Chapter one

1- Number system and codes

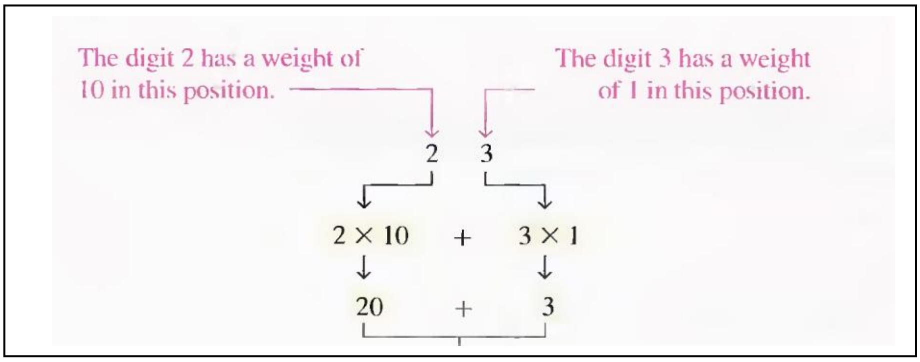
انظمه الاعداد

The binary number system and digital codes are fundamental to computers and to digital electronics in general. The binary number system and its relationship to other number systems such as decimal, hexadecimal, and octal has presented. Arithmetic operations with binary numbers have covered to provide a basis for understanding how computers and many other types of digital systems work.

الاعداد العشرية 1.1 DECIMAL NUMBERS

We are familiar with the decimal number system because we use decimal numbers every day. The decimal number system has ten digits, 0 through 9. Represent a certain Therefore, these digits not limited because used in different position. As shown in example below.

For example



The position of each digit in a decimal number indicates the magnitude of the quantity represented and could assign a weight. The weights for whole numbers are positive powers of ten that increase from right to left, beginning with $10^0 = 1$.

$$\ldots 10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$$

For fractional numbers, the weights are negative powers of ten that decrease from left to right beginning with 10^{-1}.

$$10^2\ 10^1\ 10^0.10^{-1}\ 10^{-2}\ 10^{-3}\ldots$$

Decimal point

Example:

Express the decimal number 568.23 as a sum of the values of each digit.

The whole number digit 5 has a weight of 100, which is 10^2, the digit 6 has a weight of 10, which is 10^1, the digit 8 has a weight of 1, which is 10^0, the fractional digit 2 has a weight of 0.1, which is 10^{-1}, and the fractional digit 3 has a weight of 0.01, which is 10^{-2}.

$$\begin{aligned} 568.23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0.1) + (3 \times 0.01) \\ &= \mathbf{500} + \mathbf{60} + \mathbf{8} + \mathbf{0.2} + \mathbf{0.03} \end{aligned}$$

الاعداد الثنائيه1.2 BINARY NUMBERS

The binary number has only two digits (bits) 1 and 0.

The position of a 1 or 0 in a binary number indicates its weight or value within the number,

The weights in a binary number have based on power of two.

DECIMAL NUMBER	BINARY NUMBER			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

As we have seen in Table above, four bits are required to count from zero to 15.

In general, with n bits we can count to a number equal to 2^n-1

Largest decimal number count = 2^n-1

With six bits (n = 4) you can count from zero to sixty-three.

$$2^4-1 = 16-1 = 15$$

A binary number is a weighted number. The right most bit is the **LSB** (least significant bit) in a binary whole number and has a weight of 2^0 = 1. The weights increase from **right to left** by a power of two for each bit. The left most bit is the **MSB** (most significant bit).

Frictional numbers can be represents in binary by placing bits to the right of the binary point. The **left**-most bit is the **MSB** in a binary fractional number, the fractional weights **decrease** from **left** to **right** by a negative power of two for each bit.

Figure below show the weights of binary fraction number where **n** is the **number of bits** from the binary point.

$$2^{n-1} . \; . \; . \; 2^3\, 2^2\, 2^1\, 2^0 \,.\, 2^{-1}\, 2^{-2} \; . \; . \; . \; 2^{-n}$$

Binary point

Binary weight table as shown below

POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0.5	0.25	0.125	0.0625	0.03125	0.015625

1.3 OCTAL NUMBERS الاعداد الثمانيه

The octal number system is composed of eight digits, which are

0, 1, 2, 3, 4, 5, 6, 7

Each octal number can be represented by three digits only 000 to 111

1.4 HEXADECIMAL NUMBERS الاعداد السادس عشر

The hexadecimal number system consists of digits 0-9 and letters A-F.

