Chapter Three: Number Systems

Learning Objectives:
By the end of the chapter, students should be able to:
1. Count in basic number systems using the principle of reset and carry forward.
2. Convert a number from binary to decimal, and vice versa.
3. Convert a number from binary to hexadecimal, and vice versa.
4. Convert a number from hexadecimal to decimal, and vice versa.
5. Apply knowledge of number systems in simple computer systems contexts.

Introduction

You probably already know what a number system is. Ever heard of binary numbers or hexadecimal numbers? Simply put, a number system is a way to represent numbers. We are used to using the base-10 number system, which is also called decimal. Other common number systems include base-16 (hexadecimal) and base-2 (binary). In this chapter, we will explain what these different number systems are, how to convert between one base and another, and apply these knowledge in simple computer systems contexts.

3.1 Decimal Number System

The **decimal** number system is the number system we use in daily life. It is a number system that uses 10 unique symbols, i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. With these ten symbols, we can represent any quantity.

How do we usually count in decimal? (Principle of **Reset + Carry Forward**)
When we run out of symbols, we go to the next digit placement. For example, to represent one higher than 9, we use '10' (ten), which essentially is '1' and '0' put side by side. '10' refers to one unit of ten and zero units of one. This may seem elementary, but it is crucial to understand this if you want to understand other number systems. All other number systems work in this same principle.

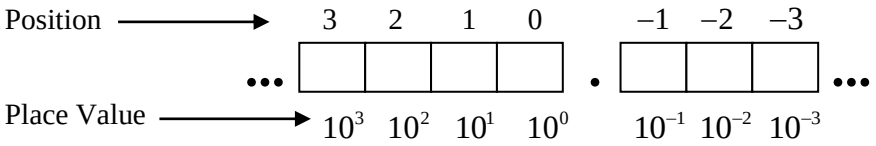
Example 3.1
List down the first 30 numbers in decimal number system. Observe the principle of 'reset and carry forward' while you are listing the numbers.

Since decimal number system uses 10 symbols, decimal is said to be "**base-10**" number system.

The placement of a symbol indicates how much it is worth. Each additional placement is an additional power of 10. Consider writing the number 1234.56 below in expanded notation:

$$\begin{aligned} 1234.56 &= 1000 + 200 + 30 + 4 + 0.5 + 0.06 \\ &= 1\times10^{3} + 2\times10^{2} + 3\times10^{1} + 4\times10^{0} + 5\times10^{-1} + 6\times10^{-2} \end{aligned}$$

In other words, starting from the decimal point, if we see to the left, the first position is the 'one' ($1 = 10^{0}$), the second position is the 'ten' ($10 = 10^{1}$), the third position is the 'hundred' ($100 = 10^{2}$), the fourth position is the 'thousand' ($1000 = 10^{3}$), and so on. To the right of the decimal point, the first position is the 'tenth' ($\frac{1}{10} = 10^{-1}$), the second position is the 'hundredth' ($\frac{1}{100} = 10^{-2}$), the third position is the 'thousandth' ($\frac{1}{1000} = 10^{-3}$), and so on.



Example 3.2
Write 286.35 in expanded notation.

3.2 Binary Number System

3.2.1 Binary Numbers

The decimal number system is fine for calculations done by humans, but it is not the easiest system for a computer to use. The binary number system is the number system used by computers.

Binary number system is also known as the "**base-2**" number system. In other words, it is a number system that uses only two symbols to represent numbers: 0 and 1. So, a binary number is just a sequence of the digits 0 and 1, such as 11011001. We use the word "**bit**" to represent **b**inary dig**it**.

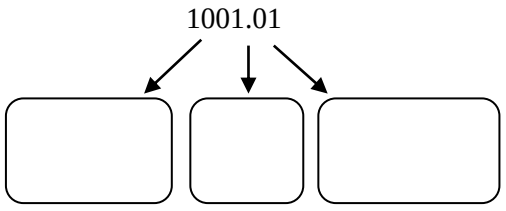
When we count binary, we run out of symbols very quickly. Like decimal, we also use the principle of 'reset and carry forward' here.

Example 3.3
List down the first 20 numbers in binary number system.

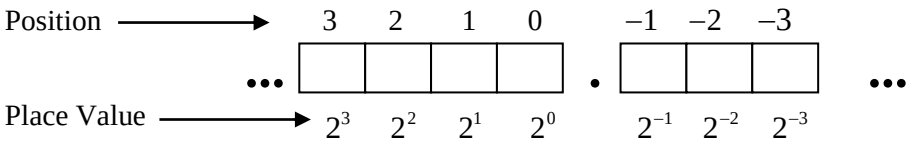
Decimal	Binary
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Decimal	Binary
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	

So, a binary number is a sequence of digits 0 and 1.
Similar to decimal number system, binary number also can have fractional parts. For example,



Similar to decimal number system, the position of a digit in the binary number system determines its value. The difference now is that the base is 2 instead of 10.



