^{K-1}	2^{K-2}	$\cdots$	2^2	2^1	2^0	Place values
$\pm$	S_{K-1}	S_{K-2}	$\cdots$	S_2	S_1	S_0	Number
	↓	↓		↓	↓	↓	
$N = \pm$	$S_{K-1} \times 2^{K-1}$ +	$S_{K-2} \times 2^{K-2}$ +	$\cdots$ +	$S_2 \times 2^2$ +	$S_1 \times 2^1$ +	$S_0 \times 2^0$	Values

Binary Number Systems: Integers [F-CS-18]

Figure 2.2 Place values in an integer in the binary system

	2^{K-1}	2^{K-2}	$\cdots$	2^2	2^1	2^0	Place values
$\pm$	S_{K-1}	S_{K-2}	$\cdots$	S_2	S_1	S_0	Number
	↓	↓		↓	↓	↓	
$N = \pm$	$S_{K-1} \times 2^{K-1}$ +	$S_{K-2} \times 2^{K-2}$ +	$\cdots$ +	$S_2 \times 2^2$ +	$S_1 \times 2^1$ +	$S_0 \times 2^0$	Values

Example 2.4

The following shows that the number $(11001)_2$ in binary is the same as 25 in decimal. The subscript 2 shows that the base is 2:

	2^4	2^3	2^2	2^1	2^0	Place values
	1	1	0	0	1	Number
$N =$	1×2^4 +	1×2^3 +	0×2^2 +	0×2^1 +	1×2^0	Decimal

Note that the equivalent decimal number is $N = 16 + 8 + 0 + 0 + 1 = 25$.

Binary Number Systems: Reals [F-CS-18]

Reals

A real—a number with an optional fractional part—in the binary system can be made of K bits on the left and L bits on the right, $\pm\ (S_{K-1} \ldots S_1\ S_0 \bullet S_{-1} \ldots S_{-L})_2$. The value can be calculated as:

	Integral part	•	Fractional part
$R = \pm$	$S_{K-1} \times 2^{K-1} \times \ldots \times S_1 \times 2^1 \times S_0 \times 2^0$	+	$S_{-1} \times 2^{-1} + \ldots + S_{-L} \times 2^{-L}$

in which S_i is a bit, $b = 2$ is the base, K is the number of bits to the left, and L is the number of bits to the right of the decimal point. Note that K starts from 0, but L starts from -1. The highest power is $K - 1$ and the lowest power is $-L$.

Example 2.5

The following shows that the number $(101.11)_2$ in binary is equal to the number 5.75 in decimal:

	2^2	2^1	2^0		2^{-1}	2^{-2}	Place values
	1	0	1	•	1	1	Number
$R =$	1×2^2 +	0×2^1 +	1×2^0	+	1×2^{-1} +	1×2^{-2}	Values

Note that the value in the decimal system is $R = 4 + 0 + 1 + 0.5 + 0.25 = 5.75$.

Octal Number Systems: Integers [F-CS-18]

Integers

We can represent an integer as $\pm\ S_{K-1} \ldots S_1\ S_0$. The value is calculated as:

$$N = \pm\ S_{K-1} \times 8^{K-1} + S_{K-2} \times 8^{K-2} + \ldots + S_2 \times 8^2 + S_1 \times 8^1 + S_0 \times 8^0$$

in which S_i is a digit, $b = 8$ is the base, and K is the number of digits.

Another way to show an octal number is to use place values $(8^0, 8^1, \ldots, 8^{K-1})$. Figure 2.4 shows a number in the octal number system using place values.

Figure 2.4 Place values in an integer in the octal system

	8^{K-1}	8^{K-2}	$\cdots$	8^2	8^1	8^0	Place values
$\pm$	S_{K-1}	S_{K-2}	$\cdots$	S_2	S_1	S_0	Number
	↓	↓		↓	↓	↓	
$N = \pm$	$S_{K-1} \times 8^{K-1}$ +	$S_{K-2} \times 8^{K-2}$ +	$\cdots$ +	$S_2 \times 8^2$ +	$S_1 \times 8^1$ +	$S_0 \times 8^0$	Values

Octal Number Systems: Integers [F-CS-18]

Figure 2.4 Place values in an integer in the octal system

	8^{K-1}	8^{K-2}	$\cdots$	8^2	8^1	8^0	Place values
$\pm$	S_{K-1}	S_{K-2}	$\cdots$	S_2	S_1	S_0	Number
	↓	↓		↓	↓	↓	
$N = \pm$	$S_{K-1} \times 8^{K-1}$ +	$S_{K-2} \times 8^{K-2}$ +	$\cdots$ +	$S_2 \times 8^2$ +	$S_1 \times 8^1$ +	$S_0 \times 8^0$	Values

Example 2.7

The following shows that the number $(1256)_8$ in octal is the same as 686 in decimal:

	8^3	8^2	8^1	8^0	Place values
	1	2	5	6	Number
N =	1×8^3 +	2×8^2 +	5×8^1 +	6×8^0	Values

Note that the decimal number is $N = 512 + 128 + 40 + 6 = 686$.

Hexadecimal Number Systems: Integers [F-CS-18]

Integers

We can represent an integer as $\pm\ S_{K-1} \ldots S_1\ S_0$. The value is calculated as:

$$N = \pm\ S_{K-1} \times 16^{K-1} + S_{K-2} \times 16^{K-2} + \ldots + S_2 \times 16^2 + S_1 \times 16^1 + S_0 \times 16^0$$

in which S_i is a digit, $b = 16$ is the base, and K is the number of digits.

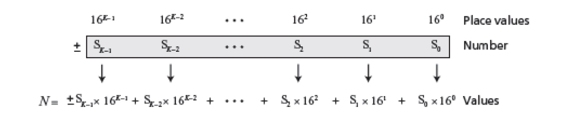
Another way to show a hexadecimal number is to use place values $(16^0, 16^1, \ldots, 16^{K-1})$. Figure 2.3 shows a number in the hexadecimal number system using place values.

Figure 2.3 Place values in an integer in the hexadecimal system

	16^{K-1}	16^{K-2}	$\cdots$	16^2	16^1	16^0	Place values
$\pm$	S_{K-1}	S_{K-2}	$\cdots$	S_2	S_1	S_0	Number
	↓	↓		↓	↓	↓	
$N = \pm$	$S_{K-1} \times 16^{K-1}$ +	$S_{K-2} \times 16^{K-2}$ +	$\cdots$ +	$S_2 \times 16^2$ +	$S_1 \times 16^1$ +	$S_0 \times 16^0$	Values

Hexadecimal Number Systems: Integers [F-CS-18]

Figure 2.3 *Place values in an integer in the hexadecimal system*



Example 2.6

The following shows that the number $(2AE)_{16}$ in hexadecimal is equivalent to 686 in decimal:

	16^2		16^1		16^0	Place values
	2		A		E	Number
N =	2×16^2	+	10×16^1	+	14×16^0	Values

Note that the value in the decimal system is $N = 512 + 160 + 14 = 686$.

Summary of Positional Number Systems [F-CS-18]

Table 2.1 Summary of the four positional number systems

System	*Base*	*Symbols*	*Examples*
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	2345.56
Binary	2	0, 1	$(1001.11)_2$
Octal	8	0, 1, 2, 3, 4, 5, 6, 7	$(156.23)_8$
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	$(A2C.A1)_{16}$

Summary of Positional Number Systems [F-CS-18]

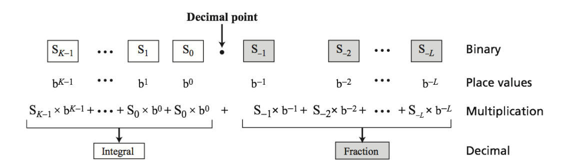
Table 2.2 Comparison of numbers in the four systems

Decimal	*Binary*	*Octal*	*Hexadecimal*
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversion between Positional Number Systems [F-CS-18]: Other Bases to Decimal

- **Any base to decimal conversion**
  - **We multiply each digit with its place value in the source system and add the results to get the number in the decimal system.**

Figure 2.5 *Converting other bases to decimal*



- **Correction: ... + $S_1 \times b^1 + S_0 \times b^0$**

Conversion between Positional Number Systems [F-CS-18]: Binary to Decimal

Example 2.8

The following shows how to convert the binary number $(110.11)_2$ to decimal: $(110.11)_2 = 6.75$:

Binary	1		1		0	•	1		1
Place values	2^2		2^1		2^0		2^{-1}		2^{-2}
Partial results	4	+	2	+	0	+	0.5	+	0.25

Decimal: 6.75

Conversion between Positional Number Systems [F-CS-18]: Octal to Decimal

Example 2.10

The following shows how to convert $(23.17)_8$ to decimal:

Octal	2		3	•	1		7
Place values	8^1		8^0		8^{-1}		8^{-2}
Partial result	16	+	3	+	0.125	+	0.109

Decimal: 19.234

This means that $(23.17)_8 \approx 19.234$ in decimal. Again, we have rounded up $7 \times 8^{-2} = 0.109375$.

