

# NUMBER SYSTEM

## Introduction to Information and Communication Technologies

Dr. Muhammad Abdullah

---



Department of Data Science  
Faculty of Computing and Information Technology (FCIT)  
University of the Punjab, Lahore, Pakistan.

## Learning Objectives

1. Distinguish among categories of numbers
2. Describe positional notation
3. Convert numbers in other bases to base 10
4. Convert base-10 numbers to numbers in other bases
5. Describe the relationship between bases 2, 8, and 16
6. Explain the importance to computing of bases that are powers of 2

# Number System

- Mainly there are two types of number systems
  - Non-Positional Number System
    - Use fingers for counting. Use stones
    - Does the direction matter?
  - Positional Number System
    - Position matters

# Non-positional Number System

- Use symbols such as I for 1, II for 2, III for 3, etc
- Each symbol represents the same value regardless of its position in the number
- Difficult to perform arithmetic with such a number system

# Positional Number System

- Value of each digit is determined by:
  - The digit itself
  - The position of the digit in the number
  - The base of the number system

Base = total number of digits in the number system

# Decimal Number system (positional system)

## Name

- “decem” (Latin) => ten

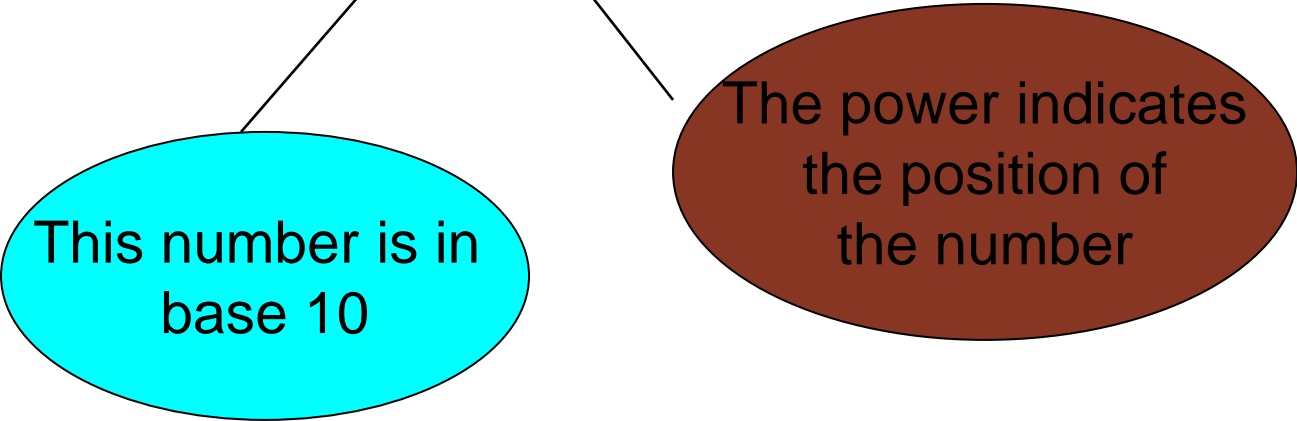
## Characteristics

- Ten symbols
  - 0 1 2 3 4 5 6 7 8 9
- Positional
  - $2945 \neq 2495$
  - $2945 = (2 \cdot 10^3) + (9 \cdot 10^2) + (4 \cdot 10^1) + (5 \cdot 10^0)$

# Positional Notation

642 in base 10 *positional notation* is:

$$\begin{aligned} 6 \times 10^2 &= 6 \times 100 = 600 \\ + 4 \times 10^1 &= 4 \times 10 = 40 \\ + 2 \times 10^0 &= 2 \times 1 = 2 \quad = 642 \text{ in base 10} \end{aligned}$$



This number is in  
base 10

The power indicates  
the position of  
the number

# Positional Notation

As a formula:

$$d_n * R^{n-1} + d_{n-1} * R^{n-2} + \dots + d_2 * R^1 + d_1 * R^0$$

R is the base  
of the number

n is the number of  
digits in the number

d is the digit in the  
 $i^{\text{th}}$  position  
in the number

$$642 \text{ is } 6 * 10^2 + 4 * 10 + 2 * 1$$



# Positional Notation

- What if 642 has a base of 13?

