

Competency 03

Investigates how instructions and data are represented in computers and exploit them in arithmetic and logic operations

2022.04.21

Computers and Programs

What is a computer?

- It is a very fast electronic machine.
- It is not intelligent. It works according to the instruction given by user.
- Then how do you give instructions to computer? By using a program.
- The computer can follow our instructions to accept the inputs that we feed, process them and to produce the output.

What is a program?

- A program is a set of instructions.
- An instruction tells the computer what to do and how to do. Then only, a computer is very useful.

What can a computer understand?

- Computer cannot understand 0,1, bits, signals.
- Computer is an electronic device. It can only understand presence of electricity and absence of electricity.
- For our convenience we use 1s and 0s. Any two symbols we can use to represent presence and absence of electricity.
Ex: ON and OFF, TRUE and FALS, HIGH and LOW, 1 and 0

ON -  = | 2 - 5 V

OFF-  =  0 - 0.8V

If the Voltage in a particular place is 0 to 0.8V, it symbolizes as 0

if the Voltage in a particular place is 2 to 5V, it symbolizes as 1

0.8v – 2.0v – uncertain area

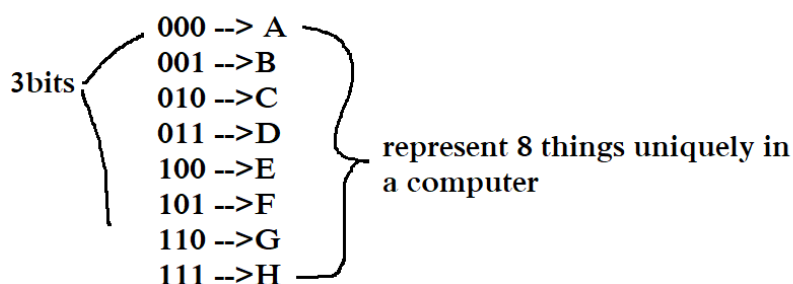
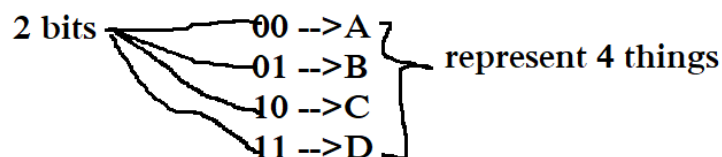
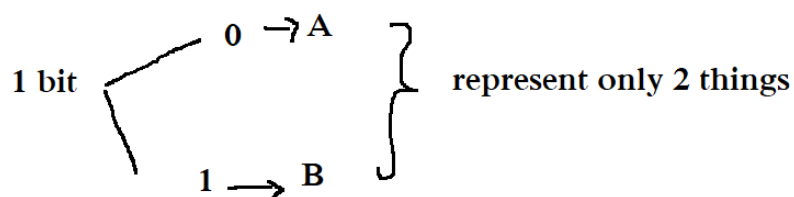
** Doesn't mean computer can understand 1s and 0s.

Bits and Byte

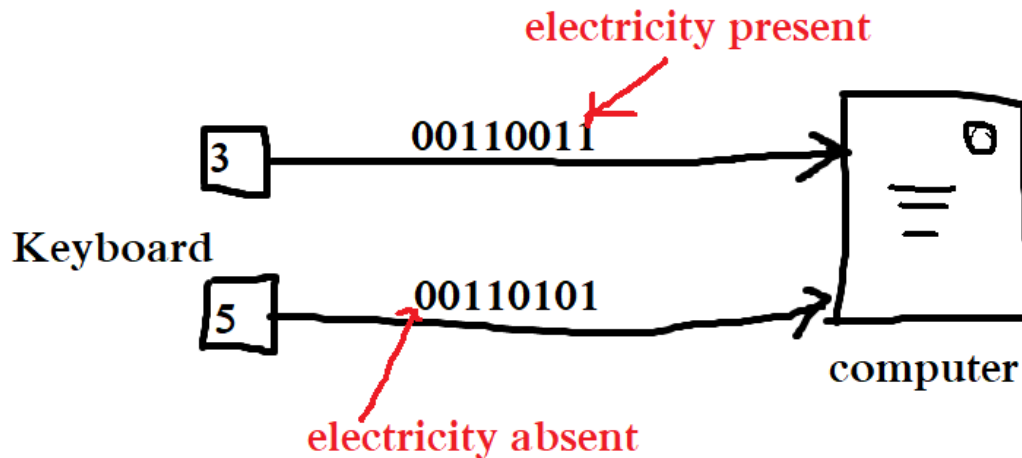
- What are Bits? Binary Digits ex:0,1
- What is a Byte? Group of 8 bits

Why are bits combined into a Byte?

Why do we combined bits together and form a byte?



2022.04.22



- We have lots of keys on our keyboard. We need more combinations. So, we need to combined 8 bits together and form a byte.

How many different symbols can be represented with a byte?

$2^8 = 256$ symbols represent uniquely with a byte

What is a character set?

- There are lots of ways to represent number 3.
 $3 \rightarrow 0011\ 0011$
 $3 \rightarrow 1100\ 1100$
 $3 \rightarrow 1010\ 1010$
- But we need to represent the number 3 in standard way.
- For that introduced **character set**.
- ANCI proposed standard character set. They proposed and agreed 3 means 00110011. This character set we called ASCII.