Conversion between Positional Number Systems [F-CS-18]: Hexadecimal to Decimal

Example 2.9

The following shows how to convert the hexadecimal number $(1A.23)_{16}$ to decimal:

Hexadecimal	1		A	•	2		3
Place values	16^1		16^0		16^{-1}		16^{-2}
Partial result	16	+	10	+	0.125	+	0.012

Decimal: 26.137

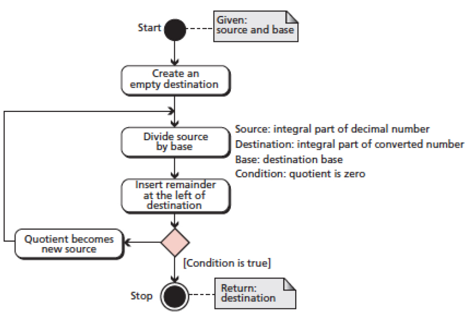
Note that the result in the decimal notation is not exact, because $3 \times 16^{-2} = 0.01171875$. We have rounded this value to three digits (0.012). In other words, $(1A.23)_{16} \approx 26.137$. When we convert a number in decimal to hexadecimal, we need to specify how many digits we allow to the right of the decimal point.

Conversion between Positional Number Systems [F-CS-18]: Decimal to Other Bases

- **Decimal to any base**
  - We need two procedures, one for the integral part and one for the fractional part.
- **Converting the integral part**
  - The integral part can be converted using repetitive division. (Figure 2.6 shows the UML diagram for the process.)
  - We call the integral part of the decimal number the source and the integral part of the converted number the destination.
  - We first create an empty destination. We then repetitively divide the source to get the quotient and the remainder. The remainder is inserted to the left of the destination. The quotient becomes a new source.
  - Figure 2.7 shows how the destination is made with each repetition.

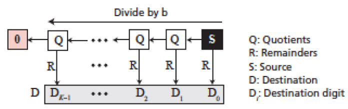
Conversion between Positional Number Systems [F-CS-18]: Decimal to Other Bases

Figure 2.6 *Algorithm to convert the integral part*



Conversion between Positional Number Systems [F-CS-18]: Decimal to Other Bases

Figure 2.7 *Converting the integral part*



Conversion between Positional Number Systems [F-CS-18]: Decimal to Binary

Example 2.11

The following shows how to convert 35 in decimal to binary. We start with the number in decimal, we move to the left while continuously finding the quotients and the remainder of division by 2. The result is $35 = (100011)_2$:

0	←	1	←	2	←	4	←	8	←	17	←	35	Decimal
		↓		↓		↓		↓		↓		↓	
		1		0		0		0		1		1	Binary

Conversion between Positional Number Systems [F-CS-18]: Decimal to Octal

Example 2.12

The following shows how to convert 126 in decimal to its equivalent in the octal system. We move to the right while continuously finding the quotients and the remainder of division by 8. The result is $126 = (176)_8$:

0	←	1	←	15	←	126	Decimal
		↓		↓		↓	
		1		7		6	Octal

Conversion between Positional Number Systems [F-CS-18]: Decimal to Hexadecimal

Example 2.13

The following shows how we convert 126 in decimal to its equivalent in the hexadecimal system. We move to the right while continuously finding the quotients and the remainder of division by 16. The result is $126 = (7E)_{16}$:

0	←	7	←	126	Decimal
		↓		↓	
		7		E	Hexadecimal

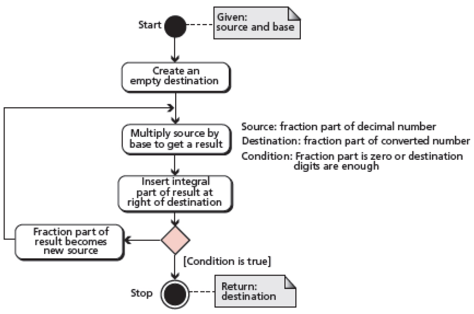
Conversion between Positional Number Systems [F-CS-18]: Decimal to Other Bases

♣ **Converting the fractional part**

- **The fractional part can be converted using repetitive multiplication.**
- **We call the fractional part of the decimal number the source and the fractional part of the converted number the destination.**
- **We first create an empty destination. We then repetitively multiply the source to get the result. The integral part of the result is inserted to the right of the destination, while the fractional part becomes the new source.**
  (Figure 2.8 shows the UML diagram for the process.)

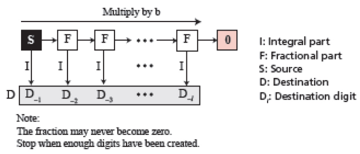
Conversion between Positional Number Systems [F-CS-18]: Decimal to Other Bases

Figure 2.8 *Algorithm to convert the fractional part*



Conversion between Positional Number Systems [F-CS-18]: Decimal to Other Bases

Figure 2.9 *Converting the fractional part*



Conversion between Positional Number Systems [F-CS-18]: Decimal to Binary

Example 2.14

Convert the decimal number 0.625 to binary.

Solution

Since the number 0.625 has no integral part, the example shows how the fractional part is calculated. The base here is 2. Write the decimal number at the left corner. Multiply the number continuously by 2 and record the integral and fractional part of the result. The fractional part moves to the right, and the integral part is recorded under each operation. Stop when the fractional part is 0 or there are enough bits. The result is $(0.101)_2$:

Decimal		0.625	→	0.25	→	0.50	→	0.00
		↓		↓		↓		
Binary	•	1		0		1		

Conversion between Positional Number Systems [F-CS-18]: Decimal to Octal/Hexadecimal

Example 2.15

The following shows how to convert 0.634 to octal using a maximum of four digits. The result is $0.634 = (0.5044)_8$. Note that we multiply by 8 (base octal):

Decimal		0.634	→	0.072	→	0.576	→	0.608	→	0.864
		↓		↓		↓		↓		
Octal	•	5		0		4		4		

Example 2.16

The following shows how to convert 178.6 in decimal to hexadecimal using only one digit to the right of the decimal point. The result is $178.6 = (B2.9)_{16}$ Note that we divide or multiply by 16 (base hexadecimal):

Decimal	0	←	11	←	178	•	0.6	→	0.6
			↓		↓		↓		
Hexadecimal			B		2	•	9		

Conversion between Positional Number Systems [F-CS-18]: Decimal to Binary

Example 2.17

An alternative method for converting a small decimal integer (usually less than 256) to binary is to break the number as the sum of numbers that are equivalent to the binary place values:

Place values	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal equivalent	128	64	32	16	8	4	2	1

Using this table, we can convert 165 to binary $(10100101)_2$ as shown below:

Decimal 165 =	128	+	0	+	32	+	0	+	0	+	4	+	0	+	1
Binary	1		0		1		0		0		1		0		1

Conversion between Positional Number Systems [F-CS-18]: Decimal to Binary

Example 2.18

A similar method can be used to convert a decimal fraction to binary when the denominator is a power of two:

Place values	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Decimal equivalent	½	¼	⅛	1/16	1/32	1/64	1/128

Using this table, we convert 27/64 to binary $(0.011011)_2$ as shown below:

Decimal = 27/64	16/64	+	8/64	+	2/64	+	1/64
	¼	+	⅛	+	1/32	+	1/64

Align these fractions according to decimal equivalent values. Note that since ½ and 1/16 are missing, we replace them with 0s:

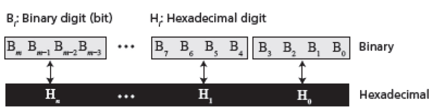
Decimal 27/64 =	0	+	¼	+	⅛	+	0	+	1/32	+	1/64
Binary	0		1		1		0		1		1

Conversion between Positional Number Systems [F-CS-18]: Binary-Hexadecimal

Binary–hexadecimal conversion

We can easily change a number from binary to hexadecimal and *vice versa*. The reason is that there is a relationship between the two bases: four bits in binary is one digit in hexadecimal. Figure 2.10 shows how this conversion can be done.

