UNIT 2

Data Representation

Learning Outcomes:

- Distinguishes the various number systems and data representation.
- Computes number system operation such addition, subtraction and complement.
- Perform number conversion, fixed point and floating-point number representation.

A. NUMBER SYSTEM

A digital system can understand positional number system only where there are a few symbols called digits and these symbols represent different values depending on the position they occupy in the number.

A value of each digit in a number can be determined using

- The digit
- The position of the digit in the number
- The base of the number system

Decimal Number System (base 10)

The number system that we use in our day-to-day life is the decimal number system. Decimal number system has base 10 as it uses 10 digits from 0 to 9. In decimal number system, the successive positions to the left of the decimal point represents units, tens, hundreds, thousands and so on.

Each position represents a specific power of the base (10). For example, the decimal number 1234 consists of the digit 4 in the unit's position, 3 in the tens position, 2 in the hundreds position, and 1 in the thousands position, and its value can be written as:

$$(1\times1000) + (2\times100) + (3\times10) + (4\times l)$$
$$(1\times10^3) + (2\times10^2) + (3\times10^1) + (4\times l0^0)$$
$$1000 + 200 + 30 + 1$$
$$1234_{10}$$

Binary Number System (base 2)

The binary number system contains numbers with a base of 2. Where 0 and 1 are the only two numbers in the binary number system. All the numbers in the other number systems can be represented using the binary number system.

When the numbers are formed by using these two digits are known as binary numbers. This type is used in computer systems, electronic devices because it can perform easily through two states like ON or 1 & OFF or 0.

The symbols of the binary number system are used to represent number in the same way as in the decimal system symbol is used individually; then the symbols are use combination. Since there are only two symbols, we can represent two numbers, 0 and 1, with individual symbols. The position of the 1 or 0 in a binary number system indicates its weight or value within the number.

Octal Number System (base 8)

The number system which has the base value '8' is known as the octal number system. This system uses eight digits from 0 to 7 to form octal numbers. The conversion of octal numbers to decimal numbers can be done by multiplying every digit with the position value & after that adding the outcome. Here the position values are 8^0, 8^1, & 8^2.

Hexadecimal Number System (base 16)

The number system which has base-16 is known as the hexadecimal number system. So the possible symbols or digit values are 16 like 0 to 9 and after that, it is A, B, C, D, E & F. Here, 10 to 15 values are represented with A to F. It needs simply four bits to signify any digit. Hexadecimal numbers are mainly represented through the addition of either a prefix '0x' or a suffix 'h'.

Each digit position has a weight like the power of 16. Every position within the Hexadecimal system is 16 times more important than the earlier position that means a hexadecimal number's numeric value can be determined by multiplying every digit of the number by the value of the position where the digit appears and then adding the products. So, it is also a positional number system.

1. Conversion of Number System

The number system conversion is the process used to change one base number to another base number like binary, decimal, hexadecimal & octal through some examples. Here, the base numbers of these systems are for decimal base -10, binary – base-2, octal base-8, and for hexadecimal –base-10.

A. Binary to Decimal

The binary number system is also a positional numbering system, the base is 2. The powers of 2 are used to find the place values. Thus, to convert a binary number into a decimal number, binary digits are to be multiplied with the powers of 2 and added.

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	*128*	*64*	*32*	*16*	*8*	*4*	*2*	*1*

Example A.1

Let us convert 1101_2 into a decimal number.

$$\begin{aligned} 1101_2 &= (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ &= (1 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) \\ &= 8 + 4 + 0 + 1 \\ &= 13_{10} \end{aligned}$$

Therefore, the decimal representation of 1101_2 is 13_{10}.

Example A.2

Let us convert 10101_2 into a decimal number.

$$\begin{aligned} 10101_2 &= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ &= (1x16) + (1\times 8) + (1\times4) + (0\times2) + (1\times1) \\ &= 16 + 0 + 4 + 0 + 1 \\ &= 21_{10} \end{aligned}$$

Therefore, the decimal representation of 10101_2 is 21_{10}.

Example A.3

Let us convert 10100011_2 into a decimal number.

$$\begin{aligned} 10100011_2 &= (1 \times 2^7) + (0 \times 2^6) + (1 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) \\ &= 128 + 0 + 32 + 0 + 0 + 0 + 2 + 1 \\ &= 163_{10} \end{aligned}$$

Therefore, binary number $10100011_2 = 163_{10}$ decimal number

B. Octal to Decimal

To convert an octal number to a decimal number we need to multiply each digit of the given octal with the reducing power of 8.

Example B.1

Let us convert 16_8 into a decimal number.

$16_8 = (1 \times 8^1) + (6 \times 8^0)$
$= (1 \times 8) + (6 \times 1)$
$= 8 + 6$
$= 14_{10}$

Therefore, the decimal representation of 16_8 is 14_{10}.

Example B.2

Let us convert 215_8 into a decimal number.

$215_8 = (2 \times 8^2) + (1 \times 8^1) + (5 \times 8^0)$
$= (2 \times 64) + (1 \times 8) + (5 \times 1)$
$= 128 + 8 + 5$
$= 141_{10}$

Therefore, the decimal representation of 215_8 is 141_{10}.

Example B.3

Let us convert 125_8 into a decimal number.

$125_8 = (1 \times 8^2) + (2 \times 8^1) + (5 \times 8^0)$
$= (1 \times 64) + (2 \times 8) + (5 \times 1)$
$= 64 + 16 + 5$
$= 85_{10}$

Therefore, the decimal representation of 125_8 is 85_{10}.

