Table of Contents

Why Numbers for Computing?

Computers are classified according to **functionality, physical size** and **purpose**. Functionality wise computers could be **analog, digital or hybrid**. Digital computers process data that is in discrete form whereas analog computers process data that is continuous in nature. Hybrid computers on the other hand can process data that is both discrete and continuous.

In digital computers, the user input is first converted and transmitted as electrical pulses that can be represented by two unique states ON and OFF. The ON state may be represented by a "1" and the off state by a "0". The sequence of ON'S and OFF'S forms the electrical signals that the computer can understand.

Why Number Systems and Logic Gates?

Data and instructions cannot be entered and processed directly into computers using human language. Any type of data be it numbers, letters, special symbols, sound or pictures must first be converted into machine-readable form i.e. binary form. Due to this reason, it is important to understand how a computer together with its peripheral devices handles data in its electronic circuits, on magnetic media and in optical devices.

Why Different Number Systems other than Binary?

- Computers not only process numbers, letters and special symbols but also complex types of data such as sound and pictures. However, these complex types of data take a lot of memory and processor time when coded in binary form.
- This limitation necessitates the need to develop better ways of handling long streams of binary digits.
- Higher number systems are used in computing to reduce these streams of binary digits into manageable form. This helps to improve the processing speed and optimize memory usage.

A number system is a set of symbols used to represent values derived from a common base or radix.

Number Systems

Properties	Decimal	Binary	Octal	Hexadecimal
Base	10	2	8	16
Values	0 ÷ 9	0 or 1	0 - 7	0 - 9, A, B, C, D, E, F
Max Per Place	9	1	7	F
Eg:	25 or 25_{10}	1011_2	26_8	$2AF_{16}$

Decimal to Binary Conversion

Even = - 0
Odd = - 1

$2 \,|\, 25 - 1$
$2 \,|\, 12 - 0$
$2 \,|\, 6 - 0$
$2 \,|\, 3 - 1$
1

11001_2 //

Binary to Decimal Conversion

11001_2

1	1	0	0	1
		2^2	2^1	2^0
16	8	4	2	1

$$16 + 8 + 1 = 25$$

25 //

Your Turn

127 → Binary	101111_2 → Decimal
2 | 127 - 1 2 | 63 - 1 2 | 31 - 1 2 | 15 - 1 2 | 7 - 1 2 | 3 - 1 1 1111111_2	1 0 1 1 1 1 1 ↓ ↓ ↓ ↓ ↓ ↓ ↓ 64 32 16 8 4 2 1 64 8 4 + 2 1 16 95 95_{4}
592 → Binary	11111111_2 → Decimal
2 | 592 - 0 2 | 292 - 0 2 | 148 - 0 2 | 74 - 0 2 | 37 - 1 2 | 18 - 0 2 | 9 - 1 2 | 4 - 0 2 | 2 - 0 1 1001010000_2	1 1 1 1 1 1 1 1 ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ 128 64 32 16 8 4 2 1 128 64 32 16 8 + 4 2 1 255 255_{4}

Octal to Binary Conversion

205_8

010 000 101 10000101_2

4 2 1
5 ← 1 0 1
0 ← 0 0 0
2 ← 0 1 0

Binary to Octal Conversion

1001110001_2

001 001 110 001
1 1 6 1

1161_8

4 2 1
1 ← 0 0 1
6 ← 1 1 0
1 ← 0 0 1
1 ← 0 0 1

Your Turn

40256_8 → Binary	1000110001001111_2 → Octal
4 2 1	4 3 0 4 7
1 1 0	43047_8
1 0 1	
0 1 0	
0 0 0	
1 0 0	
100000010101110_2	
70524106_8 → Binary	01 000001 000001111001 → Octal $_2$
4 2 1	2 0 2 0 3 7 1
1 1 0	2020371_8
0 0 0	
0 0 1	
1 0 0	
1 1 0	
1 0 1	
0 0 0	
1 1 1	
$111000101010100001000110_2$	

Hexadecimal to Binary Conversion

Ex $C401D_{16}$

	8	4	2	1
D	1	1	0	1
1	0	0	0	1
0	0	0	0	0
4	0	1	0	0
C	1	1	0	0

$C \rightarrow 1100$, $4 \rightarrow 0100$, $0 \rightarrow 0000$, $1 \rightarrow 0001$, $D_{16} \rightarrow 1101$

1100 0100 0000 0001 1101

11000100000000011101_2

Binary to Hexadecimal Conversion

Ex $\underbrace{0011}_{3}\ \underbrace{1100}_{C}\ \underbrace{1110}_{E}\ \underbrace{0001}_{1}\ \underbrace{0101}_{5}\ \underbrace{1011}_{B}{}_2$

3 C E 1 5 B

$3CE15B_{16}$

Your Turn

Hexadecimal to Binary

$E015F_{16}$

8	4	2	1
1	1	1	1
0	1	0	1
0	0	0	1
0	0	0	0
1	1	1	0

11100000000101011111_2

Hexadecimal to Binary

$B0C6_{16}$

8	4	2	1
0	1	1	0
1	1	0	0
0	0	0	0
1	0	1	1

1011000011000110_2

Binary → Hexadecimal

$\underbrace{0001}_{1}\ \underbrace{0001}_{1}\ \underbrace{1100}_{C}\ \underbrace{1001}_{9}\ \underbrace{1011}_{B}{}_2$