Figure 2.10 *Binary to hexadecimal and hexadecimal to binary*



Conversion between Positional Number Systems [F-CS-18]: Binary-Hexadecimal

Example 2.19

Show the hexadecimal equivalent of the binary number $(10011100010)_2$.

Solution

We first arrange the binary number in 4-bit patterns: 100 1110 0010. Note that the leftmost pattern can have one to four bits. We then use the equivalent of each pattern shown in Table 2.2 in section 2.2.5 to change the number to hexadecimal: $(4E2)_{16}$.

Example 2.20

What is the binary equivalent of $(24C)_{16}$?

Solution

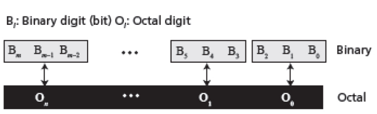
Each hexadecimal digit is converted to 4-bit patterns: 2 → 0010, 4 → 0100, and C → 1100. The result is $(001001001100)_2$.

Conversion between Positional Number Systems [F-CS-18]: Binary-Octal

Binary–octal conversion

We can easily convert a number from binary to octal and *vice versa*. The reason is that there is an interesting relationship between the two bases: three bits is one octal digit. Figure 2.11 shows how this conversion can be done.

Figure 2.11 Binary to octal conversion



Conversion between Positional Number Systems [F-CS-18]: Binary-Octal

Example 2.21

Show the octal equivalent of the binary number $(101110010)_2$.

Solution

Each group of three bits is translated into one octal digit. The equivalent of each 3-bit group is shown in Table 2.2 in section 2.2.5. The result is $(562)_8$.

Example 2.22

What is the binary equivalent of for $(24)_8$?

Solution

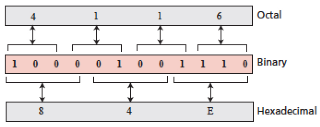
Write each octal digit as its equivalent bit pattern to get $(010100)_2$.

Conversion between Positional Number Systems [F-CS-18]: Octal-Hexadecimal

Octal–hexadecimal conversion

It is not difficult to convert a number in octal to hexadecimal or *vice versa*. We can use the binary system as the intermediate system. Figure 2.12 shows an example.

Figure 2.12 *Octal to hexadecimal and hexadecimal to octal conversion*



The following illustrates the process:

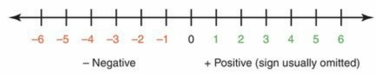
- To convert from octal to hexadecimal, we first convert the number in the octal system to binary. We then rearrange the bits in groups of four bits to find the hexadecimal equivalent.
- To convert from hexadecimal to octal, we first convert the number in the hexadecimal system to binary. We then rearrange the bits in groups of three to find the octal equivalent.

Representation of Numeric Data: Signed-magnitude Representation [DL-CS-16]

♣ **Signed-magnitude representation**

- **In the traditional decimal system, a sign (+ or –) is placed before a number's value, although the positive sign is often assumed. The sign represents the ordering, and the digits represent the magnitude of the number.**
- **Problem: There are two representations of zero -- plus zero and minus zero.**

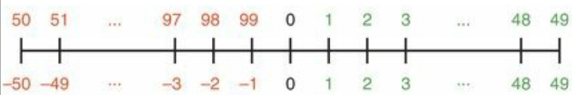
Signed-magnitude representation Number representation in which the sign represents the ordering of the number (negative and positive) and the value represents the magnitude



Representation of Numeric Data: Fixed-sized Numbers [DL-CS-16]

♣ **Fixed-sized numbers**

- **If we allow only a fixed number of values, we can represent numbers as just integer values, where half of them represent negative numbers.**
- **The sign is determined by the magnitude of the number.**
- **For example, if the maximum number of decimal digits we can represent is two, we can let 1 through 49 be the positive numbers 1 through 49 and let 50 through 99 represent the negative numbers –50 through –1.**



Representation of Numeric Data: Addition in Fixed-sized Numbers [DL-CS-16]

♣ **Addition in fixed-sized numbers**

- **We just add the numbers together and discard any carry at leftmost position.**

Signed-Magnitude	New Scheme
5 + – 6 – 1	5 + 94 99
– 4 + 6 2	96 + 6 2
– 2 + – 4 – 6	98 + 96 94

Representation of Numeric Data: Subtraction in Fixed-sized Numbers [DL-CS-16]

♣ **Subtraction in fixed-sized numbers**

- **The relationship between addition and subtraction: A – B = A + (– B).**
- **We can subtract one number from another by adding the negative of the second to the first.**

Signed-Magnitude	New Scheme		Add Negative
– 5 – 3 – 8	95 – 3	⇨	95 + 97 92

Fixed-sized Numbers: Ten's Complement Notation [DL-CS-16]

♣ **Fixed-sized numbers: ten's complement notation**

- **We can also use a formula to compute the negative representation: Negative(I) = $10^k - I$, where I is the decimal number we want to represent and k is the number of digits allowed in fixed-sized number system.**
- **Example of two-digits system: –(3) = $10^2 - 3 = 97$**
- **Example of three-digits system: –(3) = $10^3 - 3 = 997$**

♣ **Ten's complement notation**

Ten's complement A representation of negative numbers such that the negative of I is 10 raised to k minus I

Two's Complement Notation Systems

Figure 1.21 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

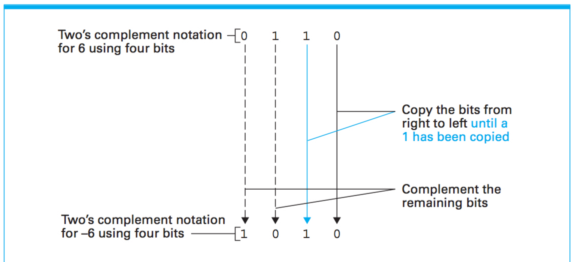
Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Two's Complement Notation Systems

- **Two's complement systems: representing positive values**
  - The system is constructed by starting with a string of 0s of the appropriate length and then counting in binary until the pattern consisting of a single 0 followed by 1s is reached.
  - These patterns represent the values 0, 1, 2, 3, ...
- **Two's complement systems: representing negative values**
  - The bit patterns representing negative values are obtained by starting with a string of 1s of the appropriate length and then counting backward in binary until the pattern consisting of a single 1 followed by 0s is reached.
  - These patterns represent the values −1, −2, −3, ...

Two's Complement Notation Systems

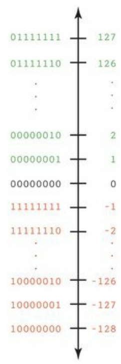
Figure 1.22 Encoding the value −6 in two's complement notation using 4 bits



Two's Complement Notation Systems [DL-CS-16]

- **Two's complement notation**
  - Let's assume that a number must be represented in eight bits, seven for the number and one for the sign.
  - The leftmost digit determines whether the number is negative or positive. A 0 bit in the leftmost digit says that the number is positive; a 1 bit says the number is negative.
  - Negative(I) = $2^7 - I$ (decimal)
  - Ex: $-(2) = 2^7 - 2 = 128 - 2 = 126$ (decimal)
    $126_{10} = 1111110_2$
  - An easier way to calculate the two's complement: invert the bits and add 1.