C. Hexadecimal to Decimal

To convert hex to a decimal manually, start by multiplying the hex numbers by 16. Then, raise it to a power of 0 and increase that power each time by 1 according to the hexadecimal equivalent system.

Hexadecimal to Decimal Table

Hexadecimal	Decimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10

B	11
C	12
D	13
E	14
F	15

While applying powers, start with the right of a hexadecimal number system and move to the left. Each time you multiply a number by 16, the power of 16 increases.

Example C.1
Let us convert $7CF_{16}$ into a decimal number.

7 = 7
C = 12
F = 15

$7CF_{16}$ $= (7 \times 16^2) + (12 \times 16^1) + (15 \times 16^0)$
$= (7 \times 256) + (12 \times 16) + (15 \times 1)$
$= 1{,}792 + 192 + 15$
$= 1{,}999_{10}$

Therefore, the decimal representation of $7CF_{16}$ is $19{,}999_{10}$.

Example C.2
Let us convert $1A7D_{16}$ into a decimal number.

1 = 1
A = 10
7 = 7
D = 13

$1A7D_{16}$ $= (1 \times 16^3) + (10 \times 16^2) + (7 \times 16^1) + (13 \times 16^0)$
$= (1 \text{ x } 4{,}096) + (10 \times 256) + (7 \times 16) + (13 \times 1)$
$= 4{,}096 + 2{,}560 + 112 + 13$
$= 6{,}781_{10}$

Therefore, the decimal representation of $1A7D_{16}$ is $6{,}781_{10}$.

Example C.3
Let us convert $E8B_{16}$ into a decimal number.

E = 14
8 = 8
B = 11

$E8B_{16}$ $= (14 \times 16^2) + (8 \times 16^1) + (1 \times 16^0)$
$= (14 \times 256) + (8 \times 16) + (11 \times 1)$
$= 3{,}584 + 128 + 11$
$= 3{,}723_{10}$

Therefore, the decimal representation of $E8B_{16}$ is $3{,}723_{10}$.

D. <u>Decimal to Binary</u>

A decimal number has base 10 and a binary number has base 2. In decimal to binary conversion, the base of the number also changes, i.e. from base 10 to base 2. All the decimal numbers have their equivalent binary numbers. These binary numbers are majorly used in computer applications, where it is used for programming or coding purposes. This is because computers understand the language of binary digits, o and 1. There are different methods in converting a given decimal to binary such as formula, division method, and so on.

<u>Method 1: Performing Short Division by Two with Remainder</u>

To convert decimal to binary numbers, proceed with the steps given below:

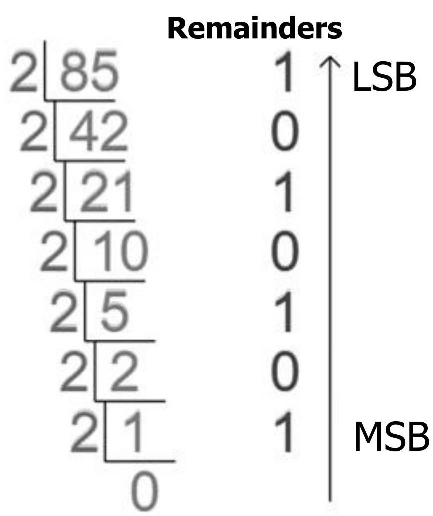
Step 1: Divide the given decimal number by "2" where it gives the result along with the remainder.

Step 2: If the given decimal number is even, then the result will be whole and it gives the remainder "0"

Step 3: If the given decimal number is odd, then the result is not divided properly and it gives the remainder "1".

Step 4: By placing all the remainders in order in such a way, the Least Significant Bit (LSB) at the top and Most Significant Bit (MSB) at the bottom, the required binary number will be obtained.

Convert 85_{10} to binary equivalent.



- When 85 is divided by 2, the quotient is 42 and the remainder is 1.
- When 42 is divided by 2, the quotient is 21 and the remainder is 0.
- When 21 is divided by 2, the quotient is 10 and the remainder is 1.
- When 10 is divided by 2, the quotient is 5 and the remainder is 0.
- When 5 is divided by 2, the quotient is 2 and the remainder is 1.
- When 2 is divided by 2, the quotient is 1 and the remainder is 0.
- When 1 is divided by 2, the quotient is 0 and the remainder is 1.
- Write the remainders from bottom to top: $\mathbf{85_{10} = 1010101_2}$

Example D.1
Convert 17_{10} into a binary number

Divide by 2	Result	Remainder	Binary Value
17 ÷ 2	8	1	1 (LSB)
8 ÷ 2	4	0	0
4 ÷ 2	2	0	0
2 ÷ 2	1	0	0
1 ÷ 2	0	1	1 (MSB)

Therefore, the binary equivalent for the given decimal number $17_{10} = 10001_2$

Example D.2
Convert the given decimal number 294 into a binary number.

