UNIT 1:

Number system

If base or radix of a number system is ‘r’, then the numbers present in that number system are ranging from zero to r-1. The total numbers present in that number system is ‘r’. So, we will get various number systems, by choosing the values of radix as greater than or equal to two.

In this chapter, let us discuss about the **popular number systems** and how to represent a number in the respective number system. The following number systems are the most commonly used.

- Decimal Number system
- Binary Number system
- Octal Number system
- Hexadecimal Number system

Decimal Number System

The **base** or radix of Decimal number system is **10**. So, the numbers ranging from 0 to 9 are used in this number system. The part of the number that lies to the left of the **decimal point** is known as integer part. Similarly, the part of the number that lies to the right of the decimal point is known as fractional part.

In this number system, the successive positions to the left of the decimal point having weights of 10^0, 10^1, 10^2, 10^3 and so on. Similarly, the successive positions to the right of the decimal point having weights of 10^{-1}, 10^{-2}, 10^{-3} and so on. That means, each position has specific weight, which is **power of base 10**

Example

Consider the **decimal number 1358.246**. Integer part of this number is 1358 and fractional part of this number is 0.246. The digits 8, 5, 3 and 1 have weights of 100, 101, 10^2 and 10^3 respectively. Similarly, the digits 2, 4 and 6 have weights of 10^{-1}, 10^{-2} and 10^{-3} respectively.

Mathematically, we can write it as

$1358.246 = (1 \times 10^3) + (3 \times 10^2) + (5 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) +$

$(4 \times 10^{-2}) + (6 \times 10^{-3})$

After simplifying the right hand side terms, we will get the decimal number, which is on left hand side.

Binary Number System

All digital circuits and systems use this binary number system. The **base** or radix of this number system is **2**. So, the numbers 0 and 1 are used in this number system.

The part of the number, which lies to the left of the **binary point** is known as integer part. Similarly, the part of the number, which lies to the right of the binary point is known as fractional part.

In this number system, the successive positions to the left of the binary point having weights of 2^0, 2^1, 2^2, 2^3 and so on. Similarly, the successive positions to the right of the binary point having weights of 2^{-1}, 2^{-2}, 2^{-3} and so on. That means, each position has specific weight, which is **power of base 2**.

Example

Consider the **binary number 1101.011**. Integer part of this number is 1101 and fractional part of this number is 0.011. The digits 1, 0, 1 and 1 of integer part have weights of 2^0, 2^1, 2^2, 2^3 respectively. Similarly, the digits 0, 1 and 1 of fractional part have weights of 2^{-1}, 2^{-2}, 2^{-3} respectively.

Mathematically, we can write it as

$1101.011 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) +$

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

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of binary number on left hand side.

Octal Number System

The **base** or radix of octal number system is **8**. So, the numbers ranging from 0 to 7 are used in this number system. The part of the number that lies to the left of the **octal point** is known as integer part. Similarly, the part of the number that lies to the right of the octal point is known as fractional part.

In this number system, the successive positions to the left of the octal point having weights of 8^0, 8^1, 8^2, 8^3 and so on. Similarly, the successive positions to the right of the octal point having weights of 8^{-1}, 8^{-2}, 8^{-3} and so on. That means, each position has specific weight, which is **power of base 8**.

Example

Consider the **octal number 1457.236**. Integer part of this number is 1457 and fractional part of this number is 0.236. The digits 7, 5, 4 and 1 have weights of 8^0, 8^1, 8^2 and 8^3 respectively. Similarly, the digits 2, 3 and 6 have weights of 8^{-1}, 8^{-2}, 8^{-3} respectively.

Mathematically, we can write it as

$1457.236 = (1 \times 8^3) + (4 \times 8^2) + (5 \times 8^1) + (7 \times 8^0) + (2 \times 8^{-1}) +$

$(3 \times 8^{-2}) + (6 \times 8^{-3})$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of octal number on left hand side.

Hexadecimal Number System

The **base** or radix of Hexa-decimal number system is **16**. So, the numbers ranging from 0 to 9 and the letters from A to F are used in this number system. The decimal equivalent of Hexa-decimal digits from A to F are 10 to 15.