+2	= 00000010
invert	11111101
add 1	00000001
−2	= 11111110



Addition in Two's Complement Notation Systems

Figure 1.23 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

Summary of Integer Representations [F-CS-18]

Table 3.1 Summary of integer representations

Contents of memory	Unsigned	Sign-and-magnitude	Two's complement
0000	0	0	+0
0001	1	1	+1
0010	2	2	+2
0011	3	3	+3
0100	4	4	+4
0101	5	5	+5
0110	6	6	+6
0111	7	7	+7
1000	8	−0	−8
1001	9	−1	−7
1010	10	−2	−6
1011	11	−3	−5
1100	12	−4	−4
1101	13	−5	−3
1110	14	−6	−2
1111	15	−7	−1

Representation of Numeric Data: Representing Real Numbers [DL-CS-16]

♣ Radix point

Radix point The dot that separates the whole part from the fractional part in a real number in any base

♣ Floating point notation

- Any real value can be described by three properties: the sign (positive or negative); the mantissa, which is made up of the digits in the value with the radix point assumed to be to the right; and the exponent, which determines how the radix point is shifted relative to the mantissa.
- Examples: sign(mantissa * 10^{exp}), sign(mantissa * 2^{exp})

Floating point A representation of a real number that keeps track of the sign, mantissa, and exponent

Representation of Numeric Data: Representing Real Numbers [DL-CS-16]

♣ Floating point representation examples

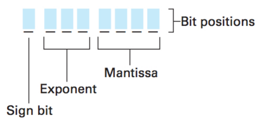
TABLE 3.1 Values in decimal notation and floating-point notation (five digits)

Real Value	Floating-Point Value
12001.00	12001 * 10^{0}
–120.01	–12001 * 10^{-2}
0.12000	12000 * 10^{-5}
–123.10	–12310 * 10^{-2}
155555000.00	15555 * 10^{4}

Representation of Numeric Data: Representing Real Numbers

♣ An example using only one byte of storage

Figure 1.26 Floating-point notation components



IEEE standards for Floating-point Representation [F-CS-18]

Figure 3.12 *IEEE standards for floating-point representation*

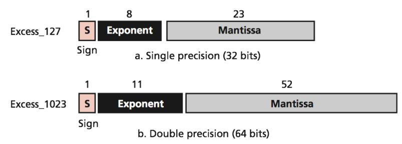


Table 3.2 Specifications of the two IEEE floating-point standards

Parameter	*Single Precision*	*Double Precision*
Memory location size (number of bits)	32	64
Sign size (number of bits)	1	1
Exponent size (number of bits)	8	11
Mantissa size (number of bits)	23	52
Bias (integer)	127	1023

Representation of Text Data

♣ Representing text

- A text document can be decomposed into paragraphs, sentences, words, and ultimately individual characters.
- To represent a text document in digital form, we simply need to be able to represent every possible character that may appear.
- The document is the continuous (analog) entity, and the separate characters are the discrete (digital) elements that we need to represent and store in computer memory.

♣ Character set [DL-CS-16]

Character set A list of the characters and the codes used to represent each one

Representing Text Symbols Using Bit Patterns [F-CS-18]

Figure 3.14 *Representing symbols using bit patterns*

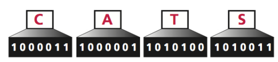


Table 3.3 Number of symbols and bit pattern length

Number of symbols	*Bit pattern length*	*Number of symbols*	*Bit pattern length*
2	1	128	7
4	2	256	8
8	3	65 536	16
16	4	4 294 967 296	32

Representation of Text Data: ASCII Character Set [DL-CS-16]

ASCII character set

- ASCII (pronounced "AS–kee") stands for *American Standard Code for Information Interchange*.
- The *American National Standards Institute* (ANSI, pronounced "AN–see") adopted the ASCII.
- The ASCII character set originally used seven bits to represent each character, allowing for 128 unique characters. The eighth bit in each character byte was originally used as a check bit, which helped ensure proper data transmission.

Latin-1 Extended ASCII character set

- Later ASCII evolved so that all eight bits were used to represent a character. This eight-bit version is formally called the *Latin-1 Extended ASCII character set*.
- The extended ASCII set allows for 256 characters and includes accented letters as well as several other special symbols.

Representation of Text Data: ASCII Character Set [DL-CS-16]

Left Digit(s) \ Right Digit	ASCII 0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	☐	!	"	#	$	%	&	'
4	(	)	*	+	,	–	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

FIGURE 3.5 The ASCII character set

Representation of Text Data: ASCII Character Set

ASCII character set

- The codes in Figure 3.5 are expressed as decimal numbers, but these values get translated to their binary equivalent for storage in the computer.

Distinct order in ASCII character set

- The ASCII characters have a distinct order based on the codes used to store them. Each character has a relative position (before or after) every other character. This property is helpful in several different ways.

Special characters in ASCII character set

- The first 32 characters in the ASCII character chart do not have a simple character representation that you could print to the screen. These characters are reserved for special purposes, such as carriage return and tab.

Representation of Text Data: ASCII Character Set

Example

- H: 072
- e: 101
- l: 108
- o: 111
- .: 046

Figure 1.13 The message "Hello." in ASCII

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

Representation of Text Data: Unicode Character Set

Unicode character set

- The extended version of the ASCII character set provides 256 characters, which is enough for English but not enough for international use.
- The goal of *Unicode character set* is nothing less than to represent every character in every language used in the entire world, including all of the Asian ideograms. It also represents many special-purpose characters such as scientific symbols.
- The Unicode uses a unique pattern of 16 bits to represent each symbol. As a result, Unicode consists of 65,536 different bit patterns -- enough to allow text written in such languages as Chinese, Japanese, and Hebrew to be represented.

ASCII character set as a subset of Unicode character set

- One convenient aspect of Unicode is that it has the ASCII characters as a subset with the same numeric values.

Representation of Text Data [DL-CS-16]

Code (Hex)	Character	Source
0041	A	English (Latin)
042F	Я	Russian (Cyrillic)
0E09	ฉ	Thai
13EA	Ꮺ	Cherokee
211E	℞	Letterlike symbols
21CC	⇌	Arrows
282F	⠯	Braille
345F	㑟	Chinese/Japanese/Korean (common)

FIGURE 3.6 A few characters in the Unicode character set

Representation of Audio Data [DL-CS-16]

♣ **Sound wave**

- We perceive sound when a series of air compressions vibrate a membrane in our ear, which sends signals to our brain.
- Thus a sound is defined in nature by the wave of air that interacts with our eardrum.

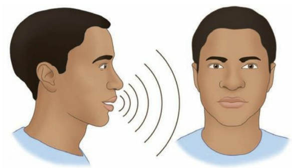


FIGURE 3.7 A sound wave vibrates our eardrums

Representation of Audio Data

♣ **Sound wave sampling**

- To represent a sound, we must somehow represent the appropriate sound wave.
- To represent audio data on a computer, we must digitize the sound wave, somehow breaking it into discrete, manageable pieces.
- An analog signal varies in voltage continuously. To digitize the signal, we periodically measure the voltage of the signal and record the appropriate numeric value. This process is called *sampling*.
- Instead of a continuous signal, we end up with a series of numbers representing distinct voltage levels.

Representation of Audio Data [DL-CS-16]

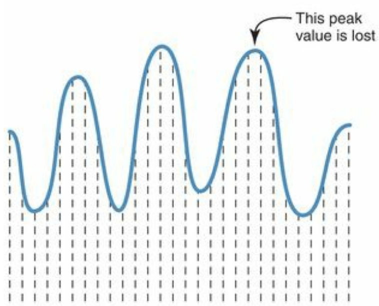


FIGURE 3.8 Sampling an audio signal

Representation of Audio Data [F-CS-18]

Figure 3.15 *An audio signal*

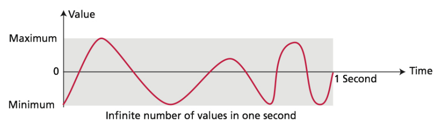
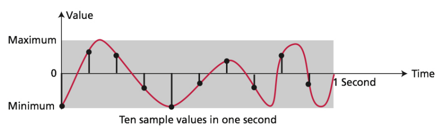
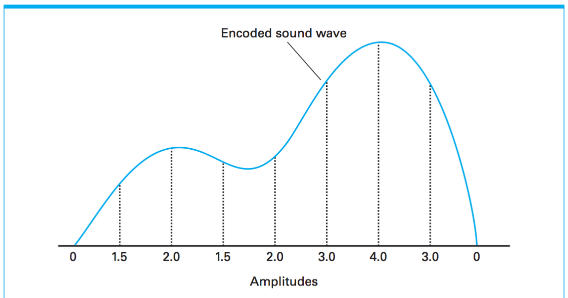


Figure 3.16 *Sampling an audio signal*



Representation of Audio Data

Figure 1.14 The sound wave represented by the sequence 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0



Representation of Audio Data

♣ **Encoding audio data**

- The most generic method of encoding audio data for computer storage and manipulation is to sample the amplitude of the sound wave at regular intervals and record the series of values obtained.
- The technique, using a sample rate of 8000 samples per second, has been used for years in long-distance voice telephone communication.
- Although 8000 samples per second may seem to be a rapid rate, it is not sufficient for high-fidelity music recordings. To obtain the quality sound reproduction obtained by today's musical CDs, a sample rate of 44,100 samples per second is used.