Divide by 2	Result	Remainder	Binary Value
294 ÷ 2	147	0	0 (LSB)
147 ÷ 2	73	1	1
73 ÷ 2	36	1	1
36 ÷ 2	18	0	0
18 ÷ 2	9	0	0
9 ÷ 2	4	1	1
4 ÷ 2	2	0	0
2 ÷ 2	1	0	0
1 ÷ 2	0	1	1 (MSB)

Therefore, the binary equivalent for the given decimal number 294_{10} is 100100110_2

Example D.3
Convert 160_{10} to binary Number

Divide by 2	Result	Remainder	Binary Value
160 ÷ 2	80	0	0 (LSB)
80 ÷ 2	40	0	0
40 ÷ 2	20	0	0
20 ÷ 2	10	0	0
10 ÷ 2	5	0	0
5 ÷ 2	2	1	1
2 ÷ 2	1	0	0
1 ÷ 2	0	1	1 (MSB)

Therefore, the binary equivalent for the given decimal number $160_{10} = 10100000_2$

How about converting a decimal value with fractional number.

- Convert 17.75_{10} to binary.

Step 1: Get the equivalent binary value of the whole number part of the decimal value, which is 17.

Divide by 2	Result	Remainder	Binary Value
17 ÷ 2	8	1	1 (LSB)
8 ÷ 2	4	0	0
4 ÷ 2	2	0	0
2 ÷ 2	1	0	0
1 ÷ 2	0	1	1 (MSB)

Therefore, the binary equivalent of the whole number part of 17.75 is 10001_2

Step 2: Get the equivalent binary value of fractional part of the decimal value which is 75 by multiplying the fractional part repeatedly by 2 until it becomes 0.

Multiply by 2	Result	Remainder	Binary Value
0.75 x 2	**1**.50	1	1 (MSB)
0.50 x 2	**1**.00	1	1 (LSB)

From top to bottom, write the integer parts of the results to the fractional part of the number in base 2.

Therefore, the binary equivalent of the decimal number fractional value $0.75_{10} = 0.11_2$

Step 3: Combine the whole number and fractional parts to obtain the overall result.

$$17.75_{10} = 10001_2 + 0.11_2 = 10001.11_2$$

Therefore, the binary equivalent of the decimal number $17.75_{10} = 10001.11_2$

Example D.4

Convert 5.375_{10} to binary Number

Divide by 2	Result	Remainder	Binary Value
5 ÷ 2	2	1	1 (LSB)
2 ÷ 2	1	0	0
1 ÷ 2	0	1	1 (MSB)

Multiply by 2	Result	Remainder	Binary Value
0.375 x 2	0.75	0	0 (MSB)
0.75 x 2	1.5	1	1
0.5 x 2	1.0	1	1 (LSB)

$$5.375_{10} = 101_2 + 0.011_2 = 101.011_2$$

Therefore, the binary equivalent of the decimal number $5.375_{10} = 101.011_2$

Example D.5
Convert 52.125_{10} to binary Number

Divide by 2	Result	Remainder	Binary Value
52 ÷ 2	26	0	0 (LSB)
26 ÷ 2	13	0	0
13 ÷ 2	6	1	1
6 ÷ 2	3	0	0
3 ÷ 2	1	1	1
1 ÷ 2	0	1	1 (MSB)

Multiply by 2	Result	Remainder	Binary Value
0.125 x 2	0.25	0	0 (MSB)
0.25 x 2	0.5	0	0
0.5 x 2	1.0	1	1 (LSB)

$$52.125_{10} = 110100_2 + 0.001_2 = 110100.001_2$$

Therefore, the binary equivalent of the decimal number $52.125_{10} = 110100.001_2$

Example D.6
Convert 3.12_{10} to binary Number

Divide by 2	Result	Remainder	Binary Value
3 ÷ 2	1	1	1
1 ÷ 2	0	1	1 (MSB)

Multiply by 2	Result	Remainder	Binary Value
0.12 x 2	0.24	0	0 (MSB)
0.24 x 2	0.48	0	0
0.48 x 2	0.96	0	0
0.96 x 2	1.92	1	1
0.92 x 2	1.84	1	1
0.84 x 2	1.68	1	1
0.68 x 2	1.36	1	1
0.36 x 2	0.75	0	0

- Multiply the fractional part repeatedly by 2 until it becomes 0. Stop after a number of steps if the fractional part does not become 0.

$$3.12_{10} = 11_2 + 0.00011110\ldots_2 = 0.00011110\ldots_2$$

Therefore, the binary equivalent of the decimal number $3.12_{10} = 11.\ 0.00011110\ldots_2$

<u>Method 2: convert a decimal to binary using the powers of 2 and subtraction</u>

Another method that can be used to convert decimal numbers into binary is with the powers of 2 and subtraction. Using this method, it may be easier to change binary numbers into decimal numbers than the first technique.

Convert 85_{10} to binary equivalent.

Step 1: Create a table with Power of 2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	*64*	*32*	*16*	*8*	*4*	*2*	*1*

- If your original decimal number is larger than 128, continue to write the powers of 2 until the result of the final power of 2 exceeds it such as $2^8 \rightarrow 256$, $2^9 \rightarrow 512$, $2^{10} \rightarrow 1024$, $2^{11} \rightarrow 2048$ and so on.

Step 2: Choose the highest power of 2 that fits into the decimal number and write the number 1 next to it.

- 85 is less than 128 but greater than 64, this means that 85 should be reduced with 64.
- 85 – 64 = 21

Step 3: Repeat step 2 until reaching 0 after subtraction.
- 21 is less than 32 but greater than 16, this means 21 – 16 = 5.
- 5 is less than 8 but greater than 4, this means 5 – 4 = 1
- 1 is equal to 1, this means 1 – 1 = 0 ← it reach 0.

