Example 1.3. *Convert 43_{10} to binary using repeated division method.*

Solution.

Repeated Division		Remainders	
2	43		
2	21	1	LSB
2	10	1	↑
2	5	0	Write in this order
2	2	1	
2	1	0	
	0	1	MSB

Reading the remainders from the bottom to the top,

$43_{10} = 101011_2$ (compare with result of example 1.1)

Example 1.4. *Convert 200_{10} to binary using repeated division method.*

Solution.

		Remainders	
2	200		
2	100	0	LSB
2	50	0	
2	25	0	
2	12	1	
2	6	0	
2	3	0	
2	1	1	
	0	1	MSB

Reading the remainders from the bottom to the top, the result is : $200_{10} = 11001000_2$

On paper you may even compute the conversion as depicted through following example. (As you can see that this is just another way of representing the repeated division.)

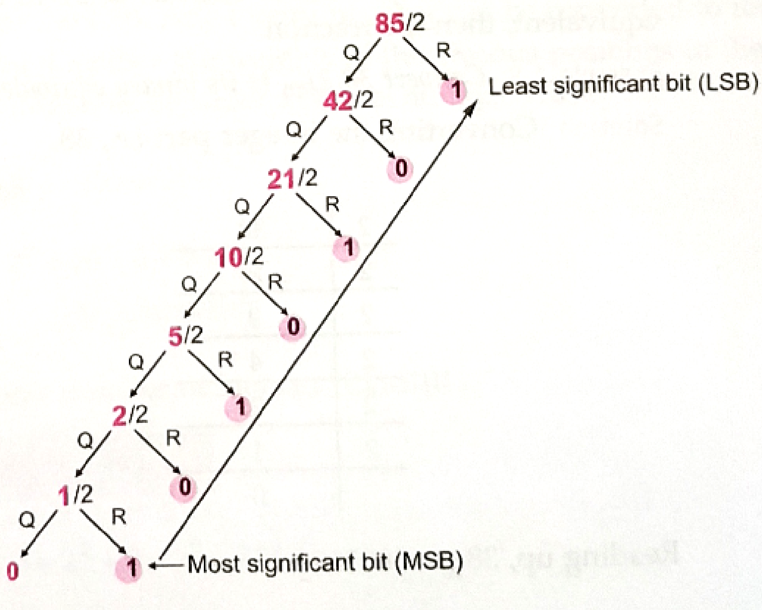
Example 1.5. *Convert 85_{10} to binary using repeated division method.*

Solution. (See on the right)

Q – Quotient and

R – Remainder

$\therefore \quad 85_{10} = 1010101_2$



1.4.1A Converting Decimal Fractions to Binary

To convert a decimal fraction into binary, the procedure is to successively multiply the decimal fraction by the radix *i.e.*, base (2 in this case) and collect all numbers to the left of the decimal point, reading down. Consider the following examples.

Example 1.6. *Convert 0.375_{10} to binary.*

Solution. Original number 0.375

			Integer Part
Multiply (fractional part) 0.375	by 2	= 0.750	0
Multiply (fractional part) 0.75	by 2	= 1.50	1
Multiply (fractional part) 0.50	by 2	= 1.00	1

Write in this order

Reading down the integer parts, $0.375 = 0.011_2$

Example 1.7. *Convert 0.54545_{10} to binary equivalent.*

Solution. Original number 0.54545

		Integer-Part
0.54545×2	= 1.0909	1
0.0909×2	= 0.1818	0
0.1818×2	= 0.3636	0
0.3636×2	= 0.7272	0
0.7272×2	= 1.4544	1
0.4544×2	= 0.9088	0
0.9088×2	= 1.8176	1
0.8176×2	= 1.6376	1
0.6376×2	= 1.2704	1
0.2704×2	= 0.5408	0

Writing order

This sequence keeps on continuing, thus, this particular number can never be expressed exactly in binary. Therefore, reading down the numbers, $0.54545 = 0.1000101110_2$ to 10 places.

As the previous example illustrates, many decimal fractions cannot be represented as exact binary numbers.

To convert a mixed number such as 38.21 to binary, first convert the integer to its binary equivalent, then the fraction.

Example 1.8. *Convert 38.21_{10} to its binary equivalent.*

Solution. Converting the integer part *i.e.*, 38.