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

تحويل الاعداد Conversion between systems

- **Decimal –to- binary conversion.تحويل العشري الى ثنائي**

We have two method discussed below

1. **Sum-of-Weights Method جمع اوزان الاعداد**

One way to find the binary number that is equivalent to a given decimal number is to determine the set of binary weights whose **sum is equal** to the decimal number.

Example:

Convert the following decimal numbers to binary:

(a) 12 **(b)** 25 **(c)** 58 **(d)** 82

Solution

(a) $12 = 8 + 4 = 2^3 + 2^2 \longrightarrow$ **1100**

(b) $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 \longrightarrow$ **11001**

(c) $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 \longrightarrow$ **111010**

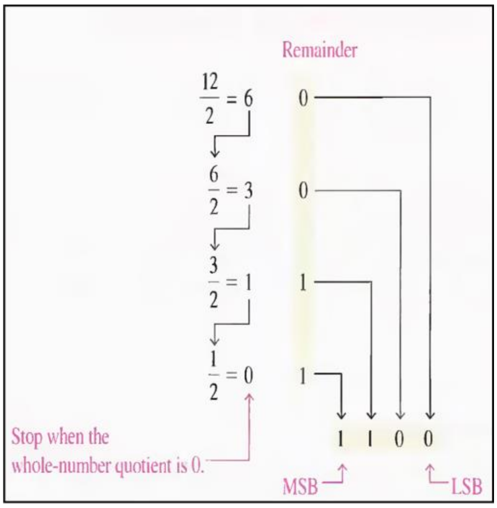
(d) $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 \longrightarrow$ **1010010**

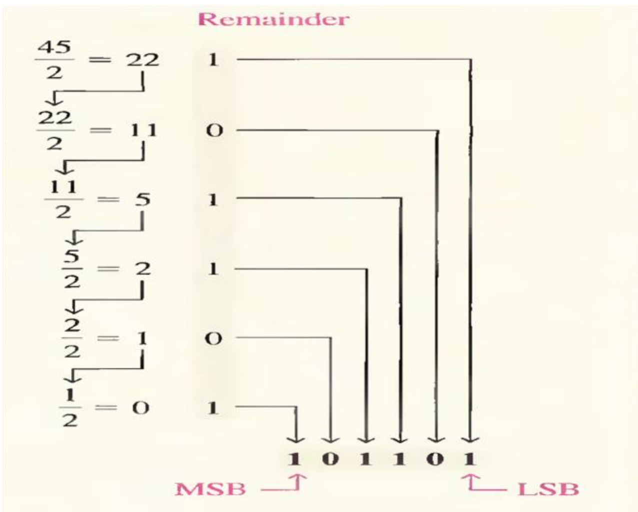
2. **Repeated Division-by-2 Method طريقه القسمة على 2**

To get the binary number for a given decimal number, divide the decimal number by 2 until the quotient is zero. Remainders form is the binary number.

Examples below explain the process for more detail.

Examples :





- **Converting Decimal Fractions -to Binaryتحويل كسور العشري الى ثنائي**

An easy way to remember fractional binary weights is that the most significant weight is 0.5, which is 2^{-1} and that by halving any weight, you get the next lower weight; thus a list of four fractional binary weights would be 0.5, 0.25, 0.125, 0.0625.