Audio Formats [DL-CS-16]

- **Audio formats**
  - Examples: WAV, AU, AIFF, VQF, and MP3.
  - All of these formats are based on the storage of voltage values sampled from analog signals, but all recognize the details of the data in different ways and all use various compression techniques to one extent or another.
- **MP3**
  - MP3 is short for MPEG-2, audio layer 3 file. MPEG is an acronym for the Moving Picture Experts Group, an international committee.
  - MP3 first analyzes the frequency spread and compares it to mathematical models of human psychoacoustics. Then, it discards information that can't be heard by humans. Finally, the bit stream is compressed using a form of Huffman encoding to achieve additional compression.

Representation of Image and Graphic Data [DL-CS-16]

- **Representing color**
  - Color is our perception of the various frequencies of light that reach the retinas of our eyes.
  - Our retinas have three types of color photoreceptor cone cells that respond to different sets of frequencies.
  - These photoreceptor categories correspond to the colors of red, green, and blue (*three-primary colors*). All other colors perceptible by the human eye can be made by combining various amounts of these three colors (*three color theory*).
- **RGB color model**
  - In a computer, color is often expressed as an *RGB* (red-green-blue) value, which is actually three numbers that indicate the relative contribution of each of these three primary colors.

Representation of Image and Graphic Data [DL-CS-16]

- **Color depth**
  - The amount of data that is used to represent a color is called the *color depth*. It is usually expressed in terms of the number of bits that are used to represent the color.
- **High color**
  - *High Color* indicates a *16-bit color depth*. With this scheme, 5 bits are used for each number in an RGB value and the extra bit is sometimes used to represent transparency.
- **True color**
  - *True Color* indicates a *24-bit color depth*. With this scheme, each number in an RGB value gets 8 bits, which gives the range of 0 to 255 for each. This results in the ability to represent more than 16.7 million unique colors.

Representation of Image and Graphic Data [DL-CS-16]

RGB VALUE			
Red	Green	Blue	Color
0	0	0	black
255	255	255	white
255	255	0	yellow
255	130	255	pink
146	81	0	brown
157	95	82	purple
140	0	0	maroon

Some Colors Defined in True-color [F-CS-18]

Table 3.4 Some colors defined in True-Color

Color	*Red*	*Green*	*Blue*	*Color*	*Red*	*Green*	*Blue*
Black	0	0	0	Yellow	255	255	0
Red	255	0	0	Cyan	0	255	255
Green	0	255	0	Magenta	255	0	255
Blue	0	0	255	White	255	255	255

Representation of Image and Graphic Data [DL-CS-16]

- **Digitized images and graphics**
  - Digitizing a picture is the act of representing it as a collection of individual dots called *pixels*, a term that stands for picture elements. Each pixel is composed of a single color.
  - The number of pixels used to represent a picture is called the *resolution*. If enough pixels are used (high resolution) and are then presented in the proper order side by side, the human eye can be fooled into thinking it's viewing a continuous picture.

Pixels Individual dots used to represent a picture; stands for *picture elements*

Resolution The number of pixels used to represent a picture

Representation of Image and Graphic Data [DL-CS-16]

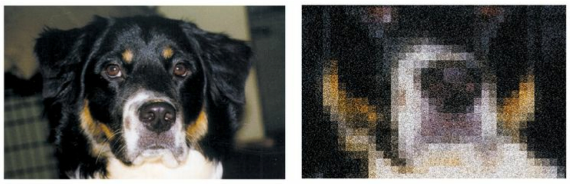


FIGURE 3.12 A digitized picture composed of many individual pixels

Representation of Image and Graphic Data [DL-CS-16]

♣ **Raster-graphics format**

- **The storage of image data on a pixel-by-pixel basis is called a *raster-graphics format*.**
- **Several popular raster-graphics file formats are currently in use, including bitmap (BMP), GIF, JPEG, and PNG.**

Raster-graphics format Storing image information pixel by pixel

Representation of Image and Graphic Data [DL-CS-16]

♣ **BMP**

- **A *bitmap* file is one of the most straightforward graphic representations.**
- **In addition to a few administrative details, a bitmap file contains the pixel color values of the image from left to right and from top to bottom.**
- **A bitmap file supports 24-bit true color, although usually the color depth can be specified to reduce the file size.**

Representation of Image and Graphic Data [DL-CS-16]

♣ **GIF**

- ***Graphics Interchange Format* (GIF) limits the number of available colors in the image to 256.**
- **A GIF image can be made up of only 256 colors, but each GIF image can be made up of a different set of 256 colors. This technique, called *indexed color*, results in smaller file sizes because there are fewer colors to reference.**
- **A version of the GIF format allows for small animations to be defined by storing a series of images that a program such as a browser displays in succession.**

Representation of Image and Graphic Data [DL-CS-16]

♣ **JPEG**

- **The *JPEG* format is designed to exploit the nature of our eyes.**
- **Humans are more sensitive to gradual changes of brightness and color over distance than we are to rapid changes. Therefore, the data that the JPEG format stores averages out the color hues over short distances.**

♣ **PNG**

- ***PNG* (pronounced “ping”) stands for *Portable Network Graphics*.**
- **PNG images can usually achieve a greater compression than GIFs, while offering a much wider range of color depths. However, PNG images do not support animation.**

Representation of Image and Graphic Data [DL-CS-16]

♣ **Vector representation of graphics**

- **A *vector-graphics format* describes an image in terms of lines and geometric shapes.**
- **A vector graphic is a series of commands that describe a line’s direction, thickness, and color.**
- **The file sizes produced with these formats tend to be small because every pixel does not have to be accounted for. The complexity of the image, such as the number of items in the picture, determines the file size.**

♣ **Resizing**

- **A raster graphic must be encoded multiple times to account for different sizes and proportions. By contrast, vector graphics can be resized mathematically, and these changes can be calculated dynamically as needed.**

Representation of Image and Graphic Data [DL-CS-16]

♣ **Drawing**

- **Vector graphics images are not good for representing real-world images, but vector graphics images are good for line art and cartoon-style drawings.**

♣ **Flash**

- **The most popular vector-graphics format used on the Web today is called Flash.**
- **Flash images are stored in a binary format and require a special editor to create.**

Representation of Video Data [DL-CS-16]

♣ **Representing video**

- **Video data is one of the most complex types of data to capture, compress, and still get a result that makes sense to the human eye.**
- **Video clips contain the equivalent of many still images, each of which must be compressed.**

♣ **Video Codecs**

- ***Codec* stands for COmpressor/DECompressor.**
- **A *video codec* refers to the methods used to shrink the size of a movie so that it can be played on a computer or over a network.**
- **Almost all video codecs use lossy compression to minimize the huge amounts of data associated with video. The goal, therefore, is not to lose information that affects the viewer's senses.**

An Introduction to Computer Science

The Question: How to Store Various Data ?

How to store various data ?

(in the electronic digital way)

George Bool (1815–1864) [DL-CS-02]

George Boole[1]

Boolean algebra is named for its inventor, English mathematician George Boole, born in 1815. His father, a tradesman, began teaching him mathematics at an early age. But Boole was initially more interested in classical literature, languages, and religion—interests he maintained throughout his life. By the time he was 20, he had taught himself French, German, and Italian. He was well versed in the writings of Aristotle, Spinoza, Cicero, and Dante, and wrote several philosophical papers himself.

At 16 he took a position as a teaching assistant in a private school to help support his family. His work there plus a second teaching job left him little time to study. A few years later, he opened a school and began to learn higher mathematics on his own. In spite of his lack of formal training, his first scholarly paper was published in the *Cambridge Mathematical Journal* when he was just 24. In 1849, he was appointed professor of mathematics at Queen's College in Cork, Ireland. He became chair of mathematics and spent the rest of his career there. Boole went on the publish over 50 papers and several major works before he died in 1864, at the peak of his career.

Boole's *The Mathematical Analysis of Logic* was published in 1847. It would eventually form the basis for the development of digital computers. In the book, Boole set forth the formal axioms of logic (much like the axioms of geometry) on which the field of symbolic logic is built. Boole drew on the symbols and operations of algebra in creating his system of logic. He associated the value 1 with the universal set (the set representing everything in the universe) and the value 0 with the empty set, and restricted his system to these quantities. He then defined operations that are analogous to subtraction, addition, and multiplication.