How characters are represented in computers?

Methods of character representation

- o BCD
- o EBCDIC
- o ASCII
- o Unicode

BCD (Binary Coded Decimal) code – This is a 4-bit code used for coding numeric values (0-9) only.

$2^4 = 16$ the remaining 6 (i.e., 1010, 1011, 1100, 1101, 1110, 1111) are invalid combinations.

E.g.: - $1000111_2 = 0100\ 0111_{(BCD)} = 47_{10}$

$357_{10} = 0011\ 0101\ 0111_{(BCD)}$

BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Decimal	0	1	2	3	4	5	6	7	8	9

EBCDIC (Extended Binary Coded Decimal Interchange Code) –

The 8-bit EBCDIC is used basically by large IBM mainframe computers and compatible equipment. It uses 256 different characters

ASCII (American Standard Codes for Information Interchange)

normally uses 8 bits (1 byte) to store each character. However, the 8th bit is used as a check digit, meaning that only 7 bits are available to store each character. This gives ASCII the ability to store a total of $2^7 = 128$ different values. The 7-bit ASCII code was originally proposed by the American National Standard Institute (ANSI). (IBM personal computers use ASCII).

Problem with this is, there are lots of characters in international language alphabets.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	 Space	64	40	100	@ @	96	60	140	` `		
1	1	001	SOH	(start of heading)	33	21	041	! !	65	41	101	A A	97	61	141	a a		
2	2	002	STX	(start of text)	34	22	042	" "	66	42	102	B B	98	62	142	b b		
3	3	003	ETX	(end of text)	35	23	043	# #	67	43	103	C C	99	63	143	c c		
4	4	004	EOT	(end of transmission)	36	24	044	$ \$	68	44	104	D D	100	64	144	d d		
5	5	005	ENQ	(enquiry)	37	25	045	% %	69	45	105	E E	101	65	145	e e		
6	6	006	ACK	(acknowledge)	38	26	046	& &	70	46	106	F F	102	66	146	f f		
7	7	007	BEL	(bell)	39	27	047	' '	71	47	107	G G	103	67	147	g g		
8	8	010	BS	(backspace)	40	28	050	((72	48	110	H H	104	68	150	h h		
9	9	011	TAB	(horizontal tab)	41	29	051))	73	49	111	I I	105	69	151	i i		
10	A	012	LF	(NL line feed, new line)	42	2A	052	* *	74	4A	112	J J	106	6A	152	j j		
11	B	013	VT	(vertical tab)	43	2B	053	+ +	75	4B	113	K K	107	6B	153	k k		
12	C	014	FF	(NP form feed, new page)	44	2C	054	, ,	76	4C	114	L L	108	6C	154	l l		
13	D	015	CR	(carriage return)	45	2D	055	- -	77	4D	115	M M	109	6D	155	m m		
14	E	016	SO	(shift out)	46	2E	056	. .	78	4E	116	N N	110	6E	156	n n		
15	F	017	SI	(shift in)	47	2F	057	/ /	79	4F	117	O O	111	6F	157	o o		
16	10	020	DLE	(data link escape)	48	30	060	0 0	80	50	120	P P	112	70	160	p p		
17	11	021	DC1	(device control 1)	49	31	061	1 1	81	51	121	Q Q	113	71	161	q q		
18	12	022	DC2	(device control 2)	50	32	062	2 2	82	52	122	R R	114	72	162	r r		
19	13	023	DC3	(device control 3)	51	33	063	3 3	83	53	123	S S	115	73	163	s s		
20	14	024	DC4	(device control 4)	52	34	064	4 4	84	54	124	T T	116	74	164	t t		
21	15	025	NAK	(negative acknowledge)	53	35	065	5 5	85	55	125	U U	117	75	165	u u		
22	16	026	SYN	(synchronous idle)	54	36	066	6 6	86	56	126	V V	118	76	166	v v		
23	17	027	ETB	(end of trans. block)	55	37	067	7 7	87	57	127	W W	119	77	167	w w		
24	18	030	CAN	(cancel)	56	38	070	8 8	88	58	130	X X	120	78	170	x x		
25	19	031	EM	(end of medium)	57	39	071	9 9	89	59	131	Y Y	121	79	171	y y		
26	1A	032	SUB	(substitute)	58	3A	072	: :	90	5A	132	Z Z	122	7A	172	z z		
27	1B	033	ESC	(escape)	59	3B	073	; ;	91	5B	133	[[123	7B	173	{ {		
28	1C	034	FS	(file separator)	60	3C	074	< <	92	5C	134	\ \	124	7C	174	|		
29	1D	035	GS	(group separator)	61	3D	075	= =	93	5D	135]]	125	7D	175	} }		
30	1E	036	RS	(record separator)	62	3E	076	> >	94	5E	136	^ ^	126	7E	176	~ ~		
31	1F	037	US	(unit separator)	63	3F	077	? ?	95	5F	137	_ _	127	7F	177	 DEL		

Source: www.LookupTables.com

UNICODE – The 16-bit code provides unique code that points to characters in many of the world's languages including Sinhala and Tamil.

$2^{16}=65536$ combinations

The Unicode standard defines UTF-8, UTF-16, and UTF-32, and several other encodings are in use.