		Remainders
2	38	
2	19	0
2	9	1
2	4	1
2	2	0
2	1	0
	0	1

Reading up, $38_{10} = 100110_2$

1.4.2 Binary-to-Decimal Conversion

The binary number system is a positional system where each binary digit (bit) carries a certain weight based on its position relative to the LSB. Any binary number can be converted to its decimal equivalent simply by summing together the weights of the various positions in the binary number which contain a 1. To illustrate, consider a binary number 11011_2

1	1	0	1	1_2

(binary)

Add positional weights for all 1s

$$\rightarrow 2^4 + 2^3 + 0 + 2^1 + 2^0 = 16 + 8 + 2 + 1$$

$$= 27_{10} \text{ (decimal)}$$

Let's try another example with a greater number of bits *i.e.*, 10110101_2

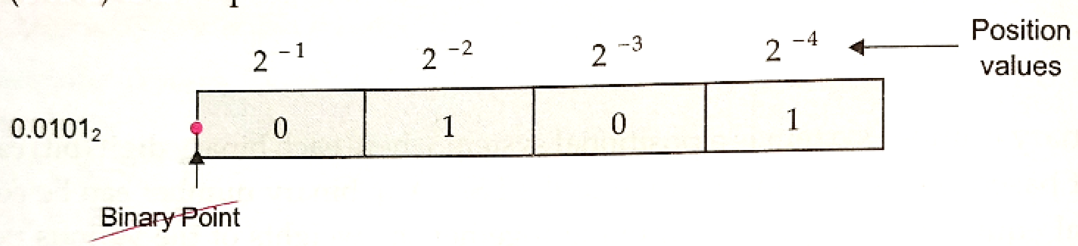
1	0	1	1	0	1	0	1_2

Adding positional weights of all 1's

$$\rightarrow 2^7 + 0 + 2^5 + 2^4 + 0 + 2^2 + 0 + 2^0 = 181_{10}$$

1.4.2A Converting Binary Fractions to Decimal

To find the decimal equivalent of binary fraction, take the sum of the products of each digit value (0 to 1) and its positional value. To illustrate :



$$= (0 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4})$$

$$= 0 + 0.25 + 0 + 0.0625 = 0.375$$

$$0.0101_2 = 0.3125_{10}.$$

Example 1.9. *Convert 1101.000101_2 to decimal.*

Solution.

$$= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5} + 1 \times 2^{-6}$$

$$= 8 + 4 + 0 + 1 + 0 + 0 + 0 + \frac{1}{16} + 0 + \frac{1}{64}$$

$$= 13 + 0.0625 + 0.015625 = 13.078125$$

$$1101.000101_2 = 13.078125_{10}$$

1.4.3 Decimal-to-Octal Conversion

A decimal integer can be converted to octal by using the same repeated-division method that we used in the decimal-to-binary conversion, but with a division factor of 8 instead of 2. An example is shown below :

		Remainders
8	266	
8	33	2
8	4	1
	0	4

Reading up, $266_{10} = 412_8$

Note that the first remainder becomes the least significant digit (LSD) of the octal number, and the last remainder becomes the most significant digit (MSD).

1.4.3A Converting Decimal Fraction to Octal

To convert a decimal fraction into octal, the procedure is to successively multiply the decimal fraction by the radix *i.e.*, base (8 in this case) and collect all the numbers to the left of the decimal point, reading down. Consider the following examples.

Example 1.10. *Convert 0.375_{10} to octal*

Solution. Original number 0.375

Integer-part

Multiply (fraction part) 0.375 by 8 = 3.0 — 3

$$0.375_{10} = 0.3_8$$

Example 1.11. *Convert 0.015625_{10} to octal.*

Solution. $0.015625 \times 8 = 0.125$ — 0

$0.125 \times 8 = 1.0$ — 1

Reading down, $0.015625_{10} = 0.01_8$

1.4.4 Octal-to-Decimal Conversion

An octal number, then, can be easily converted to its decimal equivalent by multiplying each octal digit by its positional weight. For example,

$$372_8 = 3 \times (8^2) + 7 \times (8^1) + 2 \times (8^{0)}$$
$$= 3 \times 64 + 7 \times 8 + 2 \times 1 = 250_{10}$$

Another example :

$$24.6_8 = 2 \times (8^1) + 4 \times (8^0) + 6 \times (8^{-1}) = 20.75_{10}$$

1.4.5 Octal-to-Binary Conversion

The primary advantage of the octal number system is the ease with which conversion can be made between binary and octal numbers. The conversion from octal to binary is performed by converting *each* octal digit to its 3-bit binary equivalent. The eight possible digits are converted as indicated in Table 1.3.

Table 1.3 *Binary equivalents of octal digits*

Octal Digit	0	1	2	3	4	5	6	7
Binary Equivalent	000	001	010	011	100	101	110	111

Using these conversions, any octal number is converted to binary by individually converting each digit.

