Digital System Design

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective

- Brief Introduction

Logic circuit

Digital devices

Digital System Design

Transistors

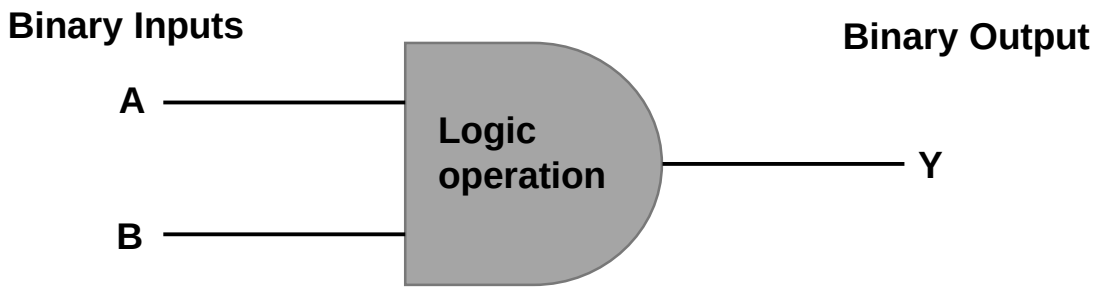
Electrons

Why do we study Digital System Design?

- Digital design deals with the design of digital electronic circuits.
  e.g., microprocessor, RAM, ALU …
- Digital circuits operate using digital signals (usually use discrete value) in contrast to analog signals (continuous range of values).
- Digital systems use **binary logic** to store, process and manipulate data.

Binary Logic

- **Binary logic or Boolean logic** deals with binary variables (input and output) and logical operations.
- Binary variables are represented using alphabets A, B, x, y…
- Logical operations are functions applied to one or more binary input and produce a single binary output



1	0
TRUE	FALSE
HIGH	LOW
5V	0V

Logical operators

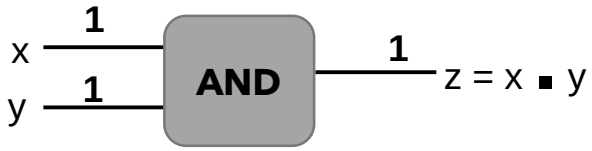
Binary variables : x , y, z with values 1 or 0

AND

Represented by dot (▪)

x ▪ y = z read as “x AND y equal to z “

z = 1 iff x = 1 and y = 1;
otherwise, z = 0

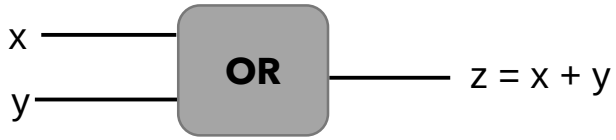


OR

Represented by plus(+)

x + y = z read as “ x OR y equal to z”

z = 1 if x = 1 or y = 1 otherwise, z = 0

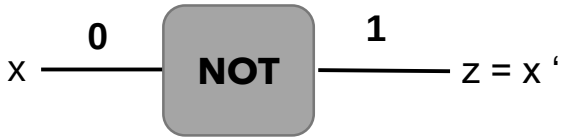


NOT

Represented by prime (‘)

x ’ = z read as “ not x is equal to z”

z = 1 if x = 0

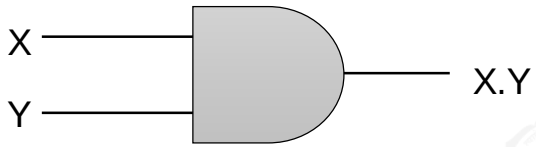


Logic Gates

- We can realize each of the Boolean function using **logic gates**.
- Logic gates are the building blocks of all circuits in a computer.
  - Produces an output based on one or more inputs.
  - Some basic logic gates are AND, OR, NOT, NAND, NOR etc.
- We can represent the logical operations using truth table, Boolean expression, venn diagram, switching circuit.

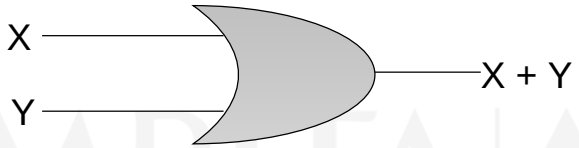
Logic gate

AND



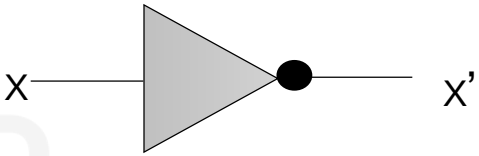
INPUTS		OUTPUT
X	Y	X.Y
0	0	0
0	1	0
1	0	0
1	1	1

OR



INPUTS		OUTPUT
X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT



INPUTS	OUTPUT
X	X'
0	1
1	0

Design a Digital System

- Understand the real-world problem.

  e.g., If out of 2 door switch (M1 and M2) any one is pressed a doorbell should be ringed.

