INTRODUCTION TO NUMBER SYSTEM

LET ME CHECK IT OUT!!

DWUMFOUR ABDULLAI ABDUL-AZIZ

NUMBER SYSTEM

- Number system is a set of values used to represent different quantities
- The total number of digits used in a number system is called its base or radix.
- The base is written after the number as subscript
- Example: $number_{radix}$

NUMBER SYSTEM

Some important number systems are as follows.

- Decimal number system
- Binary number system
- Octal number system
- Hexadecimal number system

The decimal number system is used in general. However, the computers use binary number system.

The octal and hexadecimal number systems are also used in the computer.

DECIMAL NUMBER SYSTEM

- An unsigned integer number A can be represented using n digits in base b:

$$A=(a_{n-1}a_{n-2}\ldots a_2\ a_1\ a_0\ .a_{-1}a_{-2}\ldots a_{-n})_b$$

- This representation is called positional representation
- Each number in this system consists of digits which are located at different positions.
- The position of first digit towards left side of the decimal point is 0.

- The position of second digit towards left side of the decimal point is 1
- Similarly, the position of first digit towards right side of decimal point is -1
- The position of second digit towards right side of decimal point is -2 and so on.
- Decimal number system start from 0-9

- Given n digits number A, in base *b*, the number of possible values that can be addressed is given by:

  $N = b^n$

- largest value that can be addressed A_{max}, is given by

$$A_{\max} = b^n - 1$$

- For example, the largest number that can be obtained using 3 digits in decimal is given by

$$A_{\max} = 10^3 - 1$$
$$= 1000 - 1$$
$$= 999$$

Thus we have the values in the range [000:999]

- 4 bits in base 2 is given by

$$A_{\max} = 2^4 - 1$$
$$= 16 - 1$$
$$= 15$$

- In this case, decimal numbers ranging from 0 to 15 (corresponding to binary 0000 to 1111) can be represented.

NUMBER SYSTEM

- The decimal value of the integer number A is given by

$$A = \sum_{i=0}^{n-1} a_i * b^i$$

e.g. A=451 can be represented in decimal as

$4*10^2+5*10^1+1*10^0$

4*100 +5*10 +1*1

400+50+1=451

NUMBER SYSTEM

The above formula can be generalized as:

$$X = \sum_{i=-m}^{k-1} x_i * b^i = x_{k-1}b^{k-1} + x_{k-2}b^{k-2} + ... + x_1 b^1 + x_0 b^0 + x_{-1}b^{-1} + ... + x_{-m}b^{-m}$$

Example: given the decimal number 65.375 can be written as

$$X = 6*10^1 + 5*10^0 + 3*10^{-1} + 7*10^{-2} + 5*10^{-3}$$

Example: The weights and positions of each digit of the number 542 are as follows:

Face value	5	4	2
Position	2	1	0
Weight	10^2	10^1	10^0

The above table indicates that:

The value of digit 5 = $5x10^2$ = 500

The value of digit 4 = $4x10^1$ = 40

The value of digit 2 = $2x10^0$ = 2

The actual number can be found by adding the values obtained by the digits as follows:

$500 + 40 + 2 = 542_{10}$

To do:

Find the weight presentation of each of the following numbers and compute the actual number

1. 6877
2. 357.74
3. 23.375

BINARY NUMBER SYSTEMS

- Digital computer represents all kinds of data and information in the binary system.
- Binary Number System consists of two digits 0 and 1.
- It is also referred to as base 2.
- Each bit in binary number system can be 0 or 1.

BINARY NUMBER SYSTEMS

An n-bit binary number b= $b_{n-1}\, b_{n-2} \ldots b_1 b_0$ can represent 2^n different values.

Call b_{n-1} the most significant bit (msb), b_0 the least significant bit (Lsb).

Example:

2-bit register can represent 4 different values (2^2)

3-bit register can represent 8 different values(2^3)

2-bit presentation:*10, 01, 11, 00*

3-bit presentation:*000, 001, 010, 100,101, 110, 011, 111*

OPERATIONS NUMBER SYSTEM

Binary Operations

Addition

a. $1100_2+0110_2=?$

solution

1100_2

0110_2

10010_2

b.$101110_2+1101_2=?$

101110_2

001101_2

111011_2

Subtraction:

c. 1100-0110=?

1100

0110

d. 11*10=?

11_2

10_2

00_2

$11\_\_$

110_2

EXERCISE

Compute the following

1. 10101_2+1100_2
2. $0100111_2+110011_2$
3. 1101_2-1010_2
4. 0101_2-100_2
5. 1000_2-111_2
6. 100_2*011_2
7. 1101_2*101_2
8. 10110_2*110_2

OCTAL NUMBER SYSTEM

- Octal number system has a base or radix 8.
- Eight digits are used : 0, 1, 2, 3, 4, 5, 6, 7

Addition of Octal Number

$(162)_8 + (537)_8$

Solution:

```
  1 1        <---- carry
  1 6 2
  5 3 7
-------
  7 2 1
-------
```

Therefore, sum = 721_8

OCTAL NUMBER SYSTEM

$(136)_8 + (636)_8$

Solution:

```
      1        <---- carry
    1 3 6
    6 3 6
  -------
    7 7 4
  -------
```

Therefore, sum = 774_8

OCTAL NUMBER SYSTEM

$(25.27)_8 + (13.2)_8$

Solution:

```
   1              <---- carry
   2 5 . 2 7
   1 3 . 2
  ---------
   4 0 . 4 7
 ---------
```

Therefore, sum = $(40.47)_8$

OCTAL NUMBER SYSTEM

Compute the following:

1. $(67.5)_8 + (45.6)_8$
2. $(7563)_8 + (6176)_8$
3. $(36.75)_8 + (47.64)_8$

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

A

Sum

B

SUBTRACTION OF OCTAL NUMBERS

The subtraction of octal numbers follows the same rules as the subtraction of numbers in any other number system.

The only variation is in borrowed number. In the decimal system, you borrow a group of 10_{10}.

In the binary system, you borrow a group of 2_{10}.

In the octal system you borrow a group of 8_{10}.

$(456)_8 - (173)_8$

Solution:

$$\begin{array}{rl} & {\scriptstyle 3\ 8} \quad \text{<---- borrow} \\ & 4\ 5\ 6_8 \\ - & 1\ 7\ 3_8 \\ \hline & 2\ 6\ 3_8 \end{array}$$

Compute

1. $(321)_8$-$(47)_8$
2. $(570)_8$-$(453)_8$

HEXADECIMAL NUMBER SYSTEM

- The Hexadecimal Number System consists of 16 digits from 0 to 9 and A to F
- Thus: Uses 10 digits and 6 letters 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
- Letters represents numbers starting from 10. A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.
- Also called base 16 number system.

- The first position from the right in a hexadecimal number represents a 0 power of the base (16).
- Example 16^0
- Last position in a hexadecimal number represents an x power of the base (16).
- Example 16^x where x represents the last position - 1.

