

Chapter 2 :



Computer Science

**Class XI (As per
CBSE Board)**

**Information
Representation**

**New
Syllabus
2019-20**

Visit : python.mykvs.in for regular updates

Introduction

In general term computer represent information in different types of data forms i.e. number , character ,picture ,audio , video etc.

Computers are made of a series of switches/ gates. Each switch has two states: ON(1) or OFF(0).That's why computer works on the basis of binary number system(0/1).But for different purpose different number systems are used in computer world to represent information. E.g. Octal, Decimal, Hexadecimal.

NUMBER SYSTEM		
SYSTEM	BASE	DIGIT
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Decimal Number System

Characteristics

- Ten symbols
- 0 1 2 3 4 5 6 7 8 9

Positional

- $2945 \neq 2495$
- $2945 = (2 * 10^3) + (9 * 10^2) + (4 * 10^1) + (5 * 10^0)$

(Most) people use the decimal number system Why?

THIS A POSITIONAL NUMBER SYSTEM .and that's of great advantage
..simple shifting the position of decimal.It become complex either case
to use number system <10 or >10 .

Binary Number System

Characteristics

- Two symbols
- 0 1

Positional

- Positional
- $1010_2 \neq 1100_2$

Most (digital) computers use the binary number system Why?

Computers are made of a series of switches/ gates. Each switch has two states: ON(1) or OFF(0). That's why computer works on the basis of binary number system(0/1).

Decimal-Binary Equivalence

Decimal	Binary	Decimal	Binary
0	0	16	10000
1	1	17	10001
2	10	18	10010
3	11	19	10011
4	100	20	10100
5	101	21	10101
6	110	22	10110
7	111	23	10111
8	1000	24	11000
9	1001	25	11001
10	1010	26	11010
11	1011	27	11011
12	1100	28	11100
13	1101	29	11101
14	1110	30	11110
15	1111	31	11111

Binary – Decimal Conversion

Using positional notation

$$100101_2 = (1*2^5) + (0*2^4) + (0*2^3) + (1*2^2) + (0*2^1) + (1*2^0)$$

$$= 32 + 0 + 0 + 4 + 0 + 1$$

$$= 37$$

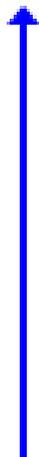
Decimal-Binary Conversion

Using the Division Method:

Divide decimal number by 2 until you reach zero, and then collect the remainders in reverse.

$$22_{10} = 10110_2$$

2) <u>22</u>	<u>Rem:</u>
2) <u>11</u>	0
2) <u>5</u>	1
2) <u>2</u>	1
2) <u>1</u>	0
0	1



Hexadecimal Number System

Characteristics

- Sixteen symbols
- 0 1 2 3 4 5 6 7 8 9 A B C D E F

Positional

$A13D_{16} \neq 3DA1_{16}$

Computer programmers often use the hexadecimal number system, Why?

Computers only work on the binary number system. The hexadecimal number system is commonly used to describe locations in computer memory. They are also used in assembly language instructions.

Decimal-Hexadecimal Equivalence

<u>Decimal</u>	<u>Hex</u>
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

<u>Decimal</u>	<u>Hex</u>
16	10
17	11
18	12
19	13
20	14
21	15
22	16
23	17
24	18
25	19
26	1A
27	1B
28	1C
29	1D
30	1E
31	1F

<u>Decimal</u>	<u>Hex</u>
32	20
33	21
34	22
35	23
36	24
37	25
38	26
39	27
40	28
41	29
42	2A
43	2B
44	2C
45	2D
46	2E
47	2F

Hexadecimal to decimal

$$\begin{aligned}25_{16} &= (2 * 16^1) + (5 * 16^0) \\ &= 32 + 5 \\ &= 37\end{aligned}$$

Decimal to hexadecimal

$$37 / 16 = 2 \text{ R } 5$$

$$2 / 16 = 0 \text{ R } 2$$

↑
Read from bottom
to top: 25_{16}

Binary - hexadecimal

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

Binary to hexadecimal

Convert 110100110_2 to hex

Starting at the right end, split into groups of 4:

1 1010 0110 \rightarrow

0110 = 6

1010 = A

0001 = 1 (pad empty digits with 0)

$110100110_2 = 1A6_{16}$

Hexadecimal to Binary

Convert $3D9_{16}$ to binary

Convert each hex digit to 4 bits:

$$3 = 0011$$

$$D = 1101$$

$$9 = 1001$$

$$0011 \quad 1101 \quad 1001 \quad \square$$

$$3D9_{16} = 111101100_{12} \quad (\text{can remove}$$

leading zeros)

Octal Number System

Characteristics

- Eight symbols
- 0 1 2 3 4 5 6 7

Positional

- $1743_8 \neq 7314_8$

Computer programmers often use the octal number system, Why?