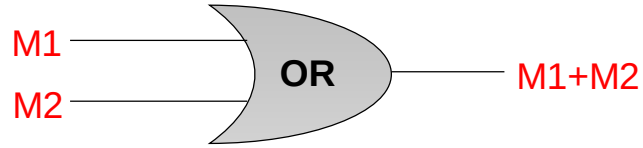
- Understand the truth table or logic behind the problem

M1	M2	Op
0	0	0
0	1	1
1	0	1
1	1	1

- Represent the truth table using a Boolean expression

  Op = M1 + M2

- Design the digital circuit i.e., hardware implementation



Summary

- We discussed about the basic concepts of digital logic.

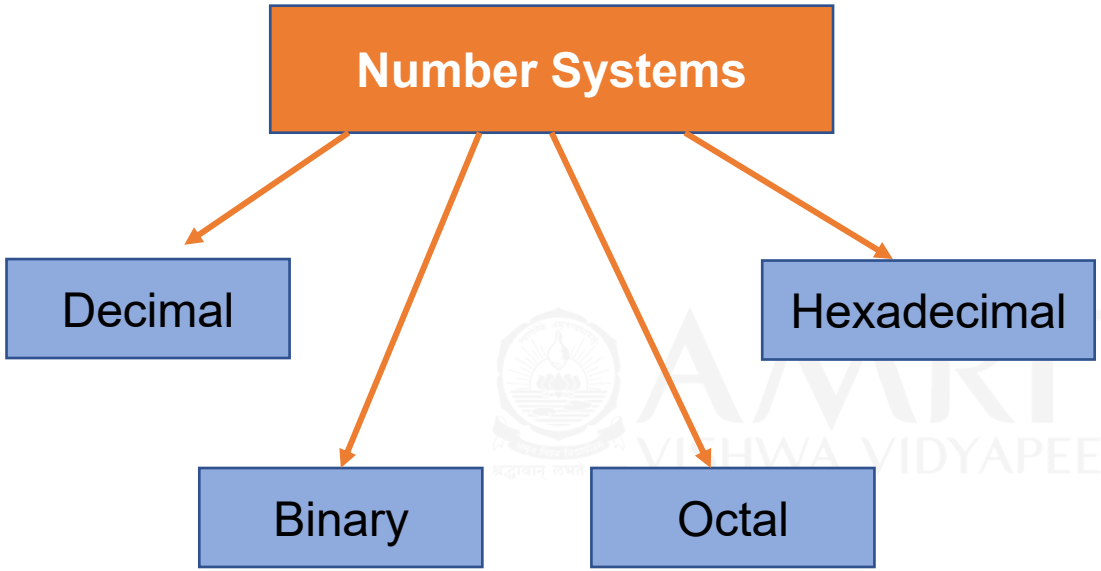
AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

Number Systems

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective



Introduction

- Digital systems including computers run on a 2-valued system called **binary system**.
- Transistors are the basic elements in digital systems with ON and OFF states.
- The binary system has only two digits **0** (represents **OFF**) and **1**(represents **ON**) called **bi**nary digi**t** or **bit**.
- Computer translates everything into binary numbers as they can understand only binary numbers.

Number Systems

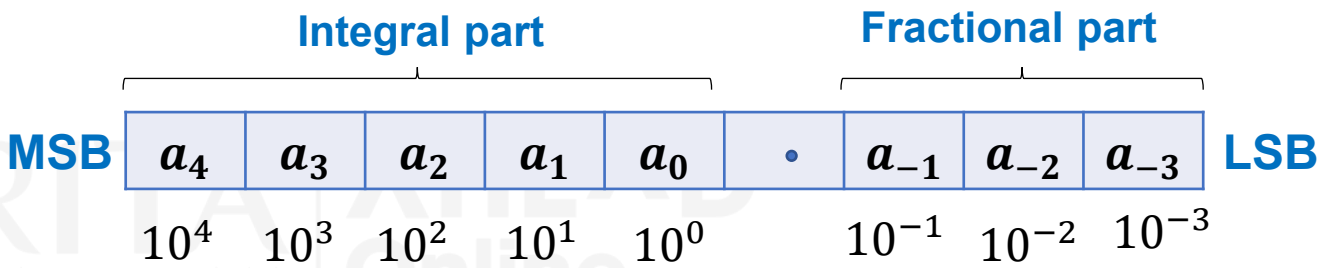
- Number system is the method to represent numbers.
- Basically, using decimal number system but introducing **binary**, **octal** and **hexadecimal** number systems.
- The base of each number system also called radix determines how many different symbols are needed to represent a number.

