COMPUTER NUMBER SYSTEM FOR PROGRAMMING

Number system as the name suggests it is all about the number game in different representations which will allow operations in Computer.

Number system representations are as follows:

1. Binary System
2. Decimal System
3. Octal System
4. Hexa-Decimal System

---

1. Binary System:

- ➢ Binary represents "1" or "0".
- ➢ Ideal Binary representation must be either in 4 bits / 8 bits / 16 bits / 32 bits / 64 bits / 128 bits..and so on as per the industrial hardware setups.
- ➢ Binary values are calculated from right to left which raises to the power of 2.

Example: $(1010)_2$ to Decimal

1 0 1 0	1010
➔ (2^3x1) + (2^2x0) + (2^1x1) + (2^0x0) ➔ 8 + 0 + 2 + 0	= 10

Binary Arithmetic

a) Binary Addition:

Binary Addition truth table is as follows:

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Binary addition:

Note: Red digit notation is to show generated "Carry"

Example 1: **(Without Carry)**

Binary	Decimal
1010 0100	10 4
1110	14

Example 2: **(With Carry)**

Binary	Decimal
0000 1010 0000 1110 1 11	10 14
0001 1000	24

Signed overflow: The Signed overflow is said to occur when a carry is generated after summing the two binary values.

Example:

1111 1111 0101 0000 1100 1100 1111 1111 1
10000 1100 0001

(Signed Overflow occurred)

---

b) Binary Subtraction :

The binary Subtraction truth table is as follows:

a	b	difference	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Binary Subtraction :

Note: Red digit notation is to show generated "Borrow"

Example-1: **(Without Borrow)**

Binary	Decimal
1110 0100	14 4
1010	10

Example-2: **(With Borrow)**

Binary	Decimal
2 0000 1100 0000 1010	 12 10
0000 0010	02

c) Binary Multiplication:

The binary multiplication operation is actually a process of addition and shifting operation and this process has to be continued until all the multiplier is done and finally the addition operation is made.

The binary Multiplication truth table is as follows:

a	**b**	**result**
0	0	0
0	1	0
1	0	0
1	1	1

Example: $(1101)_2 \times (101)_2$

1101 x 101
1101 0000 - 1101 - -
1000001

Verification:

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

$(101)_2 = (5)_{10}$

=> $13 \times 5 = (65)_{10} = (1000001)_2$

---

d) Binary Division:

i) Basically the reverse of the multiply by shift and add.

ii) Align leftmost digits in dividend and divisor

iii) **Repeat**

If that portion of the dividend above the divisor is greater than or equal to the divisor

Then subtract divisor from that portion of the dividend and Concatenate 1 to the right-hand end of the quotient

Else concatenate 0 to the right-hand end of the quotient

Shift the divisor one place right

Until dividend is less than the divisor

The quotient is correct, the dividend is remainder

Follow these rules for easy division:

1. If divisor > dividend, then multiply divisor with ‘0’

2. If divisor <= dividend, then multiply divisor with ‘1’

Example: $(1001)_2 \div (101)_2$

101	1001	1
	101	
	0100	

Verification :

=> $(1001)_2 = 9$

=> $(101)_2 = 5$

=> 9 ÷ 5 = 1 ; Remainder 4

NUMBER SYSTEM CONVERSIONS

Exercise 1: Binary To Decimal Conversion

What is the Decimal equivalent of $(11110000101010101)_2$?

1×2^{16}	65536
1×2^{15}	32768
1×2^{14}	16384
1×2^{13}	8192
0×2^{12}	0
0×2^{11}	0
0×2^{10}	0
0×2^{9}	0
1×2^{8}	256
0×2^{7}	0
1×2^{6}	64

0×2^5	0
1×2^4	16
0×2^3	0
1×2^2	4
0×2^1	0
1×2^0	1
Total	**123221**

➢ **$(11110000101010101)_2 = (123221)_{10}$**

Exercise 2: Binary To Octal Conversion :

First, convert Binary to Decimal and then Decimal to Octal.

Convert $(1101)_2$ => $(\ )_8$?

1×2^0	1
0×2^1	0
1×2^2	4
1×2^3	8
Total	**13**

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

Dividing with 8 :

13	
1	5

=> $(1101)_2 = (13)_{10} = (15)_8$

Cross Verifying :

5×8^0	5
1×8^1	8
Total	13

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

Hence, **Verified**

Exercise 3: Binary To Hexadecimal Conversion

First, convert Binary to Decimal and then Decimal to HexaDecimal.

Convert $(110110111110101)_2$ => (? $)_{16}$?

1×2^0	1
0×2^1	0
1×2^2	4
0×2^3	0
1×2^4	16
1×2^5	32
1×2^6	64
1×2^7	128
1×2^8	256
0×2^9	512
1×2^{10}	0
1×2^{11}	2048
0×2^{12}	4096
1×2^{13}	0
1×2^{14}	16384
Total	**28154**

Here, $(1101101111110101)_2$ => $(28154)_{10}$ = $(?)_{16}$

Let us convert $(28154)_{10}$ to Hexa

28154	
1759	10(A)
109	15(F)
6	13(D)

Therefore, $(1101101111110101)_2$ => **$(6DFA)_{16}$**

Exercise 4: Decimal To Binary Conversion

$(5610)_{10}$ => $(?)_2$?