➢ Place 1 to the value which is can be divide than place 0 for those value which is not divisible.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	*64*	*32*	*16*	*8*	*4*	*2*	*1*
0	1	0	1	0	1	0	1

Therefore, 85_{10} is equal to 01010101_2 (if the value of first binary value is 0, it can be omitted which means it can be written as 1010101_2)

Example D.7
Convert 17_{10} into a binary number

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	*64*	*32*	*16*	*8*	*4*	*2*	*1*
0	0	0	1	0	0	0	1

➢ 16 + 1 = 17

Therefore, 17_{10} is equal 1001_2

Example D.8
Convert 294_{10} into a binary number

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	*256*	*128*	*64*	*32*	*16*	*8*	*4*	*2*	*1*
0	1	0	0	1	0	0	1	1	0

➢ 256 + 32 + 4 + 2 + 1 = 294

Therefore, 294_{10} is 100100110_2

E. Decimal to Octal

A number with base 8 is the octal number and a number with base 10 is the decimal number. To convert decimal to octal, decimal value is divided by the number 8 and write the remainders in the reverse order to get the equivalent octal number.

Convert Decimal to Octal with Steps

Follow the steps given below to learn the decimal to octal conversion:
Step 1: Write the given decimal number
Step 2: If the given decimal number is less than 8 the octal number is the same.
Step 3: If the decimal number is greater than 7 then divide the number by 8.
Step 4: Note the remainder that been get after division
Step 5: Repeat step 3 and 4 with the quotient till it is less than 8
Step 6: Now, write the remainders in reverse order (bottom to top)
Step 7: The resultant is the equivalent octal number to the given decimal number.

For example: Convert $1{,}792_{10}$ into an octal number.

Decimal Number	Quotient	Remainder	Octal Number
1792 ÷ 8	224	0	0
224 ÷ 8	28	0	0
28 ÷ 8	3	4	4
3 ÷ 8	0	3	3

Therefore, $1{,}792_{10}$ is equal to $3{,}400_8$

Example E.1

Convert 127_{10} into a octal number

Decimal Number	Quotient	Remainder	Octal Number
127 ÷ 8	15	7	7
15 ÷ 8	1	7	7
1 ÷ 8	0	1	1

Therefore, 127_{10} is equal to 177_8

Example E.2

Convert 52_{10} into a octal number

Decimal Number	Quotient	Remainder	Octal Number
52 ÷ 8	6	4	4
6 ÷ 8	0	6	6

Therefore, 52_{10} is equal to 64_8

Example E.3

Convert 100_{10} into a octal number

Decimal Number	Quotient	Remainder	Octal Number
100 ÷ 8	12	4	4
12 ÷ 8	1	4	4
1 ÷ 8	0	1	1

Therefore, 100_{10} is equal to 144_8

F. Decimal to Hexadecimal

The base of a decimal number system is 10 and is also called a radix. Hence, it has ten symbols, more precisely the numbers from 0 to 9, i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The base or radix of a hexadecimal number system is 16. Being a base-16 numeral system, it uses 16 symbols. These include ten decimal digits, i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and the first six letters of the English alphabet, i.e. A, B, C, D, E, F. However, these letters are used to represent the values 10, 11, 12, 13, 14 and 15 respectively in one single figure.

Decimal Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Equivalent Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Decimal to Hexadecimal Conversion With Steps

Go through the steps given below to learn how to convert the numbers from decimal to hex.

Step 1: First, divide the decimal number by 16, considering the number as an integer.

Step 2: Keep aside the remainder.

Step 3: Again divide the quotient by 16 and repeat till you get the quotient value equal to zero.

Step 4: Now take the values of the remainder’s left in the reverse order to get the hexadecimal numbers.

Note: Remember, from 0 to 9, the numbers will be counted as the same in the decimal system. But from 10 to 15, they are expressed in alphabetical order such as A, B, C, D, E, F and so on.

Convert (960)10 into hexadecimal.

Step 1: First, divide 960_{10} by 16.

$960 \div 16 = 60$ and remainder $= 0$

Step 2: Again, divide quotient 60 by 16.

$60 \div 16 = 3$ and remainder 12.

Step 3: Again dividing 3 by 16, will leave quotient=0 and remainder = 3.

Step 4: Now taking the remainder in reverse order and substituting the equivalent hexadecimal value for them

$3 \rightarrow 3$, $12 \rightarrow C$ and $0 \rightarrow 0$

Divide by 16	Quotient	Remainder	Hex Value
960 ÷ 16	60	0	0
60 ÷ 16	3	12	C
3 ÷ 16	0	3	3

Therefore, $960_{10} = 3C0_{16}$

Example F.1

Convert 49_{10} into an octal number.

Divide by 16	Quotient	Remainder	Hex Value
49 ÷ 16	3	1	1
3 ÷ 16	0	3	3

Therefore, $49_{10} = 31_{16}$.

Example F.2
Convert $1{,}228_{10}$ into an octal number.

Divide by 16	Quotient	Remainder	Hex Value
1228 ÷ 16	76	12	C
76 ÷ 16	4	12	C
4 ÷ 16	0	4	4

Therefore, $1{,}228_{10} = 4CC_{16}$

Example F.3
Convert 600 into an octal number.

Divide by 16	Quotient	Remainder	Hex Value
600 ÷ 16	37	8	8
37 ÷ 16	2	5	5
2 ÷ 16	0	2	2

Therefore, $600_{10} = 258_{16}$

G. Binary to Octal

Binary numbers are commonly used in computers, in the form of bits and bytes, since the computer understand the language of 0 and 1 only. At the same time, octal numbers are used in electronics.