Number System	Base	Digits
Decimal	10	0,1,2,3,4,5,6,7,8,9
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Decimal Number System

- Referred to as base-10 with ten digits 0,1,2,3,4,5,6,7,8,9
- Number representation

$$a_4\ a_3\ a_2\ a_1\ a_0 \bullet a_{-1}a_{-2}a_{-3}$$



$$= 10^4\,a_4 + 10^3 a_3 + 10^2 a_2 + 10^1 a_1 + 10^0 a_0 + 10^{-1} a_{-1} + 10^{-2} a_{-2} + 10^{-3} a_{-3}$$

$10^2\ 10^1\ 10^0$

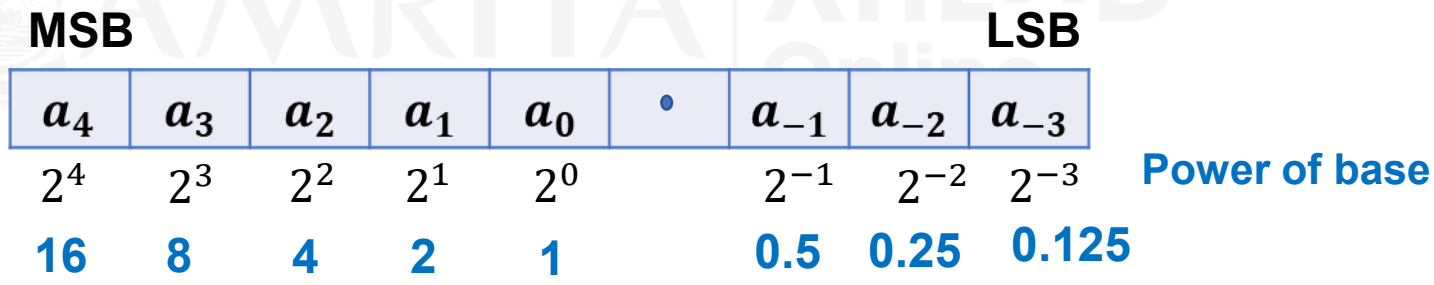
e.g., $(7\ 6\ 5)_{10} = 7\text{x}\ 10^2 + 6\text{x}\ 10^1 + 5\text{x}\ 10^0$

$= 7\text{x}100 + 6\text{x}10 + 5\text{x}1$

$= 765$

Binary Number System

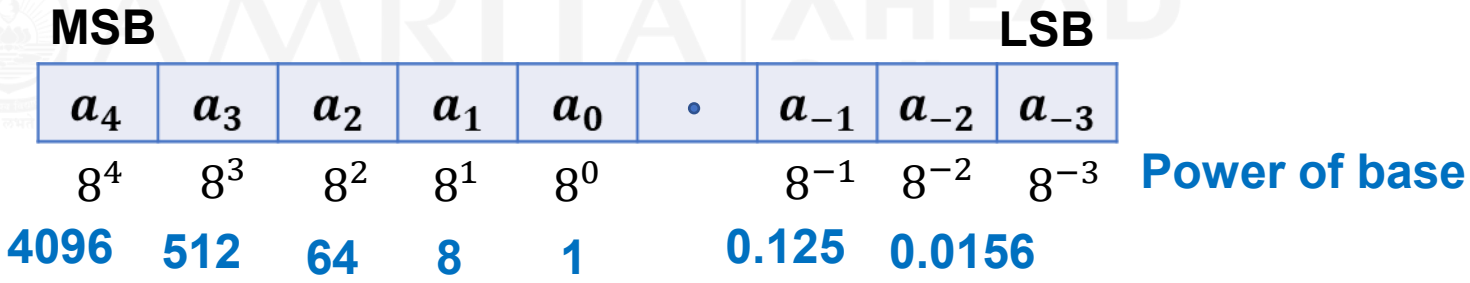
- Referred to as base-2 with two digits 0,1.
- Coefficient 'a_j' is the digit between 0 to base-1 i.e., in binary coefficient are a_j = 0 and 1.



- Example : $(11011.11)_2$, $(1001)_2$
- $(1)_{10} = (1)_2$, $(2)_{10} = (10)_2$, $(16)_{10} = (10000)_2$, $(22)_{10} = (10110)_2$

Octal Number System

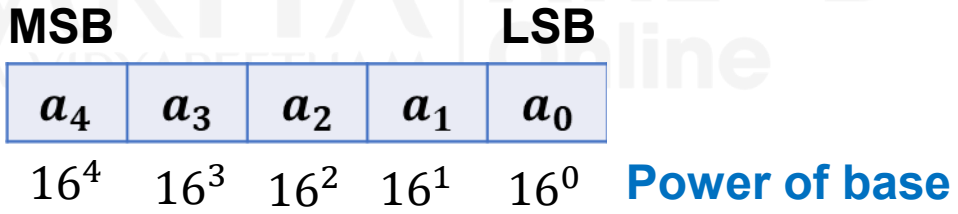
- Base-8 number system with eight digit to represent numbers
- The coefficients are one of the eight digits (0,1,2….,7) i.e., between 0 to (8-1)



- Example : $(2765)_8$, $(1435)_8$
- $(7)_{10} = (7)_8$, $(8)_{10} = (10)_8$, $(128)_{10} = (200)_8$

Hexadecimal Number System

- Base-16 number system with 16 digits for number representation.
- First ten digits are borrowed from the decimal system and letters A, B, C, D, E, F are used for digits 10, 11, 12, 13, 14, 15.