Unicode provides a unique number for every character,
No matter what the platform,
No matter what the program,
No matter what the language.

	Advantage	Disadvantage
BCD	<ul style="list-style-type: none"> • Easy to encode and decode decimals into BCD and vice versa. • Simple to implement a hardware algorithm for the BCD converter. • It is very useful in digital systems whenever decimal information is given either as inputs or displayed as outputs. • Digital voltmeters, frequency converters and digital clocks all use BCD as they display output information in decimal. 	<ul style="list-style-type: none"> • Not space efficient. • Difficult to represent the BCD form in high speed digital computers in arithmetic operations, especially when the size and capacity of their internal registers are restricted or limited. • Require a complex design of Arithmetic and logic Unit (ALU) than the straight Binary number system. • The speed of the arithmetic operations slow due to the complete hardware circuitry involved.
ASCII	<ul style="list-style-type: none"> • Uses a linear ordering of letters. • Different versions are mostly compatible. • compatible with modern encodings 	<ul style="list-style-type: none"> • Not Standardized. • Not represent world languages.
EBCDIC	<ul style="list-style-type: none"> • uses 8 bits while ASCII uses 7 before it was extended. 	<ul style="list-style-type: none"> • Does not use a linear ordering of letters.
	<ul style="list-style-type: none"> • Contained more characters than ASCII. 	<ul style="list-style-type: none"> • Different versions are mostly not compatible. • Not compatible with modern encodings
UNICODE	<ul style="list-style-type: none"> • Standardized. • Represents most written languages in the world • ASCII has its equivalent within Unicode. 	<ul style="list-style-type: none"> • Need twice memory to store ASCII characters.

2022.04.25

Data Representation

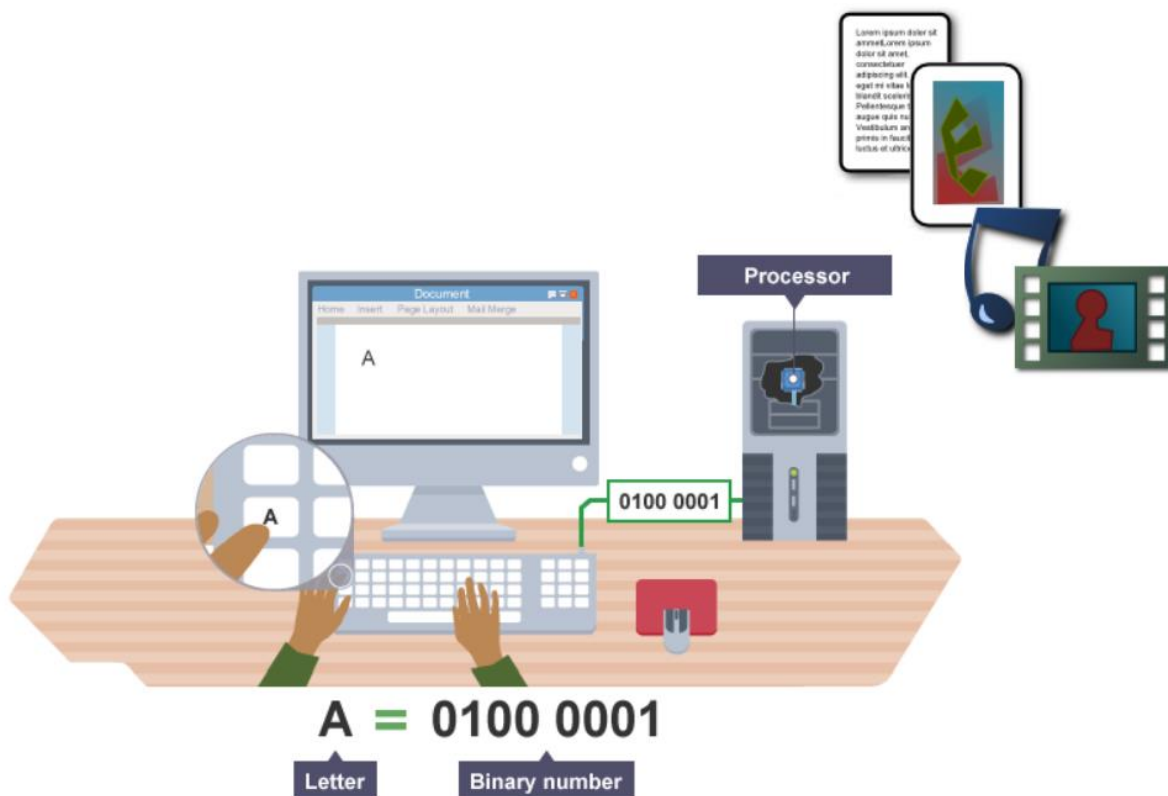
What is Data?

Unprocessed raw facts such as text, graphics, audio and video.

How to Represent Text?

When we type a character ("A"), for that assign a character code (01000001). Every character represented by a code. It is proposed by ANSI. It is called ASCII.

