**Number systems**

**Example 1.1.1: Find the magnitude of 245.378**

**D = 2.82 + 4.81 + 5.80 + 3.8-1 + 7.8-2**

= 165.48437510

A binary (base=2) number system is a special case of the positional number system in which the allowable digits are 0 and 1 that are called ìbitsî. The leftmost digit of a binary number is called the most significant bit (MSB) and the rightmost is called the least significant bit (LSB). Because the base of binary numbers is two, bit bi is associated with weight 2i .

**Example 1.1.2: Magnitude of Binary number**  
110100102 =1.27 +1.26 +0.25 +1.24 +0.23 +0.22 +1.21 +0.20

1101.00112 =1.23 +1.22 +0.21 +1.20 +0.2-1 +0.2-2 +1.2-3 +1.2-4

If the base of a number system is larger than ten, the digits exceeding 9 are expressed using alphabet letters as a convention. For example, hexadecimal number system uses 1-9 and A-F; base-32 number system uses 1-9 and A-V. This example is shown in Table 1. One may then wonder how large-base number systems such as a base-64 are expressed. Fortunately, we rarely use such a high-base number system because we find no real advantages of using them in applications. Moreover, we can always convert them from any high-base number system to a lower base number system, which is the subject of the next section.

**Table 1. Decimal, binary, hexadecimal, and base-32 Number Systems**

Decimal

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Binary Octal

00000 0 00001 1 00010 2 00011 3 00100 4 00101 5 00110 6 00111 7 01000 10 01001 11 01010 12 01011 13 01100 14 01101 15 01110 16 01111 17 10000 20 10001 21 10010 22 10011 23 10100 24 10101 25 10110 26 10111 27 11000 30 11001 31 11010 32 11011 33 11100 34 11101 35 11110 36 11111 37

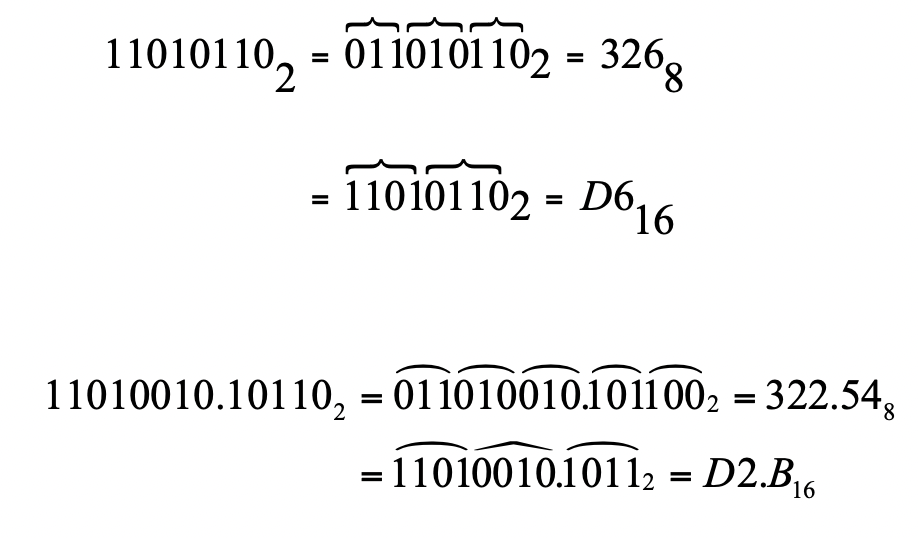
Hexadecimal

0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

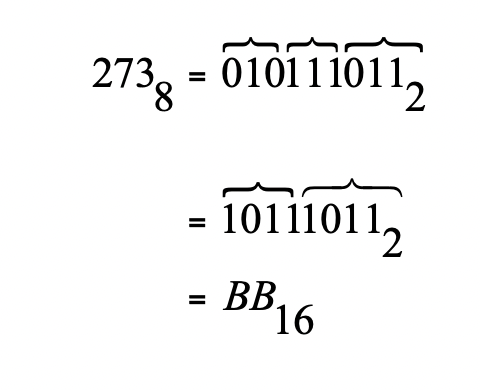
Base-32

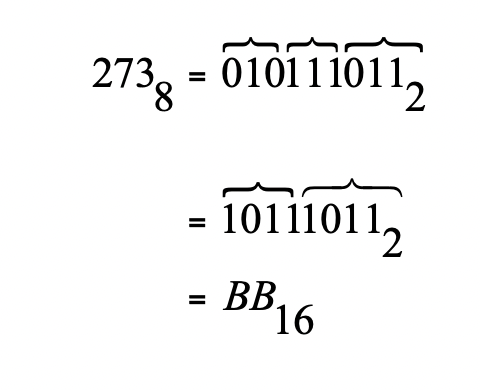
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V

**Example 1.2.1: Binary to hexadecimal or octal conversion**



**Example 1.2.2: Octal to hexadecimal or vice versa**



****

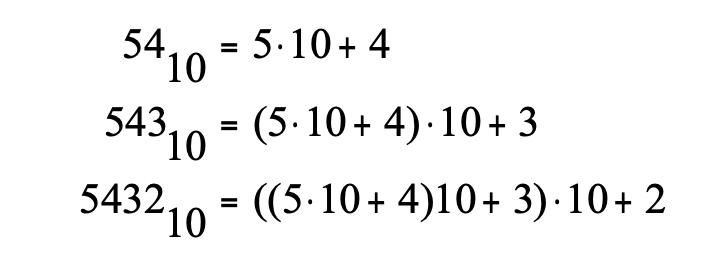
We have seen that the conversion between numbers with power of radix 2 can be readily achieved through binary expression and regrouping of bits. This convenience led to utilization of hexadecimal (or octal) numbers in representing binary numbers for many computer architecture related issues. For example, the instruction LDAA (Load Accumulator A) of 68HC11 is encoded

as the binary number 100001102 , but for convenience of writing and reading it is usually

expressed in hexadecimal 8616 , from which we save time and spaces. Very often, hexadecimal,

octal, and binary numbers are interchangeably used in the computer architecture or microprocessor related fields.