To convert binary numbers to octal numbers, follow the below steps:

1. Take the given binary number
2. Multiply each digit by 2n-1 where n is the position of the digit from the decimal
3. The resultant is the equivalent decimal number for the given binary number
4. Divide the decimal number by 8
5. Note the remainder
6. Continue the above two steps with the quotient till the quotient is zero
7. Write the remainder in the reverse order
8. The resultant is the required octal number for the given binary number

Convert 1010101_2 to octal

1. Convert given binary to decimal

$$\begin{aligned} 1010101_2 &= (1 * 2^6) + (0 * 2^5) + (1 * 2^4) + (0 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0) \\ &= 64 + 0 + 16 + 0 + 4 + 0 + 1 \\ &= 64 + 21 \\ &= 85 \text{ (Decimal form)} \end{aligned}$$

2. Convert the decimal to octal form

85 / 8 = 10 → 5
10 / 8 = 1 → 2
1 / 8 = 0 → 1

Therefore, $1010101_2 = 125_8$

H. Octal to Binary

Conversion of octal to binary number is a two-step process. First, we need to convert the given octal number into its equivalent decimal number and then convert the decimal into binary. Let us learn the complete steps here.

Step 1: Octal to Decimal Conversion
- Count the number of digits present in the given number. Let the number of digits be 'n'.
- Now multiply each digit of the number with 8n-1, when the digit is in the nth position from the right end of the number. If the number has a decimal part, multiply each digit in the decimal part by `8-m` when the digit is in the mth position from the decimal point.
- Add all the terms after multiplication.
- The obtained value is the equivalent decimal number.

Step 2: Decimal to Binary Conversion
- Take the above-produced decimal number and divide it by 2.
- Note down the remainder.
- Continue the above two steps for the quotient till the quotient is zero.
- Write the remainder in the reverse order.
- The received number is the equivalent binary number for the given octal number.

Convert 41_8 to a binary number.

Step 1.

$41_8 = (4 * 8^1) + (1 * 8^0)$
$= (4 * 8) + (1 * 1)$
$= 32 + 1$
$= 33$(Decimal number)

Step 2.

Decimal Number divided by 2	Quotient	Remainder
33 divided by 2	16	1
16 divided by 2	8	0
8 divided by 2	4	0
4 divided by 2	2	0
2 divided by 2	1	0
1 divided by 2	0	1

Therefore, 41_8 equivalent binary number is 100001_2.

I. Hexadecimal to Binary

Convert a hexadecimal number into its equivalent binary number, follow the steps:

Step 1: Take given hexadecimal number
Step 2: Find the number of digits in the decimal
Step 3: If it has n digits, multiply each digit with 16n-1 where the digit is in the nth position
Step 4: Add the terms after multiplication
Step 5: The result is the decimal number equivalent to the given hexadecimal number. Now we have to convert this decimal to binary number.

Step 6: Divide the decimal number with 2
Step 7: Note the remainder
Step 8: Do the above 2 steps for the quotient till the quotient is zero
Step 9: Write the remainders in the reverse order.
Step 10: The result is the required binary number.

Hence, from the above steps it is clear that how to convert any hexadecimal number into binary, i.e. first, it need to convert hexadecimal to decimal number and then decimal to binary.

Convert $A2B_{16}$ to an equivalent binary number.

1. Convert the given hexadecimal to the equivalent decimal number.

$A2B_{16}$ = (A × 162) + (2 × 161) + (B × 160)
= (A × 256) + (2 × 16) + (B × 1)
= (10 ×256) + 32 + 11
= 2560 + 43
= 2603(Decimal number)

2. Convert 2603_{10} to binary

Divide by 2	Result	Remainder	Binary Value
2603 ÷ 2	1301	1	1 (LSB)
1301 ÷ 2	650	1	1
650 ÷ 2	325	0	0
325 ÷ 2	162	1	1
162 ÷ 2	81	0	0
81 ÷ 2	40	1	1
40 ÷ 2	20	0	0
20 ÷ 2	10	0	0
10 ÷ 2	5	0	0
5 ÷ 2	2	1	1
2 ÷ 2	1	0	0
1 ÷ 2	0	1	1 (MSB)

The binary number obtained is 101000101011_2

Therefore, $A2B_{16} = 101000101011_2$

J. Hexadecimal to Octal

Hexadecimal numbers include binary digits; therefore, it can club these binary numbers into a pair so that we can relate it with the octal numbers. Let us check the method with steps and example:
Step 1: For each given hexadecimal number digit, write the equivalent binary number. If any of the binary equivalents are less than 4 digits, add 0's to the left side.
Step 2: Combine and make the groups of binary digits from right to left, each containing 3 digits. Add 0's to the left if there are less than 3 digits in the last group.
Step 3: Find the octal equivalent of each binary group.

Hex to Octal Conversion Table

Hexadecimal	Octal	Equivalent Decimal	Equivalent Binary
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	10	8	1000
9	11	9	1001
A	12	10	1010
B	13	11	1011
C	14	12	1100
D	15	13	1101
E	16	14	1110
F	17	15	1111

Convert $1BC_{16}$ into an octal number.