ASCII code for Capital A is 65 → 01000001

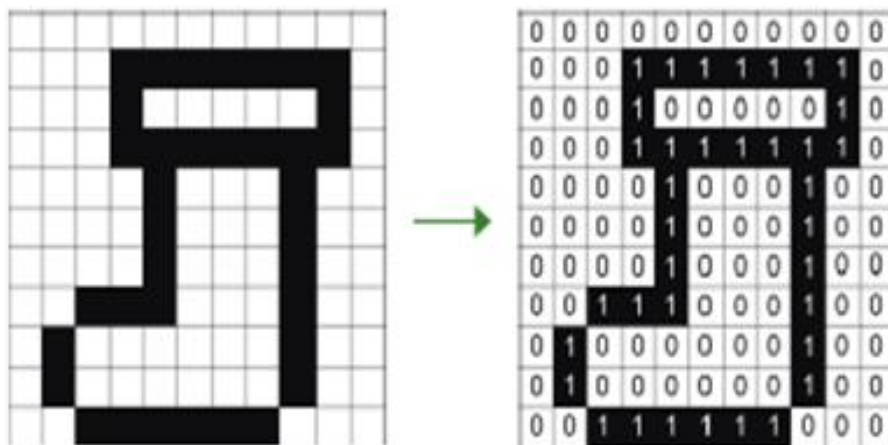


How to represent images?

(Black and White image)

- First you have to divide the image into rows and columns. Intersects forming cells. These cells are called picture elements (pixels). In the sense you have to divide the image into pixels.
- You have 2 colors (black and white). Then match the pixels into bits.
- This image becomes a collection of bits. This is called bitmap file.
- Then computer can understand:

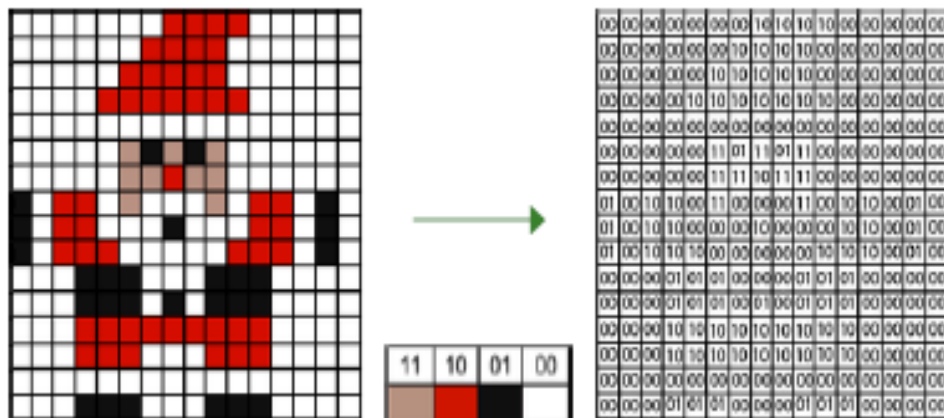
Black → 1
White → 0



Colored image
(4 color image)

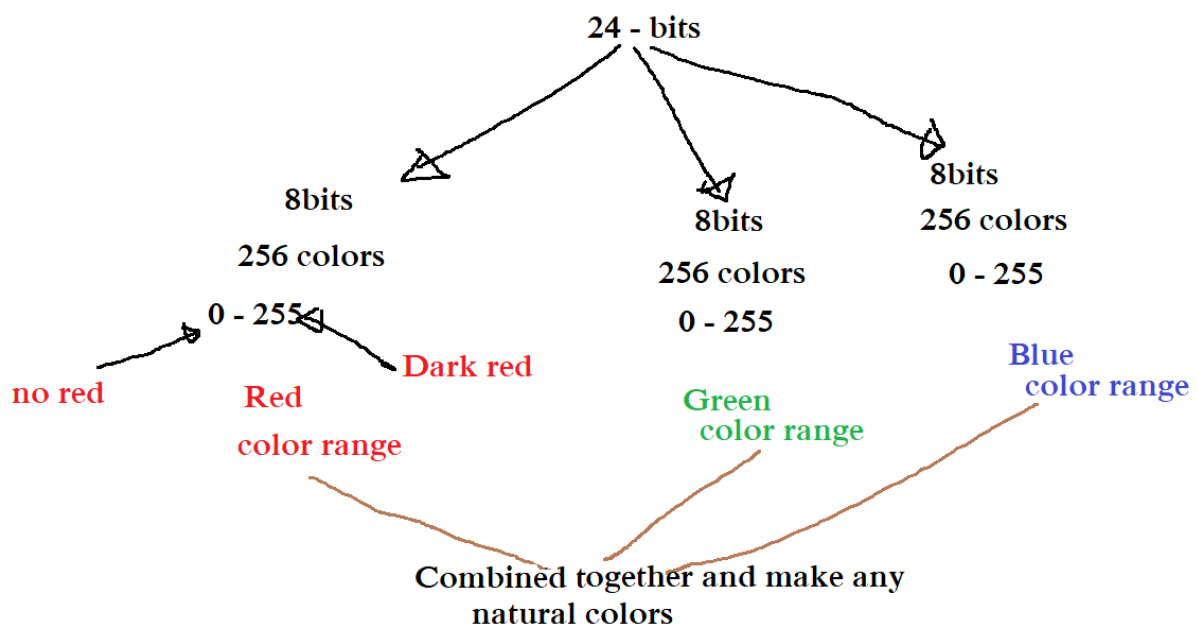
- Break the image into pixels.
- To represent 4 unique color combinations, we need 2 bits.