Example

$13A_{16}=1*16^2+3*16^1+A*16^0$

$C6F8_{16}=C*16^3+6*16^2+F*16^1+8*16^0$

HEXADECIMAL OPERATIONS

Addition in hexadecimal

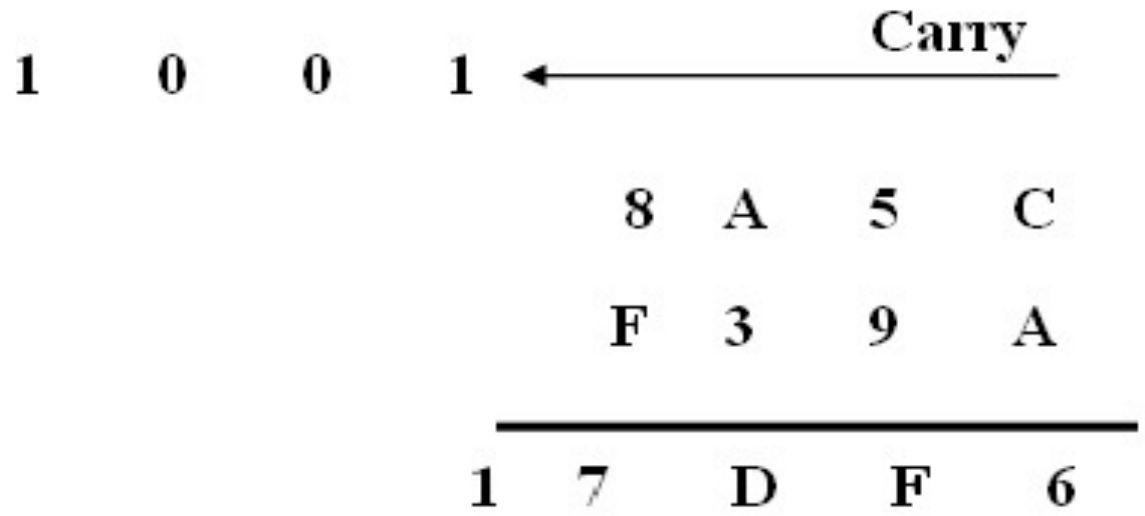
1-->$B_{16}+4_{16}=F_{16}$

2-->$C_{16}+D_{16=}19_{16}$

3-->$A3_{16}+B5_{16}=158_{16}$

4-->$4A6_{16}+1B3_{16}=659_{16}$

5-->$(8\ A\ 5\ C)_{16}$ and $(F\ 3\ 9\ A)_{16}$



HEXADECIMAL OPERATIONS

Subtraction in hexadecimal

1-->F-A=5

2-->AB-2C=7F

3-->1586_{16}-243_{16}=1343_{16}

4-->5CD2-2A0=5A32

5→4A6-1B3=?

6→ABC_{16}-$A3B_{16}$=?

HEXADECIMAL TABLE

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

RADIX CONVERSION

A radix conversion algorithm is used to convert a number representation in a given radix, r1, into another representation in a different radix, r2.

The following conversion schemes are available

- Binary to decimal
- Decimal to binary
- Octal to decimal
- Decimal to Octal
- Hexadecimal to decimal
- Decimal to hexadecimal
- Binary to Octal
- Binary to hexadecimal

DECIMAL CONVERSION

How do you understand this joke???

They say there are only 10 people in this world: those that understand binary and those that don't.

DECIMAL CONVERSION

If you don’t get that joke, you'll need a method to convert from binary to decimal.

DECIMAL CONVERSION

In order to convert from decimal to other bases

- *Step 1* – Divide the decimal number to be converted by the value of the new base.
- *Step 2* – Get the remainder from Step 1 as the rightmost digit (least significant digit) of new base number.
- *Step 3* – Divide the quotient of the previous divide by the new base.
- *Step 4* – Record the remainder from Step 3 as the next digit (to the left) of the new base number.
- Repeat Steps 3 and 4, getting remainders from right to left, until the quotient becomes zero in Step 3.
- The last remainder thus obtained will be the Most Significant Digit (MSD) of the new base number.

DECIMAL BINARY

- Conversion from decimal to binary can be performed by successively dividing the decimal number by 2 and using each remainder as a bit of the desired binary number.

Example:

$77_{10}=1001101_2$

$65_{10}=111110_2$

$127_{10}=?$

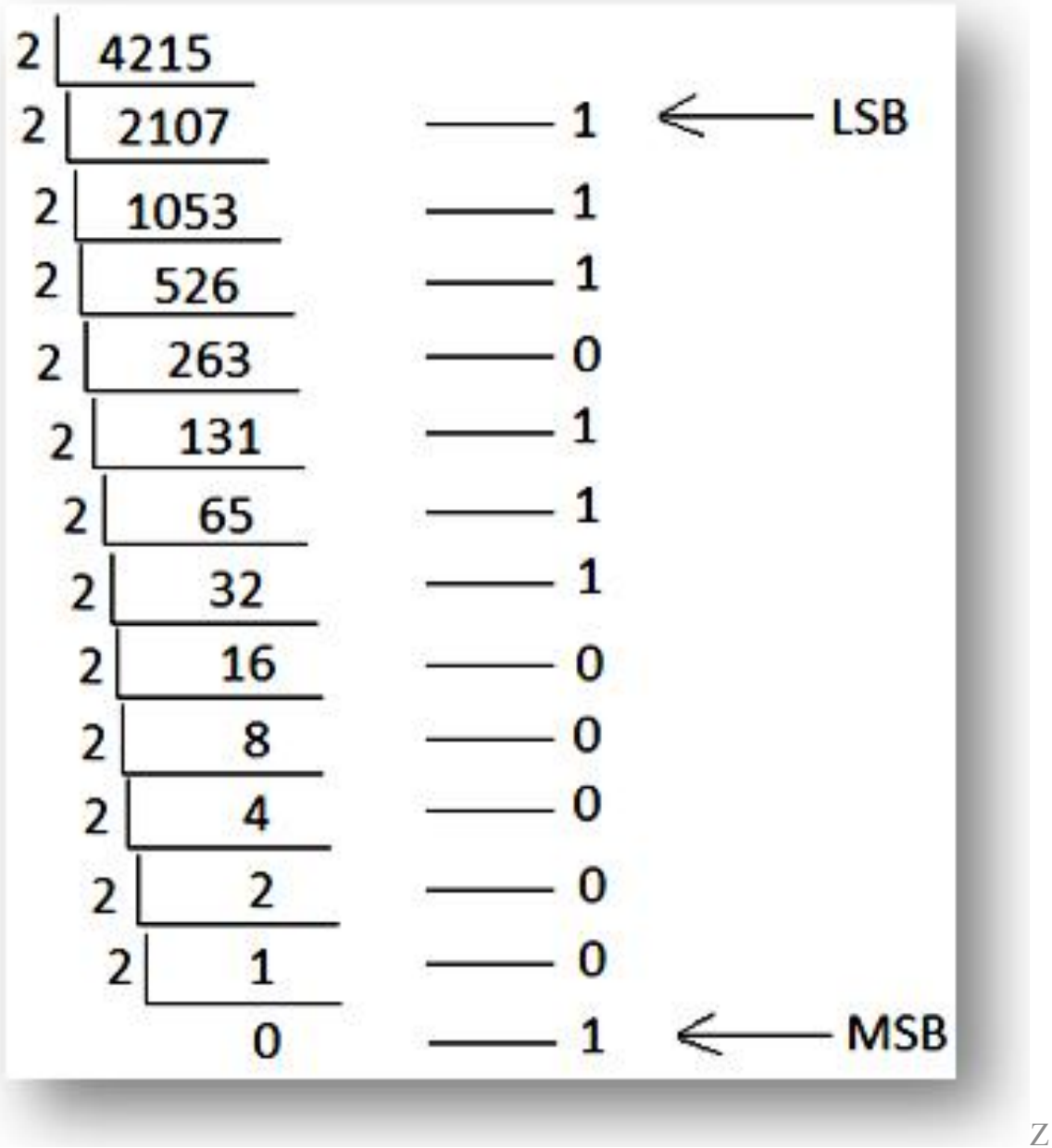
$251_{10} = ?$

Convert the following to their binary equivalent

$(25.375)_{10}$

$(56.75)_{10}$

DECIMAL BINARY

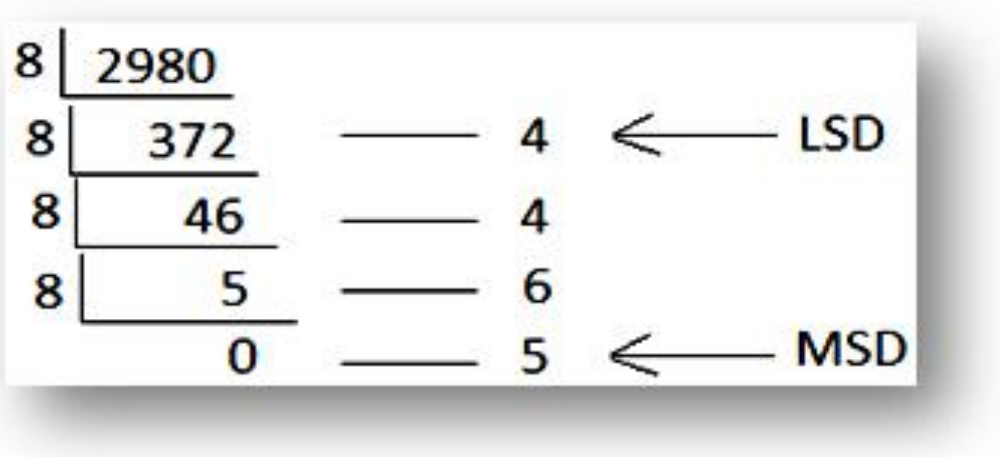


DECIMAL TO OCTAL

- Conversion from decimal to Octal can be performed by successively dividing the decimal number by 8 and using each remainder as a digit of the desired octal number.

Example:

Convert 2980_{10} to base 8



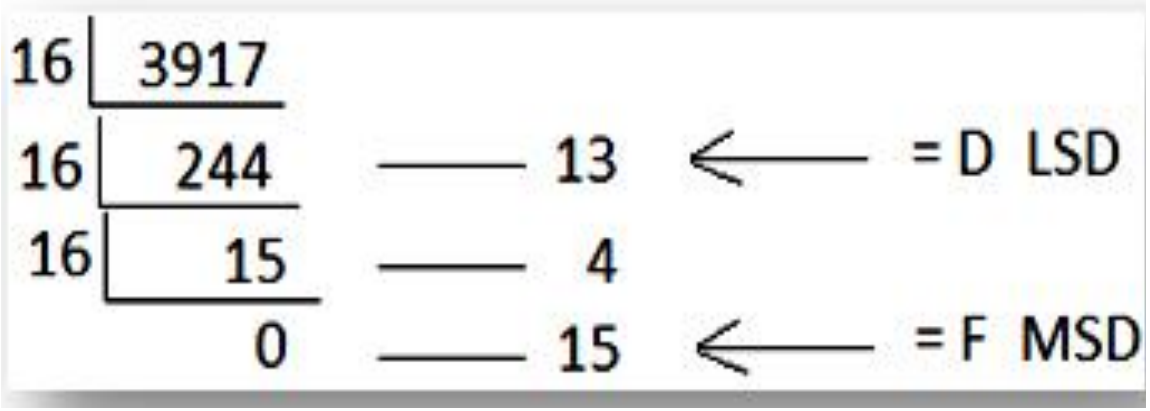
Therefore $2980_{10} = 5644_8$

DECIMAL TO HEXADECIMAL

Conversion from decimal to Hexadecimal can be performed by successively dividing the decimal number by 16 and using each remainder as a digit of the desired hexadecimal number.

Example:

Convert 3917_{10} to its hexadecimal equivalent



Therefore, $3917_{10} = F4D_{16}$

DECIMAL TO HEXADECIMAL

Convert the following decimal numerals to hexadecimal

$759_{10} = 2F7_{16}$

$1789_{10} = ??$

$356.50_{10} = ??$

$64.680_{10} = ??$

BINARY TO DECIMAL

Thus the decimal equivalent of a binary number has the general form;

$a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \ldots + a_i \times 2^i + \ldots + a_1 \times 2^1 + a_0 \times 2^0 + a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} + \ldots + a_{-p} \times 2^{-p}$

BINARY TO DECIMAL

$110_2 = (1*2^2) + (1*2^1) + (0*2^0) = 4+2+0 = 6_{10}$

OCTAL TO DECIMAL

$12570_8 = (1*8^4) + (2*8^3) + (5*8^2) + (7*8^1) + (0*8^0) = 43976_{10}$

HEXADECIMAL TO DECIMAL

Convert the following from hex to decimal

$41CF_{16} = 4*16^3 + 1*16^2 + C*16^1 + F*16^{0=} 16847_{10}$

BINARY TO HEXADECIMAL

- Since a string of 4 bits has 16 different permutations each 4 bit string represents a hexadecimal digit uniquely.
- Thus to convert a binary number to its hexadecimal equivalent we arrange the bits into groups of 4 starting at the binary point and move towards the MSB.
- We then replace each group by the corresponding hexadecimal digit

BINARY TO HEXADECIMAL

- Example

$11\ 1110\ 1101_2$ = add zeros to the left of MSB

$0011\ \ 1110\ \ 1101 = 3ED_{16}$

Convert the following into hexadecimal

a. 10110101110_2 =?

b. 100101111110_2 =?

HEXADECIMAL TO BINARY

Convert the following to binary equivalents:

$A748_{16}$

Solution:

$A748_{16}$ = 1010 0111 0100 1000

= 1010011101001000_2

Convert the following hexadecimal to binary

a. $FFDE_{16}$=?

b. $5ACF_{16}$=?

COMPLEMENT ARITHMETIC

- The complements are used to make the arithmetic operations such as subtraction in digital system easier
- The computer system use complements in most of the number system discussed in order to make subtraction operation on numbers easier.
- For each radix-r system (radix r represents base of number system) there are two types of complements.

Radix Complement

The radix complement is referred to as the r's complement

Diminished Radix Complement

The diminished radix complement is referred to as the (r-1)'s complement

BINARY SYSTEM COMPLEMENT

- As the binary system has base r = 2.
- So the two types of complements for the binary system are 2's complement and 1's complement.
- One’s complement and two’s complement are two important binary concepts.

COMPLEMENT ARITHMETIC

Example of complement

- 1's and 2's complement for binary numbers
- 7's and 8's complement for octal numbers
- 9's and 10's complement for decimal numbers
- 15's and 16's complement for hexadecimal numbers

1’s complement and 2’s complement

One’s Complement

If all bits in a binary number are inverted by changing each 1 to 0 and each 0 to 1, we have formed the one’s complement of the number

Example:

10011001 --> 01100110

10000001 --> 01111110

11110000 --> 00001111

2’s Complement

- The two’s complement is a method for representing positive and negative integer values in binary.

Rule:

- To form the two’s complement, add 1 to the one’s complement.