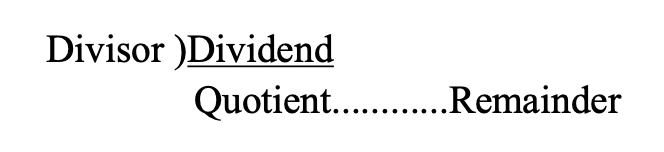
**Example 1.3.2: Integer expressions of positional numbers**

****

From Example 1.3.2, notice that if the last expression is divided by 10 the remainder is the least significant digit 2 and the quotient is ((510+ 4)10+ 3) . The next significant digit can be

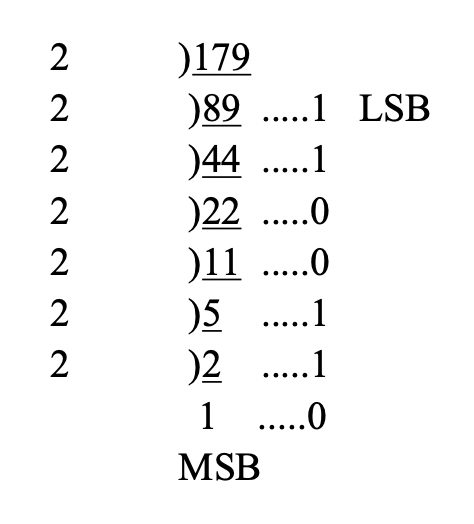
obtained by dividing 53510 again by 10. Due to this relation, the conversion to an arbitrary base

number can be obtained by repeated division of quotient and collection of remainders. A simple hand-calculation method can be devised using the above relation. Let's express the integer division by the following form.



Using this expression, Example 1.3.3 shows conversion from a decimal to a binary.

**Example 1.3.3: Convert 17910 to a binary.**

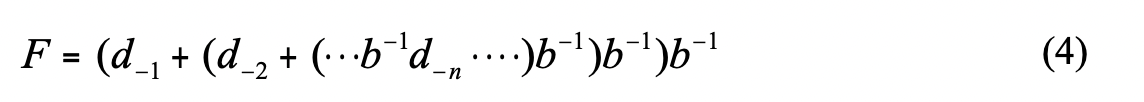
****

The final conversion result reads

17910 = 101100112.

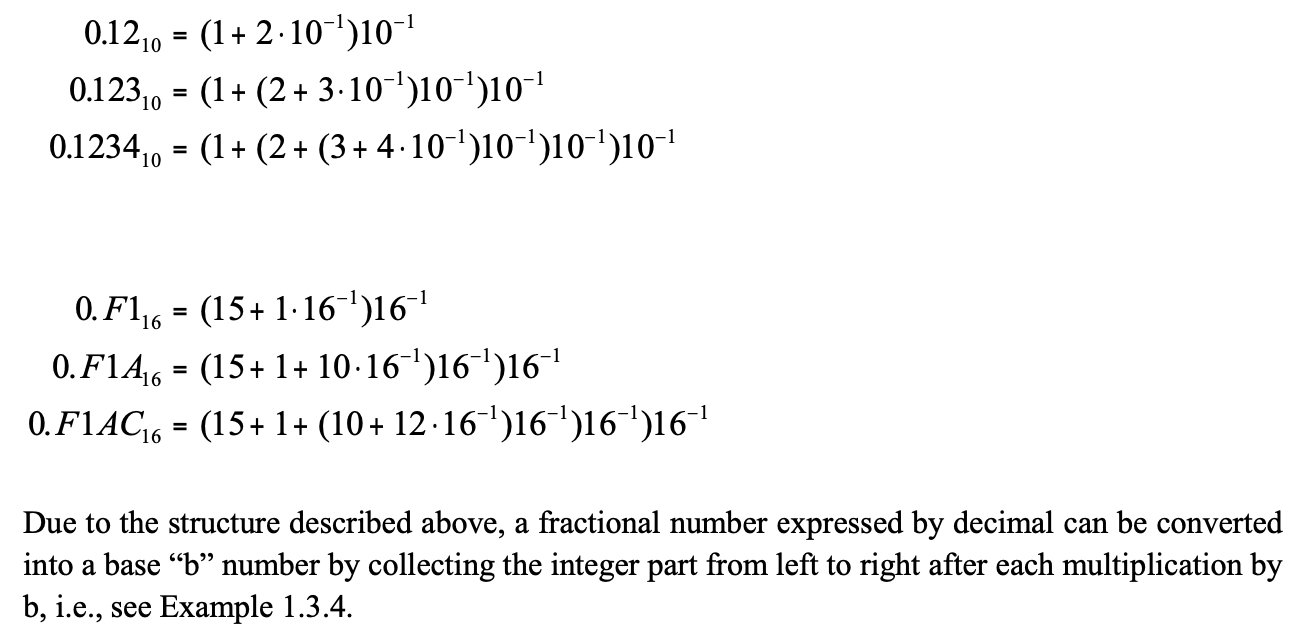
It should be noted that the above method can be extended to conversion of any other base. For example, consider that we wish to convert a hexadecimal number to a base-5 number. Then, the base-5 number can be directly converted by repeated division by 5 and collecting remainders. However, this direct division means, you must divide the base-16 number by 5, which is not simple because we are only used to decimal numbers. Thus, it is essentially wise to first convert the hexadecimal to a decimal, and then convert it to base-5.