- Example : $(C74F)_{16}$, $(93AC)_{16}$

Counting in Number System

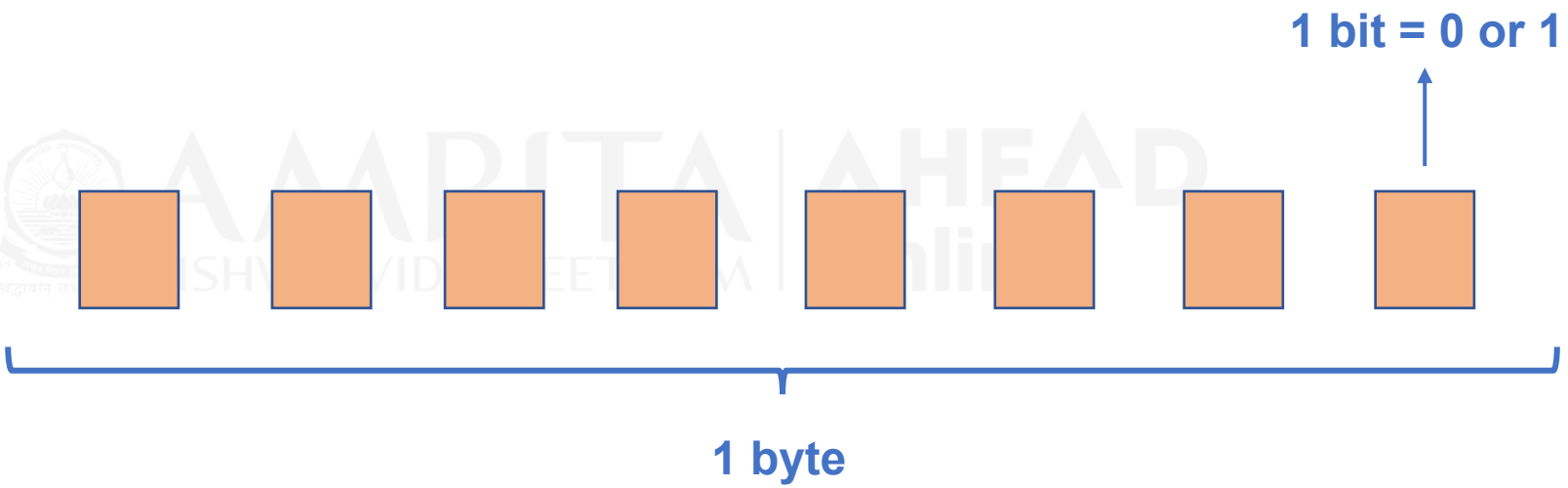
Decimal	Binary Number	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C

Bit and Byte

1 bit = single binary digit (0 or 1)

1 bit = 0

1 byte = 8 bit



- Can represent numbers **0** (00000000) to **255** (11111111)

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

Why binary not decimal?

- Transistors are the fundamental unit in digital computer with only 2 states ON and OFF.
- In decimal system : 4 transistor can represent only maximum 4 numbers by keeping all ON.
- With binary : 4 transistor can represent any number up to 15 i.e., we can represent up to 16 values from 0 to 15.

Summary

- Explains about different number systems for representing information's

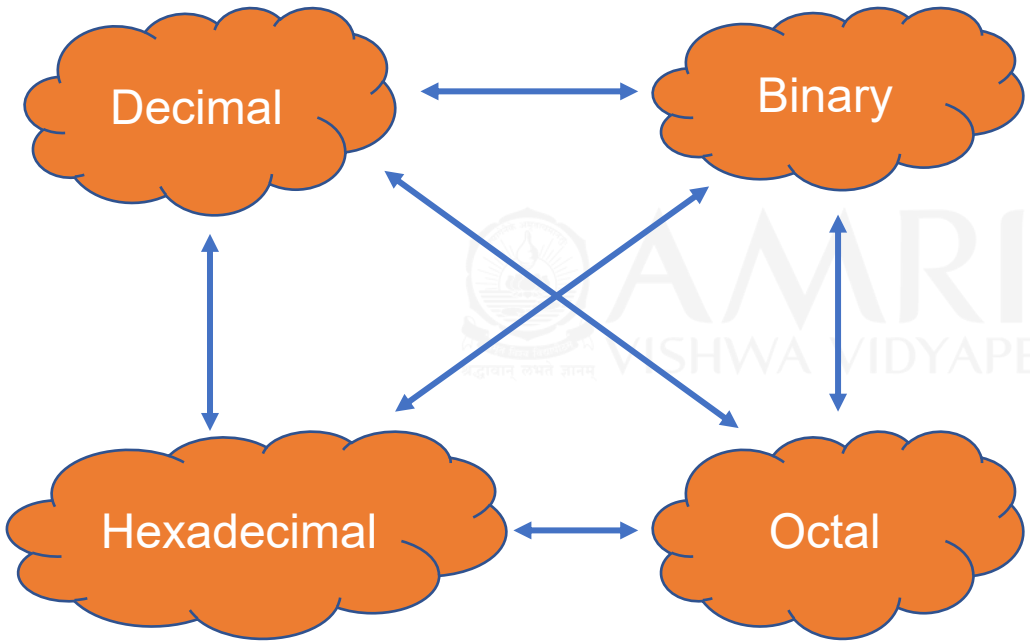
Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