The part of the number, which lies to the left of the **hexadecimal point** is known as integer part. Similarly, the part of the number, which lies to the right of the Hexa-decimal point is known as fractional part.

In this number system, the successive positions to the left of the Hexa-decimal point having weights of 16^0, 16^1, 16^2, 16^3 and so on. Similarly, the successive positions to the right of the Hexa-decimal point having weights of 16^{-1}, 16^{-2}, 16^{-3} and so on. That means, each position has specific weight, which is **power of base 16**.

Example

Consider the **Hexa-decimal number 1A05.2C4**. Integer part of this number is 1A05 and fractional part of this number is 0.2C4. The digits 5, 0, A and 1 have weights of 16^0, 16^1, 16^2 and 16^3 respectively. Similarly, the digits 2, C and 4 have weights of 16^{-1}, 16^{-2} and 16^{-3} respectively.

Mathematically, we can write it as

$1A05.2C4 = (1 \times 16^3) + (10 \times 16^2) + (0 \times 16^1) + (5 \times 16^0) + (2 \times 16^{-1}) +$

$(12 \times 16^{-2}) + (4 \times 16^{-3})$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of Hexa-decimal number on left hand side.

Number System Relationship

The following table shows the association between binary, octal, decimal, and hexadecimal number systems.

BINARY	DECIMAL	OCTAL	HEXADECIMAL
0000	0	0	0
0001	1	1	1

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

Base Conversions

As we know, the number system is a form of expressing the numbers. In **number system conversion**, we will study to convert a number of one base, to a number of another base. There are a variety of number systems such as binary numbers, decimal numbers, hexadecimal numbers, octal numbers, which can be exercised.

In this article, you will learn the conversion of one base number to another base number considering all the base numbers such as decimal, binary, octal and hexadecimal with the help of examples. Here, the following number system conversion methods are explained.

- Binary to Decimal Number System
- Decimal to Binary Number System
- Octal to Binary Number System
- Binary to Octal Number System
- Binary to Hexadecimal Number System
- Hexadecimal to Binary Number System

Get the pdf of number system with a brief description in it. The general representation of number systems are;

Decimal Number – Base 10 – N_{10}

Binary Number – Base 2 – N_2

Octal Number – Base 8 – N_8

Hexadecimal Number – Base 16 – N_{16}

Number System Conversion Table

Binary Numbers	Octal Numbers	Decimal Numbers	Hexadecimal Numbers
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9

1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Number System Conversion Methods

Number system conversions deal with the operations to change the base of the numbers. For example, to change a decimal number with base 10 to binary number with base 2. We can also perform the arithmetic operations like addition, subtraction, multiplication on the number system. Here, we will learn the methods to convert the number of one base to the number of another base starting with the decimal number system. The representation of number system base conversion in general form for any base number is;

$(Number)_b = d_{n-1} d_{n-2} —– d_1 d_0 . d_{-1} d_{-2} —- d_{-m}$

In the above expression, $d_{n-1} d_{n-2} —– d_1 d_0$ represents the value of integer part and $d_{-1} d_{-2} —- d_{-m}$ represents the fractional part.

Also, d_{n-1} is the Most significant bit (MSB) and d_{-m} is the Least significant bit (LSB).

Now let us learn, conversion from one base to another.

Related Topics	
Binary Number System	Hexadecimal Number System
Octal Number System	Number System For Class 9

Decimal to Other Bases

Converting a decimal number to other base numbers is easy. We have to divide the decimal number by the converted value of the new base.

Decimal to Binary Number:

Suppose if we have to convert decimal to binary, then divide the decimal number by 2.

Example 1. Convert $(25)_{10}$ to binary number.

Solution: Let us create a table based on this question.