Similarly to the expression of integer part in Eq. (3), the fractional part can be written in the following form:

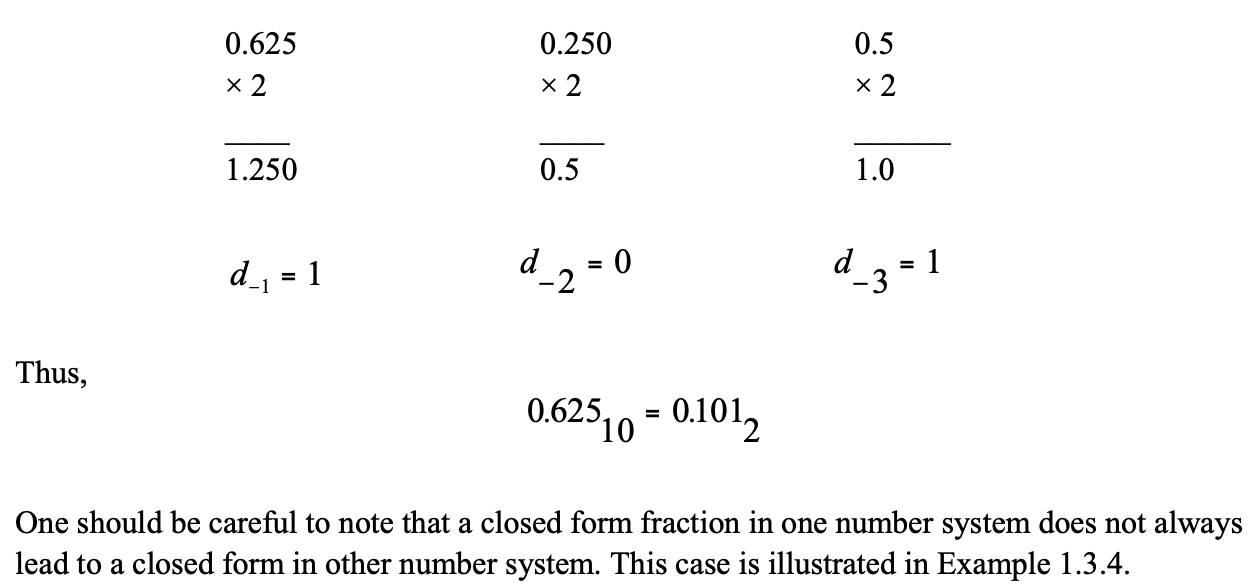
****

Note that multiplying b to F in Eq. (4) produces d-1 as a part of the product. This representation of number system is illustrated using Example 1.3.3.

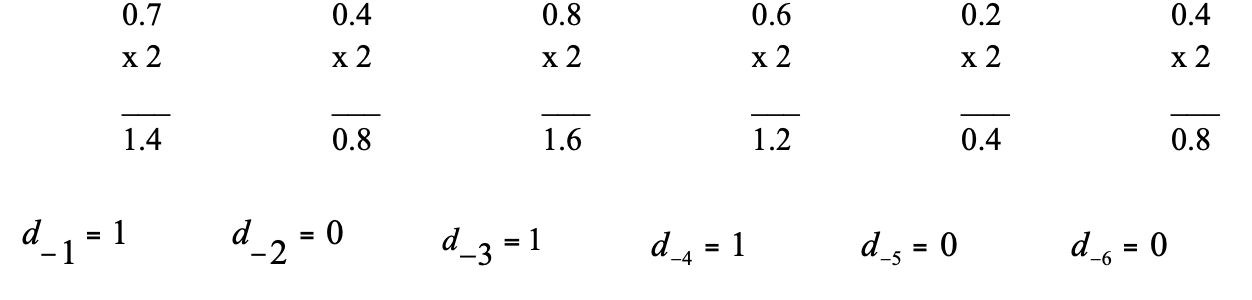
**Example 1.3.3: Fraction expression of positional numbers**