$$\begin{aligned} 6 \times 13^2 &= 6 \times 169 = 1014 \\ + 4 \times 13^1 &= 4 \times 13 = 52 \\ + 2 \times 13^0 &= 2 \times 1 = 2 \\ &= 1068 \text{ in base 10} \end{aligned}$$

- 642 in base 13 is equivalent to 1068 in base 10

# Binary Number system

## Name

- “binarius” (Latin) => two

## Characteristics

- Two symbols
  - 0 1
- Positional
  - $1010_B \neq 1100_B$

Most (digital) computers use the binary number system



## Terminology

- **Bit**: a binary digit
- **Byte**: (typically) 8 bits

# Binary counting

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



0	10	110	1110	11110	111110
1	11	111	1111	11111	111111

# Binary –Decimal Equal

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Decimal	Binary
16	10000
17	10001
18	10010
19	10011
20	10100
21	10101
22	10110
23	10111
24	11000
25	11001
26	11010
27	11011
28	11100
29	11101
30	11110
31	11111

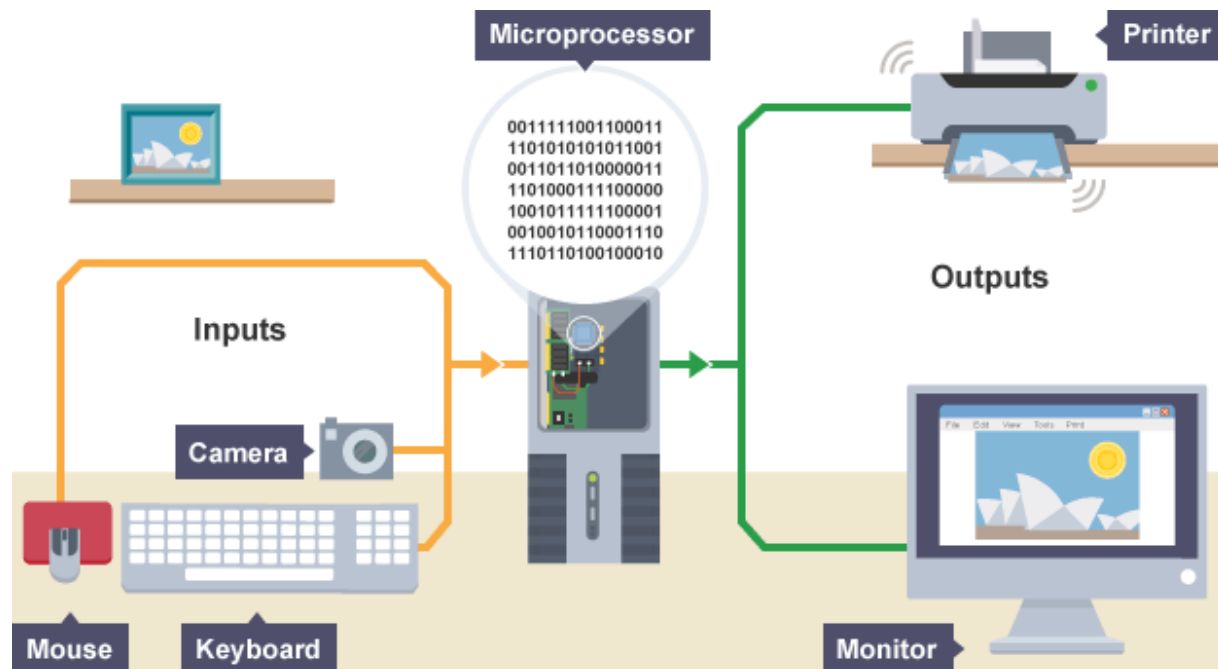
Decimal	Binary
32	100000
33	100001
34	100010
35	100011
36	100100
37	100101
38	100110
39	100111
40	101000
41	101001
42	101010
43	101011
44	101100
45	101101
46	101110
47	101111

# Why Binary in Computers?

- Binary positional systems have great advantages over decimals in electronic computing
  - Circuits in computers have billions of transistors that can turn on and off
- Two reasons computer use binary
  - Two clearly distinct states that provide a safe range for reliability.
  - Least amount of necessary circuitry, which results in the least amount of space, energy consumption, and cost.