Operation	Output	Remainder
25 ÷ 2	12	1(MSB)
12 ÷ 2`	6	0
6 ÷ 2	3	0
3 ÷ 2	1	1
1 ÷ 2	0	1(LSB)

Therefore, from the above table, we can write,

$(25)_{10} = (11001)_2$

Decimal to Octal Number:

To convert decimal to octal number we have to divide the given original number by 8 such that base 10 changes to base 8. Let us understand with the help of an example.

Example 2: Convert 128_{10} to octal number.

Solution: Let us represent the conversion in tabular form.

Operation	Output	Remainder
128÷8	16	0(MSB)
16÷8	2	0
2÷8	0	2(LSB)

Therefore, the equivalent octal number = 200_8

Decimal to Hexadecimal:

Again in decimal to hex conversion, we have to divide the given decimal number by 16.

Example 3: Convert 128_{10} to hex.

Solution: As per the method, we can create a table;

Operation	Output	Remainder
128÷16	8	0(MSB)
8÷16	0	8(LSB)

Therefore, the equivalent hexadecimal number is 80_{16}

Here MSB stands for a Most significant bit and LSB stands for a least significant bit.

Other Base System to Decimal Conversion

Binary to Decimal:

In this conversion, binary number to a decimal number, we use multiplication method, in such a way that, if a number with base n has to be converted into a number with base 10, then each digit of the given number is multiplied from MSB to LSB with reducing the power of the base. Let us understand this conversion with the help of an example.

Example 1. Convert $(1101)_2$ into a decimal number.

Solution: Given a binary number $(1101)_2$.

Now, multiplying each digit from MSB to LSB with reducing the power of the base number 2.

$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 8 + 4 + 0 + 1$

$= 13$

Therefore, $(1101)_2 = (13)_{10}$

Octal to Decimal:

To convert octal to decimal, we multiply the digits of octal number with decreasing power of the base number 8, starting from MSB to LSB and then add them all together.

Example 2: Convert 22_8 to decimal number.

Solution: Given, 22_8

$2 \times 8^1 + 2 \times 8^0$

$= 16 + 2$

$= 18$

Therefore, $22_8 = 18_{10}$

Hexadecimal to Decimal:

Example 3: Convert 121_{16} to decimal number.

Solution: $1 \times 16^2 + 2 \times 16^1 + 1 \times 16^0$

$= 16 \times 16 + 2 \times 16 + 1 \times 1$

$= 289$

Therefore, $121_{16} = 289_{10}$

Hexadecimal to Binary Shortcut Method

To convert hexadecimal numbers to binary and vice versa is easy, you just have to memorize the table given below.

Hexadecimal Number	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011

C	1100
D	1101
E	1110
F	1111

You can easily solve the problems based on hexadecimal and binary conversions with the help of this table. Let us take an example.

Example: Convert $(89)_{16}$ into a binary number.

Solution: From the table, we can get the binary value of 8 and 9, hexadecimal base numbers.

8 = 1000 and 9 = 1001

Therefore, $(89)_{16} = (10001001)_2$

Octal to Binary Shortcut Method

To convert octal to binary number, we can simply use the table. Just like having a table for hexadecimal and its equivalent binary, in the same way, we have a table for octal and its equivalent binary number.

Octal Number	**Binary**
0	000
1	001
2	010

3	011
4	100
5	101
6	110
7	111

Example: Convert $(214)_8$ into a binary number.

Solution: From the table, we know,

2 → 010

1 → 001

4 → 100

Therefore,$(214)_8 = (010001100)_2$

Gray Code

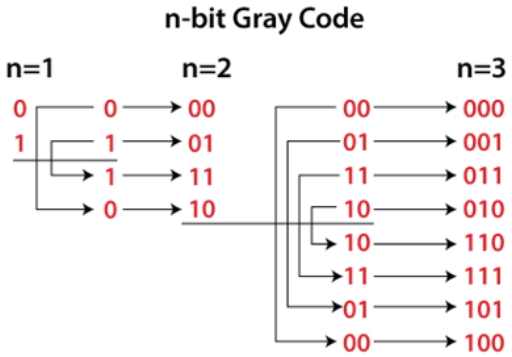
The **Gray Code** is a sequence of binary number systems, which is also known as **reflected binary code**. The reason for calling this code as reflected binary code is the first N/2 values compared with those of the last N/2 values in reverse order. In this code, two consecutive values are differed by one bit of binary digits. Gray codes are used in the general sequence of hardware-generated binary numbers. These numbers cause ambiguities or errors when the transition from one number to its successive is done. This code simply solves this problem by changing only one bit when the transition is between numbers is done.