White	- 00
Black	- 01
Red	- 10
Gray	- 11
- Then assign these combinations into each pixel.
- Now it is called bitmap. Computer can understand bitmap.



True color imagers

- New modern computers use 24-bit true color imagers.
- 24 bits can break up into 3 groups.



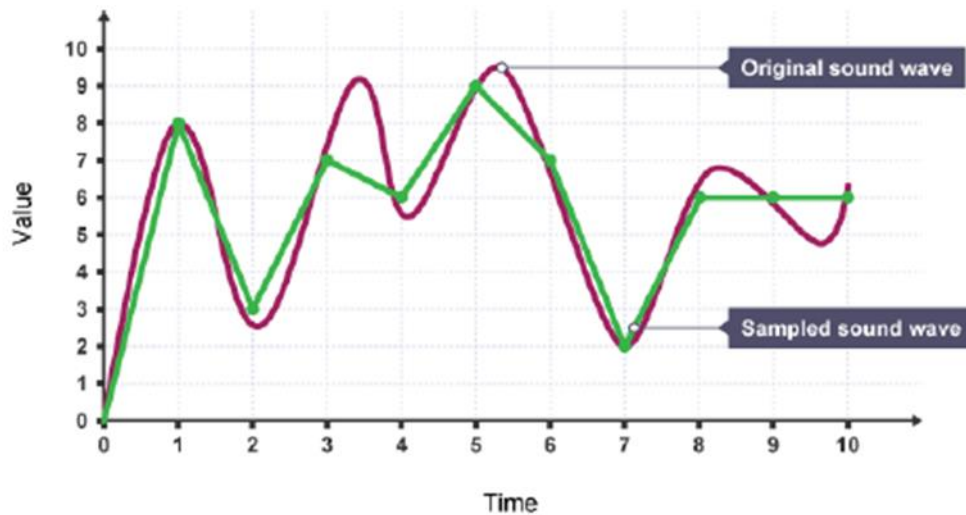
How to represent a Video?

- Video is a series of images, which display rapidly (24 frame in one second).
- That's why video break into a frame.
- When we come to the film industry called motion imagers (imagers are moving very fast).

**How to represent an Audio?**

- Audio signal is Continuous (Analog). Computer cannot understand Analog signals. It can only understand Desecrate (Digital) signals.
- We need to convert Analog signal into Desecrate signal. But it is not possible. Because we cannot digitize all the analog values into Digital values. Because Analog signal has infinite number of values. So, we take sample values then digitize them.
- Reproduced the signal using sample values. It is closed to the original signal but not identical. If you take more samples

then can have a signal that is very closed to the original signal. Sample signal we can represent by binary.



Time sample	1	2	3	4	5	6	7	8	9	10
Decimal	8	3	7	6	9	7	2	6	6	6
Binary	1000	0011	0111	0110	1001	0111	0010	0100	0110	0110

*****Whatever data represent inside the computer using numbers*****

2022.04.26

Number Systems

- Humans need different number systems to interact with computers.
- Humans can't remember large set of ones and zeros ("1" and "0").
- Binary files are difficult for humans to read and edit.
- Binary files can get confusing when transferring between computers with different architectures.
- By using octal and hexadecimal numbers humans can easily read.

Decimal Numbers

- The number system is extremely close to our day to day life.
- It is made up of the digits of 0,1,2,3,4,5,6,7,8,9
- It is called the decimal number system because ten digits available in this number system
- It can be believed that it has become a popular number system because humans have ten fingers in their hand.
- However, a large number can be represented using these ten digits.
- The decimal point is used to separate a fractional part of a number
- Plus”+” sign used to show positive values and negative “-” sign is used to show negative values.
- E.g.:- +10.235, -25.321

Integers

- Integers are a number set.
- Natural numbers (0, 1, 2, 3...) and (-1,-2,-3...) are integers.
- Neither decimal number nor fractions are integers.
- E.g. :- 0.75, 8.5

Decimal number system

- This number system has 10 digits of 0, 1, 2, 3, 4, 5, 6, 7, 8,9.
- When the value of a particular number exceeds the largest number 9 in that number set, the multiples of 10 of the number of values are transferred to the next (left) place value.
- Every place value is multiplied by ten to get the next place value.

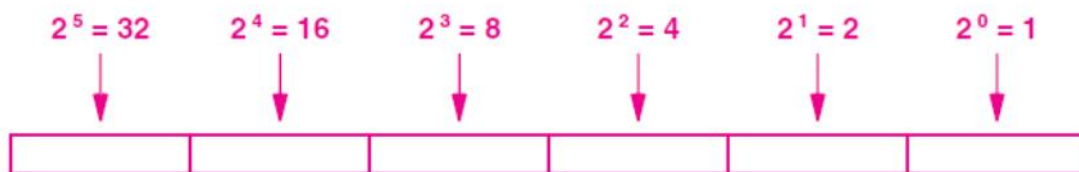
- E.g. :- $3456 = 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$
 $= 3000 + 400 + 50 + 6$
 $= 3456$
- The place values in decimal number are multiple values of 10.
- Therefore, the base value of the decimal number system is 10.

Binary number system

- The binary number system has two digits which can be represent two states.
- These two states are, represented by digits “0” and “1”.
- Therefore a number system with the two digits can be used here.
- There are multiplications of 2 in place values of the binary number system. They are as follows.

$$\begin{array}{cccccccc}
 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & 2^{-1} & 2^{-2} & 2^{-3} \\
 16 & 8 & 4 & 2 & 1 & 1/2 & 1/4 & 1/8
 \end{array}$$

Place values



The value associated with each of the first six positions of the binary number system. Each binary digit corresponds to the next power of two.

- Therefore, the base value of the binary number system is 2.
- As the computer works on electricity and is an electronic device, its functions are controlled by two states.

- These two states are, where the power is ON and OFF (As two different levels of voltage)
- The every place value is multiply by 0 or 1 (digits of binary number system) to get the value of binary number.

E.g. :- $110102 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
 $= 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$
 $= 26_{10}$

Therefore, $110102 = 26_{10}$

- One location (one digits) is call a bit. There are 5 bits in the above number.

Octal number system

- The base value of the octal number system is 8.
- The digits are 0, 1, 2, 3, 4, 5, 6 and 7.
- The place values are as follows.

8^2	8^1	8^0	8^{-1}	8^{-2}	
64	8	1	1/8	1/64	place values

E.g. :- $673_8 = 6 \times 8^2 + 7 \times 8^1 + 3 \times 8^0$
 $= 6 \times 64 + 7 \times 8 + 3 \times 1$
 $= 443_{10}$

Therefore, $673_8 = 443_{10}$

Hexadecimal number system

- The base value of hexadecimal number system is 16.
- There are 16 digits in the hexadecimal number system. Digits over value 9 need two digits. Therefore, A, B, C, D, E, F also used as remaining digits. All digits are as follows.
- Digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- The minimum value is 0 and maximum value is F ($=15_{10}$).
- The values represented by the digits are as follows

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

E.g. :- $BC12_{16}$

$$\begin{aligned}
 &= B(11) \times 16^3 + C(12) \times 16^2 + 1 \times 16^1 + 2 \times 16^0 \\
 &= 11 \times 16^3 + 12 \times 16^2 + 1 \times 16^1 + 2 \times 16^0 \\
 &= 11 \times 4096 + 12 \times 256 + 1 \times 16 + 2 \times 1 \\
 &= 45056 + 3072 + 16 + 2 \\
 &= 48146
 \end{aligned}$$

Therefore, $BC12_{16} = 48146_{10}$

1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1

D E C 9 0 9 4 9

Illustration of the relationship between binary and hexadecimal.
Each hex digit represents four bits.

2022.04.28

Conversions between number systems

Decimal

Example

$$473_{10} = 4 \times 10^2 + 7 \times 10^1 + 3 \times 10^0$$

Binary

Decimal to binary

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

Binary to decimal

$$101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5_{10}$$

		Remainder
2	25	
2	12	1 (LSB)
2	6	0
2	3	0
	1	1
	(MSB)	

Octal

Decimal to octal

$$461_{10} = 715_8$$

Octal to decimal

$$112_8 = 1 \times 8^2 + 1 \times 8^1 + 2 \times 8^0 = 74_{10}$$

Binary to octal

$$1110111001_2 = 001 \ 110 \ 111 \ 001 = 1671_8$$

Octal to binary

$$1671_8 = 001 \ 110 \ 111 \ 001 = 001110111001_2$$

		Remainder
8	461	
8	57	5 (LSD)
	7	1
	(MSD)	

Hexadecimal

Decimal to hexadecimal

$$10767_{10} = 2A0F_{16}$$

Hexadecimal to decimal

$$2F3_{16} = 2 \times 16^2 + 15 \times 16^1 + 3 \times 16^0 = 755_{10}$$

Binary to hexadecimal

11011110110010010000100101001001

D E C 9 0 9 4 9

Hexadecimal to Binary

D E C 9 0 9 4 9

11011110110010010000100101001001

16	10767	Remainder
16	672	15 = F (LSD)
16	42	0
	2	10 = A
	(MSD)	

Converting fractions to Binary

- Multiply the given decimal fraction by 2.
- Multiply by 2 until the decimal part becomes 0.
- Write the values in front of decimal point from beginning to end.

E.g.:- convert 0.3125_{10} to binary


	0.3125	x2
0	.625	x2
1	.25	x2
0	.50	x2
1	.00	

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

Converting fractions to Octal

- Multiply the given decimal fraction by 8.
- Multiply the decimal by 8 until it becomes 0.
- Write from the beginning to end, the values in front of the decimal point.

E.g. :- convert 0.3125_{10} to binary



0	0.3125	x8
2	.50	x8
4	.0	x8

$$0.3125_{10} = 0.24_8$$

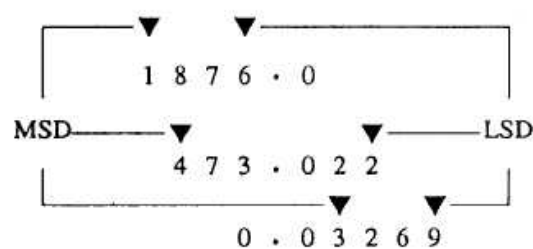
2022.04.29

Most Significant Digit (MSD) and Least Significant Digit (LSD)

MSD - The Digit that contain the most positional value in a number.

LSD - The Digit that contains the least positional value in a number.