# Why Binary in Computers?

- It is easy for machines to store electrical signals as **ON** or **OFF**
- Binary math is easier for computers than anything else
- Binary numbers help in digital electronic circuits using **logic gates**
  - Logic gates are the basic building blocks of any digital system
  - Common logic gates are AND, OR, NOT, XOR, NOR, NAND



# Binary

- Recall, in decimal each position of digital is multiplied case by base ten (10)
  - $273 = 2*10^2 + 7*10^1 + 3*10^0$
  - $543.21 = 5*10^2 + 4*10^1 + 3*10 + 2*10^{-1} + 1*10^{-2}$
  - Similarly binary equivalence is
  - $5 = 101 = 1*2^2 + 0*2^1 + 1*2^0 = 4 + 0 + 1$
  - $26 = 11010 = 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 16 + 8 + 0 + 2 + 0$

# Binary Number

- Binary number is generally much longer than its corresponding decimal number
  - $256,058 = 111\ 11010\ 00001\ 11010$
  - Reason: binary system has only two possibilities
  - In other words, binary digit carries less information than in decimal digit



# Octal Number System

- The numbers with base 8 are called octal
  - 0, 1, 2, 3, 4, 5, 6, 7
  - There is no such thing as 8 and 9
  - We count the same way as we do for decimal, but we stop at 7

# Octal-Decimal

Decimal	Octal
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

Decimal	Octal
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

Decimal	Octal
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 Number System

- We count the same way as we do for decimals, but we stop at 7
- Means, we will have 10 after 7.
- Similarly, 20 after 17
- $(35)_{10} = (43)_8$

$$(43)_8 = 4 \times 8^1 + 3 \times 8^0 = (4 \times 8) 32 + 3 = (35)_{10}$$

$$(605)_8 = 6 \times 8^2 + 0 \times 8^1 + 5 \times 8^0 = 6 \times 64 + 0 + 5 = 384 + 5 = (389)_{10}$$

# Why we are discussing Octal

- The fact that there is a short cut method to convert binary number into octal
- Method: make group of 3 digits start from right to left

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

It saves us from calculation of lot of terms of 0's & 1's individually and respective powers of 2's for calculation. See example:

$$10110101 = \underline{10} \underline{110} \underline{101} = (2 \ 6 \ 5)_8 = \underline{2 \times 8^2 + 6 \times 8^1 + 5 \times 8^0} = 2 \times 64 + 6 \times 8 + 5 \times 1 = 128 + 48 + 5 = (181)_{10}$$

$$1011010111 = \underline{1} \underline{011} \underline{010} \underline{111} = (1327)_8 = \underline{1 \times 8^3 + 3 \times 8^2 + 2 \times 8^1 + 7 \times 8^0} = 512 + 192 + 16 + 7 = (727)_{10}$$

# Hexadecimal Number System

- Hexadecimal numbers is also important in computers
- Hexadecimal has based 16 and basic symbols are
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - A is 10 and F is 15

# Decimal-Hexadecimal

Decimal	Hexa
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

Decimal	Hexa
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

Decimal	Hexa
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

# Why Hexadecimal?

- $(43)_{16} = 4 \times 16^1 + 3 \times 16^0 = 64 + 3 = (67)_{10}$
- $(A5B)_{16} = A \times 16^2 + 5 \times 16^1 + B \times 16^0 = 10 \times 256 + 5 \times 16 + 11 = 2560 + 80 + 11 = (2651)_{10}$

○ There is a shortcut method to convert binary into hexadecimal

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

# Decimal to Any Base

For converting a given decimal number to any base, we need to divide the number with the target base. Lets see how to convert from Decimal to Binary

$$\begin{array}{r} 2 \overline{) 29} \\ 2 \overline{) 14} \\ 2 \overline{) 7} \\ 2 \overline{) 3} \\ 2 \overline{) 1} \\ 0 \end{array}$$