First, find the binary equivalent of each hexadecimal value (*refer to the table above*)

1 → 0001
B → 1011
C → 1100

Second, group them from right to left, each having 3 digits.

a. 0001 + 1011 + 1100
b. 000110111100
c. 000, 110, 111, 100
d. 000→0, 110 →6, 111→7, 100→4

Therefore, $1BC_{16} = 674_8$

Practice Exercises 2.1

A. Convert the following Decimal Number (Base10) to Binary Number (Base2), Octal Number (Base8) and Hexadecimal Number (Base16).
   1. 100_{10}
   2. 53_{10}
   3. 201_{10}
B. Convert the Binary Number (Base10) to Decimal Number (Base10) and Octal Number (Base8).
   4. 10101011_2
   5. 11101110_2
C. Convert the Hexadecimal Number (Base16) into Decimal Number (Base10) and Octal Number (Base8).
   6. F41
   7. 1D2

Answer to Practice Exercises 2.1

A.	1.) Binary = 1100100_2	Octal = 144_8	Hexadecimal = 64_{16}
	2.) Binary = 10011001_2	Octal = 65_8	Hexadecimal = 35_{16}
	3.) Binary = 11001001_2	Octal = 311_8	Hexadecimal = $C9_{16}$
B.	4.) Decimal = 171_{10}	Octal = 253_8	
	5.) Decimal = 238_{10}	Octal = 356_8	
C.	6.) Decimal = 3905_{10}	Octal = 7501_8	
	7.) Decimal = 466_{10}	Octal = 722_8	

2. Binary Arithmetic Operation

The binary addition operation works similarly to the base 10 decimal system, except that it is a base 2 system. The binary system consists of only two digits, 1 and 0. Most of the functionalities of the computer system use the binary number system. The binary code uses the digits 1's and 0's to make certain processes turn off or on. The process of the addition operation is very familiar to the decimal system by adjusting to the base 2.

Before attempting the binary addition process, we should have complete knowledge of how the place works in the binary number system. Because most of the modern digital computers and electronic circuits perform the binary operation by representing each bit as a voltage signal. The bit 0 represents the "OFF" state, and the bit 1 represents the "ON" state.

A. Addition of Positive and Negative Number

Rules of Binary Addition

Binary addition is much easier than the decimal addition when you remember the following tricks or rules. Using these rules, any binary number can be easily added. The four rules of binary addition are:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 with a carry-over of 1

Carry-overs of binary addition are performed in the same manner as in decimal addition. With the help of the above rules addition of three or more binary numbers can be worked out but this has little use in digital computers.

Add 101 + 101

Procedure for Binary Addition of Numbers:

```
     101
(+)  101
```

- **Step 1:** First consider the 1's column, and add the one's column, (1+1) and it gives the result 10 as per the condition of binary addition.

- **Step 2:** Now, leave the 0 in the one's column and carry the value 1 to the 10's column.

```
        1 carry over
      101
(+)   101
------
        0
```

- **Step 3:** Now add 10's place, 1 + (0 + 0) = 1**.** So, nothing carries to the 100's place and leave the value 1 in the 10's place

```
       1
      101
(+)   101
-----
       10
```

- **Step 4:** Now add the 100's place (1 + 1) = 10. Leave the value 0 in the 100's place and carries 1 to the 1000's place.

```
       1
      101
(+)   101
-----
     1010
```

Therefore, the resultant of the addition operation is 1010.

When you cross-check the binary value with the decimal value, the resultant value should be the same.

The binary value 101 is equal to the decimal value 5

So, 5 + 5 = 10

The decimal number 10 is equal to the binary number 1010.

Binary Addition Table

The table of adding two binary numbers 0 and 1 is given below:

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	0 (where 1 is carried over)

You can see from the above table, x and y are the two binary numbers. So when the input for x = 0 and y = 0, then the output is equal to 0. When x = 0 or 1 and y = 1 or 0, then x + y = 1. But when both x and y are equal to 1, then their addition equals to 0, but the carryover number will equal to 1, which means basically 1 + 1 = 10 in binary addition, where 1 is carry forwarded to the next digit.

Example 1: 10001 + 11101

Solution:

```
         1
   1 0 0 0 1
(+) 1 1 1 0 1
—————————
  1 0 1 1 1 0
```

Example 2: 10111 + 110001

Solution:

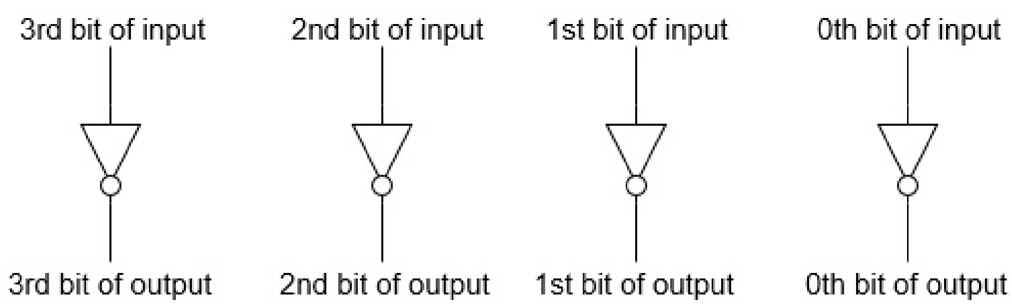
```
   1   1 1 1
     1 0 1 1 1
(+) 1 1 0 0 0 1
—————————
  1 0 0 1 0 0 0
```

<u>Binary Addition Using 1's Complement</u>

To get 1's complement of a binary number, simply invert the given number.

- The number **0** represents the positive sign
- The number **1** represents the negative sign

There is a simple algorithm to convert a binary number into 1's complement. To get 1's complement of a binary number, simply invert the given number. You can simply implement logic circuit using only NOT gate for each bit of Binary number input.