For example, we can convert 472_8 to binary using 3 bits for each octal digit as follows :

$$\begin{array}{ccc} 4 & 7 & 2 \\ \downarrow & \downarrow & \downarrow \\ 100 & 111 & 010 \end{array}$$

Hence, octal 472 is equivalent to binary 100111010. As another example, consider converting 5431 to binary :

$$\begin{array}{cccc} 5 & 4 & 3 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 101 & 100 & 011 & 001 \end{array}$$

Thus, $5431_8 = 101100011001_2$.

The same process applies equally on fractions. For example,

$$3.1_8 = \begin{array}{ccc} 3 & \cdot & 1 \\ \downarrow & & \downarrow \\ 011 & \cdot & 001 \end{array}$$

$$3.1_8 = 011.001_2$$

1.4.6 Binary-to-Octal Conversion

Converting from binary integers to octal integers is simply the reverse of the foregoing process. The bits of the binary integer are grouped into groups of *three* bits starting at the LSB. Then each group is converted to its octal equivalent (Table 1.3). To illustrate, consider the conversion of 100111010_2 to octal.

$$\begin{array}{ccc} \underbrace{1\ 0\ 0} & \underbrace{1\ 1\ 1} & \underbrace{0\ 1\ 0} \\ \downarrow & \downarrow & \downarrow \\ 4 & 7 & 2_8 \end{array}$$

Sometimes the binary number will not have even groups of 3 bits. For those cases, we can add one or two 0s to the left of MSB of the binary number to fill out the last group. This is illustrated below for the binary number 11010110.

0 added here to make it group of 3 bits

$$\begin{array}{ccc} \underbrace{0\ 1\ 1} & \underbrace{0\ 1\ 0} & \underbrace{1\ 1\ 0} \\ \downarrow & \downarrow & \downarrow \\ 3 & 2 & 6_8 \end{array}$$

Note that a 0 was placed to the left of the MSB in order to produce complete groups of 3.

The same process applies on fractions. But after the binary point, zeros are added to the right. For example

$$10110.0101_2 = \underbrace{010}_{2}\ \underbrace{110}_{6}\ .\ \underbrace{010}_{2}\ \underbrace{100}_{4}$$

2 zeros added here to make it a group of 3 bits

$$10110.0101_2 = 26.24_8$$

Note that, after the binary point, the groups of 3 bits are made starting from left-to-right. That is why, we added two zeros to make a group of three bits as the last group had only 1.

1.4.7 Decimal-to Hex Conversion

Recall that we did decimal-to-binary conversion using repeated division by 2, and decimal-to-octal using repeated division by 8. Likewise, decimal-to-hex conversion can be done using repeated division by 16. For example, to convert 423_{10} to hex,

		Remainders	
16	423		
16	26	7	
16	1	A	$(10_{10} = A_{16})$
	0	1	

Reading up, $423_{10} = 1A7_{16}$

Similarly, to convert 214_{10} to hex,

		Remainders	
16	214		
16	13	6	
	0	D	$(13_{10} = D_{16})$

Reading up, $214_{10} = D6_{16}$

Note that any remainders that are greater than 9 are represented by letters A through F.

1.4.7A Converting Decimal Fractions to Hex

The same old procedure applies here also, that is, successively multiply the decimal fraction by the radix (base) (16 here) and collect all the numbers to the left of the decimal point, reading down. For instance, 0.03125_{10} is converted as

$$0.03125 \times 16 = 0.5 \quad 0$$
$$0.5 \times 16 = 8.0 \quad 8$$
$$0.03125_{10} = 0.08_{16}$$

1.4.8 Hex-to-Decimal Conversion

A hex number can be converted to its decimal equivalent by using the fact that each hex digit position has a weight that is a power of 16. The LSD has a weight of $16^0 = 1$, the next higher digit has a weight of $16^1 = 16$, the next higher digit has a weight of $16^2 = 256$, and so on. The conversion process is demonstrated in the examples below :

$$
\begin{aligned}
356_{16} &= 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 \\
&= 768 + 80 + 6 \\
&= 854_{10}
\end{aligned}
$$

$$
\begin{aligned}
2AF_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\
&= 512 + 160 + 15 \\
&= 687_{10}
\end{aligned}
$$

Note that in the second example the value 10 was substituted for A and the value 16 for F in the conversion to decimal.

To convert a fractional number,

$$
\begin{aligned}
56.08_{16} &= 5 \times 16^1 + 6 \times 16^0 + 0 \times 16^{-1} + 8 \times 16^{-2} \\
&= 80 + 6 + 0 + 8/256 = 86 + 0.03125 \\
&= 86.03125_{10}
\end{aligned}
$$

1.4.9 Binary-to-Hex Conversion

Binary numbers can be easily converted to hexadecimal by grouping in groups of four starting at the binary point.

Example 1.12. *Convert 1010111010_2 to hexadecimal.*

Solution. Group in fours 10,1011,1010

Convert each number 2 B A

Thus, the solution is 2 BA.

Example 1.13. *Convert 10101110.010111_2 to hexadecimal.*

Solution. Groups in fours (inserting zeros before MSB or in the end, wherever required)

1010, 1110 . 0101, 1100

A E . 5 C

$10101110.010111_2 = AE.5C_{16}$

1.4.10 Hex to Binary Conversion

Like the octal number system, the hexadecimal number system is used primarily as a "shorthand" method for representing binary numbers. It is a relatively simple matter to convert a hex number to binary. *Each* hex digit is converted to its 4-bit binary equivalent (Table 1.1). This is illustrated below for $9F2_{16}$.

$$
\begin{array}{lccc}
9F2_{16} = & 9 & F & 2 \\
 & \downarrow & \downarrow & \downarrow \\
 & 1001 & 1111 & 0010 \\
\end{array}
$$

$= 100111110010_2$

Similarly, $3A6_{16}$ = 3 → 0011, A → 1010, 6 → 0110

$= 001110100110_2$

> **Note** The hexadecimal and octal codes are used as shorthand means of expressing large binary numbers.

Consider another example

$3BF.5C_{16}$ = 3 → 0011, B → 1011, F → 1111, 5 → 0101, C → 1100

$= 001110111111.01011100_2$

Converting from Any Base to Any OTHER Base

As demonstrated in earlier examples and the table below, there is a direct correspondence between the number systems : with three binary digits corresponding to one octal digit ; four binary digits corresponding to one hexadecimal digit, which you can use to convert from one number system to another.

Table 1.4 *Correspondence of binary and octal number systems*

BIN	OCT	DEC
000	0	0
001	1	1
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7

For conversion from *base* 2 to *base 8*, we **use groups of three bits** and vice-versa.

Table 1.5 *Correspondence of Binary, Octal and Hexadecimal number systems.*

BIN	HEX	OCT	DEC
0000	0	00	0
0001	1	01	1
0010	2	02	2
0011	3	03	3
0100	4	04	4
0101	5	05	5
0110	6	06	6
0111	7	07	7
1000	8	10	8
1001	9	11	9
1010	A	12	10
1011	B	13	11
1100	C	14	12
1101	D	15	13
1110	E	16	14
1111	F	17	15

For conversion from *base* 2 to *base 16*, we **use groups of four bits** and vice-versa.

Let us consider some more examples regarding the same, *i.e.*, converting from any number system to another.

Example 1.14. *Convert* $1948.B6_{16}$ *to Binary and Octal equivalents.*

Solution.

Hexadecimal	1	9	4	8	.	B	6
Binary	0001	1001	0100	1000	.	1011	0110

(By converting each individual Hex digit to equivalent 4 digit binary from above table 1.5)

$\therefore \quad 1948.B6_{16} = 0001100101001000.10110110_2$

New Octal number can be generated from above Binary equivalent *i.e.*, as follows :

Binary	0	001	100	101	001	000	.	101	101	100
Octal		1	4	5	1	0	.	5	5	4

(By creating groups of 3 binary digits and converting them into equivalent octal no.)

$\therefore \quad 1948.B6_{16} = 14510.554_8.$

Example 1.15. *Convert* 75643.5704_8 *to hexadecimal and binary numbers.*

Solution. We shall convert it in following way :

(*i*) From octal to binary — by representing each octal digit to 3 digit binary number.

(*ii*) From the complete binary number, we shall create groups of 4 binary digits around the decimal point.

(*iii*) Convert each 4-digit-binary group to equivalent hex digit.

Octal	7	5	6	4	3	.	5	7	0	4
Binary	111	101	110	100	011	.	101	111	000	100

$\therefore \quad 75643.5704_8 = 111101110100011.101111000100_2$

Binary	0111	1011	1010	0011	.	1011	1100	0100
Hexadecimal	7	B	A	3	.	B	C	4

$\therefore \quad 75643.5704_8 = 7BA3.BC4_{16}$

After learning about various digital number systems, one must know how the data is represented in computers. Following sections illustrate the same.

1.5 Binary Representation of Numbers

While working with numbers, you have mainly worked with **integers** (*e.g.*, +17, –23 etc.) or with **real numbers** (*e.g.*, +17.015, –23.214 etc.)

Integers are *whole numbers* or *fixed-point numbers* with the radix point *fixed* after the least-significant bit. They are contrast to *real numbers* also called *floating-point numbers*, where the position of the radix point varies. It is important to take note that integers and floating-point