The gray code is a very light weighted code because it doesn't depend on the value of the digit specified by the position. This code is also called a cyclic variable code as the transition of one value to its successive value carries a change of one bit only.

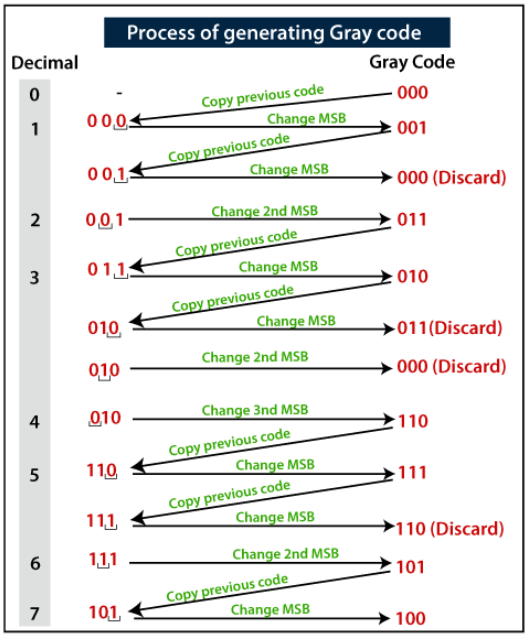
How to generate Gray code?

The prefix and reflect method are recursively used to generate the Gray code of a number. For generating gray code:

1. We find the number of bits required to represent a number.
2. Next, we find the code for 0, i.e., 0000, which is the same as binary.
3. Now, we take the previous code, i.e., 0000, and change the most significant bit of it.
4. We perform this process reclusively until all the codes are not uniquely identified.
5. If by changing the most significant bit, we find the same code obtained previously, then the second most significant bit will be changed, and so on.



Process of generating Gray Code



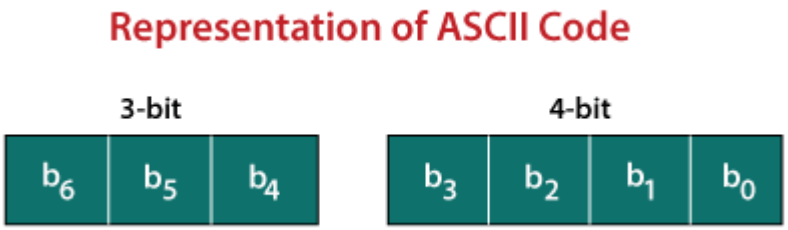
Gray Code Table

Decimal Number	Binary Number	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011

14	1110	1001
15	1111	1000

ASCII Code

The ASCII stands for American Standard Code for Information Interchange. The ASCII code is an alphanumeric code used for data communication in digital computers. The ASCII is a 7-bit code capable of representing 2^7 or 128 number of different characters. The ASCII code is made up of a three-bit group, which is followed by a four-bit code.



- The ASCII Code is a 7 or 8-bit alphanumeric code.
- This code can represent 127 unique characters.
- The ASCII code starts from 00h to 7Fh. In this, the code from 00h to 1Fh is used for control characters, and the code from 20h to 7Fh is used for graphic symbols.
- The 8-bit code holds ASCII, which supports 256 symbols where math and graphic symbols are added.
- The range of the extended ASCII is 80h to FFh.

The ASCII characters are classified into the following groups:

Control Characters

The non-printable characters used for sending commands to the PC or printer are known as control characters. We can set tabs, and line breaks functionality by this code. The control characters are based on telex technology. Nowadays, it's not so much popular in use. The character from 0 to 31 and 127 comes under control characters.