Therefore, in binary number system, we will deal with a lot of 'powers of two'. You are recommended to be familiar with them, as you need to use them a lot in the future, especially in the field of IT.

Position	Place Value
10	$2^{10} =$
9	$2^{9} =$
8	$2^{8} =$
7	$2^{7} =$
6	$2^{6} =$
5	$2^{5} =$
4	$2^{4} =$
3	$2^{3} =$
2	$2^{2} =$
1	$2^{1} =$
0	$2^{0} =$
-1	$2^{-1} =$
-2	$2^{-2} =$
-3	$2^{-3} =$
-4	$2^{-4} =$
-5	$2^{-5} =$

Example 3.4
Write 1101.101_2 in expanded notation.

Note:

1. As a convention, the base of the number is written with a subscript beside the number. If the number is a binary number, we write down a subscript "2" beside the number. If the number is decimal, we write down a subscript "10" beside a number. For example, 110_2 refers to the binary number "one-one-zero" which is equivalent to 6 in decimal number system, while 110_{10} refers to the decimal number "one hundred and ten".

2. If the base of a number is not written, by default, we assume the number has a base of 10 (decimal).

3. Long binary numbers are sometimes written in groups of four for easier reading. For example, the number 100100010100.001001 can be written as 1001 0001 0100.0010 01 instead. The number on the extreme left is the **most significant bit** (MSB) and the number on the extreme right is the **least significant bit** (LSB).

3.2.2 Conversion from Binary to Decimal

To convert a binary number to decimal, use the following steps:

- Write the binary number in expanded notation
- Omit the terms where the bits are zero
- Add the resulting values

Example 3.5

Convert the following binary numbers to decimal:

(a) 1001.011_2

(b) 1011101.1011_2

(9.375; 93.6875)

3.2.3 Conversion from Decimal to Binary

Given a decimal number, say 13, how do we find its binary representation? In other words, suppose the binary number is represented by $b_3b_2b_1b_0$ where $b_3, b_2, b_1, b_0 \in \{0,1\}$, how do we find the binary digits b_0, b_1, b_2 and b_3? The binary expansion of the decimal number 13 is

$$13 = b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 2^0,$$

which can be rewritten as

$$13 = \underbrace{(b_3 2^2 + b_2 2^1 + b_1)}_{quotient} 2 + \underbrace{b_0}_{remainder}.$$

Hence, we obtain the value of the binary digit b_0 which is the remainder.
If we rewrite the quotient in a similar way,

$$b_3 2^2 + b_2 2^1 + b_1 = \underbrace{(b_3 2^1 + b_2)}_{new\ quotient} 2 + \underbrace{b_1}_{new\ remainder}$$

we obtain the next binary digit b_1 which is the new remainder. If this process is repeated until the quotient is zero, we will obtain all the binary digits in sequence starting from the least significant bit b_0 to the most significant bit b_3.

How do we convert a decimal fraction, say 0.546875, into its binary representation?
Suppose the binary fractional part has n bits and has a binary representation $0.b_{-1}b_{-2}b_{-3}\ldots b_{-n}$, its value is given by

$$0.546875 = b_{-1}2^{-1} + b_{-2}2^{-2} + b_{-3}2^{-3} + \ldots + b_{-n}2^{-n}.$$

If we multiply this by 2, we get

$$1.09375 = b_{-1} + b_{-2}2^{-1} + b_{-3}2^{-2} + \ldots + b_{-n}2^{-n+1}$$

Hence, we obtain the binary digit b_{-1}, which is the coefficient of 2^0 or the integral part of the product. If we multiply the fractional part of the product by 2, we can get b_{-2} and so on, until we get all the binary digits.

In summary, to convert a decimal number to binary, use the following steps:

- For the integral part, divide the number by 2, and take note of the remainder. Repeat this step until you get 'zero' as the result of your division. The result is read by taking the remainders "bottom-up".
- For the fractional part, multiply the number by 2, and take note of the integral part of the result. Take the fractional part of the result and repeat this step until you get 'zero' as the fractional part. The binary equivalent is then obtained by taking the integral parts "top-down".

Example 3.6
Convert the following decimal numbers to binary:

(a) 9.375
(b) 93.6875

$(1001.011_2 ; 1011101.1011_2)$

Example 3.7
Convert 0.1 to binary.

$(0.0\overline{0011}_2)$

It is becoming clear that this computation can continue indefinitely. This shows that **not all decimal fractions can be exactly converted to binary**. This inability to make exact conversion is an unavoidable source of inaccuracy in some computations. In Example 3.7, the repetitive pattern becomes obvious very early after a few iterations. In such cases, we indicate the repetition with a horizontal line above the repeating digits (as shown above). Let us look at an example where the repetitive pattern is not obvious after a few iterations.

Example 3.8
Convert 0.743 to binary.

$(0.101111100\cdots_2)$

In Example 3.8, it is clear that the repetitive pattern does not become obvious after a few iterations (in fact, for this example, it repeats only after hundreds of iterations, which is extremely tedious to compute by hand). In this case, to decide how far to carry out computations, we use the rule of thumb that **each decimal digit requires three binary bits to give the same accuracy**. Thus our original three-digit number requires about nine bits. The result of our conversion is therefore:

$$0.743 \approx 0.101111100_2$$

3.2.4 Range of numbers obtainable with *n* bits

Example 3.9
What is the range of numbers obtainable with:

(a) 4 bits?
(b) 8 bits?
(c) 16 bits?
(d) n bits? (where $n \in \mathbb{Z}^+$)

(0-15; 0-255; 0-65,535; 0-[2^n–1])

Supplementary Material (for your own reading)

Application of Binary Numbers in Computer Systems: Bits and Bytes

In computer systems, a **b**inary dig**it** is often abbreviated as "**bit**". A bit is simply one digit in the binary number system, i.e. 0 or 1. For example, the binary number 10101 consists of 5 bits, while the binary number 1010101.10 consists of 9 bits.

A **byte** is a group of 8 bits. Historically, the byte was the number of bits used to encode a single character of text in a computer (the maximum number of unique characters supported were 256). Because the size of one bit/byte is particularly small, in practice the SI (International System of Units) prefixes (such as kilo, mega, or giga) are often used with bits and bytes.

There has been some confusion in the usage of these prefixes, though. Historically (before 1998), one "kilobyte" refers to the power of two that is closest to 1000 (i.e. $2^{10} = 1024$), one "megabyte" refers to the power of two that is closest to 1,000,000 (i.e. $2^{20} = 1,048,576$) and so on. The following table shows some commonly used SI prefixes for bits and bytes:

Prefix	Abbreviation	Meaning (before 1998)
Kilo	k	$2^{10} = 1024$
Mega	M	$2^{20} = 1,048,576$
Giga	G	$2^{30} = 1,073,741,824$
Tera	T	$2^{40} = 1,099,511,627,776$
Peta	P	$2^{50} = 1,125,899,906,842,624$
Exa	E	$2^{60} = 1,152,921,504,606,846,976$
Zetta	Z	$2^{70} = 1,180,591,620,717,411,303,424$
Yotta	Y	$2^{80} = 1,208,925,819,614,629,174,706,176$

In 1998, the convention on prefixes was standardized, and the meaning was changed. The SI unit was made more intuitive, i.e. kilobyte means 1000 (10^3) bytes, and Megabyte means 1,000,000 (10^6) bytes. The following table shows the modified meaning for bits and bytes from 1998 onwards:

Prefix	Abbreviation	Meaning (from 1998)
Kilo	k	$10^3 = 1,000$
Mega	M	$10^6 = 1,000,000$
Giga	G	$10^9 = 1,000,000,000$
Tera	T	$10^{12} = 1,000,000,000,000$
Peta	P	$10^{15} = 1,000,000,000,000,000$
Exa	E	$10^{18} = 1,000,000,000,000,000,000$
Zetta	Z	$10^{21} = 1,000,000,000,000,000,000,000$
Yotta	Y	$10^{24} = 1,000,000,000,000,000,000,000,000$

If you are interested to know more about the full details of this standardization, you may want to visit the following Wikipedia link: https://en.wikipedia.org/wiki/Binary_prefix#Inconsistent_use_of_units

3.3 Hexadecimal Number System

3.3.1 Hexadecimal Numbers

Hexadecimal numbers (or HEX for short) are base-16 numbers. There are 16 valid symbols in this number system, i.e. digits from 0 to 9, and the capital letters of A to F.

Example 3.10
List down the first 20 numbers in hexadecimal number system.

The following table summarizes the first 16 decimal, binary and hexadecimal integers:

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

3.3.2 Conversion from Binary to Hexadecimal

To convert from binary to hexadecimal, use the following steps:

- Group the bits into four starting at the binary point.
- Add zeros (where necessary) to form a group of 4 bits.
- Assign each group the appropriate hexadecimal symbol using the table above.