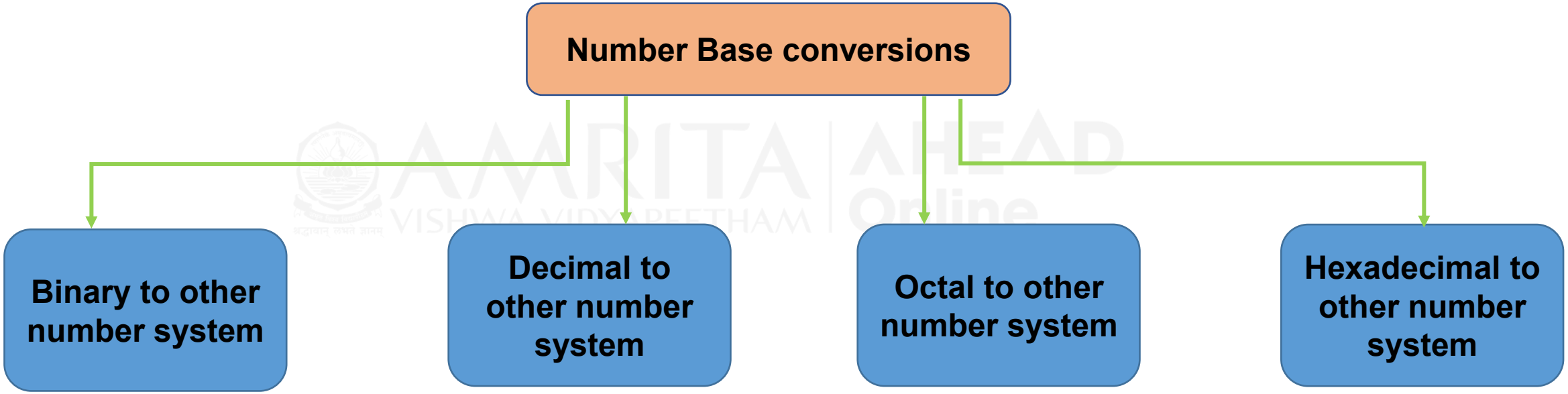
Number System Conversions

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective



Introduction



Number base conversions

➢ Converting from any number system to decimal system

- If the base ‘r’ and the coefficients are given, then any number can be converted to decimal equivalent by multiplying each coefficient with the corresponding power of r and adding.

- $a_n * r^n + a_{n-1} * r^{n-1} + \ldots\ldots + a_1 * r^1 + a_0 + a_{-1} * r^{-1} \ldots\ldots a_{-m} * r^{-m}$