Dividing the value with 2 each time.

5610	
2805	0
1402	1
701	0
350	1
175	0
87	1
43	1
21	1
10	1
5	0
2	1
1	0

Answer : **$(1010111101010)_2$**

Exercise 5: Decimal To Octal Conversion

Convert **$(5610)_{10}$ => (? $)_8$?**

Dividing the value with 8 each time.

5610	
701	2
87	5
10	7
1	2

$(5610)_{10}$ = > $(12752)_8$

Exercise 6: Decimal To Hexa-Decimal Conversion

Convert **$(5610)_{10}$ => (? $)_{16}$?**

Dividing the value with 16 each time.

5610	
350	10(A)
21	14(E)
1	5

(5610)10 = > $(15EA)_{16}$

Exercise 7: Octal To Binary Conversion

$(231)_8$ => (? $)_2$?

$1x8^0$	1
$3x8^1$	24
$2x8^2$	128
Total	153

$(231)_8$ => $(153)_{10}$

Converting to base 2

Dividing the value with 2 each time.

153	
76	1
38	0
19	0
9	1
4	1
2	0
1	0

Therefore, **$(231)_8$ => $(10011001)_2$**

Cross Verifying :

Converting to base 10 :

$1x2^0$	1
$0x2^1$	0
$0x2^2$	0
$1x2^3$	8
$1x2^4$	16
$0x2^5$	0
$0x2^6$	0
$1x2^7$	128

$(10011001)_2 = (153)_{10}$

=> Converting base 10 to base 8

$(153)_{10}$ => $(?)_8$

Dividing the value with 8 each time.

153	

19	1
2	3

$(153)_{10}$ => $(231)_8$

Hence, verified

Exercise 8: Octal To Decimal Conversion

$(76)_8$ =>$(?)_{10}$?

$6x8^0$	6
$7x8^1$	56
Total	62

Therefore, **$(76)_8$ => $(62)_{10}$**

Cross Verifying:

62	
7	6

Therefore, **$(62)_{10}$ = $(76)_8$**

Hence, verified

Exercise 9: Octal To Hexa-Decimal Conversion

$(76)_8$ =>$(?)_{16}$?

$(76)_8$ =>$(?)_{10}$ = $(?)_{16}$

$6x8^0$	6
$7x8^1$	56
Total	62

$(76)_8$ = $(62)_{10}$ = $(?)_{16}$

62	14	E
3		3

Therefore, **$(76)_8 = (3E)_{16}$**

Exercise 10: Hexa-Decimal to Binary Conversion

$(0xF2)_{16} = (?)_2$?

=> $(0xF2)_{16} = (?)_{10} = (?)_2$

$2x16^0$	2
$15x16^1$	240
Total	242

Here, $(0XF2)_{16} = (242)_{10} = (?)_2$

242	
121	0
60	1
30	0
15	0
7	1
3	1
1	1

Therefore, **$(0XF2)_{16} = (11110010)_2$**

Exercise 11: Hexa-Decimal to Decimal Conversion

$(0xF2)_{16} = (?)_{10}$?

$2x16^0$	2
$15x16^1$	240
Total	242

Therefore, $\mathbf{(0XF2)_{16}=(242)_{10}}$

Exercise 12: Hexa-Decimal to Octal Conversion

$\mathbf{(0xF2)_{16}= (?)_8 ?}$

$(0xF2)_{16}= (?)_{10} = (?)_8$

$2x16^0$	2
$15x16^1$	240
Total	242

Here, $(0XF2)_{16}=(242)_{10} = (?)_2$

242	
30	2
3	6

Therefore, $\mathbf{(0XF2)_{16}= (362)_8}$

Complements in Binary Number System :

So far we have seen arithmetic operations with binary numbers. Now let's try to understand the magnitude or sign of a binary number.

- **Unsigned Number:**

  An unsigned number only stores **positive numbers** in binary.

  Example: Let's take a positive number '+10' in 5 bits.

→ '+10' in binary $(01010)_2$

This is known as an unsigned number.

- **Signed number:**

Signed Number is used to represent **negative numbers** in binary.

The process of finding a negative binary number is nothing but calculating the **2's complement** of that number.

Sign (or) Magnitude:

0	Positive	+
1	Negative	-

The complement system is used to represent negative numbers.

There are two types of Complements = **R's** Complement and **R-1's** Complement

→ R's Comp = R-1 comp + 1

For Binary Numbers : Radix(R) = 2 ; 2's comp and 1's comp are possible.

For Ternary Numbers : Radix(R) = 3; 3's comp and 2's comp are possible.

For Decimal Numbers : Radix(R) = 10; 10's comp and 9's comp are possible.