Two's Complement

We obtain 2's complement by adding 1 to the 1's complement

Example:

01100110+1=01100111(2's complement)

01111110+1=01111111(2's complement)

00001111+1=00010000 (2's complement)

1’s complement and 2’s complement

Convert the following into 2’s complement

- -12_{10}
- -65_{10}

First we convert to binary (forget about sign for now)

$12_{10}=1100_2$

find 1’s complement==>0011

Find 2’s complement==> [0011+1] =0100

Let consider the following problems

Using 1's complement compute the value of

a-->110101_2-100101_2=010000_2

b-->101011_2-111001_2=-001110_2

Using 2's complement compute the value of the problem above

Find 1's and 2's complement of 15_{10}-10_{10}=……$_2$

DECIMAL COMPLEMENT ARITHMETIC

9's and 10's complement

- This complement system provides an easy way of performing subtraction operation on decimal numbers
- To work in complement arithmetic , we need to convert the decimal number to its 9's or 10's complement equivalence.
- 9′s complement of decimal number can be obtained by $((10^n - 1)$-number) where n represents the number of digits in given number.
- 10′s complement can be obtained by $(10^n$ – number) where n represents the number of digits in given number.

DECIMAL COMPLEMENT ARITHMETIC

Example 9's complement of 115 is given by

$((10^3-1)-115)=884$

10's complement of 115 is given by

884+1=885

9's complement of 1984 is given by

$((10^4-1)-1984)=8015$

10's complement of 1984 is given by

8015+1=8016

Find 9's and 10's complement of the following

a. 2000
b. 1234
c. 48

Using 9's and 10's complement arithmetic to compute 251-185

251-185=>251+(-185)

Convert 185 to 9's complement=>(999-185)=814

10's complement of 185=>814+1=815

251+815

$$\begin{array}{r} 251 \\ 815 \\ \hline 1066 \end{array}$$

Drop the carry (1) so we have 66 which is equivalent to 251-185

Note: If the final answer of the summation does not have carry, we find 10's complement of the answer and add negative sign

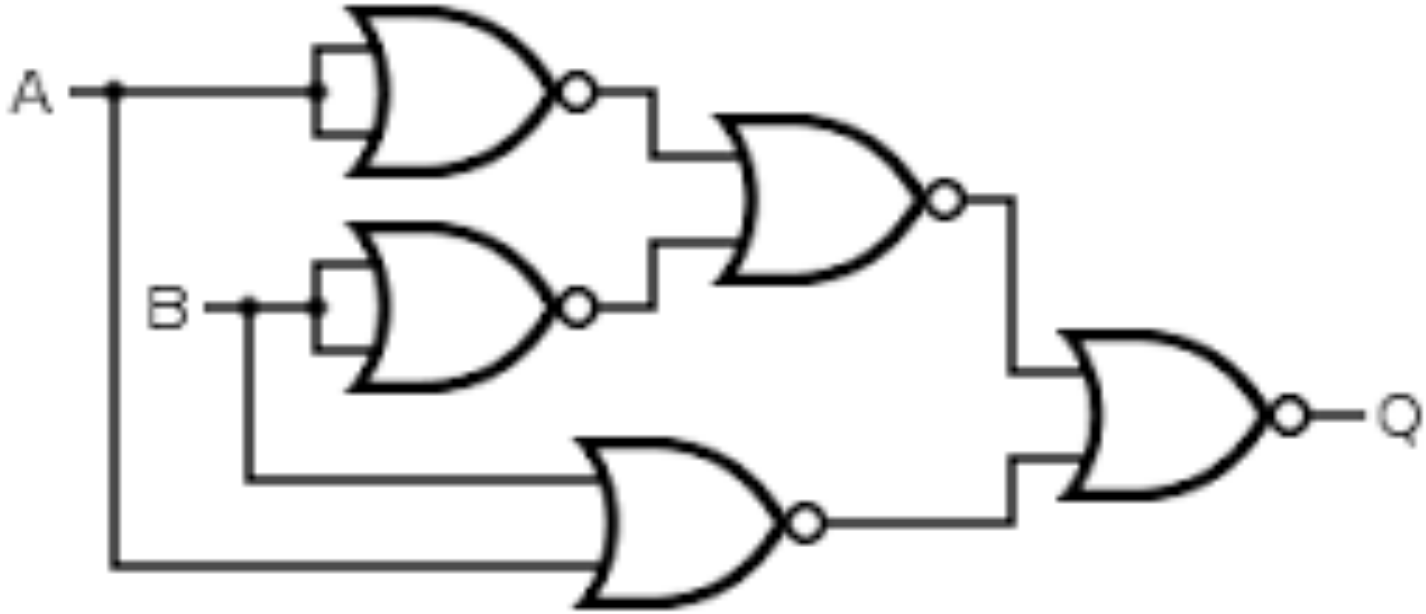
Class assignment

Discuss the following complements with illustrative examples

7's and 8's complement

15' and 16's complement

INTRODUCTION TO DIGITAL LOGIC DESIGN



DIGITAL SYSTEMS

- The fundamental idea of digital systems is to represent data in discrete form (Binary: 1's and 0's) and processing that information.
- Digital systems have led to many scientific and technological advancements.
- Mobile phones, computers, calculators, Microwaves etc. are examples of digital systems widely used for commercial and business data processing.

DIGITAL SYSTEMS

- The most important property of a digital system is the ability to follow a sequence of instructions to perform a task .
- Representing data in 1's and 0's, i.e. in binary system is the root of the digital systems.
- All the digital system store data in binary format, hence the need to understand binary number system.
- We have already discussed binary number systems

BINARY LOGIC/BOOLEAN ALGEBRA

- The design and behavior of circuitry in digital systems is analyzed, with the use of a mathematical discipline called *Boolean algebra.*
- Boolean algebra makes use of logical variables and logical operators.
- Variable may take on the value 1 (TRUE) or 0 (FALSE).

BOOLEAN OPERATORS

- The basic logical operators are AND, OR, XOR and NOT
  - AND is represented by dot (.)
  - OR is represented by plus sign(+)
  - NOT is represented by overbar (bar on top of the variable)
  - XOR is represented by (⊕)

BOOLEAN VARIABLES AND OPERATORS

Given the logic variables A and B with Logical operators AND, OR, XOR and NOT we have

$$A \; AND \; B = A \; . \; B$$

$$A \; OR \; B = A + B$$

$$NOT \; A = \overline{A}$$

$$NOT \; B = \overline{B}$$

$$A \; XOR \; B = A \oplus B$$

RULES OF BOOLEAN ALGEBRA

AND: Given two inputs x, y the expression x.y or simply xy represents "x AND y" and equals to 1 if both x and y are 1, otherwise 0.

OR: Given two inputs x, y the expression x+y represents "x OR y" and equals to 1 if at least one of x and y is 1, otherwise 0.

XOR: Given two inputs x, y the expression $x \oplus y$ represents "x XOR y" and equals to 1 if only one of x and y is 1 (not both).

NOT: Given x, the expression x' represents NOT(x) equals to 1 if x is 0, otherwise 0. NOT(x) is x complement.