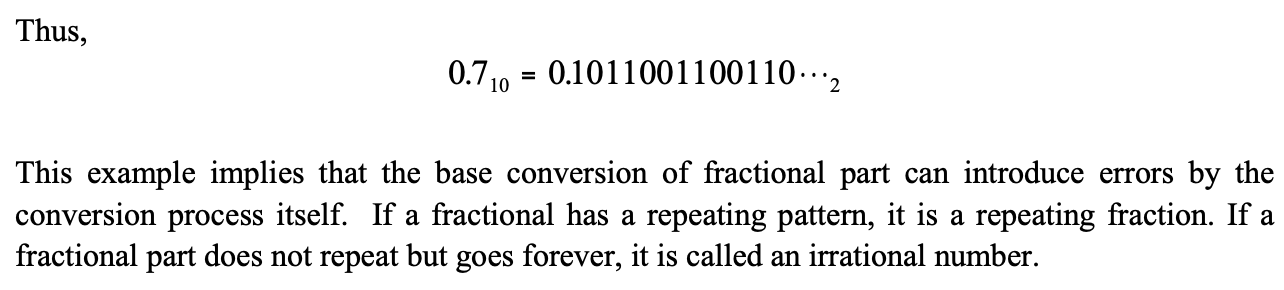


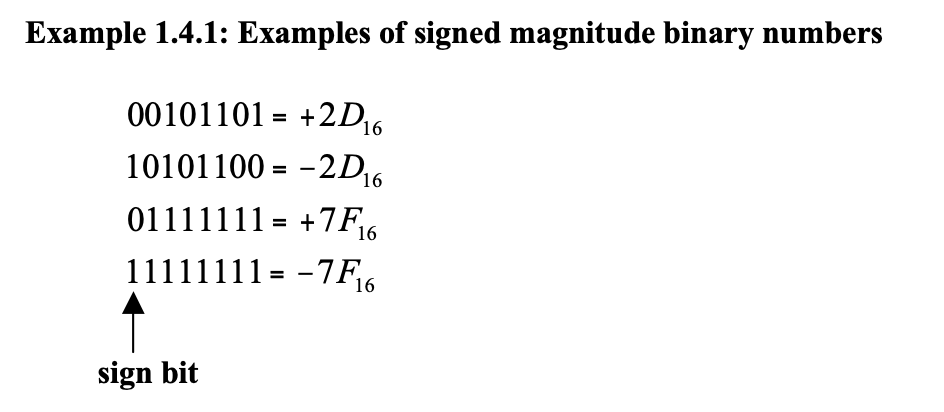
**Example 1.3.4: Convert a decimal number 0.625 to binary.**