Special Characters

All printable characters that are neither numbers nor letters come under the special characters. These characters contain technical, punctuation, and mathematical characters with space also. The character from 32 to 47, 58 to 64, 91 to 96, and 123 to 126 comes under this category.

Numbers Characters

This category of ASCII code contains ten Arabic numerals from 0 to 9.

Letters Characters

In this category, two groups of letters are contained, i.e., the group of uppercase letters and the group of lowercase letters. The range from 65 to 90 and 97 to 122 comes under this category.

Binary	Hexadecimal	Decimal	ASCII Symbol	Description
0000000	0	0	NUL	The null character encourage the device to do nothing
0000001	1	1	SOH	The symbol SOH(Starts of heading) Initiates the header.
0000010	2	2	STX	The symbol STX(Start of Text) ends the header and marks the beginnir a message.
0000011	3	3	ETX	The symbol ETX(End of Text) indicates the end of the message.
0000100	4	4	EOT	The EOT(end of text) symbol marks the end of a completes transmissio
0000101	5	5	ENQ	The ENQ(Enquiry) symbol is a request that requires a response
0000110	6	6	ACK	The ACK(Acknowledge) symbol is a positive answer to the request.
0000111	7	7	BEL	The BEL(Bell) symbol triggers a beep.
0001000	8	8	BS	Lets the cursor move back one step (Backspace)
0001001	9	9	TAB (HT)	A horizontal tab that moves the cursor within a row to the next predef position (Horizontal Tab)
0001010	A	10	LF	Causes the cursor to jump to the next line (Line Feed)

0001011	B	11	VT	The vertical tab lets the cursor jump to a predefined line (Vertical Tab)
0001100	C	12	FF	Requests a page break (Form Feed)
0001101	D	13	CR	Moves the cursor back to the first position of the line (Carriage Return)
0001110	E	14	SO	Switches to a special presentation (Shift Out)
0001111	F	15	SI	Switches the display back to the normal state (Shift In)
0010000	10	16	DLE	Changes the meaning of the following characters (Data Link Escape)
0010001	11	17	DC1	Control characters assigned depending on the device used (Device Con
0010010	12	18	DC2	Control characters assigned depending on the device used (Device Con
0010011	13	19	DC3	Control characters assigned depending on the device used (Device Con
0010100	14	20	DC4	Control characters assigned depending on the device used (Device Con
0010101	15	21	NAK	The negative response to a request (Negative Acknowledge)

0010110	16	22	SYN	Synchronizes a data transfer, even if no signals are transmitted (Synchro Idle)
0010111	17	23	ETB	Marks the end of a transmission block (End of Transmission Block)
0011000	18	24	CAN	Makes it clear that transmission was faulty and the data must be disca (Cancel)
0011001	19	25	EM	Indicates the end of the storage medium (End of Medium)
0011010	1A	26	SUB	Replacement for a faulty sign (Substitute)
0011011	1B	27	ESC	Initiates an escape sequence and thus gives the following characte special meaning (Escape)
0011100	1C	28	FS	File separator.
0011101	1D	29	GS	Group separator.
0011110	1E	30	RS	Record separator.
0011111	1F	31	US	Unit separator.
0100000	20	32	SP	Blank space

0100001	21	33	!	Exclamation mark
0100010	22	34		Only quotes above
0100011	23	35	#	Pound sign
0100100	24	36	$	Dollar sign
0100101	25	37	%	Percentage sign
0100110	26	38	&	Commercial and
0100111	27	39		Apostrophe
0101000	28	40	(	Left bracket
0101001	29	41	)	Right bracket
0101010	2A	42	*	Asterisk
0101011	2B	43	+	Plus symbol

0101100	2C	44	,	Comma
0101101	2D	45	-	Dash
0101110	2E	46	.	Full stop
0101111	2F	47	/	Forward slash
0110000	30	48	0	
0110001	31	49	1	
0110010	32	50	2	
0110011	33	51	3	
0110100	34	52	4	
0110101	35	53	5	
0110110	36	54	6	
0110111	37	55	7	
0111000	38	56	8	
0111001	39	57	9	