EXAMPLE OF BOOLEAN OPERATIONS

X	Y	X . Y	X + Y	NOT X	NOT Y	X $\oplus$ Y
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	1	0	0	0

MORE BOOLEAN OPERATORS

The AND, OR and XOR operators can be extended using the NOT operator to produce other three operators including

- NAND
- NOR
- XNOR

NAND

This is a negation of AND Operator : NOT (AND)=>NAND

NOR

This is a negation of OR operator: NOT (OR)=>NOR

XNOR

This is a negation of XOR operator: NOT (XOR)=>XNOR

MORE BOOLEAN VARIABLES AND OPERATORS

Example

$$A\ NAND\ B = \overline{A.\ B}$$

$$A\ NOR\ B = \overline{A + B}$$

$$A\ XNOR\ B = \overline{A \oplus B}$$

- In the absence of parenthesis, the AND Operator takes precedence over the OR operator.
- Logical operators play key role is creating digital circuit

BASIC IDENTITIES OF BOOLEAN ALGEBRA

AND	OR	IDENTITY
$A.B=B.A$	$A+B=B+A$	Commutative
$A.(B+C)=(A.B)+(A.C)$	$A+(B.C)=(A+B).\ (A+C)$	Distributive
$A.(B.C)=(A.B).C$	$A+(B+C)=(A+B)+C$	Associative
$A.1=A$	$A+0=A$	Identity Element
$A.\ \overline{A}=0$	$A+\overline{A}=1$	Inverse Element
$A.0=0$	$A+0=A$	
$A.A=A$	$A+A=A$	
$A+AB=A$	$A(A+B)=A$	Absorption
$\overline{A\ .\ B}=\ \overline{A}+\ \overline{B}$	$\overline{A+B}=\ \overline{A}\ .\ \overline{B}$	De Morgan's Theorem

TRY

Prove that

1. a+(a′.b)=a+b

2.a.(a′+b)=a.b

3.a.b′+a.(b+c)′+b.(b+c)′=a.b′

TRY

Prove that

1. a+(a′.b)

2.a.(a′+b)

3.a.b′+a.(b+c)′+b.(b+c)′

Truth Table to Verify DeMorgan's

$$\overline{X + Y} = \overline{X} \cdot \overline{Y} \qquad \overline{X \cdot Y} = \overline{X} + \overline{Y}$$

X	Y	X·Y	X+Y	$\overline{X}$	$\overline{Y}$	$\overline{X+Y}$	$\overline{X} \cdot \overline{Y}$	$\overline{X \cdot Y}$	$\overline{X}+\overline{Y}$
0	0	0	0	1	1	1	1	1	1
0	1	0	1	1	0	0	0	1	1
1	0	0	1	0	1	0	0	1	1
1	1	1	1	0	0	0	0	0	0

- Generalized DeMorgan's Theorem:

$$\overline{X_1 + X_2 + \ldots + X_n} = \overline{X_1} \cdot \overline{X_2} \cdot \ldots \cdot \overline{X_n}$$

$$\overline{X_1 \cdot X_2 \cdot \ldots \cdot X_n} = \overline{X_1} + \overline{X_2} + \ldots + \overline{X_n}$$

INTRODUCTTION TO LOGIC GATE

- A gate is an electronic circuit that produces an output signal, which is a simple Boolean operation on its input signals.
- Logic gates are the elementary building blocks for digital systems
- Logic gates are connected to form logic circuit which is found in all digital systems

INTRODUCTTION TO LOGIC GATE

- The basic gates used in digital logic are AND, OR, NOT, XOR, NOR, NAND, XNOR
- Each gate is defined in three ways:
  - Graphic symbol,
  - Algebraic notation,
  - truth table.

LOGIC GATE

- The symbology used here is the IEEE standard, IEEE Std 91.
- Each gate shown in has one or two inputs and one output.
- However, as indicated in all of the gates except NOT can have more than two inputs.

AND GATE

A basic AND gate has two inputs and one output.

Let's call the two inputs A and B, and the output Y.

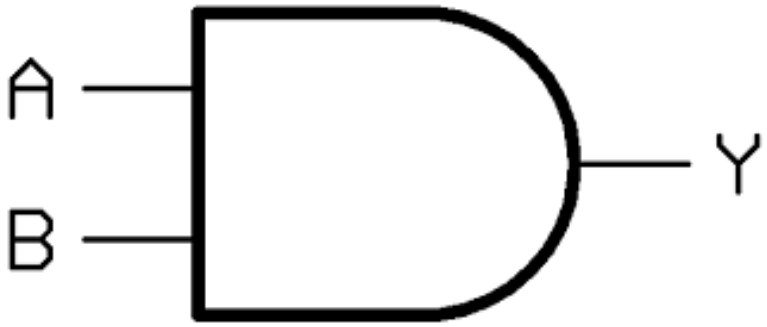
Then Y = 1 if and only if A = 1 and B = 1, hence the name "AND."

The AND operation is represented in **Boolean expression** settings by multiplication, i.e. we write

Y=A . B : algebraic / Boolean expression

AND GATE

- The standard Graphical Notation of the AND gate is represented as



- Truth Table representation is

Input		Output
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE

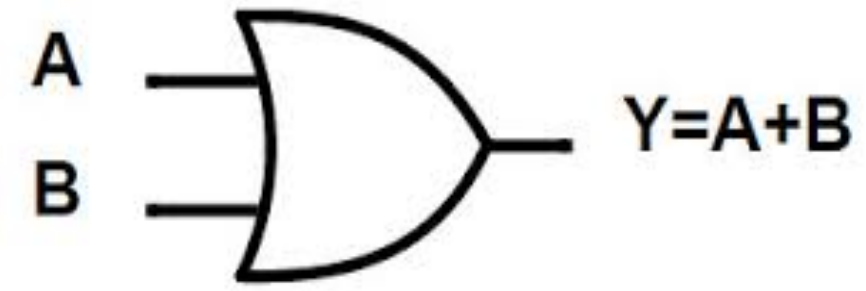
A basic OR gate will have two inputs, but in this case Y = 1 if and only if A = 1 or B = 1 (which includes the case in which both A and B are 1).

The OR operation is represented in **Boolean equation** settings by plus, i.e. we write

Y=A+B : algebraic / boolean equation

OR GATE

- Graphical Notation of the OR gate is represented as



- Truth Table representation is

Input		Output
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE

A NOT gate has one input A and one output Y, with the output being the logical negation of the input.

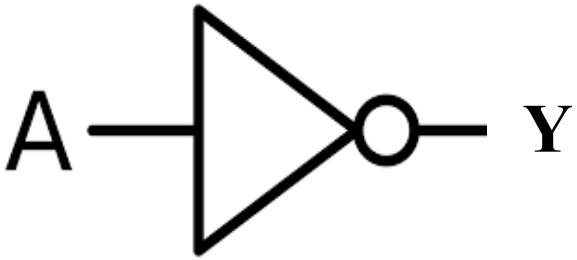
In other words, an input of 1 produces an output of 0, and vice versa.

In boolean equations, a NOT operation is indicated by an overbar or complement operator (‘) :

Y= $\overline{A}$ ***or Y= A’ :*** ***algebraic / boolean equation***

NOT GATE

- Graphical Notation of the NOT gate is represented as



- Note that the inversion (NOT) operation is indicated by a circle.
- Truth Table representation is

Input	Output
A	Y
0	1
1	0

NAND GATE

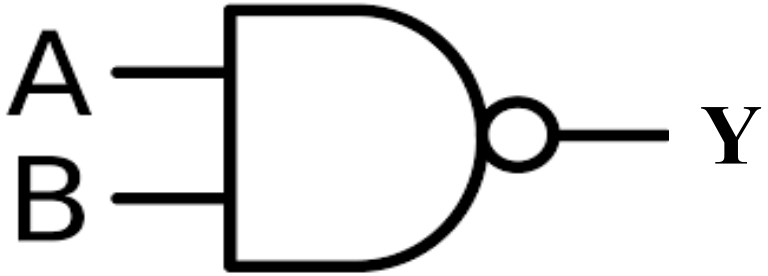
Here there are two inputs A and B, and one output Y.