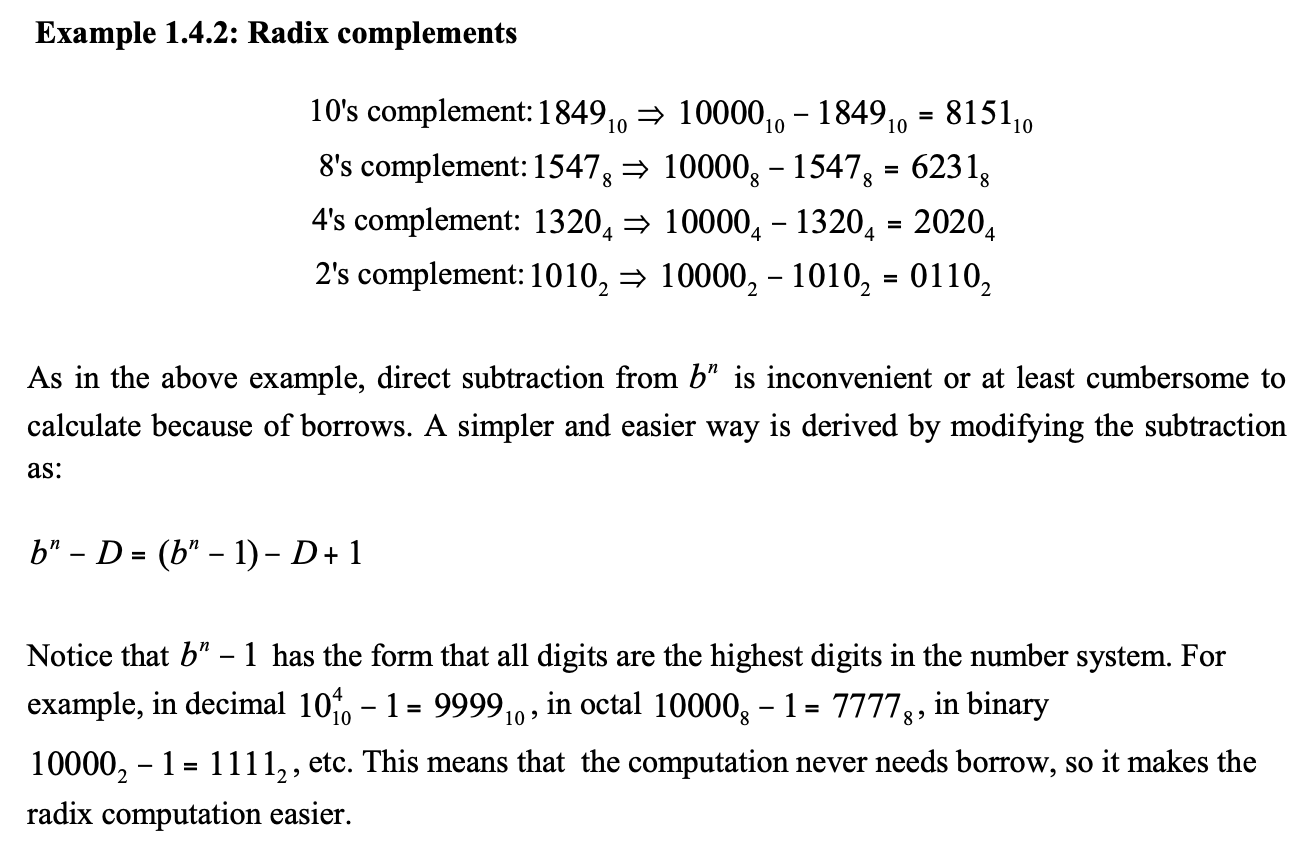
****

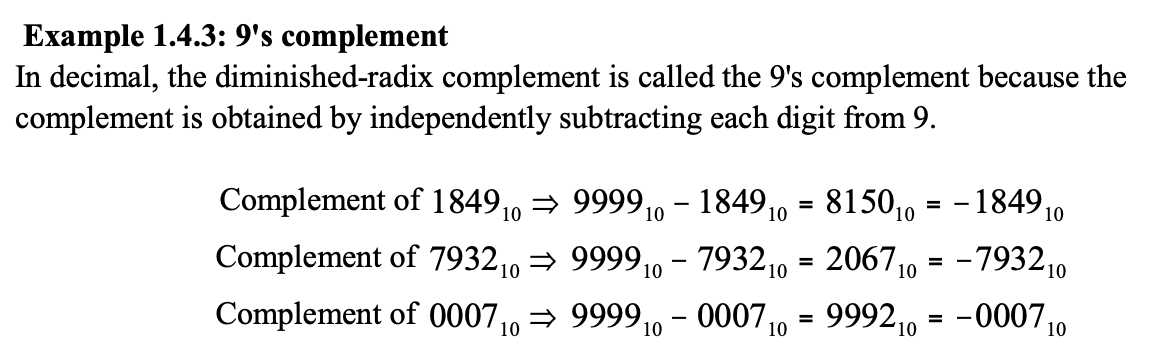
**Example 1.3.4: Decimal to base-x conversion: Convert 0.710 to a binary.**