Example 3.11

Convert the following binary numbers into hexadecimal:

(a) 10110100111001_2

(b) 101111.0011111_2

3.3.3 Conversion from Hexadecimal to Binary

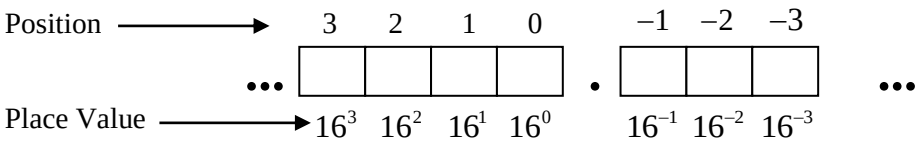
To convert from hexadecimal to binary, we simply need to reverse the procedure.

Example 3.12

Convert $3B25.E_{16}$ into binary.

3.3.4 Conversion from Hexadecimal to Decimal

As with decimal and binary numbers, each hex digit has a place value, equal to the base (16), raised to the position number. Thus, the place value of the positions are as follows:



To convert from hexadecimal to decimal, use the following steps:

- Replace each letter in the hex number by its decimal equivalent.
- Write the number in expanded notation, multiplying each hex digit by its place value.
- Add the resulting numbers.

Example 3.13

Convert $3B.F_{16}$ to decimal.

(59.9375)

3.3.5 Conversion from Decimal to Hexadecimal

To convert from decimal to hexadecimal, use the following steps:

- Repeatedly divide the given number by 16, and convert each remainder to hex.
- Read the remainders in the "bottom-up" fashion. The last remainder is the most significant digit, while the first remainder is the least significant digit.

Example 3.14

Convert 83579.125 into hexadecimal.

$(1467B.2_{16})$

Tips: Instead of converting directly between decimal and hexadecimal, many find it easier to first convert to binary.

Online Supplementary Resources

(1) Fundamentals of decimal, binary, octal, and hexadecimal number systems
http://tibasicdev.wikidot.com/binandhex

(2) Bits and bytes
http://computer.howstuffworks.com/bytes.htm

Tutorial 3 – Number Systems

Section A (Basic)

1. Convert the following binary numbers to decimal.
   (a) 10_2 (b) 111_2 (c) 1011_2 (d) $10\,0100_2$
   (e) 0.1_2 (f) 0.0101_2 (g) 10.01_2 (h) 11001.11101_2

2. Convert the following decimal numbers to binary.
   (a) 50 (b) 72 (c) 274 (d) 8375
   (e) 0.75 (f) 5.5 (g) 4.375 (h) 0.3

3. Convert the following binary numbers to hexadecimal.
   (a) 10_2 (b) 111_2 (c) 1011_2 (d) $10\,0100_2$
   (e) 0.1_2 (f) 0.0101_2 (g) 10.01_2 (h) 11001.11101_2

4. Convert the following hexadecimal numbers to binary.
   (a) 123_{16} (b) $E5_{16}$ (c) $6F_{16}$ (d) $2D3_{16}$
   (e) $37A4_{16}$ (f) $3.F_{16}$ (g) $9.AA_{16}$ (h) $ABC.DE_{16}$

5. Convert the following hexadecimal numbers to decimal.
   (a) 123_{16} (b) $E5_{16}$ (c) $6F_{16}$ (d) $2D3_{16}$
   (e) $37A4_{16}$ (f) $3.F_{16}$ (g) $9.AA_{16}$ (h) $ABC.DE_{16}$

6. Convert the following decimal numbers to hexadecimal.
   (a) 50 (b) 72 (c) 274 (d) 8375
   (e) 0.75 (f) 5.5 (g) 4.375 (h) 0.3

7. List down the first 20 **positive** integers in base-3 and base-5.

8. **(1213S1/B3)** In computer display systems, the red, green and blue (RGB) colour model is used to display a range of colours. Modern display systems use an 8-bit binary number to represent each of the values of red (R), green (G) and blue (B).
   (a) How many possible colours can this display produce?
   (b) It is known that the 'skyblue' colour can be produced by setting the decimal values representing R, G and B to 135, 206 and 255 respectively.
      (i) What is the value of R in binary?
      (ii) What is the value of G in hexadecimal?

Section B (Intermediate/Challenging)

9. Convert $13\times16^3+11\times16^2+9\times16^1+3$ to hexadecimal, and hence find the number of 1's in its binary representation.