Example: **Now let's try to find the binary number of '-10' which is a signed number.**

- 2's complement of Number N = 1s complement of N + 1
- Now 2's complement of -10 : 1s comp of 10 +1
- 1s complement : Inverting bits

  Consider $(10)_{10} = (1010)_2$
- 1s complement + 1 :

10101 10001 1
10110

So the binary of **-10** would be $\mathbf{(10110)_2}$

General Formula for finding R's Complement and (R-1)'s Complement

R's Complement ==> ($[R^n]$ base 10) - (N base 10)

(R-1)s Complement ==> ($[[R^n]$ base 10 - 1]) - (N base 10)

N ==> Number

R ==> Radix

n ==> No. of Digits

FLOATING POINT NUMBERS CONVERSION :

Exercise 1: Converting Decimal Floating Numbers to Binary

$(34.625)_{10} = (?)_2$

Here, **34** is called as **'Exponent'** and .**625** is called as **'Mantissa'**

Conversion of Exponent to Binary:

34	
17	0
8	1
4	0
2	0
1	0

$(34)_{10} = (100010)_2$

Conversion of Mantissa to Binary :

Multiplying Exponent with 2

Mantissa Part :

Mantissa part is multiplied with 2 until the exponent gets 0.

In each step, consider the exponent as the solution value to append and mantissa as a multiplier for the next consecutive value.

0.625 X 2	1.25 (Take the Exponent out i.e. 1)
0.25 X 2	0.5 (Take the Exponent out i.e. 0)
0.5 x 2	1.0 (Take the Exponent out i.e. 1)

Therefore, **$(34.625)_{10} = (100010.101)_2$**

Exercise 2 : Converting Binary Number to Decimal Floating Number

$(1010.1111)_2$ = (? $)_{10}$

Mantessa = .1111

Exponent = 1010

Converting Exponent to Binary

$0x2^0$	0
$1x2^1$	2
$0x2^2$	0
$1x2^3$	8
Total	10

Exponent in Decimal : $(10)_{10}$

Mantessa = 1111

	0.5
$(1x2^{-2})$	0.25
$1x2^{-3}$	0.125
$1x2^{-4}$	0.0625
Total	0.9375

Therefore, Ordinary fraction = **10.9375 = (175/16)**

Storage Units and Maximum, Minimum Values of 'N' Bits:

Bit:

The smallest unit of data in a computer is called **Bit** (Binary Digit). A bit has a single binary value, either '**0**' or '**1**'.

Nibble	Half a Byte	4 bits
Byte	8 bits	1 Byte
KiloByte	1024 Bytes	1 KB
MegaByte	1024 KiloBytes	1 MB
GigaByte	1024 MegaBytes	1 GB
TeraByte	1024 GigaBytes	1 TB

Minimum No of Bits Required from Decimal Number 0-18

Decimal	Binary	No.of Bits
0	0	1
1	1	1
2	10	2
3	11	2
4	100	3
5	101	3
6	110	3
7	111	3
8	1000	4
9	1001	4
10	1010	4
11	1011	4
12	1100	4
13	1101	4

14	1110	4
15	1111	4
16	10000	5
17	10001	5
18	10010	5

Exercise 1 :

Given 4 bits, what are the minimum and maximum numbers you can store?

(Unsigned)

Minimum Number in 4 bits : 0000 = 0

Maximum Number in 4 bits : 1111 = 15

(Signed)

Minimum Number in 4 bits: 1000 = -8 (Since its signed, use 2's comp)

Maximum Number in 4 bits: 0111 = +7 (Since its signed, use 2's comp)

Exercise 2 :

Given 8 bits, what are the minimum and maximum numbers you can store?

(**Unsigned**)

Minimum Number in 8 bits : 0000 0000 = 0

Maximum Number in 8 bits : 1111 1111 = 255

(**Signed**)

Minimum Number in 8 bits: 1000 0000= -128 (Since its signed, use 2's comp)

Maximum Number in 8 bits: 0111 1111 = +127 (Since its signed, use 2's comp)

Exercise 3: Biggest Binary Number

Sign bit must be **'0'** for an n-bit number in order to be **a positive number** whereas **'1'** represents a **negative number:**

Description	Binary	Decimal	General form
Biggest binary number with 1 bit is	$(1)_2$	0	$[\ 2^1-1]$
Biggest binary number with 2 bits is	$(11)_2$	3	$[\ 2^2-1]$
Biggest binary number with 3 bits is	$(111)_2$	7	$[\ 2^3-1]$
-	-	-	-
-	-	-	-
- (and so on)	-	-	-
Biggest binary number with n bits is	$(111..n)_2$	-	$[\ 2^n - 1\ ]$

Therefore, the biggest binary number with n bits is derived as **"$2^{(n)} - 1$".**

Example: Consider n = 3 bits;

=> $(\ 2^3-1\ ) = 7$

Biggest number with 3 bits will be 7 which is $(111)_2$

ASCII

ASCII stands for **"American Standard Code for Information Interchange"**, is a character encoding standard universally.

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Image Source: Google Images

Exercise 1: Represent string ***CODE*** in Binary Format

- Find ASCII numbers of individual characters
- Convert ASCII number from Decimal to Binary

Character	ASCII Value	Binary Value
C	67	0100 0011

O	79	0100 1111
D	68	0100 0100
E	69	0100 0101