The term "NAND" stands for "not-and," meaning that Y = 1 if the statement "A = 1 and B = 1" is not true.

Y= $\overline{AB}$ ***: algebraic / boolean equation***

NAND GATE

- Graphical Notation of the NAND gate is represented as

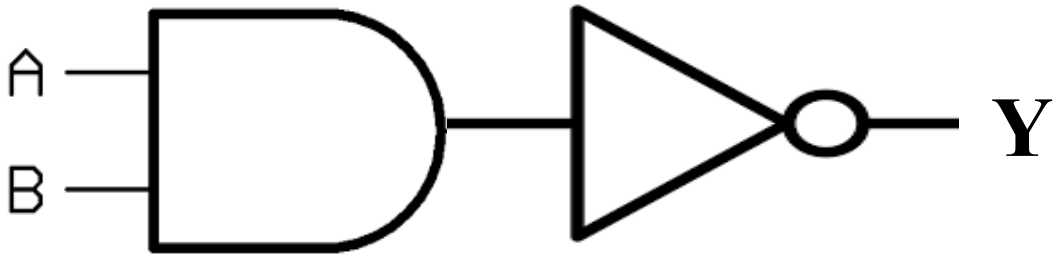


- Note that the little circle here means "not."
- Truth Table representation is

Input		Output
A	B	Y= $\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

SYNTHESIZING NAND GATE

- If we run out of NAND gates, we could synthesize a NAND Gate by using an AND together with a NOT:



- This would not be so desirable as using a real NAND.
- The synthesized version would probably have more transistors than the real one
- would be slower and take up more space on a chip,

NOR GATE

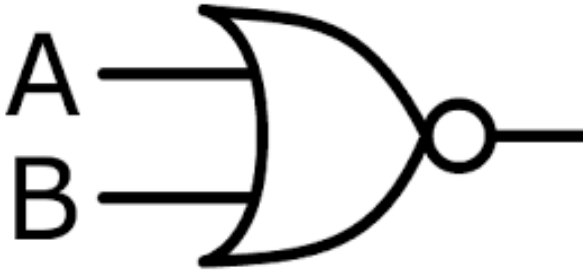
Again, inputs A and B, output Y, with Y being equal to 1 if the statement "A = 1 or B = 1" is not true.

The boolean equation is

Y= $\overline{A + B}$ ***: algebraic / boolean equation***

NOR GATE

- Graphical Notation of the NOR gate is represented as



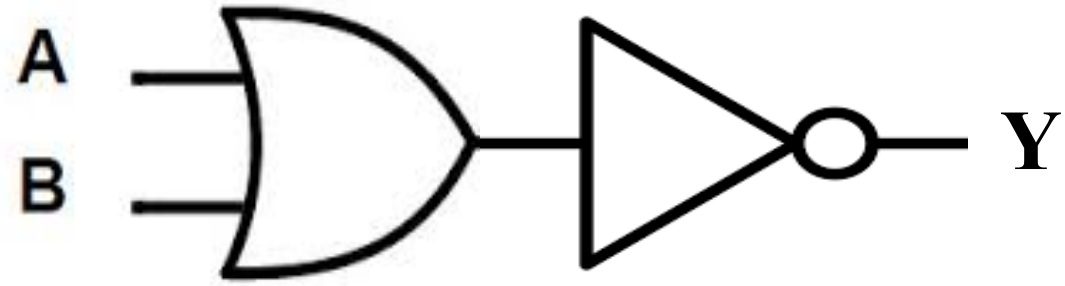
$$Y \quad \overline{A + B}$$

- Note that the little circle here means "not."
- Truth Table representation is

Input		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

SYNTHESIZING NOR GATE

- If we run out NOR gates, we could synthesize a NOR Gate by using an OR together with a NOT:



- This would not be so desirable as using a real NOR.
- The synthesized version would probably have more transistors than the real one
- would be slower and take up more space on a chip,

XOR GATE

Here we have inputs A and B, output Y, with Y being equal to 1 if the statement "A = 1 or B = 1 but not both" is true.

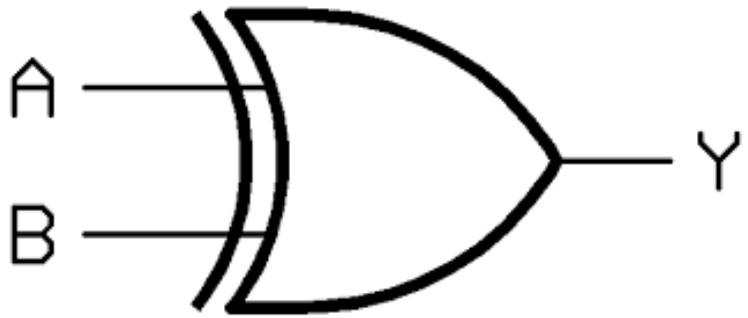
The term used for this is "exclusive-or," abbreviated to XOR.

The boolean equation is

$$Y = A \oplus B = \overline{A}B + A\overline{B}$$

XOR GATE

- Graphical Notation of the XOR gate is represented as

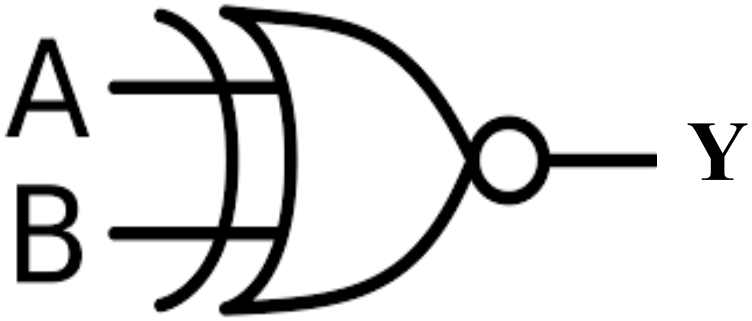


- Truth Table representation is

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

XNOR GATE

- The *XNOR (exclusive-NOR) gate* is a combination XOR gate followed by an inverter.
- Its output is "true" if the inputs are the same, and "false" if the inputs are different.
- Graphical Notation of the XNOR gate is represented as