10. Find the number of 1's in the binary representation of $3\times4096+12\times256+5\times16+11$.

11. What is the **minimum** decimal equivalent of the number $11B.0_e$, where base e is unknown?

12. **(1516S1/C2)** A 12-bit (binary digit) number, consisting of 8 integer bits and 4 fractional bits, is to be sent from a sender to a receiver via a wireless communication channel.
    (a) How many different numbers can be sent via this channel?
    (b) What is the maximum number (in decimal) that can be sent?
    (c) List all the numbers (in decimal) that can be sent via this channel, between 1.25 and 1.5 inclusive.
    (d) Alex wants to send a decimal value of 37.625 to Betty via this communication channel. How will the transmitted message look like?

13. (a) Write down the first ten fractional digits of the binary representation of $\frac{1}{10}$.
    (b) *From your result in part (a), it should be clear that $\frac{1}{10}$ cannot be represented exactly in binary using a finite number of bits because it has a non-terminating binary expansion. If this value is stored using a 24-bit register, what is the error caused by truncation of the non-terminating bits? (Hint: The value stored would be $0.0001\ 1001\ 1001\ 1001\ 1001\ 100_2$)
    (c) *A Patriot missile has an internal clock that converts $\frac{1}{10}$ second to binary every $\frac{1}{10}$ second and stores it in a 24-bit register. If the clock has been running for 100 hours, what is the error in the computation of time?
    (Note: This error caused 28 soldiers to be killed and 100 others to be injured on 25th Feb 1991, during the Gulf War. Read more at: https://www.ima.umn.edu/~arnold/disasters/patriot.html)

Section C (MCQ)

14. **(1415S1/A3)** How many unique symbols does a base-*n* number system have?
    (a) $n-1$ (b) n (c) $n+1$ (d) 2^n

15. **(1415S2/A3)** The largest decimal number that can be represented using a *n*-digit base-*k* number is:
    (a) n^k (b) n^k-1 (c) k^n (d) k^n-1

16. What is the largest decimal number that can be represented by a 4-digit octal number? (Hint: The base of octal numbers is 8.)
    (a) 4095 (b) 4096 (c) 65535 (d) 65536

17. **(1314S1/A3)** Which of the following decimal numbers cannot be exactly represented using 8-bit binary?
    (a) 247 (b) 128.5 (c) 60.75 (d) 6.21875

18. **(1213S2/A5)** Which of the following decimal numbers cannot be exactly represented in binary using a finite number of bits?
    (a) 0.125 (b) 0.1875 (c) 0.825 (d) 0.96875

Tutorial 3 – Answers

1. (a) 2 (b) 7 (c) 11 (d) 36 (e) 0.5 (f) 0.3125 (g) 2.25 (h) 25.90625
2. (a) $11\,0010_2$ (b) $100\,1000_2$ (c) $1\,0001\,0010_2$ (d) $10\,0000\,1011\,0111_2$
   (e) 0.11_2 (f) 101.1_2 (g) 100.011_2 (h) $0.0\overline{1001}_2$
3. (a) 2_{16} (b) 7_{16} (c) B_{16} (d) 24_{16} (e) 0.8_{16} (f) 0.5_{16} (g) 2.4_{16} (h) $19.E8_{16}$
4. (a) $1\,0010\,0011_2$ (b) $1110\,0101_2$ (c) $0110\,1111_2$ (d) $10\,1101\,0011_2$
   (e) $11\,0111\,1010\,0100_2$ (f) 11.1111_2 (g) $1001.1010\,1010_2$ (h) $1010\,1011\,1100.1101\,1110_2$
5. (a) 291 (b) 229 (c) 111 (d) 723
   (e) 14244 (f) 3.9375 (g) 9.6640625 (h) 2748.8671875
6. (a) 32_{16} (b) 48_{16} (c) 112_{16} (d) $20B7_{16}$ (e) $0.C_{16}$ (f) 5.8_{16} (g) 4.6_{16} (h) $0.4\overline{C}_{16}$
7. Base-3: 1, 2, 10, 11, 12, 20, 21, 22, 100, 101, 102, 110, 111, 112, 120, 121, 122, 200, 201, 202
   Base-5: 1, 2, 3, 4, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24, 30, 31, 32, 33, 34, 40
8. (a) 16,777,216 (b) (i) $1000\,0111_2$ (ii) CE_{16}
9. $DB93_{16}$; 10
10. 9
11. 167
12. (a) 4096 (b) 255.9375 (c) 1.25; 1.3125; 1.375; 1.4375; 1.5 (d) $0010\,0101.1010_2$
13. (a) 0.0001100110_2 (b) 0.00000009536743164062 5 (c) 0.343 seconds
14. b 15. d 16. a 17. b 18. c