Example

2^{nd} position - 2^2

- $(1101)_2$ to decimal

0^{th} position - 2^0

$= 1*2^3 + 1* 2^2 + 0* 2^1 + 1* 2^0$

$= 8 + 4 + 0 + 1 = (13)_{10}$

- $(630.4)_8$ to decimal

1^{st} position - 8^1

$= 6*8^2 + 3* 8^1 + 0* 8^0 + 4* 8^{-1}$

$= 384 + 24 + 0 + 0.5 = (408.5)_{10}$

- $(C76F)_{16}$ to decimal

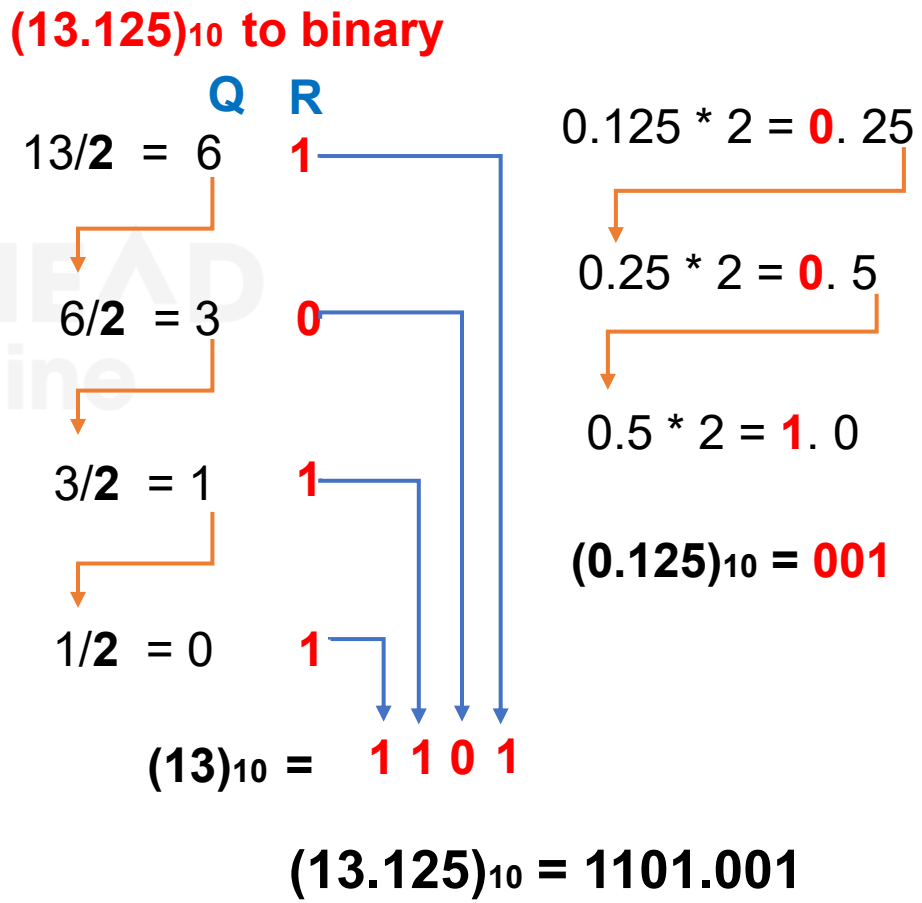
$= 12*16^3 + 7* 16^2 + 6* 16^1 + 15* 16^0$

$= 49{,}152 + 1{,}792 + 96 + 15 = (51055)_{10}$

Number base conversions

➢ Convert decimal numbers to other number systems.

- Divide the decimal number repeatedly with the base until no longer possible to divide.
- Divide the quotient and note the remainder.
- Remainders of the division put together in reverse order is the result.
- For fractional part, multiply the decimal fraction with base repeatedly until we reach zero or close to zero. The integer part together is the result of decimal fraction.



Example

- $(25.13)_{10}$ to binary

25/2 quotient = 12 ; remainder = **1**
12/2 quo = 6 ; rem = **0**
6/2 quo = 3 ; rem = **0**
3/2 quo = 1 ; rem = **1**
1/2 quo = 0 ; rem = **1**

hexadecimal

Fractional Part binary equivalent
0.13 * 2 = **0**.26
0.26 *2 = **0**.52
0.52 *2 = **1**.04

$(25.13)_{10}$ = 11001 + .001
= **$(11001.001)_2$**

$(153)_{10}$ to octal

153/8 quo = 19 ; rem = **1**
19/8 quo = 2 ; rem = **3**
2/8 quo = 0 ; rem = **2**

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

$(2545)_{10}$ to

2545/16 quo = 159 ; rem = **1**
159/16 quo = 9 ; rem = 15 (**F**)
9/16 quo = 0; rem = **9**

$(2545)_{10}$ = **$(9F1)_{16}$**

Number base conversions

➢ Octal and Hexadecimal numbers

- Octal digit corresponds to three binary digit (111 = 7 is the maximum range of octal coefficient)
- $(256)_8$ to binary = 2 5 6 = $(010\ 101\ 110)_2$

  010 101 110
- $(1110.0101)_2$ to octal = 001 110 . 010 1 = $(16.21)_8$

  1 6 2 1

Number base conversions

- Hexadecimal digit corresponds to four binary digits.
- Binary to hexadecimal is by partitioning the binary number into groups of four digits each.
- $(10\ 1100\ 0110\ 1011.1111\ 0010)_2$ $= (2C6B.F2)_{16}$

  2 C 6 B F 2
- $(306.D)_{16}$ = 3 0 6 . D= $(0011\ 0000\ 0110.1101)_2$

  0011 0000 0110 1101

Summary

- Explained about the number system conversions

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

Binary Operations

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective

- Explains how to perform various operations in binary number system.