0111010	3A	58	:	Colon
0111011	3B	59	;	Semicolon
0111100	3C	60	<	Small than bracket
0111101	3D	61	=	Equals sign
0111110	3E	62	>	Bigger than symbol
0111111	3F	63	?	Question mark
1000000	40	64	@	At symbol
1000001	41	65	A	
1000010	42	66	B	
1000011	43	67	C	
1000100	44	68	D	

1000101	45	69	E	
1000110	46	70	F	
1000111	47	71	G	
1001000	48	72	H	
1001001	49	73	I	
1001010	4A	74	J	
1001011	4B	75	K	
1001100	4C	76	L	
1001101	4D	77	M	
1001110	4E	78	N	
1001111	4F	79	O	

1010000	50	80	P	
1010001	51	81	Q	
1010010	52	82	R	
1010011	53	83	S	
1010100	54	84	T	
1010101	55	85	U	
1010110	56	86	V	
1010111	57	87	W	
1011000	58	88	X	
1011001	59	89	Y	
1011010	5A	90	Z	

1011011	5B	91	[	Left square bracket
1011100	5C	92	\	Inverse/backward slash
1011101	5D	93	]	Right square bracket
1011110	5E	94	^	Circumflex
1011111	5F	95	_	Underscore
1100000	60	96	`	Gravis (backtick)
1100001	61	97	A	
1100010	62	98	B	
1100011	63	99	C	
1100100	64	100	D	
1100101	65	101	E	

1100110	66	102	F	
1100111	67	103	G	
1101000	68	104	H	
1101001	69	105	I	
1101010	6A	106	J	
1101011	6B	107	K	
1101100	6C	108	L	
1101101	6D	109	M	
1101110	6E	110	N	
1101111	6F	111	O	
1110000	70	112	P	

1110001	71	113	Q	
1110010	72	114	R	
1110011	73	115	S	
1110100	74	116	T	
1110101	75	117	U	
1110110	76	118	v	
1110111	77	119	w	
1111000	78	120	x	
1111001	79	121	y	
1111010	7A	122	z	
1111011	7B	123	{	Left curly bracket

1111100	7C	124	|	Vertical line
1111101	7D	125	}	Right curly brackets
1111110	7E	126	~	Tilde
1111111	7F	127	DEL	The DEL(Delete) symbol deletes a character. This is a control character consists of the same number in all positions.

ASCII Table

The values are typically represented in ASCII code tables in decimal, binary, and hexadecimal form.

Example 1: (1001010110000111101101100001101010011100001101111101001 1101110111010010000000110001011001001100 11)$_2$

Step 1: In the first step, we we make the groups of 7-bits because the ASCII code is 7 bit.

1001010 1100001 1110110 1100001 1010100 1110000 1101111 1101001 1101110 1110100 1000000 0110001 0110010 0110011

Step 2: Then, we find the equivalent decimal number of the binary digits either from the ASCII table or **64 32 16 8 4 2 1** scheme.

Binary	Decimal
64 32 16 8 4 2 1 1 0 0 1 0 1 0	64+8+2=74

64 32 16 8 4 2 1 1 1 0 0001	64+32+1=94
64 32 16 8 4 2 1 1 1 1 0110	64+32+16+4+2=118
64 32 16 8 4 2 1 1 1 0 0001	64+32+1=97
64 32 16 8 4 2 1 1 0 1 0100	64+16+4=84
64 32 16 8 4 2 1 1 1 1 0000	64+32+16=112
64 32 16 8 4 2 1 1 1 0 1111	64+32+8+4+2+1=111
64 32 16 8 4 2 1 1 1 0 1001	64+32+8+1=105
64 32 16 8 4 2 1 1 1 0 1110	64+32+8+4+2=110
64 32 16 8 4 2 1 1 1 1 0100	64+32+16+4=116
64 32 16 8 4 2 1 1 0 0 0000	64
64 32 16 8 4 2 1 0 1 1 0001	32+16+1=49

64 32 16 8 4 2 1 0 1 1 0 0 1 0	32+16+2=50
64 32 16 8 4 2 1 0 1 1 0 0 1 1	32+16+2+1=51

Step 3: Last, we find the equivalent symbol of the decimal number from the ASCII table.