Number	MSD	LSD
2975.0	2	5
56.034	5	4
0.03145	3	5
0031.0060	3	6



Sign Integers

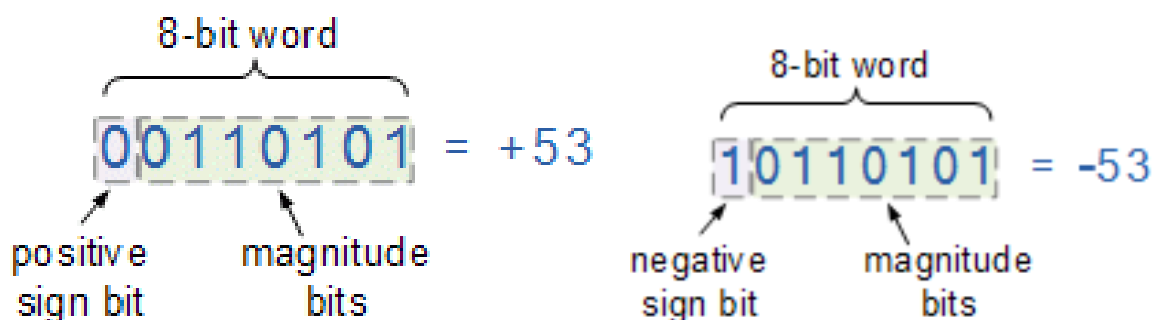
To represent negative numbers, we need an alternative interpretation of bit values. Three interpretations have been used:

1. Signed Magnitude Representation
2. 1's Complement
3. 2's Complement

Signed Magnitude Representation

- Used leftmost bit for the sign.

Mathematical representation	Binary representation
3	0011
-3	1011



- Maximum range that we can represent only 0-7 with 4-bit computer. (Max: -7 to +7)

	+	-
0	0000	1000
1	0001	1001
2	0010	1010
3	0011	1011
4	0100	1100
5	0101	1101
6	0110	1110
7	0111	1111

**** If you want to extend this range you need more bits.

- This representation is used in early days and it has some problems.

Problems of sign magnitude

- This leads to **having two representations for the number zero**. We can have a positive result for zero, **+0 or 0000₂**, and a negative result for zero, **-0 or 1000₂**. Both are valid but which one is correct?

- You can't do subtraction by addition of negative values.

$$3 + (-3) = 0$$

Answer should be 0. But adding negative numbers give wrong answer.

$$\begin{array}{r}
 0011 \quad (+3) \\
 + \\
 1011 \quad (-3) \\
 \hline
 1110 \\
 \hline
 \end{array}$$

↓
- 6

To do all 4 mathematical operations in decimal number system, can do by using adder.

Ex:

$$3 + 5 = 8$$

$$5 - 3 = 2 \rightarrow 5 + (-3) = 2$$

$$5 * 3 = 15 \rightarrow 5 + 5 + 5 = 15$$

$$15 / 3 = 5 \rightarrow 15 - 5 - 5 - 5 = 5$$

- Solutions for these problems, computer Architects proposed 1's complement representation.

1's complement representation

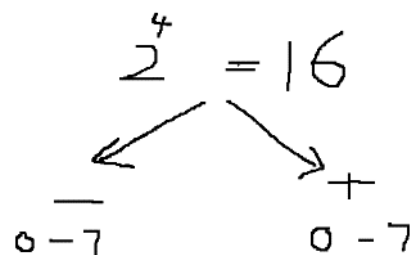
- In one's complement, positive numbers (also known as non-complements) remain unchanged as before with the sign-magnitude numbers.
- If it is a negative number
 - First take the binary value without considering the sign.
 - Invert bits (flip the bits).

Ex: represent -3 in 1's complement

$$3 = 0011 \rightarrow 1100 = -3$$

	+	-
0	0000	1111
1	0001	1110
2	0010	1101
3	0011	1100
4	0100	1011
5	0101	1010
6	0110	1001
7	0111	1000

- In the 4-bit computer, we can represent maximum -7 to +7 values.



- If we want more values need to extend bits.
- If our computer is 32-bits or 64-bits then range is big

Problems of 1's complement representation

- Still, we have two 0's problem (-0 and +0)
- Solved the second problem. Now we can do subtractions as addition of negative numbers.

①

$$\begin{array}{r} 3 + \\ -3 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0011 \text{ (3)} \\ 1100^+ \text{ (-3)} \\ \hline 1111 \text{ (-0)} \end{array}$$

②

$$\begin{array}{r} 4 + \\ -6 \\ \hline -2 \end{array}$$

$$\begin{array}{r} 0100 \text{ (4)} \\ 1001^+ \text{ (-6)} \\ \hline 1101 \text{ (-2)} \end{array}$$

③

$$\begin{array}{r} 4 + \\ -2 \\ \hline 2 \end{array}$$

$$\begin{array}{r} 0100 \\ 1101 \\ \hline 10001^+ \\ \hline 0010 \text{ (2)} \end{array}$$

Overflow bit (Carry bit)
Bring the overflow bit and add to the result.

- To solve the two 0's problem, proposed 2's complement representation.