Example-1: Find 1's complement of binary number 10101110.

Simply invert each bit of given binary number, so 1's complement of given number will be 01010001.

Example-2: Find 1's complement of binary number 10001.001.

Simply invert each bit of given binary number, so 1's complement of given number will be 01110.110.

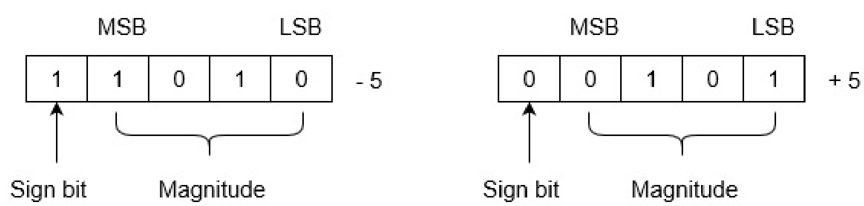
Find 1's complement of each 3 bit binary number.

Simply invert each bit of given binary number, so 1's complement of each 3 bit binary number will be:

Binary number	1's complement
000	111
001	110
010	101
011	100
100	011
101	010
110	001
111	000

1's complement binary numbers are very useful in Signed number representation. Positive numbers are simply represented as Binary number. There is nothing to do for positive binary number. But in case of negative binary number representation, we represent in 1's complement. If the number is negative then it is represented using 1's complement. First represent the number with positive sign and then take 1's complement of that number.

Example: Let we are using 5 bits register. The representation of -5 and +5 will be as follows:



+5 is represented as it is represented in sign magnitude method. -5 is represented using the following steps:

Step 1: +5 = **<u>0</u>** 0101
Step 2: Take 1's complement of **<u>0</u>** 0101 and that is **<u>1</u>** 1010. MSB is 1 which indicates that number is negative.

MSB is always 1 in case of negative numbers.

- *Case 1: Addition of the positive number with a negative number when the positive number has a greater magnitude.*

In this case addition of numbers is performed after taking 1's complement of the negative number and the end-around carry of the sum is added to the least significant bit.

Initially, calculate the 1's complement of the given negative number. Sum up with the given positive number. If we get the end-around carry 1, it gets added to the LSB.

Add 1101 and -1001

1. First, find the 1's complement of the negative number 1001. So, for finding 1's complement, change all 0 to 1 and all 1 to 0.

   -1001 $\Rightarrow$ 0110

2. Now, add both the numbers, i.e., 1101 and 0110;

```
     1
     1 1 0 1
 (+) 0 1 1 0
   1 0 0 1 1
```

3. By adding both numbers, we get the end-around carry 1. We add this end around carry to the LSB of 0011.

```
     1 1
   0 0 1 1
(+)      1
   0 1 0 1
```

Checking:
1101 → 13
(+) - 1001 → 9
4 ← 0100

Example A.1: Add 1110 and -1101

Solution:

```
                    1 1
 + 1 1 1 0          1 1 1 0
 - 1 1 0 1   ⇒  (+) 0 0 1 0
                    0 0 0 0
                          1 ← carry
                    0 0 0 1
```

Checking:
1110 → 14
(+) - 1101 → 13
1 ← 0001

Example A.2: Add 1010 and -0101

Solution:

```
                      1
 + 1 0 1 0          1 0 1 0
 - 0 1 0 1   ⇒  (+) 1 0 1 0
                    0 1 0 0
                          1 ← carry
                    0 1 0 1
```

Checking:
1010 → 10
(+) - 0101 → 5
5 ← 0101

- *Case 2: Adding a positive value with a negative value in case the negative number has a higher magnitude.*

In this case the addition is carried in the same way as in case 1 but there will be non end-around carry. The sum is obtained by taking 1's complement of the magnitude bits of the result and it will be negative.

Initially, calculate the 1's complement of the negative value. Sum it with a positive number. In this case, we did not get the end-around carry. So, take the 1's complement of the result to get the final result.

Add 1101 and -1110

1. First find the 1's complement of the negative number 1110. So, for finding 1's complement, we change all 0 to 1, and all 1 to 0.

   -1110 ⇒ 0001

2. Now, add both the numbers, i.e., 1101 and 0001;

```
      1
    1 1 0 1
(+) 0 0 0 1
    -------
    1 1 1 0
```

3. Now, find the 1's complement of the result 1110 that is the final result. So, the 1's complement of the result 1110 is 0001, and we add a negative sign before the number so that we can identify that it is a negative number.

   1 1 1 0 ⇒ (-) 0 0 0 1

Checking:

```
      1101 → 13
(+) - 1110 → 14
             --
             -1 ← (-) 0101
```

Example A.3: Add 1010 and -1100

Solution:

```
                      1
   1 0 1 0          1 0 1 0
 - 1 1 0 0  ⇒  (+)  0 0 1 1
                    -------
                    1 1 0 1  ⇒ (-) 0 0 1 0
```

Checking:

```
      1010 → 10
(+) - 1100 → 12
             --
             -2 ← (-) 0010
```

Example A.4: Add 0011 and -1111

Solution:

```
   0 0 1 1          0 0 1 1
 - 1 1 1 1  ⇒  (+)  0 0 0 0
                    -------
                    0 0 1 1  ⇒ (-) 1 1 0 0
```

Checking:

```
      0011 →  3
(+) - 1111 → 15
             --
            -12 ← (-) 1100
```

- *Case 3: Addition of two negative numbers*

In this case, first find the 1's complement of both the negative numbers, and then we add both these complement numbers. In this case, we always get the end-around carry, which get added to the LSB, and for getting the final result, we take the 1's complement of the result.