Decimal	Symbol
74	J
94	a
118	v
97	a
84	T
112	p
111	o
105	i
110	n
116	t
64	@

49	1
50	2
51	3

Introduction of Floating Point Representation

1. To convert the floating point into decimal, we have 3 elements in a 32-bit floating point representation:

i) Sign
ii) Exponent
iii) Mantissa

- **Sign** bit is the first bit of the binary representation. '1' implies negative number and '0' implies positive number.
  Example: 11000001110100000000000000000000 This is negative number.
- **Exponent** is decided by the next 8 bits of binary representation. 127 is the unique number for 32 bit floating point representation. It is known as bias. It is determined by $2^{k-1} -1$ where 'k' is the number of bits in exponent field. There are 3 exponent bits in 8-bit representation and 8 exponent bits in 32-bit representation.

  Thus

  bias = 3 for 8 bit conversion ($2^{3-1} -1 = 4-1 = 3$)
  bias = 127 for 32 bit conversion. ($2^{8-1} -1 = 128-1 = 127$)
  Example: 01000001110100000000000000000000
  $10000011 = (131)_{10}$
  131-127 = 4
  Hence the exponent of 2 will be 4 i.e. $2^4 = 16$.
- **Mantissa** is calculated from the remaining 23 bits of the binary representation. It consists of '1' and a fractional part which is determined by:
  Example:
  01000001110100000000000000000000

  The fractional part of mantissa is given by:

  1*(1/2) + 0*(1/4) + 1*(1/8) + 0*(1/16) +……. = 0.625

  Thus the mantissa will be 1 + 0.625 = 1.625

The decimal number hence given as: Sign*Exponent*Mantissa = $(-1)^0$*(16)*(1.625) = 26

2. To convert the decimal into floating point, we have 3 elements in a 32-bit floating point representation:

i) Sign (MSB)
ii) Exponent (8 bits after MSB)
iii) Mantissa (Remaining 23 bits)

- **Sign bit** is the first bit of the binary representation. '1' implies negative number and '0' implies positive number.
  Example: To convert -17 into 32-bit floating point representation Sign bit = 1
- **Exponent** is decided by the nearest smaller or equal to 2^n number. For 17, 16 is the nearest 2^n. Hence the exponent of 2 will be 4 since 2^4 = 16. 127 is the unique number for 32 bit floating point representation. It is known as bias. It is determined by 2^{k-1} -1 where 'k' is the number of bits in exponent field.
  Thus bias = 127 for 32 bit. (2^{8-1} -1 = 128-1 = 127)
  Now, 127 + 4 = 131 i.e. 10000011 in binary representation.
- **Mantissa:** 17 in binary = 10001.
  Move the binary point so that there is only one bit from the left. Adjust the exponent of 2 so that the value does not change. This is normalizing the number. 1.0001 x 2^4. Now, consider the fractional part and represented as 23 bits by adding zeros.
  00010000000000000000000

Advantages:

Wide range of values: Floating factor illustration lets in for a extensive variety of values to be represented, along with very massive and really small numbers.
Precision: Floating factor illustration offers excessive precision, that is important for medical and engineering calculations.
Compatibility: Floating point illustration is extensively used in computer structures, making it well matched with a extensive variety of software and hardware.
Easy to use: Most programming languages offer integrated guide for floating factor illustration, making it smooth to use and control in laptop programs.

Disadvantages:

Complexity: Floating factor illustration is complex and can be tough to understand, mainly for folks that aren't acquainted with the underlying mathematics.
Rounding errors: Floating factor illustration can result in rounding mistakes, where the real price of a number of is barely extraordinary from its illustration inside the computer.
Speed: Floating factor operations can be slower than integer operations, particularly on older or much less powerful hardware.
Limited precision: Despite its excessive precision, floating factor representation has a restrained number of sizeable digits, which could restrict its usefulness in some programs.