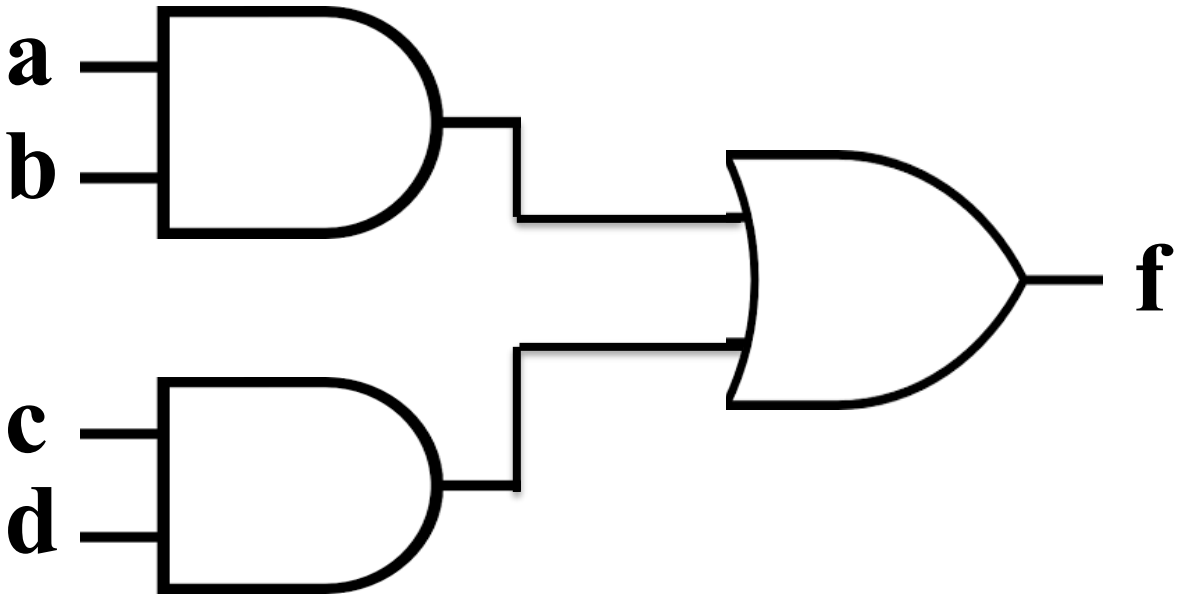


$$Y = \overline{A \oplus B} = A.B + \overline{A}.\overline{B}$$

- Truth Table representation is

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Implement the function : $f=a.b + c.d$ using AND and OR gate

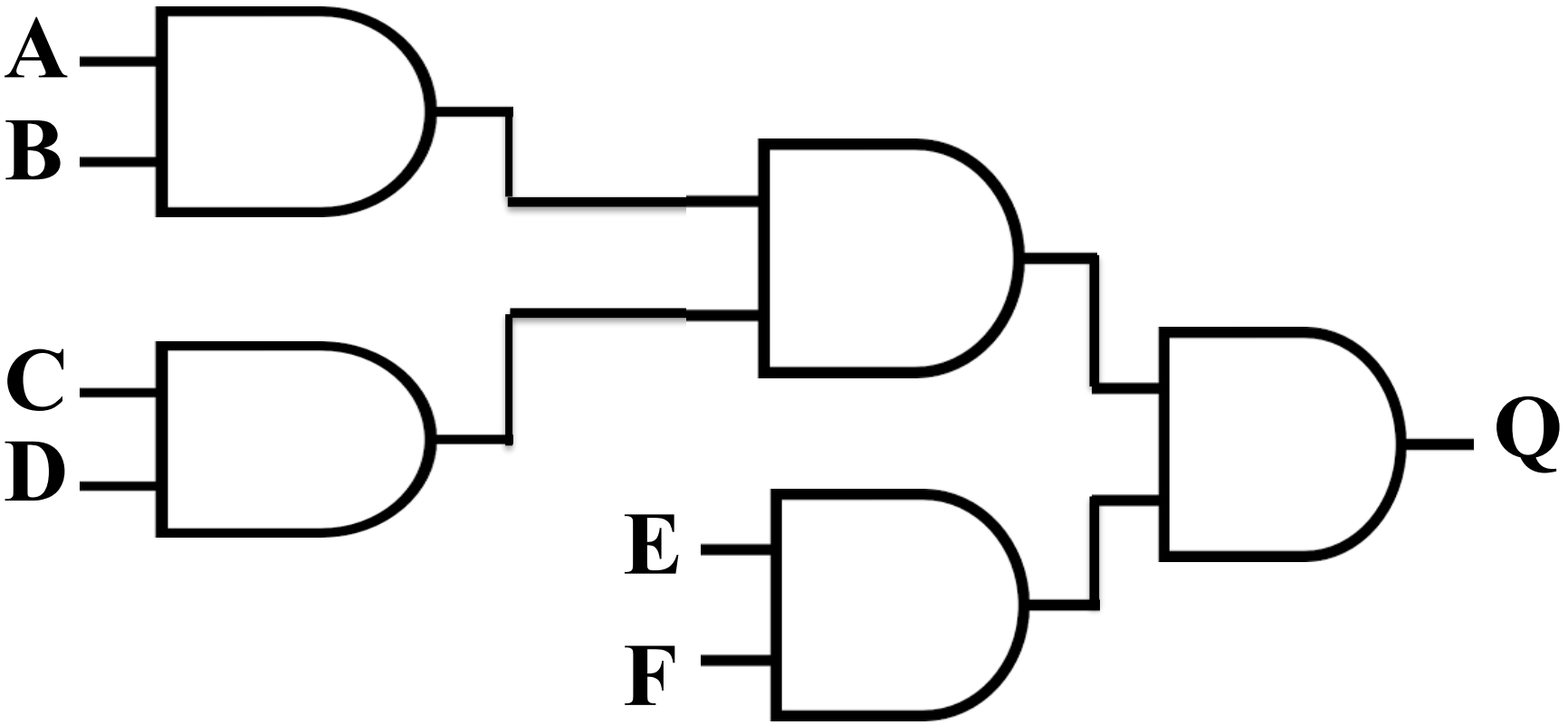


implement the function using two levels and three levels respectively

Q = (A.B).(C.D).(E.F)

implement the function using two levels and three levels respectively

Q = (A.B).(C.D).(E.F)



LOGIC DESIGN PROCESS

A simple logic design process involves the following activities

- Problem specification
- Truth table derivation
- Derivation of logical expression
- Simplification of logical expression
- Implementation of Gate

PROBLEM SPECFICATION

- The design of a combination circuit start from a verbal problem and end in a logic circuit diagram or a set of Boolean functions from which the logic diagram can easily be obtained
- Problem specification involves stating and analyzing the problem to clearly understand it.
- The nature of inputs and outputs are determined
- The number of inputs and number of outputs are also determined

TRUTH TABLE DERIVATION

- The input and output requirements are analyzed and truth table is formed
- The truth table would be used to assess the behavior of the expected circuit diagram
- The truth table defines the required relationship between the input and expected output

DERIVING LOGICAL EXPRESSION FROM TRUTH TABLE

Definition of terms

Literal: complemented or non-complemented variable (E.g. A OR A′)

Product term: a series of literals related to one another through the AND operator (E.g. A′.B.C).

Sum term: A series of literals related to one another through the OR operator (E.g. A′+B+C′)

SOP form: (Sum –of-products form)

This represent the some of product terms

The logic-circuit simplification require the logic expression to be in SOP form;

E.g. (AB′C+A′BD′+AC′D)

DERIVING LOGICAL EXPRESSION FROM TRUTH TABLE

Definition of terms

POS form (product-of-sum form):

This represents the product of sum terms

This form sometimes used in logic circuit; Example;

`(A′+D′).(A+C′).(C+D′)`

Canonical and standard form

Product terms that consist of the variables of function are called "Canonical product terms" or "Minterms ".

The term AB'C is a Minterm in a three variable logic function, but will be a non-Minterm in a four variable logic function.

DERIVING LOGICAL EXPRESSION FROM TRUTH TABLE

Canonical and standard form

Sum terms which contain all the variables of a Boolean function are called "Canonical sum terms" or Maxterms".

`A'+B+C` is a Maxterm in a three variable logic function

FORMING LOGICAL EXPRESSION

- A Boolean function may be expressed algebraically from a given truth table by forming a Minterm for each combination of variable that produce a 1 and taken the OR of those terms(Disjunctive Normal Form)
- Similarly, the same function can be obtained by forming the Maxterm for each combination of variable that produces 0 and then taken the AND of those term (Conjunctive Normal Form)

LOGICAL EXPRESSION DERIVATION

✧ Derivation of logical expressions from truth tables

- Sum-of-products (SOP)/Minterms form
- Product-of-sums (POS) / Maxterms form

Minterms

- A product term in which all the variables appear exactly once, either complemented or uncomplemented, is called a Minterm
- It represent exactly one combination of the binary variables in a truth table. It has the value of 1 for that combination and 0 for the others

DERIVE SOP/MINTERMS LOGICAL EXPRESSION

- Boolean function can be represented algebraically from a given truth table by forming the logical sum of all the minterms that produce a 1 in the function.
- This expression is called a sum of minterms
- When the sum of minterms is simplified by reducing the number of product terms the simplified expression is referred to as *Sum- of- Product(SOP)*

LOGICAL EXPRESSION DERIVATION

Forming Minterms

- Write an AND term for each input combination that produces a 1 output
- The product of all the combination must be 1
- Write the variable if its value is 1;
- Complement the variable if its value is 0
- OR the AND terms to get the final expression
- With n variable, 2^n minterms can be formed.
- Also referred to as Disjunctive Normal Form (DNF)

DERIVE SOP/MINTERMS LOGICAL EXPRESSION

3-INPUT FUNCTION

A	B	C	F	Minterms	Symbol
0	0	0	0	$\overline{A}.\overline{B}.\overline{C}$	m0
0	0	1	0	$\overline{A}.\overline{B}.C$	m1
0	1	0	0	$\overline{A}.B.\overline{C}$	m2
0	1	1	1	$\overline{A}.B.C$	m3
1	0	0	0	$A.\overline{B}.\overline{C}$	m4
1	0	1	1	$A.\overline{B}.C$	m5
1	1	0	1	$A.B.\overline{C}$	m6
1	1	1	1	$A.B.C$	m7

$$F = (\overline{A}\ B\ C) + (A\ \overline{B}\ C) + (A\ B\ \overline{C}) + (A\ B\ C)$$

- A Minterm represents exactly one combination of the binary variables in a truth table.
- It has the value of 1 for that combination and 0 for the others

DERIVE MINTERMS LOGICAL EXPRESSION

3-INPUT FUNCTION

A	B	C	F	minterms
0	0	0	1	m_0
0	0	1	0	m_1
0	1	0	1	m_2
0	1	1	0	m_3
1	0	0	0	m_4
1	0	1	1	m_5
1	1	0	0	m_6
1	1	1	1	m_7

$$F = (\overline{A}\ \overline{B}\ \overline{C}) + (\overline{A}\ B\ \overline{C}) + (A\ \overline{B}\ C) + (A\ B\ C)$$

DERIVE MINTERMS LOGICAL EXPRESSION

The sum of minterms from the truth table above is given by

$$F = (\overline{A}\ \overline{B}\ \overline{C}) + (\overline{A}\ B\ \overline{C}) + (A\ \overline{B}\ C) + (A\ B\ C)$$

Which can be defined by the minterms symbols

$F = m_0 + m_2 + m_5 + m_7$

$$F(X,Y,Z) = \sum_{M}(0,2,5,7)$$

DERIVE POS/MAXTERMS LOGICAL EXPRESSION

- A Boolean function can be represented algebraically from a given truth table by forming the logical product of all the Maxterms that produce a 0 in the function.
- This expression is called a product of Maxterms
- When we simplify a function in product of Maxterms form by reducing the number of sum terms or by reducing the number of literals in the terms we obtained Product-of-Sums form (POS)

LOGICAL EXPRESSION DERIVATION

Maxterms

- A sum term in which all the variables appear exactly once, either complemented or uncomplemented, is called a Maxterm
- A Maxterm represents exactly one combination of the binary variables in a truth table.
- It has the value of 0 for that combination and 1 for the others

LOGICAL EXPRESSION DERIVATION

Forming Maxterms

- Each Maxterm is obtained by writing the OR term for each input combination that produce 0,
- Each variable is complemented if the corresponding bit is 1 and uncomplemented if bit is 0.
- The sum of all the combination must be 0
- Note that each Maxterm is the complement of its corresponding Minterm and vice versa
- In a similar fashion, n variables forming an OR term provide 2^n possible combinations
- Also referred to as Conjunctive Normal Form(CNF)

- A Maxterm represents exactly one combination of the binary variables in a truth table.
- It has the value of 0 for that combination and 1 for the others
- A Minterm and Maxterm with the same subscript are the complements of each other, i.e., $Mj = \overline{Mj}$

DERIVE MAXTERMS LOGICAL EXPRESSION

3-INPUT FUNCTION

A	B	C	F	minterms
0	0	0	1	m_0
0	0	1	0	m_1
0	1	0	1	m_2
0	1	1	0	m_3
1	0	0	0	m_4
1	0	1	1	m_5
1	1	0	0	m_6
1	1	1	1	m_7

$$F = (A + B + \overline{C}).(A + \overline{B} + \overline{C}).(\overline{A} + B + C).(\overline{A} + \overline{B} + C)$$

DERIVE MAXTERMS LOGICAL EXPRESSION

3-INPUT FUNCTION

The Product of Maxterms from the truth table above is given by

$$F = (A + B + \overline{C}).(A + \overline{B} + \overline{C}).(\overline{A} + B + C).(\overline{A} + \overline{B} + C)$$

Which can be defined by the Maxterms symbols

$$F = m_1 m_3 m_4 m_6$$

$$F(X,Y,Z) = \prod_{M}(1,3,4,6)$$

DERIVE MAXTERMS LOGICAL EXPRESSION

3-INPUT FUNCTION

A	B	C	F	Maxterms	Symbol
0	0	0	0	$A+B+C$	m0
0	0	1	0	$A+B+\overline{C}$	m1
0	1	0	0	$A+\overline{B}+C$	m2
0	1	1	1	$A+\overline{B}+\overline{C}$	m3
1	0	0	0	$\overline{A}+B+C$	m4
1	0	1	1	$\overline{A}+B+\overline{C}$	m5
1	1	0	1	$\overline{A}+\overline{B}+C$	m6
1	1	1	1	$\overline{A}+\overline{B}+\overline{C}$	m7

$$F=(A+B+C).(A+B+\overline{C}).(A+\overline{B}+C).(\overline{A}+B+C)$$

Example: *minterms and maxterms for 3 variables*:

inputs			Minterm	Designation	Maxterm	Designation
A	B	C				
0	0	0	$\bar{A}\bar{B}\bar{C}$	m0	$A+B+C$	M0
0	0	1	$\bar{A}\bar{B}C$	m1	$A+B+\bar{C}$	M1
0	1	0	$\bar{A}B\bar{C}$	m2	$A+\bar{B}+C$	M2
0	1	1	$\bar{A}BC$	m3	$A+\bar{B}+\bar{C}$	M3
1	0	0	$A\bar{B}\bar{C}$	m4	$\bar{A}+B+C$	M4
1	0	1	$A\bar{B}C$	m5	$\bar{A}+B+\bar{C}$	M5
1	1	0	$AB\bar{C}$	m6	$\bar{A}+\bar{B}+C$	M6
1	1	1	ABC	m7	$\bar{A}+\bar{B}+\bar{C}$	M7