Binary Addition

- Binary addition obeys four basic rules :

```
  0        0        1        1
 +0       +1       +0       +1
---      ---      ---      ---
  0        1        1       1 0
```

0 with carry 1

7 5 = 12

➢ Add 111 and 101

```
   1 1
   1 1 1
 + 1 0 1
 -------
 1 1 0 0
```

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Binary addition rules

Binary Subtraction

- Binary subtraction obeys four basic rules :

0	0	1	1
- 0	- 1	- 0	- 1
0	1 with borrow 1	1	0

4 3 1

➢ Subtract 100 and 011

```
    1
  0 10 10
  1 0 0
- 0 1 1
-------
  0 0 1
```

Binary multiplication

- Binary multiplication obeys the four basic rules :

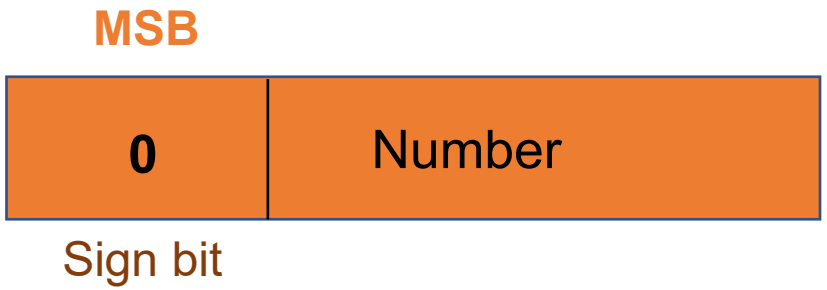
0 x 0 = 0 1 x 0 = 0 0 x 1 = 0 1 x 1 = 1

- Multiply binary numbers 101 x 11

```
    1 0 1
      1 1
-----------
    1 0 1
  1 0 1
-----------
  1 1 1 1
```

Binary representation of Positive numbers

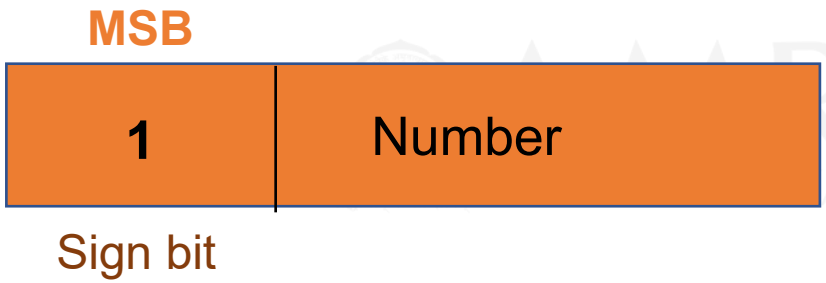
- No sign bit in unsigned binary numbers so it can only represent its magnitude.
- Binary numbers having MSB **0** are called "Positive signed binary numbers".



Positive Number	Binary equivalent
0	000
1	001
5	0101
3	011

Binary representation of Negative numbers

- The binary number with their MSB **1** is called "Negative signed binary number".



Negative Number	Binary equivalent
0	1000
1	1001
5	1101
3	1011

- Signed number representation : Sign Magnitude

  1's complement

  2's complement

Sign Magnitude

- Most significant bit (bit with greatest value in the left most column) is used to store the sign (positive or negative) of number.

Sign is 1: Negative
Sign is 0 :Positive

- Assume a 4-bit representation,

+5 = **0** 101

Sign bit

magnitude

-5 = **1** 101

Sign bit

magnitude

1's Complement

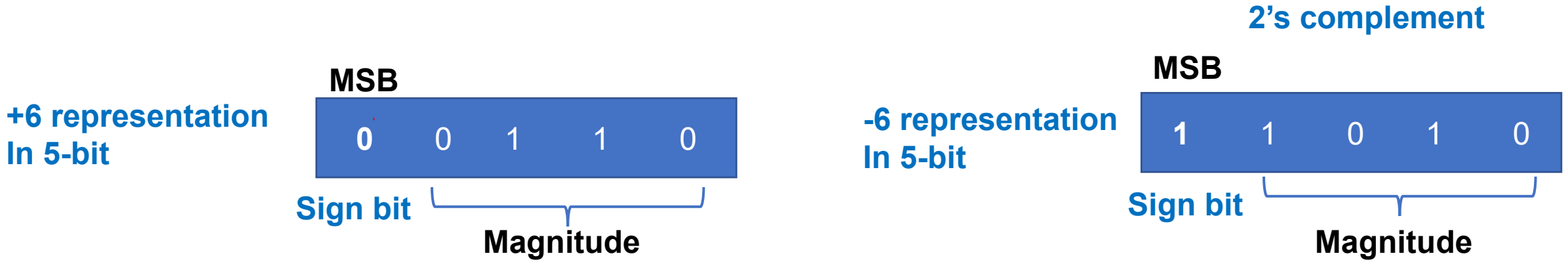
- Negative numbers can be represented using 1's complement.
- By inverting each bit of a number, we can obtain the 1's complement of a number (change all 0's to 1 and all 1's to 0).
- **1's complement** of $(11010.1101)_2 = (00101.0010)_2$
- e.g. $(100110.1001)_2 = (011001.0110)_2$

2's complement

- Find the 1's complement of the binary number and then add 1 to the least significant bit .