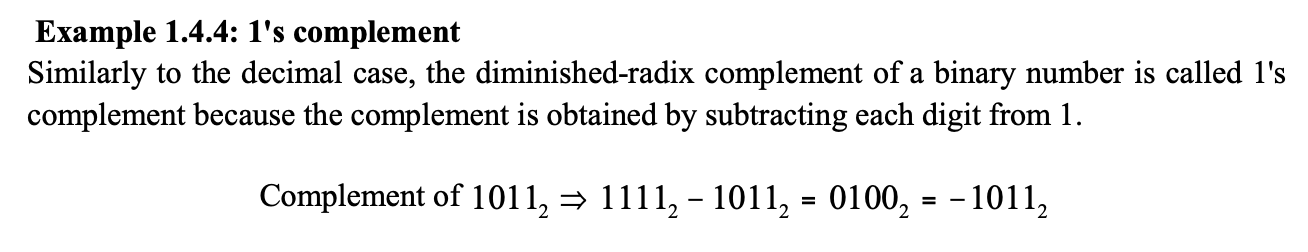


****

****

****

****

****

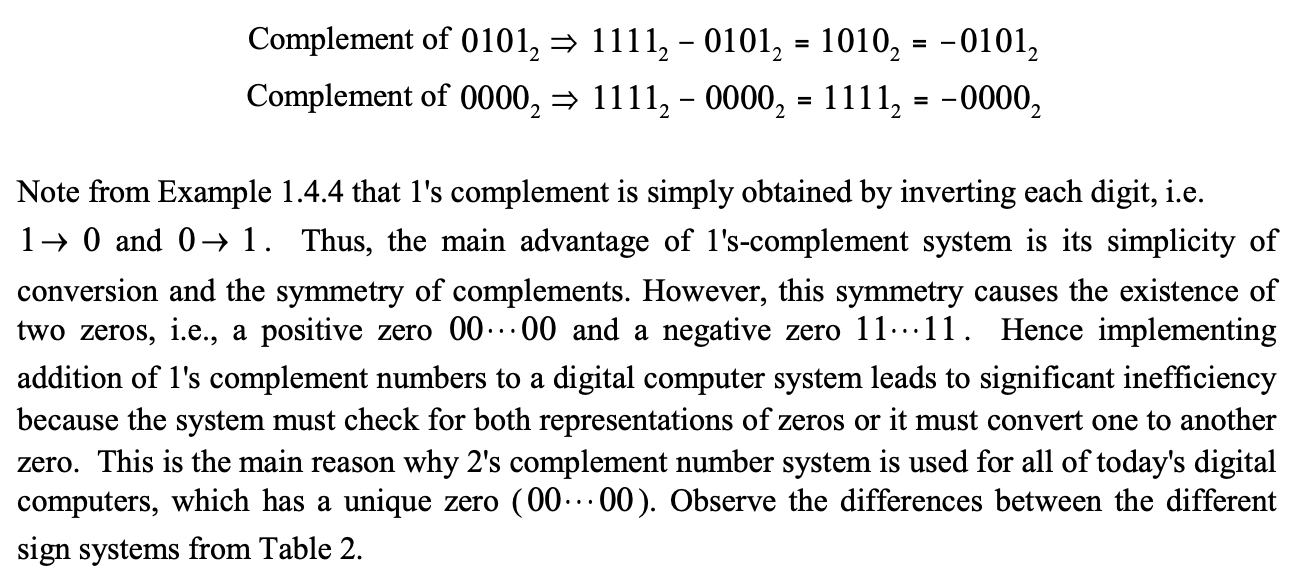
****

Table 2. 4-bit Numbers in Different Signed Systems

Decimal

-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6

7

2’s Complement 1000

1001 1010 1011 1100 1101 1110 1111 0000 0001 0010 0011 0100 0101 0110

0111

1’s complement -