Octal and hex use the human advantage that they can work with lots of symbols while it is still easily convertible back and forth between binary.

Decimal-Octal Equivalence

<u>Decimal</u>	<u>Octal</u>
----------------	--------------

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17

<u>Decimal</u>	<u>Octal</u>
----------------	--------------

16	20
17	21
18	22
19	23
20	24
21	25
22	26
23	27
24	30
25	31
26	32
27	33
28	34
29	35
30	36
31	37

<u>Decimal</u>	<u>Octal</u>
----------------	--------------

32	40
33	41
34	42
35	43
36	44
37	45
38	46
39	47
40	50
41	51
42	52
43	53
44	54
45	55
46	56
47	57

Octal to decimal

357_8

positional powers of 8:	8^2	8^1	8^0
decimal positional value:	64	8	1
Octal number:	3	5	7

$$(3 \times 64) + (5 \times 8) + (7 \times 1)$$

$$= 192 + 40 + 7 = 239_{10}$$

Decimal to octal

Using the Division Method:

Example 1: $214_{10} = 326_8$

8)	214	Rem:	
8)	26	6	
8)	3	2	
		0	3	

Binary-Octal Conversion

E.g.

001010000100111101₂
1 2 0 4 7 5₈

Octal to binary

1 2 0 4 7 5₈
001010000100111101₂

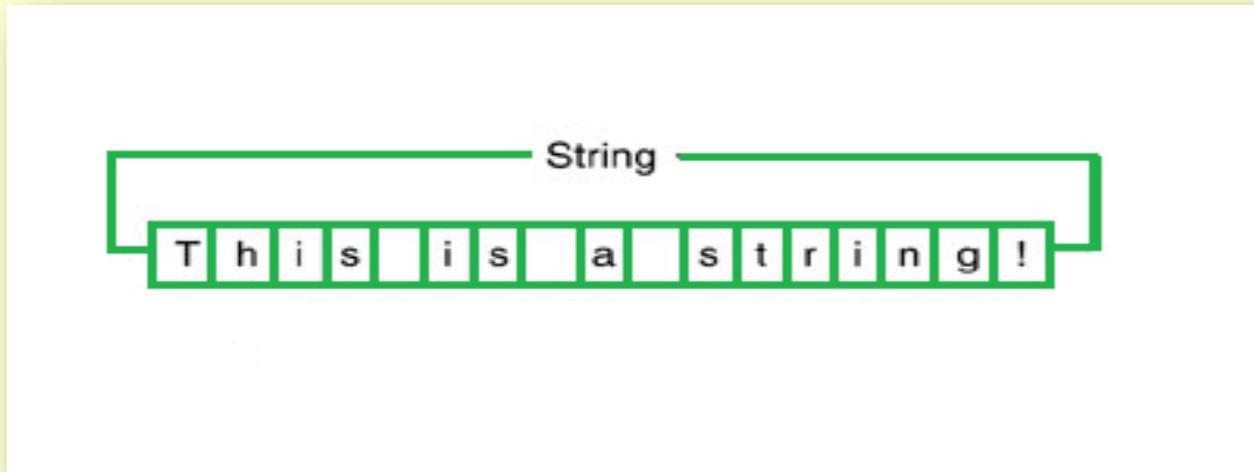
Adding Unsigned Integers

$$\begin{array}{r} 3 \\ + 10 \\ \hline 13 \end{array} \qquad \begin{array}{r} 1 \\ 0011_2 \\ + 1010_2 \\ \hline 1101_2 \end{array}$$

Start at right column
Proceed leftward Carry 1
when necessary

String representation

String is any finite sequence of characters. Any string includes letters, numerals, symbols and punctuation marks.



Computers are designed to work internally with numbers. In order to handle characters, we need to choose a number for each character. There are many ways to do this

String representation

Following are some form of character set

- **ASCII**
- **UNICODE**
- **ISCII**

String representation

ASCII

It is most common coding system (Pronounced *ass-key*).