1. **Sum-of-Weightsجمع الاوزان**

The sum-of-weights method could apply to fractional decimal numbers, we determine the fraction binary wait whose sum equal to decimal number as shown in the following example:

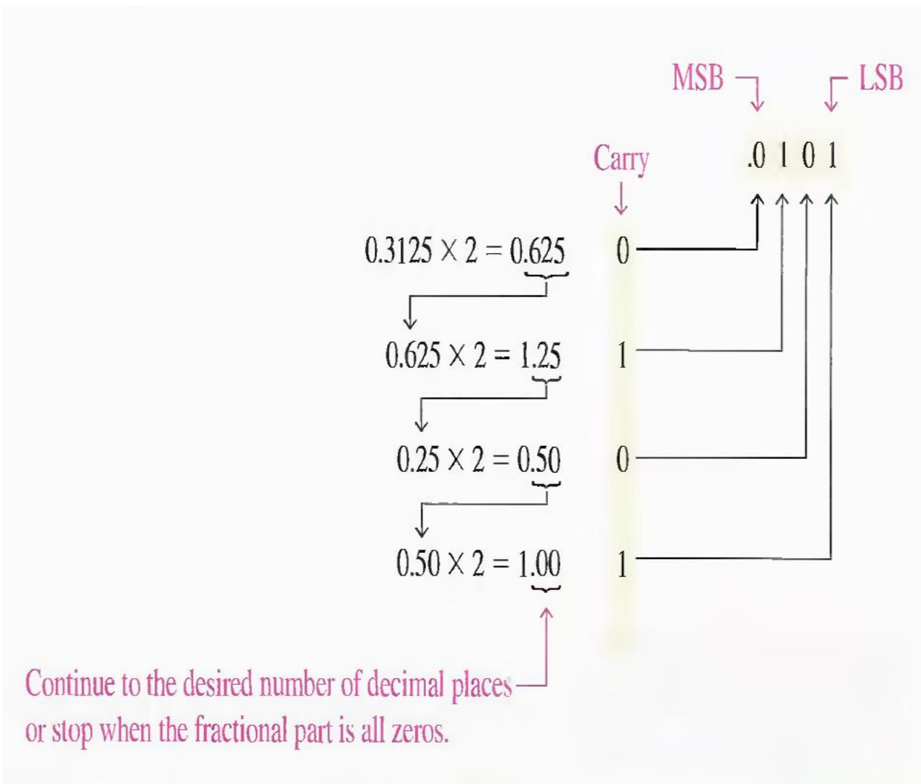
Example:

$$0.625 = 0.5 + 0.125 = 2^{-1} + 2^{-3} = 0.101$$

There is a 1 in the 2^{-1} position, a 0 in the 2^{-2} position, and a 1 in the 2^{-3} position.

2. **Repeated Multiplication by 2اعاده ضرب الاعداد**

As you have seen, decimal whole numbers can be converted to binary by repeated division by 2. Decimal fractions can be converted to binary by repeated multiplication by 2, the carry digits are the binary number we stop multiplication when the fraction part of multiplication equal to zero For example,



Binary-to-Decimal Conversionتحويل الثنائي الى عشري

The decimal value of any binary number can be founds by **adding** the weights of all bits that are **1** and **discarding** the weights of all bits that are **zero**. Examples below more detail to conversion.

Example1:

Convert the binary whole number 1101101 to decimal.

Solution Determine the weight of each bit that is a 1, and then find the sum of the weights to get the decimal number.

$$\begin{array}{rccccccc} \text{Weight:} & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ \text{Binary number:} & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{array}$$

$$\begin{aligned} 1101101 &= 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\ &= 64 + 32 + 8 + 4 + 1 = \mathbf{109} \end{aligned}$$

Example2:

Convert the fractional binary number 0.1011 to decimal.

Solution Determine the weight of each bit that is a 1, and then sum the weights to get th decimal fraction.

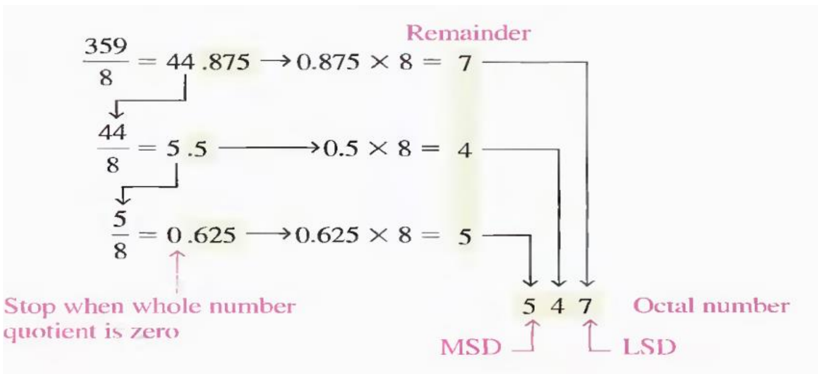
$$\begin{array}{rcccc} \text{Weight:} & 2^{-1} & 2^{-2} & 2^{-3} & 2^{-4} \\ \text{Binary number:} & 0\,.\,1 & 0 & 1 & 1 \end{array}$$

$$\begin{aligned} 0.1011 &= 2^{-1} + 2^{-3} + 2^{-4} \\ &= 0.5 + 0.125 + 0.0625 = \mathbf{0.6875} \end{aligned}$$