1000 1001 1010 1011 1100 1101 1110

1111 or 0000 0001 0010 0011 0100 0101 0110

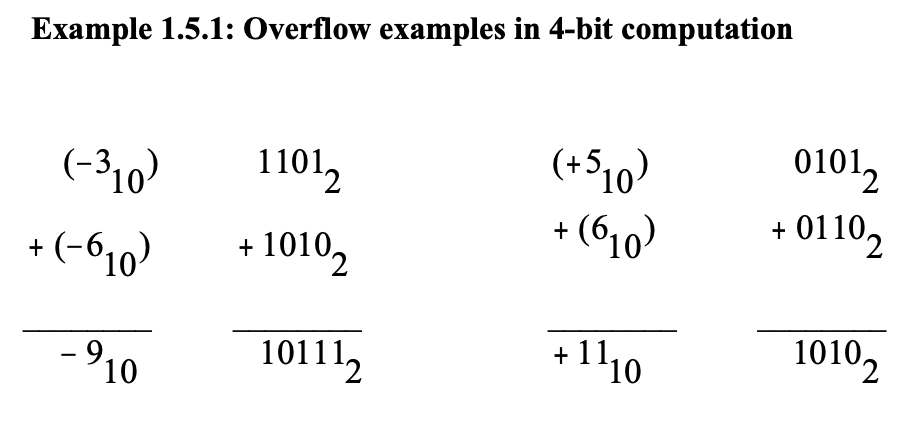
0111

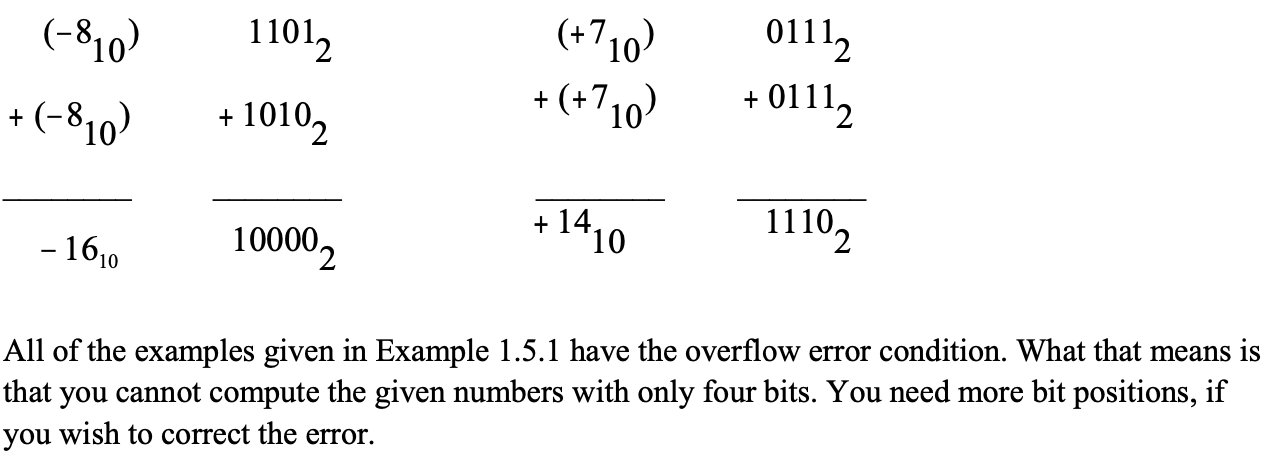
Signed Magnitude -

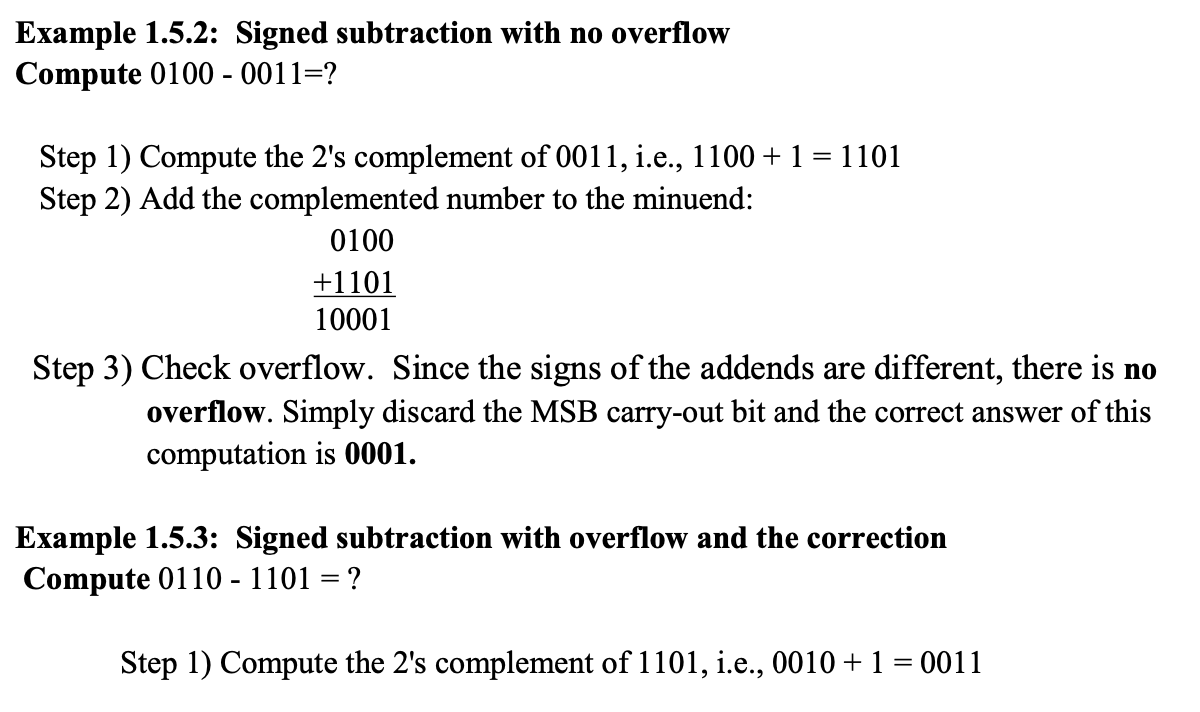
1111 1110 1101 1100 1011 1010 1001

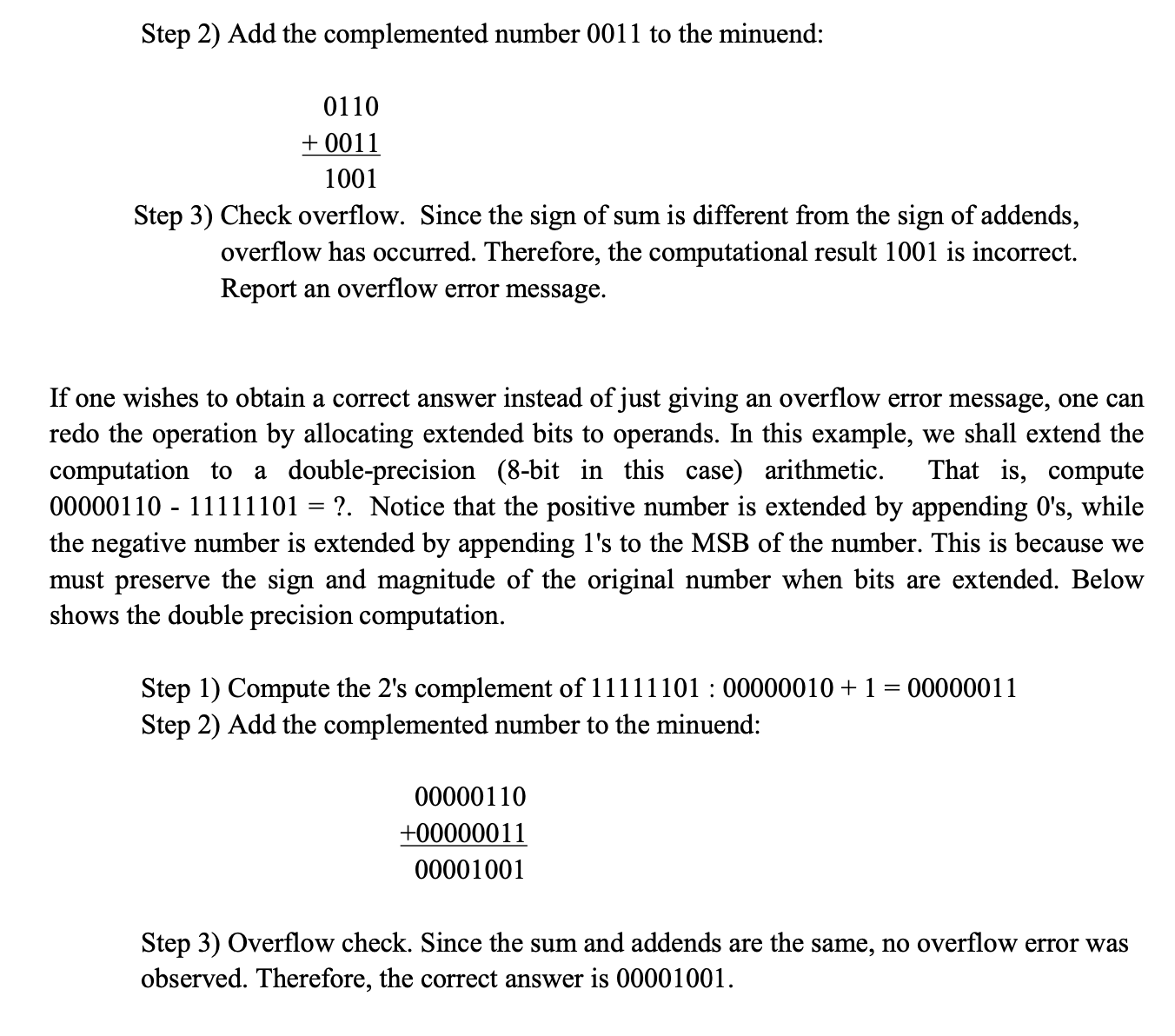
1000 or 0000 0001 0010 0011 0100 0101 0110