1 1 C 9 B

$11C9B_{16}$

Binary → Hexadecimal

$\underbrace{0100}_{4}\ \underbrace{0111}_{7}\ \underbrace{1001}_{9}{}_2$

4 7 9_{16}

Conversions Vice Versa

Decimal to Hexadecimal

Ex 25

2 | 25 - 1
2 | 12 - 0
2 | 6 - 0
2 | 3 - 1
1

0001 1001

↓ ↓

1 9

19_{16}

Hexadecimal → Decimal

Ex $2A_{16}$

0010 1010

128 64 32 16 8 4 2 1

32
8
2

42

42_{10}

Hexadecimal → Octal

Ex $D0B_{16}$

110100001011_2

↓ ↓ ↓ ↓

6 4 1 3

6413_8

Octal → Hexadecimal

Ex 207_8

010000111_2

↓ ↓

8 7

87_{16}

1023 → Octal

2 | 1023 - 1
2 | 511 - 1
2 | 255 - 1
2 | 127 - 1
2 | 63 - 1
2 | 31 - 1
2 | 15 - 1
2 | 7 - 1
2 | 3 - 1
1

001111111111

↓ ↓ ↓ ↓

1 7 7 7

1777_8

ABC_{16} → Octal

↓ ↓ ↓

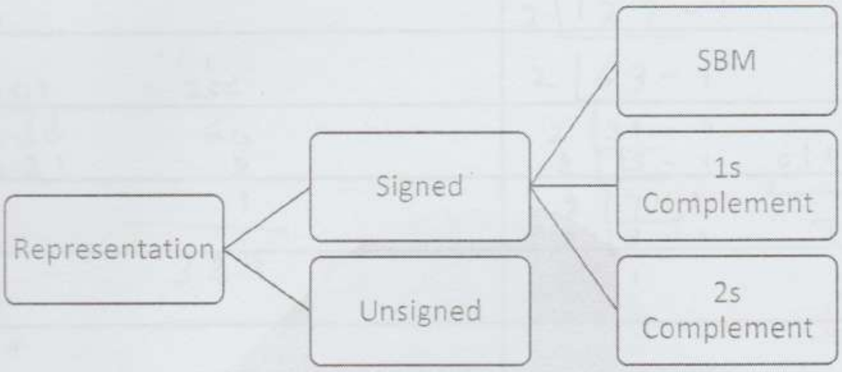
101010111100

↓ ↓ ↓ ↓

5 2 7 4

5274_8

Representation of Decimal Numbers



Unsigned Representation

Is used to represent positive numbers only.

Signed Representation

Is used to represent both **positive** and **negative** numbers. There are three approaches.

Sign Bit Magnitude Representation

Ex +27

2 | 27 - 1
2 | 13 - 1
2 | 6 - 0
2 | 3 - 1
1

Add 3 zeros to make 8 bits

11011_2

00011011_2

+

−27

10011011_2

−

• Two representation for zero.

One's Complement (Invert All)

Ex +27

2 | 27 - 1
2 | 13 - 1
2 | 6 - 0
2 | 3 - 1
1

00011011_2

−27

00011011_2

11100100_2

Invert all bits

−

• Two representations for zero.

Two's Complement (Invert All and Add 1)

Ex $+27$

$$\begin{array}{r|l} 2 & 27 - 1 \\ 2 & 13 - 1 \\ 2 & 6 - 0 \\ 2 & 3 - 1 \\ & 1 \end{array}$$

00011011_2

-27

00011011_2

Invert all the bits and add 1

11100100_2

$+1$

11100101_2

$-$

Your Turn

How Characters are Represented in Computers?

Computers and digital circuits processes information in the binary format. Each character is assigned 7 or 8-bit binary code to indicate its character which may be numeric, alphabet or special symbol. Example - Binary number 1000001 represents 65(decimal) in straight binary code, alphabet A in ASCII code and 41(decimal) in BCD code. Following are character representation methods:

BCD (Binary-Coded Decimal) code

- Four-bit code that represents one of the ten decimal digits from 0 to 9.
- Example - $(37)_{10}$ is represented as 0011 0111 using BCD code, rather than $(100101)_2$ in straight binary code.
- Thus, BCD code requires more bits than straight binary code.
- Still it is suitable for input and output operations in digital systems.

Note: 1010, 1011, 1100, 1101, 1110, and 1111 are INVALID CODE in BCD code.

ASCII (American Standard Code Information Interchange) code

- It is 7-bit or 8-bit alphanumeric code.
- 7-bit code is standard ASCII supports 127 characters.
- Standard ASCII series starts from 00_{16} to $7F_{16}$, where 00_{16}-$1F_{16}$ are used as control characters and 20_{16}-$7E_{16}$ as graphics symbols.
- 8-bit code is extended ASCII supports 256 symbols where special graphics and math's symbols are added.
- Extended ASCII series starts from 80_{16} to FF_{16}.

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	Escape	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	File Separator	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	Group Separator	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	Record Separator	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	Unit Separator	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	Del

EBCDIC (Extended Binary Coded Decimal Interchange Code) code

- 8-bit alphanumeric code developed by IBM, supports 256 symbols.
- It was mainly used in IBM mainframe computers.
- It is an 8 bit code and therefore can accommodate 256 characters.

Unicode

- The pre-Unicode world was populated with hundreds of different encoding schemes that assigned a number to each letter or character.
- Many such schemes included code pages that contained only 256 characters - each character requiring 8 bits of storage.
- it was insufficient to hold ideographic character sets containing thousands of characters such as Chinese and Japanese, and also did not allow the character sets of many languages to co-exist with each other.
- Unicode is an attempt to include all the different schemes into one universal text-encoding standard.
- It is platform, program, and language independent.

Basic Arithmetic & Logic Operations on Binary Numbers