- 2's complement of $(1011001)_2$

Find 1's complement : $(0100110)_2$
Add 1 to LSB : $(0100111)_2$



Summary

- Explained about Binary operations like addition, subtraction and complements.

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

Boolean Algebra

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective

- Boolean Algebra

Introduction

- Digital inputs and outputs are binary values i.e., 0 and 1.
- Binary operations performed by digital circuits are called logical functions or operations.
- Logic gates are the implementations of the logical operations.
- Multiple logic gates can be connected from output to input to design complex circuits.
- **Boolean algebra** can be used to reduce or simplify the complex circuits.

Boolean Algebra

- Mathematics of logic design, introduced by **George Boole** in 1854.
- **Boolean algebra** deals with logical operations on binary variables with a set of laws or rules.
- **Laws of Boolean algebra** can be used to reduce the number of logic gates by reducing the Boolean expressions.
- One application of Boolean algebra is in the analysis and design of digital systems.

Boolean Function

- Expression formed with binary variables, logical operators , parentheses and an equal sign.

  e.g., **F (x, y, z) = x + y.z'**

- Function can be either 1 or 0 based on the binary variable value.
- In the above e.g., F = 1 if x = 1 *and* y = 1*and* z' = 1 otherwise F = 0.
- Function can be represented as a **Boolean expression** (same as e.g.,)or in a ***truth table*** *or using* ***logic diagram****.*
- Operator Precedence order: Parentheses, NOT, AND , OR

Truth Table

- Tabular representation of a Boolean Function.
- A truth table shows all the possible inputs and outputs of a function.
- A function with n variables has 2^n possible combinations of inputs.
- **Truth table of AND, OR, NOT with 2 inputs**

2 inputs - 2^2 = 4 outputs

inputs		output		
x	**y**	**x.y**	**x +y**	**x'**
0	0	0	0	1
0	1	0	1	
1	0	0	1	0
1	1	1	1	

$F(x, y) = 2^2 = 4$

2 2

$F(x, y, z) = 2^3 = 8$

2 2 2

Logic Circuit

- Logic circuit is the hardware implementation of Boolean function using the logic gates.

- **Literals** – a variable or its complement in the function.

  e.g., F(x,y,z) = x.y.z' 3 literals x, y, z'

  F(A,B,C) = A'.C + B'.C 4 literals A', C, B', C

  term

- Each **literal** in the function is an input to a logic gate and each **term** is implemented with the logic gate.

- Minimizing the number of literals and terms results in a simpler circuit by reducing the number of logic gates.

Laws of Boolean Algebra

THEOREMS	AND	OR
Identity law	A . 1 = A	A + 0 = A
Annulment law	A . 0 = 0	A + 1 = 1
Idempotent law	A . A = A	A + A = A
Complement law	A . A’ = 0	A + A’ = 1
Commutative law	A . B = B . A	A + B = B+ A
Associative law	A.(B.C) = (A.B).C	A+(B+C) = (A+B)+C
Distributive law	A.(B+C) = A.B+A.C	A+(B.C) = (A+B).(A+C)
Absorption law	A.(A+B) = A	A+(A.B) = A
DeMorgans law	(A.B)’ = A’ + B'	(A+B)’ = A’ . B'
Double Negation	$\overline{\overline{A}} = A$	

Duality & Complement of a Function

Duality Principle : Interchange binary operator and identity element

Imagine a Boolean function of 2 variables, **F(A,B) = A'.B + A.B'**

$$F_{dual} = (A' + B).(A + B')$$

AND : convert to OR , OR : convert to AND
1: convert to 0 , 0: convert to 1
No change to variables

Complement : derive using DeMorgans law

$$\mathbf{F'} \text{ or } \bar{F} = (A + B')(A' + B)$$

AND : convert to OR , OR : convert to AND
1: convert to 0 , 0: convert to 1
Complement all variables

Simplify the expressions?

- **x + xy = x**

  x + xy = x.1 + xy ---------- identity law, x.1 =x

  = x.(1 + y) ---------- distributive law

  = x. 1 ---------- annulment law , y+1=1

  = **x** ---------- identity law, x.1 =x

x	y	xy	x+xy
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

- **x + x'y = x+y**

  x + x'y = x + xy +x'y ---------- absorption law, x+xy = x

  = x + y(x + x') ---------- distributive law

  = x + y.1 ---------- complement law, x+x' = 1

  = **x + y** ---------- identity law, y.1 = y

Summary

- Basics of logic design called Boolean Algebra.
- Laws and rules to define the operations.

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition