Chapter 2
Data Representation on Bio-molecular Computer

The two terms "information" and "data" are alternatively used in this chapter. As discussed in Chap. 1, computing characteristic for bio-molecular computer satisfies the von Neumann architecture. Therefore, bio-molecular computer is an information processing machine. Before we can talk about how to deal with data, you need to fully understand the nature of data. In this chapter, we introduce the different information and how they are stored in tubes in bio-molecular computer.

2.1 Introduction to Data Types

Today information can be represented in different forms of data such as audio, image, video, text, number and so on. Figure 2.1 is used to explain that information is made of five different types of data. From Fig. 2.1, music can be represented in the form of audio data and your voice also can be represented in the form of audio data. Similarly, photos can be regarded as image data and movies can also be regarded as video data. Documents can be generally regarded as text data, and integers that are made of digits are regarded as number data. The term "multimedia" is applied to denote information that includes audio, image, video, text, and number.

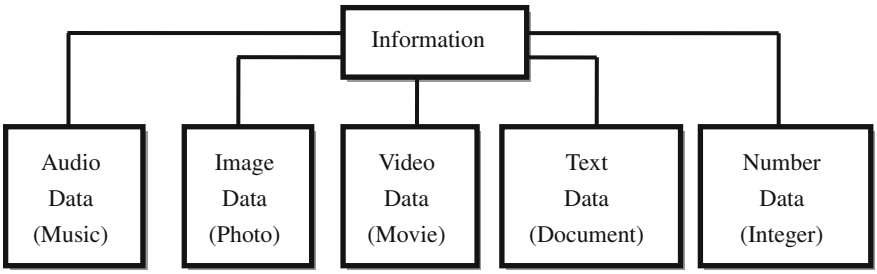

Fig. 2.1 Information is made of five different types of data

W.-L. Chang and A. V. Vasilakos, *Molecular Computing*, Studies in Big Data 4,
DOI: 10.1007/978-3-319-05122-2_2,

2.2 Data Representation for Bio-molecular Computer

The interesting question is to how to handle all these data types in Fig. 2.1. Of course, the most efficient solution for the interesting question is to use a uniform representation of data. All data types from outside bio-molecular computer are transformed into this uniform representation when they are stored in tubes in bio-molecular computer, and then transformed back when leaving tubes in bio-molecular computer. This universal format is called a *bit pattern*.

Before further discussion of bit patterns, we must define a bit. A bit (binary digit) is the smallest unit of data that can be stored in tubes in bio-molecular computer; it is either 0 or 1. For a bit in tubes in bio-molecular computer, different sequences of bio-molecules can be used to represent its two states (either 0 or 1). For example, two different sequences of bio-molecules can be regarded as the on state and the off state of a switch in a digital computer. The convention is to represent the on state as 1 and the off state as 0. Therefore, it is very clear that two different sequences of bio-molecules can be applied to represent a bit. In other words, two different sequences of bio-molecules can be employed to store one bit of information. Today, data can be represented different sequences of bio-molecules and also stored in tubes in bio-molecular computer.

A single bit cannot possibly solve the data representation problem. Hence, a bit pattern or a string of bits is used to solve the problem. A bit pattern made of 16 bits is shown in Fig. 2.2. It is a combination of 0s and 1s. This is to say that if a bit pattern made of 16 bits can be stored in a tube in bio-molecular computer, then 32 different sequences of bio-molecules are needed.

1010101001010101

Fig. 2.2 A bit pattern is made of 16 bits

A tube in bio-molecular computer is just used to store the data as bit patterns. It does not know what type of data a stored bit pattern represents. The designer for a bit pattern is responsible for interpreting a bit pattern as number, text or some other

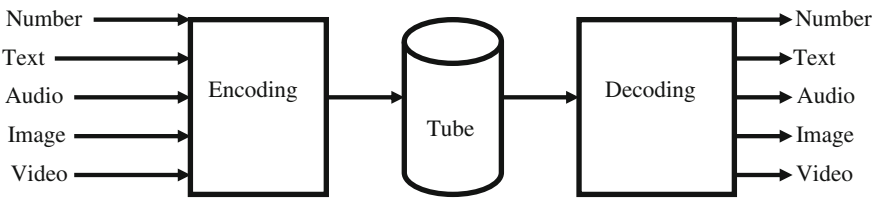

Fig. 2.3 Examples of bit patterns

type of data. In other words, data are encoded when they are stored in a tube and decoded when they are presented to the designer (Fig. 2.3). By tradition, a bit pattern of length 8 is called a byte. This term is used to measure the size of data stored in a tube in bio-molecular computer. For example, generally speaking, a tube in bio-molecular computer that can be applied to store 10^{15} bits of information is said to have 1.25×10^{14} bytes of information.

2.3 Hexadecimal Notation

The bit pattern is designed to represent data when they are stored in tubes in bio-molecular computer. However, to manipulate bit patterns is found to be difficulty for people. Using a long stream of 0s and 1s is tedious and prone to error. Hexadecimal notation is applied to improve this situation. Hexadecimal notation is based on 16 (hexadec is Greek for 16). This implies that 16 symbols (hexadecimal digits): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Each hexadecimal digit can be represented by four bits, and four bits also can be represented by a hexadecimal digit. The relationship between a bit pattern and a hexadecimal digit is shown in Table 2.1.

Table 2.1 The corresponding table has the relation among hexadecimal digits and binary digits

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

Converting from a bit pattern to hexadecimal is done by organizing the pattern into groups of four and finding the hexadecimal value for each group of 4 bits. For hexadecimal to bit pattern conversion, convert each hexadecimal digit to its 4-bit equivalent (Fig. 2.4). Generally speaking, hexadecimal notation is written in two formats. In the first format, a lowercase (or uppercase) x is added before the digits to show that the representation is in hexadecimal. For example, xCFD8 is applied to represent a hexadecimal value in this convention. In another format, the base of the number (16) is indicated as the subscript after the notation. For example, $(CFD8)_{16}$ shows the same value in the second convention. In this book, we use both conventions.

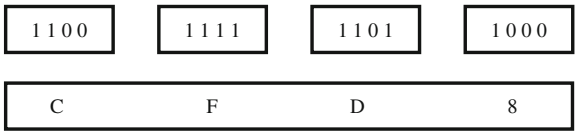

Fig. 2.4 Binary to hexadecimal and hexadecimal to binary transformation

2.4 Octal Notation

Another notation used to group bit patterns together is octal notation. Octal notation is based on 8 (*oct* is Greek for 8). This implies that there are eight symbols (octal digits): 0, 1, 2, 3, 4, 5, 6, and 7. An octal digit can represent three bits, and three bits can be represented by an octal digit. The relationship between a bit pattern and an octal digit is shown in Table 2.2. From Table 2.2, the first bit pattern 000 corresponds to the first octal digit 0, the second bit pattern 001 corresponds to the second octal digit 1, and so on with that the last bit pattern 111 corresponds to the last octal digit 7.

Table 2.2 The corresponding table has the relation among octal digits and binary digits

Bit pattern	Octal digit	Bit pattern	Octal digit
000	0	100	4
001	1	101	5
010	2	101	6
011	3	111	7

Converting from a bit pattern to octal is performed through organizing the pattern into groups of three and finding the octal value for each group of three bits. For octal to bit pattern conversion, convert each octal digit to its 3-bit equivalent (Fig. 2.5). Generally speaking, octal notation is written in two formats. In the first format, a 0 (zero) is added before the digits to show that the representation is in octal. For example, the value, 04756, is applied to represent an octal value in this convention. In another format, the base of the number (8) is indicated as the subscript after the notation. For example, $(4756)_8$ shows the same value in the second convention. In this book, we use both conventions.

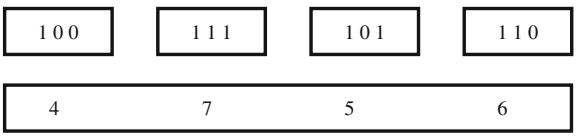

Fig. 2.5 Binary to octal and octal to binary transformation

2.5 Summary

In this chapter, an introduction to data representation of bio-molecular computer was provided. We described information that was made of different types of data and used a bit pattern as a uniform representation of data. We then introduced a bit that is the smallest unit of data that can be stored in tubes in bio-molecular computer, and used different sequences of bio-molecules to encode its two states (either 0 or 1). We also introduced a designer to a bit pattern that was responsible for interpreting a bit pattern. We then described a hexadecimal system in which its base is 16 and we used sixteen symbols to represent numbers. Simultaneously, we also introduced conversion from a binary system to a hexadecimal system and from a hexadecimal system to a binary system. We then described an octal system in which its base is 8 and we used eight symbols to represent numbers. Similarly, we also introduced conversion from a binary system to an octal system and from an octal system to a binary system.

2.6 Bibliographical Notes

In this chapter for more details about data types in a digital computer, the recommended books are Forouzan and Mosharraf (2008); Koren (2001); Marques and Silva (2012); Miano (1999). For a more detailed introduction to data representation in bio-molecular, the recommended books are Amos (2005); Ehrenfeucht et al. (2004); Paun et al. (1998). For more details about the subjects of a number system discussed in a digital computer, the recommended books are Forouzan and Mosharraf (2008); Mano (1979); Reed (2008); Shiva (2008).

2.7 Exercises

2.1. For a digital computer and bio-molecular computer, a bit is the smallest unit of data in which its vale either 1 or 0. Answer the following questions about how to encoding a bit:

   a. How are to a bit its values 0 and 1 encoded in a digital computer?
   b. How are to a bit its values 0 and 1 encoded in bio-molecular computer?

2.2. A bit pattern is a uniform representation of information. Answer the following questions about how to encoding a bit pattern:

   a. How is a bit pattern encoded in a digital computer?
   b. How is a bit pattern encoded in bio-molecular computer?

2.3 Write a program of a digital computer to convert a *hexadecimal* number to its corresponding *binary* number.

2.4 Write a program of a digital computer to convert a *binary* number to its corresponding *hexadecimal* number.
2.5 Write a program of a digital computer to convert an *octal* number to its corresponding *binary* number.
2.6 Write a program of a digital computer to convert a *binary* number to its corresponding *octal* number.

References

A. Ehrenfeucht, T.H.I. Petre, D.M. Prescott, G. Rozenberg, *Computation in Living Cells: Gene Assembly in Ciliates*. (Springer, Hidelberg, 2004). ISBN: 3540407952
B. Forouzan, F. Mosharraf, *Foundations of Computer Science*, 2nd edn. (Thomson, London, 2008). ISBN: 978-1-84480-700-0)
D. Reed, *A Balanced Introduction to Computer Science*. (Pearson Prentice Hall, New Jersy, 2008). ISBN: 9780136017226
G. Paun, G. Rozenberg, A. Salomaa, *DNA Computing: New Computing Paradigms*. (Springer, Hidelberg, 1998). ISBN: 3540641963
I. Koren, *Computer Arithmetic Algorithms*. (A k Peters, Natick, 2001). ISBN: 1568811608
J. Miano, *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP*. (Addison Wesley, Boston, 1999). ISBN: 0201604434
M. Marques, D. Silva, *Multimedia Communications and Networking*. (CRC Press, New York, 2012). ISBN: 978-1439874844
M. Amos, *Theoretical and Experimental DNA Computation*. (Springer, Hidelberg, 2005). ISBN: 9783540657736
M.M. Mano, *Digital Logic and Computer Design*. (Prentice-Hall, New Jersy, 1979). ISBN: 0-13-214510-3
S.G. Shiva, *Computer Organization, Design, and Architecture*. (CRC Press, Boca Raton, 2008). ISBN: 9780849304163

http://www.springer.com/978-3-319-05121-5

Molecular Computing
Towards a Novel Computing Architecture for Complex Problem Solving
Chang, W.-L.; Vasilakos, A.V.
2014, XIII, 271 p. 229 illus., Hardcover
ISBN: 978-3-319-05121-5