In 1854, Boole published *An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities*. This book described theorems built on his axioms of logic and extended the algebra to show how probabilities could be computed in a logical system. Five years later, Boole published *Treatise on Differential Equations*, followed by *Treaties on the Calculus of Finite Differences*. The latter is one of the cornerstones of numerical analysis, which deals with the accuracy of computations.

Boole received little recognition and few honors for his work. Given the importance of Boolean algebra in modern technology, it is hard to believe that his system of logic was not taken seriously until the early twentieth century. George Boole was truly one of the founders of computer science.

Bits and Their Storage: Logical (Boolean) Operations

♣ **Truth-values and bits**

- **To understand how individual bits are stored and manipulated inside a computer, it is convenient to imagine that the bit 0 represents the *truth-value* "*FALSE*" and the bit 1 represents the *truth-value* "*TRUE*" because that allows us to think of manipulating bits as manipulating *TRUE/FALSE* truth-values.**

♣ **Logical (Boolean) operations**

- **Operations that manipulate *TRUE/FALSE* truth-values are called *logical operations* or *Boolean operations* (in honor of the logician/mathematician George Boole (1815–1864)).**
- **Three of the basic logical (Boolean) operations are *AND* (*conjunction*), *OR* (*disjunction*), and *XOR* (*exclusive or, exclusive disjunction*).**

Bits and Their Storage: Logical (Boolean) Operations

Figure 1.1 The Boolean operations AND, OR, and XOR (exclusive or)

The AND operation

$$\begin{array}{r}0\\ \text{AND}\;\underline{0}\\ 0\end{array}\quad \begin{array}{r}0\\ \text{AND}\;\underline{1}\\ 0\end{array}\quad \begin{array}{r}1\\ \text{AND}\;\underline{0}\\ 0\end{array}\quad \begin{array}{r}1\\ \text{AND}\;\underline{1}\\ 1\end{array}$$

The OR operation

$$\begin{array}{r}0\\ \text{OR}\;\underline{0}\\ 0\end{array}\quad \begin{array}{r}0\\ \text{OR}\;\underline{1}\\ 1\end{array}\quad \begin{array}{r}1\\ \text{OR}\;\underline{0}\\ 1\end{array}\quad \begin{array}{r}1\\ \text{OR}\;\underline{1}\\ 1\end{array}$$

The XOR operation

$$\begin{array}{r}0\\ \text{XOR}\;\underline{0}\\ 0\end{array}\quad \begin{array}{r}0\\ \text{XOR}\;\underline{1}\\ 1\end{array}\quad \begin{array}{r}1\\ \text{XOR}\;\underline{0}\\ 1\end{array}\quad \begin{array}{r}1\\ \text{XOR}\;\underline{1}\\ 0\end{array}$$

Bits and Their Storage: Logic Gates

♣ **Logic gates**

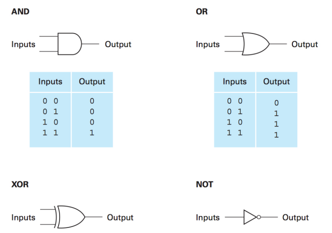
- A device that produces the output of a logical (Boolean) operation when given the operation's input truth-values is called a ***logic gate***.
- Logic gates can be constructed from a variety of technologies such as gears, relays, and optic devices.
- Today, logic gates are usually implemented as small electronic circuits in which the digits **0** and **1** are represented as voltage levels.

♣ **Logic gates as the basic building blocks for logic circuits**

- Logic gates provide the basic building blocks from which logic circuits are constructed.

Bits and Their Storage: Logic Gates

Figure 1.2 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

Logic Gates [DL-CS-16]

Gate A device that performs a basic operation on electrical signals, accepting one or more input signals and producing a single output signal

Circuit A combination of interacting gates designed to accomplish a specific logical function

Boolean algebra A mathematical notation for expressing two-valued logical functions

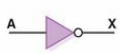
Logic diagram A graphical representation of a circuit; each type of gate has its own symbol

Truth table A table showing all possible input values and the associated output values

Logic Gates [DL-CS-16]

Boolean Expression: $X = A'$

Logic Diagram Symbol:



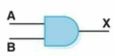
Truth Table:

A	X
0	1
1	0

FIGURE 4.1 Representations of a NOT gate

Boolean Expression: $X = A \cdot B$

Logic Diagram Symbol:



Truth Table:

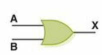
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

FIGURE 4.2 Representations of an AND gate

Logic Gates [DL-CS-16]

Boolean Expression: $X = A + B$

Logic Diagram Symbol:



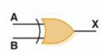
Truth Table:

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

FIGURE 4.3 Representations of an OR gate

Boolean Expression: $X = A \oplus B$

Logic Diagram Symbol:



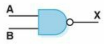
Truth Table:

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

FIGURE 4.4 Representations of an XOR gate

Logic Gates [DL-CS-16]

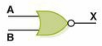
Boolean Expression	Logic Diagram Symbol	Truth Table
X = (A · B)'		



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

FIGURE 4.5 Representations of a NAND gate

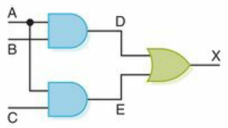
Boolean Expression	Logic Diagram Symbol	Truth Table
X = (A + B)'		



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

FIGURE 4.6 Representations of a NOR gate

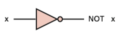
Combinational Circuits [DL-CS-16]



A	B	C	D	E	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

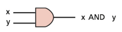
Logic Operations at the Bit Level [F-CS-18]

Figure 4.1 *Logic operations at the bit level*



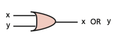
NOT

x	NOT x
0	1
1	0



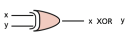
AND

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1



OR

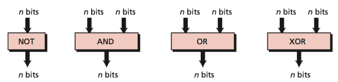
x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1



XOR

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

Figure 4.2 *Logic operators applied to bit patterns*



Bits and Their Storage: Flip-Flops

♣ **The key to storing a bit**

- ◆ **Have two possible stable states, to represent 1 and 0 respectively.**
- ◆ **Keep the stable states until some particular time point.**
- ◆ **Can switch between the two stable states according to control instruction.**

♣ **Flip-flops**

- ◆ **A *flip-flop* is a logic circuit that produces an output value of 1 or 0 as *two possible stable states*, which *remains constant* until a pulse is applied to one control input and causes it *to shift to the other value*.**
- ◆ **In other words, the output of a flip-flop will flip or flop between two values under control of external stimuli.**

Bits and Their Storage: Flip-Flops

♣ **Flip-flops as the fundamental building blocks for computers**

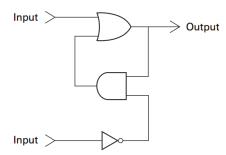
- ◆ **A flip-flop can be set to have the output value of either 1 or 0.**
- ◆ **Other circuits can adjust this value by sending pulses to the flip-flop's inputs, and still other circuits can respond to the stored value by using the flip-flop's output as their inputs.**
- ◆ **Thus many flip-flops, constructed as circuits, can be used inside a computer as a means of recording data that are encoded as patterns of 1s and 0s.**

Bits and Their Storage: An Example of Flip-Flop

♣ **Flip-flop example**

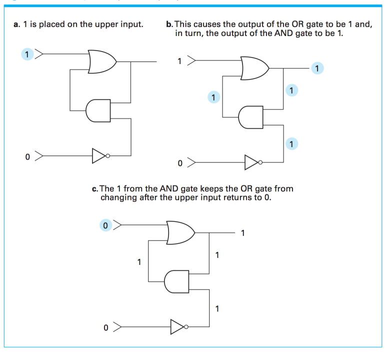
- ◆ **If both inputs in the flip-flop circuit remain 0, then the output (whether 0 or 1) will not change.**
- ◆ **However, temporarily placing a 1 on the upper input will force the output to be 1, whereas temporarily placing a 1 on the lower input will force the output to be 0.**

Figure 1.3 A simple flip-flop circuit



Bits and Their Storage: An Example of Flip-Flop

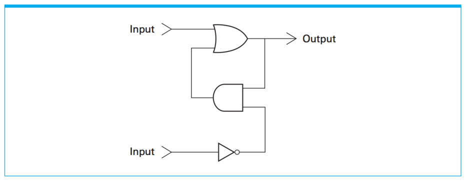
Figure 1.4 Setting the output of a flip-flop to 1



Bits and Their Storage: An Example of Flip-Flop

- The upper input: 0
- The lower input: 1
- The output of the NOT gate: 0
- The output of AND gate: 0
- The output of OR gate: 0 (this output value will persist after the lower input value returns to 0)

Figure 1.3 A simple flip-flop circuit



Bits and Their Storage: Hexadecimal Notation

- **Representing string of bits**
  - A long string of bits is often called a ***stream***.
  - Streams are difficult for the human mind to comprehend.
  - To simplify the representation of streams, we usually use a shorthand notation called ***hexadecimal notation***, which takes advantage of the fact that bit patterns within a machine tend to have lengths in multiples of four.
  - In particular, hexadecimal notation uses a single symbol to represent a pattern of four bits.
- **Examples**
  - A string of twelve bits can be represented by three hexadecimal symbols.
  - The 16-bit pattern 1010010011001000 can be reduced to the more palatable form A4C8.