Remainders

1    LSB

0

1

1

1    MSB

Read the remainders  
from the bottom up

29 decimal = 11101 binary



# Decimal to Hexadecimal

$$(423)_{10} = ( ? )_{16}$$

# Any Base to Decimal?

We can express any given **n** digit number of any **base** as

$$d_n * R^{n-1} + d_{n-1} * R^{n-2} + \dots + d_2 * R^1 + d_1 * R^0$$

$$(642)_{10} = 6 * 10^2 + 4 * 10^1 + 2 * 10^0$$

$$(1011)_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$

$$(A12)_{16} = A * 16^2 + 1 * 16^1 + 2 * 16^0$$

# Binary to Hexa and Hexa to Binary?

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

$$(BC11)_{16} = ( \quad )_2$$

$$(1110011101)_2 = ( \quad )_{16}$$

# How many numbers can be represented using different number of bits?

1 bit

2 bits

3 bits

4 bits

# Computer Arithmetic

# Binary Addition

- It's a key for binary subtraction, multiplication, division
- Four Rules

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

# Binary Addition

## ○Example

$$0011010 + 001100 = 00100110$$

$$\begin{array}{rcccccccc} & & 1 & 1 & & & & \text{carry} \\ & 0 & 0 & 1 & 1 & 0 & 1 & 0 = (26)_{10} \\ + & 0 & 0 & 0 & 1 & 1 & 0 & 0 = (12)_{10} \\ \hline & 0 & 1 & 0 & 0 & 1 & 1 & 0 = (38)_{10} \end{array}$$

# Binary Addition

## ○Example

$$\begin{array}{r} 00010011 + 00111110 = 01010001 \\ \begin{array}{ccccccccc} & & & 1 & 1 & 1 & 1 & 1 & \text{carry} \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & = 19_{10} \\ + 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & = 62_{10} \\ \hline 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & = 81_{10} \end{array} \end{array}$$



# Binary Subtraction

- It's a key for binary subtraction, multiplication, division
- Four Rules

Case	A - B	Subtract	Carry
1	0 - 0	0	0
2	1 - 0	1	0
3	1 - 1	0	0
4	0 - 1	1	1

# Binary Subtraction

## ○Example

$$00100101 - 00010001 = 00010100$$

0 1

borrows

$$00100101 = 37_{10}$$

$$- 00010001 = 17_{10}$$

-----

$$00010100 = (20)_{10}$$

# Binary Subtraction

## ○ Example

$$00110011 - 00010110 = 00011101$$

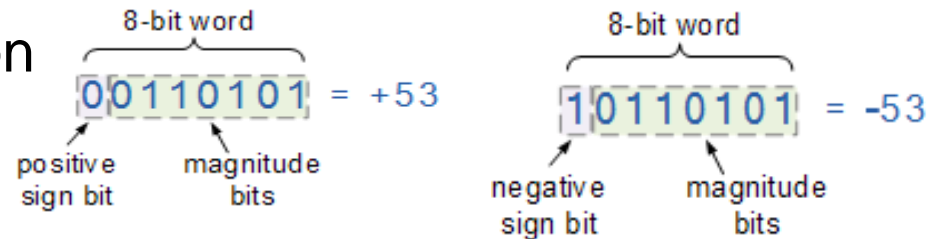
				1	1	1	1			borrows
	0	0	<del>1</del>	<del>1</del>	0	0	1	1	=	51 <sub>10</sub>
-	0	0	0	1	0	1	1	0	=	22 <sub>10</sub>
-----										
	0	0	0	1	1	1	0	1	=	29 <sub>10</sub>

# Issues in Binary Subtraction

- The classical subtraction (discussed above)
  - Works when we subtract small number from large number (27-9)
  - However, it does not work when we subtract a large number from small number
- Alternatively, we use two methods
  - 1's complement
  - 2's complement

# Negative Binary Number Representation

- Sign-magnitude Representation



- One's Complement: Flip the bits to get the negative number representation

$$+53 = 00110101$$

$$-53 = 11001010$$

- Two's Complement: Find One's complement and then add 1 to get the negative number representation

$$+53 = 00110101$$

$$-53 = 11001011$$

# Issues in Binary Subtraction

- 1's complement is obtained by replacing 0's with 1's and 1's with 0's
  - For example, 1's complement of 10110 is 01001
- 2's Complement
  - Write the number in binary (say, 10001)
  - Take 1's complement (01110)
  - Add 1 to it (01111)

# Subtraction using 2's complement

- 2's Complement

- First of all 2's complement of the subtrahend (In  $a-b$ ,  $b$  is subtrahend) is found
- Then it is added to minuend (In  $a-b$ ,  $a$  is minuend)
- If the final carry over of the sum is 1, it is dropped and the result is positive
- If there is no carry, the 2's complement of the sum will be the result and it is negative

Signed-magnitude  
number for adding two  
number (+3 and -3) using  
4 bits representation

Two's Complement for adding +3  
and -3

One's Complement for  
adding +3 and -3