Decimal-to-Octal Conversion :تحويل العشري الى ثماني

A method of converting a decimal number to an octal number is the repeated division-by- 8

Example



Octal-to-Decimal Conversionتحويل الثماني الى عشري

The evaluation of an octal number in terms of its decimal equivalent has accomplished by multiplying each digit by its weight and summing the products.

$$\begin{aligned} 2374_8 &= (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) \\ &= (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \\ &= 1024 + 192 + 56 + 4 = 1276_{10} \end{aligned}$$

Decimal to Hexadecimal Conversion تحويل العشرى الى سادس عشر

Repeated division of a decimal number by 16 will produce the equivalent hexadecimal number, formed by the remainders of the divisions. The first remainder produced is the least significant digit (LSD). Each successive division by 16 yields a remainder that becomes digit in the equivalent hexadecimal number. Note that when a quotient has a fractional part, the fractional part has multiplied by the divisor to get the remainder.

Convert the decimal number 650 to hexadecimal by repeated division by 16.

Solution

Hexadecimal remainder

$\frac{650}{16} = 40.625 \rightarrow 0.625 \times 16 = 10 =$ A

$\frac{40}{16} = 2.5 \rightarrow 0.5 \times 16 = 8 =$ 8

$\frac{2}{16} = 0.125 \rightarrow 0.125 \times 16 = 2 =$ 2

Stop when whole number quotient is zero.

2 8 A Hexadecimal number

MSD LSD

Hexadecimal-to-Decimal Conversion تحويل السادس عشر الى عشرى

One way to find the decimal equivalent of a hexadecimal number is to first convert the hexadecimal number to binary and then convert from binary to decimal.

Convert the following hexadecimal numbers to decimal:

(a) $1C_{16}$ (b) $A85_{16}$

Solution Remember, convert the hexadecimal number to binary first, then to decimal.

(a) 1 C

$\underbrace{0001}_{1}\underbrace{1100}_{C} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28_{10}}$

(b) A 8 5

$\underbrace{1010}_{A}\underbrace{1000}_{8}\underbrace{0101}_{5} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693_{10}}$

Another way to convert a hexadecimal number to its decimal equivalent is to multiply

The decimal value of each hexadecimal digit by its weight and then take the sum of these products, the weights of a hexadecimal number are increasing powers of 16 (from right to left). For a 4-digit hexadecimal number, the weights are

16^3	16^2	16^1	16^0
4096	256	16	1

Example:

Convert the following hexadecimal numbers to decimal:

(a) $E5_{16}$ (b) $B2F8_{16}$

Solution Recall from Table 2–3 that letters A through F represent decimal numbers 10 through 15, respectively.

(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229_{10}}$

(b) $B2F8_{16} = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1)$
$= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1)$
$= 45{,}056 + 512 + 240 + 8 = \mathbf{45{,}816_{10}}$

Binary-to-Octal Conversion تحويل الثنائي الى ثماني of a binary number to an octal number is the reverse of the octal-to-binary conversion

Example

Convert each of the following binary numbers to octal:

(a) 110101 (b) 101111001 (c) 100110011010 (d) 11010000100

Solution

(a) $\underbrace{110}\underbrace{101}$ → 6 5 $= \mathbf{65_8}$

(b) $\underbrace{101}\underbrace{111}\underbrace{001}$ → 5 7 1 $= \mathbf{571_8}$

(c) $\underbrace{100}\underbrace{110}\underbrace{011}\underbrace{010}$ → 4 6 3 2 $= \mathbf{4632_8}$