Bits and Their Storage: Hexadecimal Notation

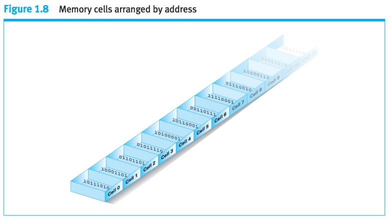
Figure 1.6 The hexadecimal encoding system

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: Memory Organization

- **Memory organization**
  - For the purpose of storing data, a computer contains a large collection of circuits (such as flip-flops), each capable of storing a single bit.
  - This bit reservoir is known as the machine's ***main memory***.

Figure 1.8 Memory cells arranged by address



Main Memory: Memory Organization

- **Memory units (cells)**
  - A computer's main memory is organized in manageable units called ***cells***, with a typical cell size being ***eight bits*** (***one byte***).
- **High-order end and low-order end of a cell**
  - Although there is no left or right within a computer, we normally envision the bits within a memory cell as being arranged in a row.
  - The left end of this row is called the ***high-order end***, and the right end is called the ***low-order end***.
  - The leftmost bit is called either the ***high-order bit*** or the ***most significant bit***, and the rightmost bit is referred to as the ***low-order bit*** or the ***least significant bit***.

Main Memory: Memory Cell

Figure 1.7 The organization of a byte-size memory cell

High-order end 0 1 0 1 1 0 1 0 Low-order end

Most significant bit

Least significant bit

Main Memory: Memory Organization

♣ **Memory unit address**

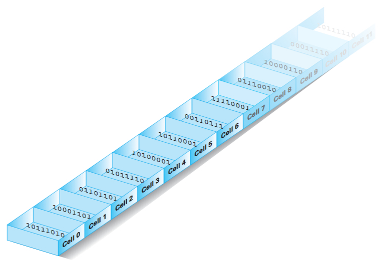
- **To identify individual cells in a computer's main memory, each cell is assigned a unique "*name*," called its *address*.**
- **We envision all the cells being placed in a single row and numbered in this order starting with the value zero.**

♣ **Ordering cells**

- **Such an addressing system not only gives us a way of uniquely identifying each cell but also associates an order to the cells, giving us phrases such as "*the next cell*" or "*the previous cell*."**
- **An important consequence of assigning an order to both the cells in main memory and the bits within each cell is that the entire collection of bits within a computer's main memory is essentially ordered in one long row.**
- **Pieces of this long row can therefore be used to store bit patterns that may be longer than the length of a single cell.**

Main Memory: Memory Cell Address

Figure 1.8 Memory cells arranged by address



Main Memory: Measuring Memory Capacity

♣ **KB**

- **The size of the memories in early computers were often measured in 1024 (which is 2^{10}) cell units.**
- **Since 1024 is close to the value 1000, the computing community adopted the prefix *kilo* in reference to this unit.**
- **The term *kilobyte* (abbreviated *KB*) was used to refer to 1024 *bytes*.**

♣ **MB, GB, TB and their misuse**

- **As memories became larger, this terminology grew to include *MB* (*megabyte*), *GB* (*gigabyte*), and *TB* (*terabyte*).**
- **Unfortunately, this application of prefixes kilo-, mega-, and so on, represents a misuse of terminology because these are already used in other fields in reference to units that are powers of a thousand.**
- **As a general rule, terms such as kilo-, mega-, etc. refer to powers of two when used in the context of a computer's memory, but they refer to powers of a thousand when used in other contexts.**

Main Memory [F-CS-18]

Figure 5.3 *Main memory*

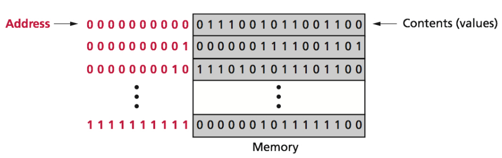


Table 5.1 Memory units

Unit	*Exact Number of Bytes*	*Approximation*
kilobyte	2^{10} (1024) bytes	10^3 bytes
megabyte	2^{20} (1048576) bytes	10^6 bytes
gigabyte	2^{30} (1073741824) bytes	10^9 bytes
terabyte	2^{40} bytes	10^{12} bytes

Main Memory: Measuring Memory Capacity [DL-CS-16]

Power of 10	Power of 2	Value of Power of 2	Prefix	Abbreviation	Derivation
10^{-12}			pico	p	Italian for *little*
10^{-9}			nano	n	Greek for *dwarf*
10^{-6}			micro	μ	Greek for *small*
10^{-3}			milli	m	Latin for *thousandth*
10^{3}	2^{10}	1024	kilo	K	Greek for *thousand*
10^{6}	2^{20}	1,048,576	mega	M	Greek for *large*
10^{9}	2^{30}	1,073,741,824	giga	G	Greek for *giant*
10^{12}	2^{40}	not enough room	tera	T	Greek for *monster*
10^{15}	2^{50}	not enough room	peta	P	Greek prefix for *five*

Main Memory: Memory Organization

- **RAM**
  - ***Random Access Memory (RAM)*** **makes up most of the main memory in a computer.**
  - **Because a computer's main memory is organized as individual, addressable cells, the cells can be accessed independently as required.**
  - **Note: The term "RAM" is confusing, because ROM can also be accessed randomly. What distinguishes RAM from ROM is that RAM can be read from and written to.**
- **ROM**
  - **In addition to RAM, most computers contain a second kind of memory, called *ROM*. ROM stands for *Read-Only Memory*.**

Main Memory: RAM vs. ROM

- **RAM**
  - **The CPU can write data to RAM and later overwrite it.**
  - **RAM is volatile, i.e., the data is lost if the computer is powered down.**
  - **RAM technology is divided into two broad categories: SRAM and DRAM.**
- **ROM**
  - **The contents in ROM are permanent and cannot be changed by a stored operation.**
  - **ROM is non-volatile, i.e., its contents are not lost if you turn off the computer.**
  - **Placing the bit pattern in ROM is called *burning* (either at the time the ROM is manufactured or at the time the computer parts are assembled).**

Main Memory: SRAM vs. DRAM [F-CS-18]

- **SRAM**
  - ***Static RAM (SRAM)*** **technology uses traditional flip-flop gates to hold data. The gates hold their state (1 or 0), which means that data is stored as long as the power is on and there is no need to refresh memory locations.**
  - **SRAM is fast but expensive.**
- **DRAM**
  - ***Dynamic RAM (DRAM)*** **technology uses capacitors, electrical devices that can store energy, for data storage. If a capacitor is charged, the state is 1; if it is discharged, the state is 0. Because a capacitor loses some of its charge with time, DRAM memory cells need to be refreshed periodically.**
  - **DRAM is slow but inexpensive.**

Mass (Secondary) Storage Systems

- **Mass storage**
  - **Due to the volatility and limited size of a computer's main memory, most computers have additional memory devices called *mass storage* (or *secondary storage*) systems, including magnetic disks, magnetic tapes, CDs, DVDs, and flash drives.**
- **Advantages**
  - **Less volatility, large storage capacities, low cost, and the ability to remove the storage medium from the machine for archival purposes.**
- **Disadvantages**
  - **Require mechanical motion, and therefore, require significantly more time to store and retrieve data than a machine's main memory.**

