Number system

Number system is used for writing numbers. Every number system has **literals.** The number of literals is called **radix** or **base**.

IDC 101: Introduction to computers

Number system (Integers)

Position Number	3	2	1	0

Decimal number system (10)

Positional value	10^3	10^2	10^1	10^0

Decimal Number $(1729)_{10}$

In decimal number $1x\ 10^3 + 7 \times 10^2 + 2x10^1 + 9x10^0$

Binary number system (2)

Positional value	2^3	2^2	2^1	2^0

Binary Number $(1110)_2$

In decimal number $1x\ 2^3 + 1x\ 2^2 + 1 \times 2^1 + 0 \times 2^0 = 14$

IDC 101: Introduction to computers

Number system (Integers)

Octal number system (8)

Positional value	8^3	8^2	8^1	8^0

Octal Number $(1024)_8$

Decimal Number $1 \times 8^3 + 0 \times 8^2 + 2 \times 8^1 + 4 \times 8^0 = 532$

Hexadecimal number system (16)

Positional value	16^3	16^2	16^1	16^0

Hexadecimal Number $(1C3)_{16}$

In decimal Number $0 \times 16^3 + 1 \times 16^2 + 12 \times 16^1 + 3 \times 16^0 = 451$

IDC 101: Introduction to computers

Octal	Decimal	Binary
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

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

IDC 101: Introduction to computers

Number system (Fractional part)

Position Number	3	2	1	0	.	-1	-2

Decimal number system (10)

Positional value	10^3	10^2	10^1	10^0	.	10^{-1}	10^{-2}

Decimal Number $(1729.21)_{10}$

In decimal number $1x\ 10^3 + 7x\ 10^2 + 2x\ 10^1 + 9 x\ 10^0 + 2x\ 10^{-1} + 1x\ 10^{-2}$

Binary number system (2)

Positional value	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}

Binary Number $(1110.11)_2$

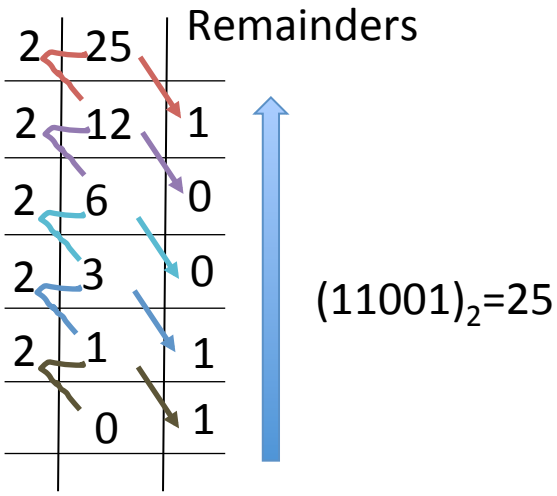
In decimal number $1x\ 2^3 + 1x\ 2^2 + 1 x\ 2^1 + 0 x\ 2^0 + 1 x\ 2^{-1} + 1 x\ 2^{-2} = 14.75$

IDC 101: Introduction to computers

Decimal to Binary/Octal/Hexadecimal number conversion (Integer)

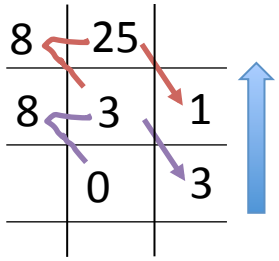
- **Convert 25 to binary or any base number**
  1. Divide the number by 2 (or proper base) and write **Remainder.**
  2. Next, keep on dividing the quotient by the 2 (or proper base) and note down the **Remainder**. Do this until quotient if **Zero.**
  3. Read the remainder from bottom to top to get binary (base) equivalent of decimal number

To Binary conversion



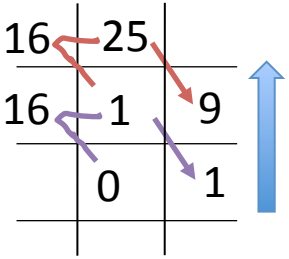
$(11001)_2=25$

To Octal conversion



$(31)_8=25$

To Hexadecimal conversion



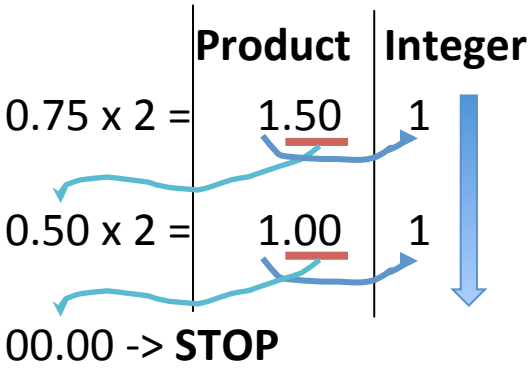
$(19)_{16}=25$

IDC 101: Introduction to computers

Decimal to Binary/Octal/Hexadecimal number conversion (Fraction)

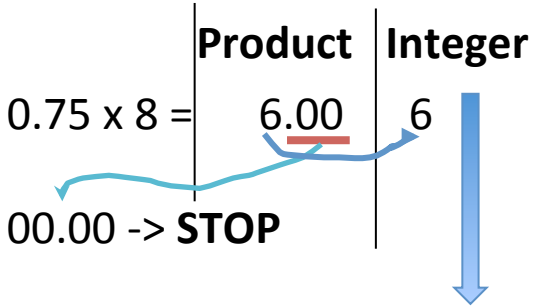
- **Convert 0.75 to binary or any base number**
  1. Multiply the fractional part of the number with 2 (or proper base) and from the product keep **Integer part.**
  2. Next multiply the **fractional part of product from previous step** with 2 (or proper base) and follow the same rule of writing Integer part.
  3. Repeat the multiplication till the fractional part becomes **zero**.
  4. If the fractional part does not become Zero and the fractional part becomes repeating, the multiplication is stopped. If the number non-recurring, then STOP at maximum bit size available for storing the number. The result is base representation for the binary number (See next slide for example).

Binary conversion



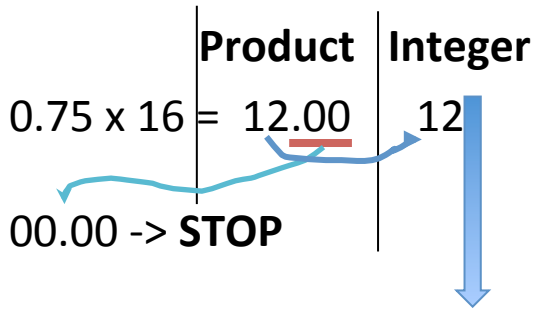
$(0.11)_2$=0.75

Octal conversion



$(0.6)_8$=0.75

Hexadecimal conversion

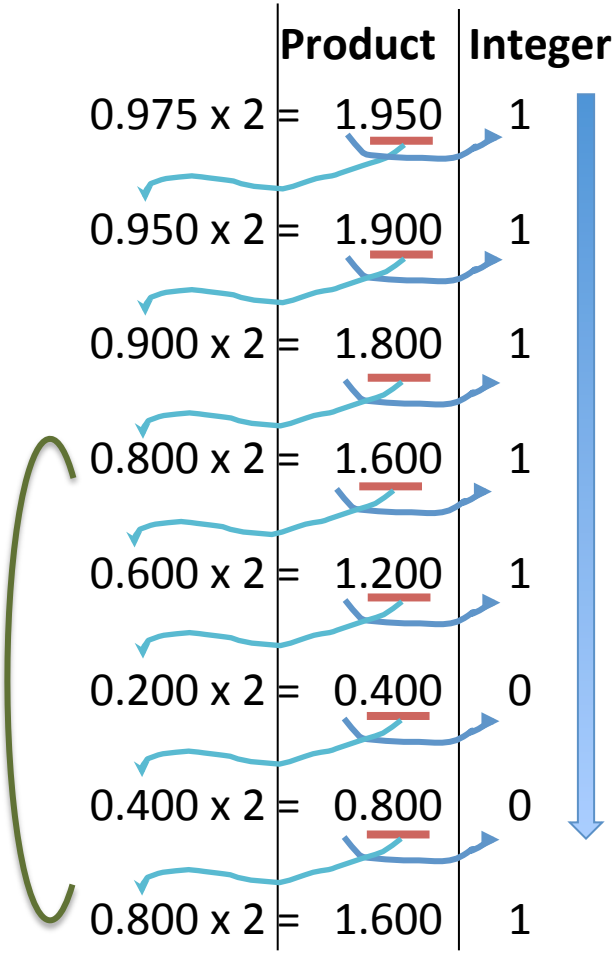


$(0.C)_{16}$=0.75

IDC 101: Introduction to computers

Decimal to Binary/Octal/Hexadecimal number conversion (Fraction)

Binary conversion



$(0.1111100)_2=0.975$

OR if continued for a next steps after STOP

$(0.111110011001100)_2=0.975$

-> **STOP depending on bit size used for representation**

IDC 101: Introduction to computers

BINARY to Octal/Hexadecimal number conversion

- **Binary number to octal (Integer)**

10010110 -> Split into a perfect group of 3 bits starting from LSB (Least significant bit) to MSB (Most significant bit)

010 010 110 If required, **0** can be added to MSB to make group of 3 bits

2 2 6 Convert 3 bit binary into decimal number, which will range from 0 to 7

Therefore, $(226)_8 = (10010110)_2$

- **Binary number to Hexadecimal (Integer)**

10010110 -> Split into a perfect group of 4 bits starting from LSB (Least significant bit) to MSB (Most significant bit)

1001 0110 If required, **0** can be added to MSB to make group of 4 bits

9 6 Convert 4 bit binary into decimal number, which will range from (0-9,A-F)

Therefore, $(96)_{16} = (10010110)_2$

IDC 101: Introduction to computers

BINARY to Octal/Hexadecimal number conversion

- **Binary number to octal (Fraction)**

10010110.10 **->** Split into a perfect group of 3 bits starting from LSB (Least significant bit) to MSB (Most significant bit)

010 010 110 . 100 If required, **0** can be added to MSB to make group of 3 bits **or**
0 can be added to LSB after decimal to make a group of 3 bit

2 2 6 . 4 Convert 3 bit binary into decimal number, which will range from 0 to 7

Therefore, $(226.4)_8 = (10010110.10)_2$

Same rules are used in conversion to Hexadecimal numbers

ASCII and Unicode system

American Standard Code for Information Interchange (ASCII) is a character encoding standard that was developed for electronic communications. It was initially designed to have 7-bit unique character (0-127). Subsequently, it was extended to 8-bit and now it is from 0-256.