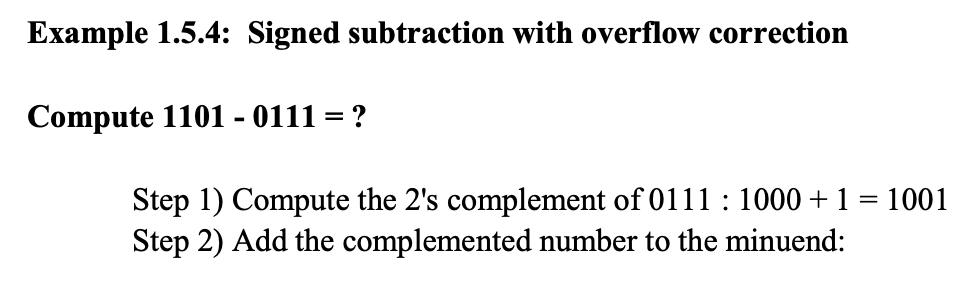
0111

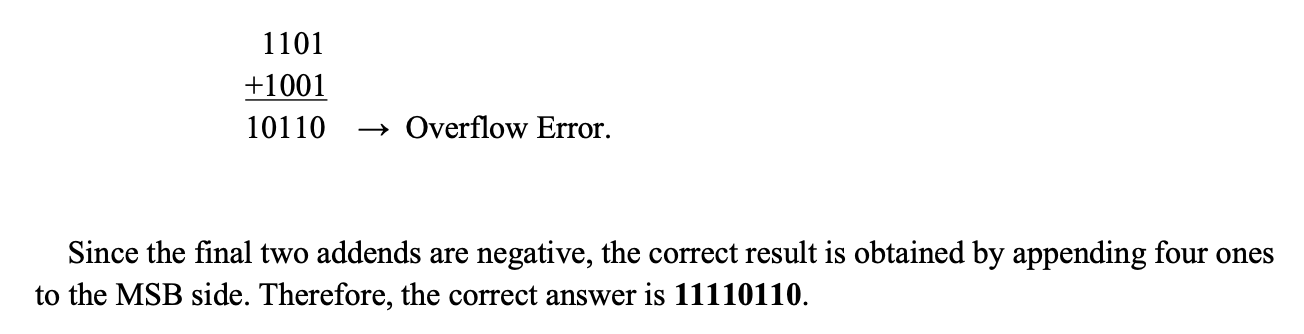
****

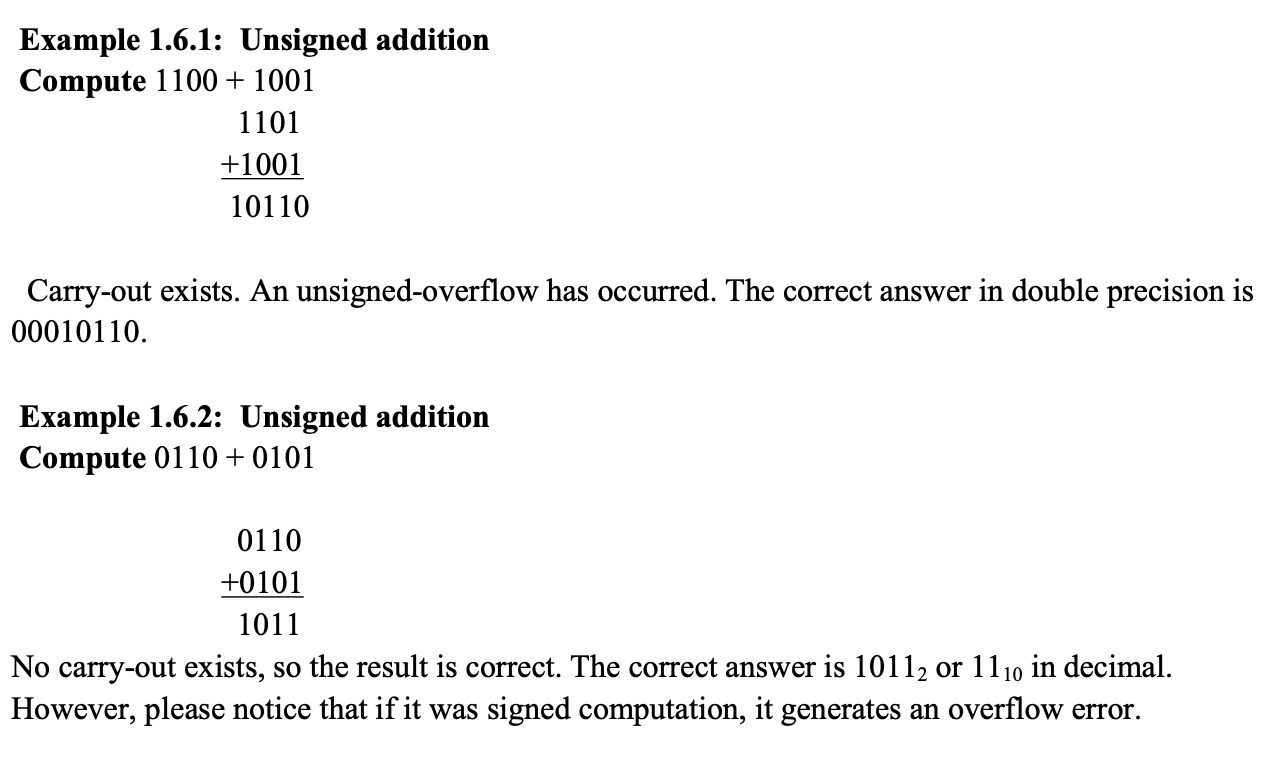
****

****

****

****

****

****

**Exercise 1: Decimal – Binary**

Convert these IP addresses from dotted decimal notation to binary notation:   
a. 123.160.16.178  
b. 221.255.31.117

**Exercise 2: Binary – Decimal**

Convert these IP addresses from binary notation to dotted decimal notation:   
a. 11010000.01010000.01011100.00010010(2)  
b. 11110100.00110010.00001110.11010001(2)

**Exercise 3:**

Convert 120(10) to binary

Convert 1010101(2) to decimal

Convert 1023(10) to binary

Convert 10001110(2) to decimal, octal and hexadecimal.

Convert the following to binary form:

810

4010

10110

Convert the following to decimal form:

11002

001100102

011112

**Exercise 4: Binary – Octal**

Convert binary to octal number and octal to binary number

a. 11010000.01010000.01011100.00010010(2)  
b. 11110100.00110010.00001110.11010001(2)

c. 1456762(o)

d. 54278423(o)

**Exercise 5: Binary – HexaDecimal**

a. 11010000.01010000.01011100.00010010(2)  
b. 11110100.00110010.00001110.11010001(2)

c. 56A5DB38929(H)

d. F56EFFE7667(H)

e. 47F8D2EE67F(H)