Mass Storage: Magnetic Disk

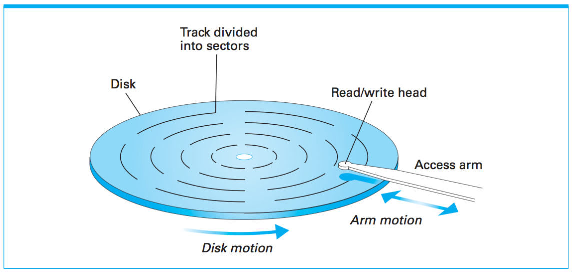
- **Magnetic disks: Track**
  - **In a *magnetic disk*, a thin spinning disk with magnetic coating is used to hold data.**
  - **Read/write heads are placed above and/or below the disk so that as the disk spins, each head traverses a circle, called a *track*.**
  - **By repositioning the read/write heads, different concentric tracks can be accessed.**
- **Magnetic disks: Cylinder**
  - **In many cases, a disk storage system consists of several disks mounted on a common spindle, one on top of the other, with enough space for the read/write heads to slip between the platters.**
  - **In such cases, the read/write heads move in unison. Each time the read/ write heads are repositioned, a new set of tracks, which is called a *cylinder*, becomes accessible.**

Mass Storage: Magnetic Disk

- **Magnetic disks: Sector**
  - **Since a track can contain more data than we would normally want to manipulate at any one time, each track is divided into small arcs called *sectors* on which data is recorded as a continuous string of bits.**
  - **All sectors on a disk contain the same number of bits (typical capacities are in the range of 512 bytes to a few KB), and in the simplest disk storage systems each track contains the same number of sectors.**
  - **In high capacity disk storage systems, the tracks near the outer edge are capable of containing significantly more sectors than those near the center, and this capability is often utilized by applying a technique called *zoned-bit recording*.**
- **Magnetic disks: Formatting**
  - **The location of tracks and sectors is not a permanent part of a disk's physical structure. Instead, they are marked magnetically through a process called *formatting* (or *initializing*) the disk.**

Mass Storage: Magnetic Disk Storage System

Figure 1.9 A disk storage system



Mass Storage: Magnetic Disk Storage System [DL-CS-16]

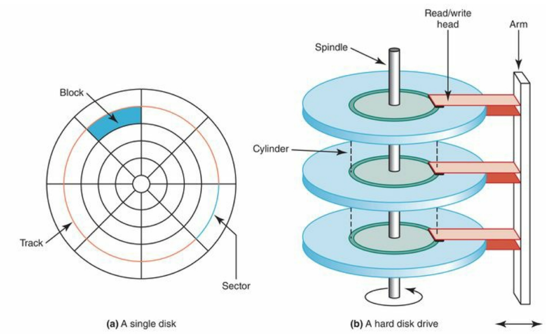
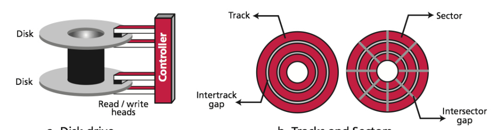


FIGURE 5.5 The organization of a magnetic disk

Mass Storage: Magnetic Disk Storage System [F-CS-18]

Figure 5.6 *A magnetic disk*



Mass Storage: Magnetic Disk

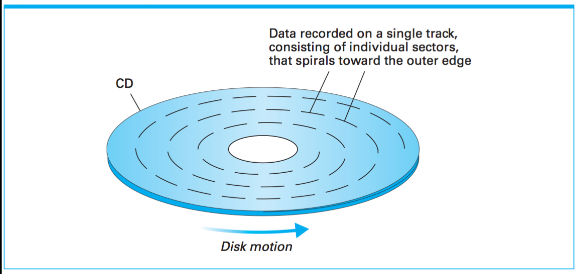
- **Measurements for evaluating a disk system's performance**
  - **(1) seek time (the time required to move the read/write heads from one track to another);**
  - **(2) rotation delay or latency time (half the time required for the disk to make a complete rotation, which is the average amount of time required for the desired data to rotate around to the read/write head once the head has been positioned over the desired track);**
  - **(3) access time (the sum of seek time and rotation delay); and**
  - **(4) transfer rate (the rate at which data can be transferred to or from the disk).**

Mass Storage: Compact Disk Storage System

- **Compact disk (CD)**
  - ***Compact disks* are 12 centimeters (approximately 5 inches) in diameter and consist of reflective material covered with a clear protective coating.**
  - **Data is recorded on them by creating variations in their reflective surfaces.**
  - **This data can then be retrieved by means of a laser beam that detects irregularities on the reflective surface of the CD as it spins.**
- **CD-DA**
  - **CD technology was originally applied to audio recordings using a recording format known as *CD-DA* (*compact disk-digital audio*), and the CDs used today for computer data storage use essentially the same format.**

Mass Storage: Compact Disk Storage System

Figure 1.11 CD storage format



Mass Storage: Flash Memory

- **Physical motion leads to slow speed**
  - A common property of mass storage systems based on magnetic or optic technology is that physical motion, such as spinning disks, moving read/write heads, and aiming laser beams, is required to store and retrieve data. This means that data storage and retrieval is slow compared to the speed of electronic circuitry.
- **Flash memory**
  - In a ***flash memory*** system, bits are stored by sending electronic signals directly to the storage medium where they cause electrons to be trapped in tiny chambers of silicon dioxide, thus altering the characteristics of small electronic circuits.
  - Since these chambers are able to hold their captive electrons for many years, this technology is suitable for off-line storage of data.

File Storage and Retrieval

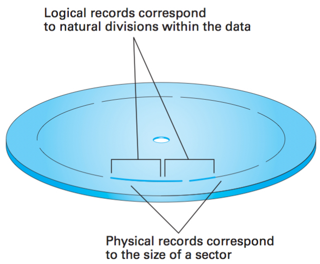
- **File**
  - A ***file*** is a named collection of related data. A file may be considered a sequence of bits, bytes, lines, or ***records***, depending on how you look at it.
  - From the user's point of view, a file is the smallest amount of data that can be written to secondary memory.
  - Organizing everything into files presents a uniform view for data storage.
- **File system**
  - A ***file system*** is the logical view that an operating system provides so that users can manage data as a collection of files.
  - A file system is often organized by grouping files into ***directories***.

File Storage and Retrieval

- **Physical record**
  - A file stored on a magnetic disk must be manipulated by sectors, each of which is a fixed predetermined size.
  - A block of data conforming to the specific characteristics of a storage device is called a ***physical record***.
  - Thus, a large file stored in mass storage will typically consist of many physical records.
- **Logical record**
  - In contrast to this division into physical records, a file often has natural divisions determined by the data represented.
  - These naturally occurring blocks of data are called ***logical records***.

File Storage and Retrieval

Figure 1.12 Logical records versus physical records on a disk



File Storage and Retrieval

- **Field**
  - Logical records often consist of smaller units called ***fields***.
- **Key field and key**
  - Sometimes each logical record within a file is uniquely identified by means of a particular field within the record. Such an identifying field is called a ***key field***.
  - The value held in a key field is called a ***key***.

File System [DL-CS-16]

- **File system**
  - A file may be considered a sequence of bits, bytes, lines, or records, depending on how you look at it.
  - The creator of a file decides how the data in a file is organized, and any users of the file must understand that organization.
- **File types**
  - All files can be broadly classified as either text files or binary files.

File A named collection of data, used for organizing secondary memory

File system The operating system's logical view of the files it manages

Directory A named group of files

File System: Text File vs. Binary File [DL-CS-16]

- **Text file**
  - In a text file, the bytes of data are organized as characters from the ASCII or Unicode character sets.
- **Binary file**
  - In a binary file, the bytes of data are organized as bit patterns.
  - A binary file requires a specific interpretation of the bits based on the data in the file.

Text file A file that contains characters

Binary file A file that contains data in a specific format, requiring a special interpretation of its bits

File System: Text File vs. Binary File [DL-CS-16]

- **Note**
  - The terms "text file" and "binary file" are somewhat misleading.
  - They seem to imply that the data/information in a text file is not stored as binary data.
- **An important fact**
  - Ultimately, all data on a computer is stored as binary digits.
  - These terms refer to how those bits are formatted: as chunks of 8 or 16 bits, interpreted as characters, or in some other special format.

An Introduction to Computer Science

- Computer Science: What Is It and Why Study It?
- Computation: What Is It and Why Study It?
- Computability
- Computational Complexity (CS101A class only)
- Algorithms
- Data, Information, and Knowledge, and Their Representations
- Data Storage
- Computer Architecture
- Data Manipulation in Computer Systems
- Programming Languages and Compilers
- Operating Systems
- System Software and Application Software
- Software Engineering (CS101A class only)
- Knowledge Engineering and Artificial Intelligence (CS101A class only)
- Information Security Engineering (CS101A class only)