ASCII = American National Standard Code for Information Interchange

It is Defined in ANSI document X3.4-1977. It is a 7-bit code. Its 8th bit is unused (or used for a parity bit)

$$2^7 = 128 \text{ codes}$$

Two general types of codes:

95 are “Graphic” codes (displayable on a console)

33 are “Control” codes (control features of the console or communications channel)

String representation

ASCII

	000	001	010	011	100	101	110	111
0000	NULL	DLE		0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EDT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

String representation

MOST SIGNIFICANT BIT

ASCII CHART

	000	001	010	011	100	101	110	111
0000	NULL	DLE		0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EDT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

LEAST SIGNIFICANT BIT

Visit : python.mykvs.in for regular updates

String representation

ASCII

“Hello, world” Example

	Binary		Hexadecimal		Decimal
H	= 01001000	=	48	=	72
e	= 01100101	=	65	=	101
l	= 01101100	=	6C	=	108
l	= 01101100	=	6C	=	108
o	= 01101111	=	6F	=	111
,	= 00101100	=	2C	=	44
	= 00100000	=	20	=	32
w	= 01110111	=	77	=	119
o	= 01100111	=	67	=	103
r	= 01110010	=	72	=	114
l	= 01101100	=	6C	=	108
d	= 01100100	=	64	=	100

LEAST SIGNIFICANT BIT

String representation

UNICODE

It is a worldwide character-encoding standard .Its main objective is to enable a single, unique character set that is capable of supporting all characters from all scripts, as well as symbols, that are commonly utilized for computer processing throughout the world.

String representation

VARIOUS UNICODER ENCODING

Name	UTF-8	UTF-16	UTF-32
Smallest code point	0000	0000	0000
Largest code point	10FFFF	10FFFF	10FFFF
Code unit size	8 bits	16 bits	32 bits
Byte order	N/A	<BOM>	<BOM>
Fewest bytes per character	1	2	4
Most bytes per character	4	4	4

LEAST SIGNIFICANT BIT

Visit : python.mykvs.in for regular updates

String representation

UTF-8

It is most popular type of Unicode encoding. It uses one byte for standard English letters and symbols, two bytes for additional Latin and Middle Eastern characters, and three bytes for Asian characters

Any additional characters can be represented using four bytes

It is backwards compatible with ASCII, since the first 128 characters are mapped to the same values.

LEAST SIGNIFICANT BIT

Visit : python.mykvs.in for regular updates

String representation

UTF-8 REPRESENTATION

Representation	Representation format	Example
1 Octet Representation	<p>Control Bit(0) Data Bit</p>	<p>Letter 'A'[U+41]</p> <p>Control Bit Binary code for 'A'</p>
2 Octet Representation	<p>Control Bit(110) Data Bit Control Bit(10) Data Bit</p>	<p>Letter 'A'[U+41]</p> <p>Control Bit(110) Data Bit Control Bit(10) Data Bit</p>
3 Octet Representation	<p>1110xxxx 10xxxxxx 10xxxxxx</p> <p>Control Bit(1110) Data Bit Control Bit(10) Data Bit Control Bit(0) Data Bit</p>	<p>Letter 'A'[U+41]</p> <p>Bold (as control bit) others are data bit</p>
4 Octet Representation	<p>11110xxx 10xxxxxx 10xxxxxx 10xxxxxx</p>	<p>Letter 'A'[U+41]</p>

LEAST SIGNIFICANT BIT

Visit : python.mykvs.in for regular updates

String representation

UTF-32

It is a multi-byte encoding that represents each character with 4 bytes

* Makes it space inefficient

Its main use is in internal APIs where the data is single code points or glyphs, rather than strings of characters

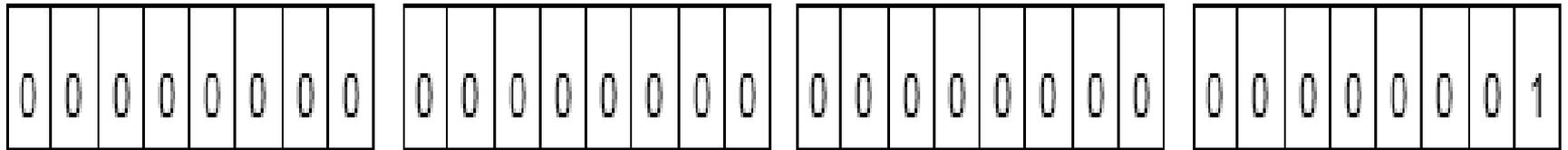
Used on Unix systems sometimes for storage of information

String representation

UTF-32

It is a fixed length encoding scheme that uses exactly 4 bytes to represent all Unicode code points. E.g.

Letter 'A' [U+41]



LEAST SIGNIFICANT BIT

Visit : python.mykvs.in for regular updates

String representation

ISCII

ISCII stands for Indian Script Code for Information Interchange for Indian languages. It is an 8-bits code to represent Indian scripts.

The Department of Electronics (DOE) has established standard and standard are in action from 1983.

These codes are used for 10 Indian scripts- Devanagiri, Punjabi, Gujrati, Udia, Bengali, Asami, Telgu, Kannad, Malayalam and Tamil. C-DAC (established in August-September, 1988) developed standard for font coding in 1990 is called ISFOC (Indian Standards for Font Coding).

LEAST SIGNIFICANT BIT