2's complement representation

- The positive numbers (also known as non-complements) remain unchanged as before with the sign-magnitude numbers.
- For the negative numbers:

○ Decimal to Binary

- Convert to binary, invert bits and then add 1 to LSB.
- Ex: how to represent -3 in 2's complement

First convert the decimal number into binary without considering the sign.

And then flip the bits and add 1 to the LSB

$$3 = 0011 \rightarrow 1100 + 1 = 1101 = -3$$

	+	-
0	0000	
1	0001	1111
2	0010	1110
3	0011	1101
4	0100	1100
5	0101	1011
6	0110	1010
7	0111	1001
8		1000

- In 4-bit computer, we can have $2^4 = 16$ combinations.
- 1 for 0 other 15 for other numbers.
- 15 cannot divide into same parts.

So here, no +8, only have +7 to -8 including one 0.

If you want to represent +8 then you have to increase the number of bits

$$3 - 3 = 3 + (-3)$$

3	0011 (3)
-3	1101 (-3)
<u>0</u>	<u>10000 (0)</u>

Simply discard the overflow bit

$$4 - 6 = 4 + (-6)$$

4	0100 (4)
-6	1010 (-6)
<u>-2</u>	<u>1110 (-2)</u>

- **Binary to Decimal**

- Take sign bit from MSB.
 - If it is 0, the number is positive. Then keep it as same and convert to decimal as usual.
 - If it is 1, invert bits, add 1 to LSB and then convert to decimal.
- **Ex: 1110 → convert to decimal**
 - MSB is 1, then it is negative number
 - Then invert bits → 0001
 - Then add 1 to LSB → 0010
 - Convert the answer into decimal. The answer is 2.
 - Then get the sign from MSB (early recognized)
 - The answer is -2

Advantage of 2's complement representation.

- Operations are much simpler.
- Two 0's problem is over.
- In 1's complement addition, if there is an overflow bit, we have to add it into LSB of the result. That is a problem. But in 2's complement addition, if there is an overflow bit, we can simply discard it.
- In our modern computers, to represent negative numbers used 2's complement representation.

Usage of sign magnitude, 1's complement and 2's complement

	Usage
Sign Magnitude	Used only when we do not add or subtract the data. They are used in analog to digital conversions. They have limited use as they require complicated arithmetic circuits.
One's Complement	Simpler design in hardware due to simpler concept.
Two's Complement	Makes it possible to build low-cost, high-speed hardware to perform arithmetic operations.

2022.05.02

Uses basic arithmetic and logic operations on binary numbers**Works out additions (multiple numbers with or without carryovers) – in binary numbers**

- The binary number system uses only two digits 0 and 1 due to which their addition is simple.

$$0+0=0$$

$$0+1=1$$

$$1+0=1$$

$$1+1=10$$

- The above first three equations are very identical to the binary digit number. The column-by-column addition of binary is applied below in details. Let us consider the addition of 11101 and 11011.

$$\begin{array}{r}
 \\
 \\
 (+) \\
 \hline
 1 \\
 \hline
 \end{array}$$

← carry

Circuit Globe

Works out subtraction (with or without borrowing) – in binary numbers

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$10 - 1 = 1$$

- The above first three operations are easy to understand as they are identical to decimal subtraction. The fourth operation can be understood with the logic two minus one is one.
- For a binary number with two or more digits, the subtraction is carried out column by column as in decimal subtraction. Also, sometimes one has to borrow from the next higher column. Consider the following example.

$$\begin{array}{r}
 010 \\
 1100 \\
 (-) 1010 \\
 \hline
 0010
 \end{array}$$

Circuit Globe

Works out NOT, AND, OR, XOR bitwise operations**Bitwise operations**

In arithmetic-logic unit (which is within the CPU), mathematical operations like: addition, subtraction, multiplication and division are done in bit-level. To perform bit-level operations bitwise operations are used.

1. NOT operation

A	NOT A
0	1
1	0

E.g. :- **NOT** 0111_2 (7_{10}) = 1000_2 (8_{10})

Example

35 = 00100011 (In Binary)

Bitwise NOT Operation of 35

NOT 00100011

11011100 = 220 (In decimal)

2. Bitwise AND operation

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

E.g. :- 0101_2 (5_{10}) AND 0011_2 (3_{10})

$$\begin{array}{r}
 0101_2 \\
 0011_2 \\
 \hline
 0001_2 \text{ (} 1_{10} \text{)}
 \end{array}$$

Therefore 0101_2 AND 0011_2 is 0001_2

For an example, suppose the bitwise AND operation of two integers 12 and 25.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bit Operation of 12 and 25

00001100

AND 00011001

00001000 = 8 (In decimal)

3. Bitwise OR operation

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

E.g. :- 0101_2 (5_{10}) **OR** 0011_2 (3_{10})

0 1 0 1₂

0 0 1 1₂

0 1 1 1₂ (7_{10})

Therefore 0101_2 OR 0011_2 is 0111_2

Example

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25

00001100

OR 00011001

00011101 = 29 (In decimal)

4. Bitwise XOR (Exclusive OR) operation

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

E.g. :- $0010_2 (2_{10}) \text{ XOR } 1010_2 (10_{10})$

1010_2

0010_2

$= 1000_2 (8_{10})$

Therefore $0010_2 \text{ XOR } 1010_2$ is 1000_2

Example

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25

00001100

00011001

00010101 = 21 (In decimal)