0-31 are non-printable codes
48-57 encode character from 0-9
65-90 encode A-Z
97-122 encode a-z

ISCII for Indian languages

IDC 101: Introduction to computers

ASCII and Unicode system

Unicode is universal character encoding standard. It assigns a code to every character and symbol in every language.

UTF-8 (Unicode Transformation Format). It used 1-4 byte to encode a character

UTF-16 and UTF-32 uses 2 and 4 byte to represent character.

IDC 101: Introduction to computers

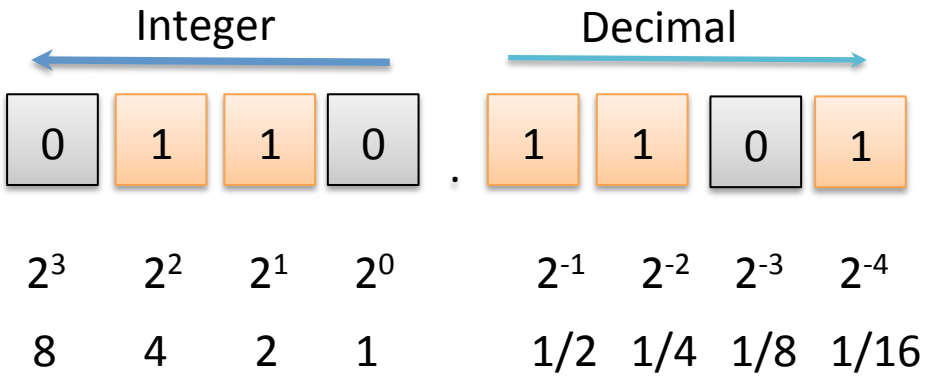
Integer number representation and bit size

Type	Storage size (in bits)	Storage in bytes	Value range
Unsigned Int	16	2	0 – 65,535
Signed Int	16	2	-32,768 to 32,767
Unsigned Int	32	4	0 – 4,294,967,295
Signed Int	32	4	-2,147,483,648 to 2,147,483,647
Unsigned Int	64	8	0 to 18446744073709551615
Signed Int	64	8	-9223372036854775808 to 9223372036854775807

IDC 101: Introduction to computers

Number representation

- **Fixed point number representation:** Fixed number of bits for integer and fraction parts.



Using 8 bit the maximum unsigned decimal number that can be represented is limited!!

IDC 101: Introduction to computers

- **Floating point number representation (32 bit)**

$$\text{Decimal number} = (-1)^{\text{Sign}} \times (1 + \text{Mantissa}) \times 2^{(\text{exponent}-127)}$$

Example

Sign = $(-1)^0 = 1$

Exponent = 1001 0100 = $(133)_{10}$

Actual Exponent of base = Exponent−127 = 133−127 = **6**

IDC 101: Introduction to computers

- **Floating point number representation (32 bit)**

Decimal number = $(-1)^{Sign} \times (1 + Mantissa) \times 2^{(exponent-127)}$

Example

Sign = $(-1)^0 = +1$

Exponent = 1001 0100 = $(133)_{10}$

Actual Exponent of base = Exponent−127 = 133−127 = **6**

$$\text{Mantissa} = \frac{1}{2^4} + \frac{1}{2^7} + \frac{1}{2^{12}} = \frac{289}{4096}$$

$$= +\left[1 + \frac{289}{4096}\right] \times 2^6$$

$$= +\left[\frac{4385}{64}\right] = 68.515625$$

IDC 101: Introduction to computers

How to represent 68.515625 in 32 bit representation:

68 = 1000100
0.515625 = 0.100001
68.515625 = 1000100.100001

Convert to standard form by moving decimal to left…….

$68.515625 = 1.000100100001 \times 2^{6}$

Mantissa = **1**+0.000100100001 (1 is hidden bit)
Actual exponent (base) = Exponent – 127
Exponent = Actual exponent + 127
Exponent = 127+6 = $(133)_{10}$ = $(10000101)_{2}$

68.515625 in 32 bit floating point is represented as:

0 10000101 00010010000100000000000

IDC 101: Introduction to computers

How to represent 0.3125 in 32 bit representation:

0.3125 = 0.0101

Convert to standard form by moving decimal to right…….

$0.0003125 = 1.01 \times 2^{-2}$

Mantissa = **1**+0.01 (1 is hidden bit)

Actual exponent (base) = Exponent – 127

Exponent = Actual exponent + 127

Exponent = $-2+127 = (125)_{10} = (01111101)_2$

0.3125 in 32 bit floating point is represented as:

0 01111101 01000000000000000000000

IDC 101: Introduction to computers

- **Floating point number representation (64bit)**