For the addition of two negative numbers 1’s complements of both the numbers are to be taken and then added. In this case an end-around carry will always appear. This along with a carry from the MSB (i.e. the 4th bit in the case of sign-plus-magnitude 5-bit register) will generate a 1 in the sign bit. 1's complement of the magnitude bits of the result of addition will give the final sum.

Add -1101 and -1110

1. Firstly find the 1's complement of the negative numbers 1101 and 01110. So, for finding 1's complement, we change all 0 to 1, and all 1 to 0.

   - 1110 ⇒ **1** 0001
   - 1101 ⇒ **1** 0010

2. Now, we add both the complement numbers, i.e., 10001 and 10010;

```
     1 0 0 0 1
(+)  1 0 0 1 0
   1 0 0 0 1 1
```

3. By adding both numbers, we get the end-around carry 1. We add this end-around carry to the LSB of 00011.

```
    0 0 0 1 1
(+)         1
    0 0 1 0 0
```

4. Now, find the 1's complement of the result 00100 that is the final answer. So, the 1's complement of the result 00100 is 11011, and add a negative sign before the number so that we can identify that it is a negative number.

   00100 ⇒ (-) 11011

Checking:

```
     - 1101 → - 13
(+)  - 1110 → - 14
              - 27 ← (-) 11011
```

Example A.5: Add -1010 and -0101

Solution:

```
- 1 0 1 0          1 0 1 0 1
- 0 1 0 1  ⇒ (+)  1 1 0 1 0
                   0 1 1 1 1
                           1
                   1 0 0 0 0 ⇒ (-) 01111 or (-) 1111
```

Checking:

```
     - 1010 → - 10
(+)  - 0101 → -  5
              - 15 ← (-) 1111
```

Example A.6: Add -0110 and -0111

Solution:

```
- 0 1 1 0          1 1 0 0 1
- 0 1 1 1  ⇒ (+)  1 1 0 0 0
                   1 0 0 0 1
```

Checking:

```
     - 0110 → -  6
(+)  - 0111 → -  7
              - 13 ← (-) 1101
```

$$\begin{array}{r} \underline{\qquad\qquad 1} \\ 1\,0\,0\,1\,0 \Rightarrow (-)\ 01101\ \textit{or}\ (-)\ 1101 \end{array}$$

B. Subtraction of Positive and Negative Number

Binary subtraction is also similar to that of decimal subtraction with the difference that when 1 is subtracted from 0, it is necessary to borrow 1 from the next higher order bit and that bit is reduced by 1 (or 1 is added to the next bit of subtrahend) and the remainder is 1.

Thus the rules of binary subtraction are as follows:

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 with a borrow of 1

Procedure to do Binary Subtraction:

Subtract 101 from 1010

$$\begin{array}{lr} & 1010 \\ (-) & 101 \end{array}$$

Step 1: First consider the 1's column, and subtract the one's column,(0 – 1) and it gives the result 1 as per the condition of binary subtraction with a borrow of 1 from the 10's place.

Step 2: After borrowing 1 from the 10's column, the value 1 in the 10's column is changed into the value 0

$$\begin{array}{lcccc} & & & \textit{0} & \textit{10}\ \text{Borrow} \\ & 1 & 0 & \cancel{1} & \cancel{0} \\ (-) & & 1 & 0 & 1 \\ \hline & & & & 1 \end{array}$$

Step 3: So, subtract the value in the 10's place, (0 – 0) = 0.

$$\begin{array}{lcccc} & & & \textit{0} & \textit{10}\ \text{Borrow} \\ & 1 & 0 & \cancel{1} & \cancel{0} \\ (-) & & 1 & 0 & 1 \\ \hline & & & 0 & 1 \end{array}$$

Step 4: Now subtract the values in 100's place. Borrow 1 from the 1000's place (0 – 1) = 1.

$$\begin{array}{l|c|c|c|c} & \textit{0} & \textit{10} & \textit{0} & \textit{10}\ \text{Borrow} \\ & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{0} \\ (-) & & 1 & 0 & 1 \\ \hline & 0 & 1 & 0 & 1 \end{array}$$

Checking:

$$\begin{array}{lrcr} & 1010 & \rightarrow & 10 \\ (-) & 101 & \rightarrow & \underline{5} \\ & 0101 & \leftarrow & 5 \end{array}$$

So, the resultant of the subtraction operation is 0101.

Example B.1: Subtract 1010 from 1100

```
          0 10
      1  ̶1̶  ̶0̶  0
  (-) 1  0  1  0
      0  0  1  0
```

Checking:

```
     1100 → 12
(-)  1010 → 10
             2 ← 0010
```

Example B.2: Subtract 111 from 1000

```
          1   1
      0 1̶0̶ 1̶0̶ 10
      1̶  0̶  0̶  0
  (-)    1  1  1
      0  0  0  1
```

Checking:

```
     1000 → 8
(-)   111 → 7
            1 ← 0001
```

Practice Exercises 2.2

A. Add the following:
   1. 11011 + 10101
   2. 10001 + 1101

B. Perform subtraction operation on the following:
   3. 11101 – 1011
   4. 1000 – 10

Answer to Practice Exercises 2.2

A. 1.) 110000 2.) 11110

B. 3.) 10010 4.) 0110