(d) $\underbrace{011}\underbrace{010}\underbrace{000}\underbrace{100}$ → 3 2 0 4 $= \mathbf{3204_8}$

Octal-to-Binary Conversion تحويل الثنائي الى ثماني : Because each octal digit can be represented by a 3-bit binary number,

OCTAL DIGIT	0	1	2	3	4	5	6	7
BINARY	000	001	010	011	100	101	110	111

Example:

Convert each of the following octal numbers to binary:

(a) 13_8 (b) 25_8 (c) 140_8 (d) 7526_8

Solution

(a) 1 3 → 001011

(b) 2 5 → 010101

(c) 1 4 0 → 001100000

(d) 7 5 2 6 → 111101010110

Binary-to-Hexadecimal Conversion تحويل من الثنائي الى السادس عشر

Converting a binary number to hexadecimal is a straightforward procedure. Simply break the binary number into 4-bit groups. Starting at the right-most bit, replace each 4-bit group with the equivalent hexadecimal symbol.

Convert the following binary numbers to hexadecimal:

(a) 1100101001010111 **(b)** 111111000101101001

Solution **(a)** 1100101001010111 → C A 5 7 $= CA57_{16}$

(b) 00111111000101101001 → 3 F 1 6 9 $= 3F169_{16}$

Two zeros have been added in part (b) to complete a 4-bit group at the left.

Hexadecimal-to-Binary Conversion تحويل السادس عشر الى الثنائي

To convert from a hexadecimal number to a binary number, reverse the process and replace each hexadecimal symbol with the appropriate four bits.

Example:

Determine the binary numbers for the following hexadecimal numbers:

(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(c)** 9742_{16}

Solution **(a)** 1 0 A 4 → 1000010100100 **(b)** C F 8 E → 1100111110001110 **(c)** 9 7 4 2 → 1001011101000010

In part (a), the MSB is understood to have three zeros preceding it, thus forming a 4-bit group.

Octal to Hexadecimal Conversion تحويل الثماني الى السادس عشر

To convert the octal to hex number by the fallowing steps

1- Convert the octal number to binary
2- Make group 4 digit and we add 0 to MSB
3- Convert the number to Hex

Example: convert 754_8 to Hexadecimal number

Octal	7	5	4
Binary	00 (1 1 1	101	100)
	0001	1110	1100
HEX	**(1**	**E**	**$C)_{16}$**

Hexadecimal to Octal Conversion تحويل السادس عشر الى الثماني

To convert the hex number to octal by

1- Convert the Hex number to binary
2- Make group for 3 digit and add 0 to MSB
3- Convert the number to octal

Example: convert the Hex $(FD4)_{16}$ to octal

(F	D		4)
(1111	1101		0100)
(111	111	010	100)
(7	**7**	**2**	**4)$_8$**

Seven segment display : شاشة الأقسام السبع

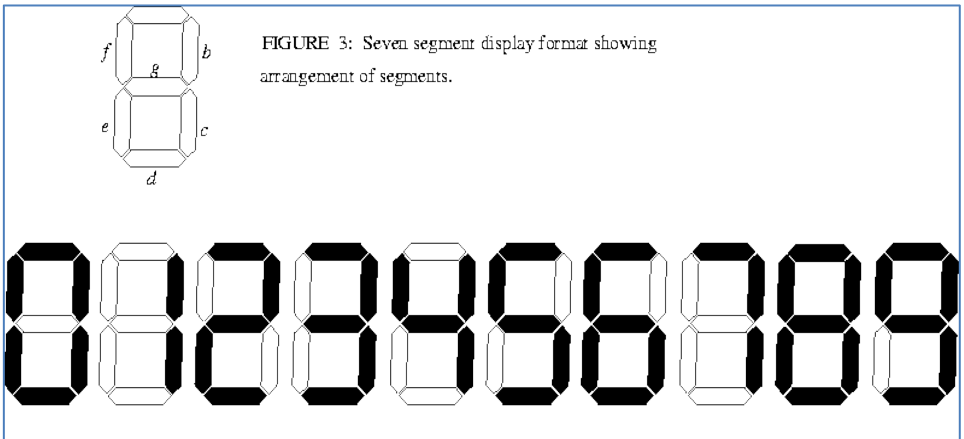
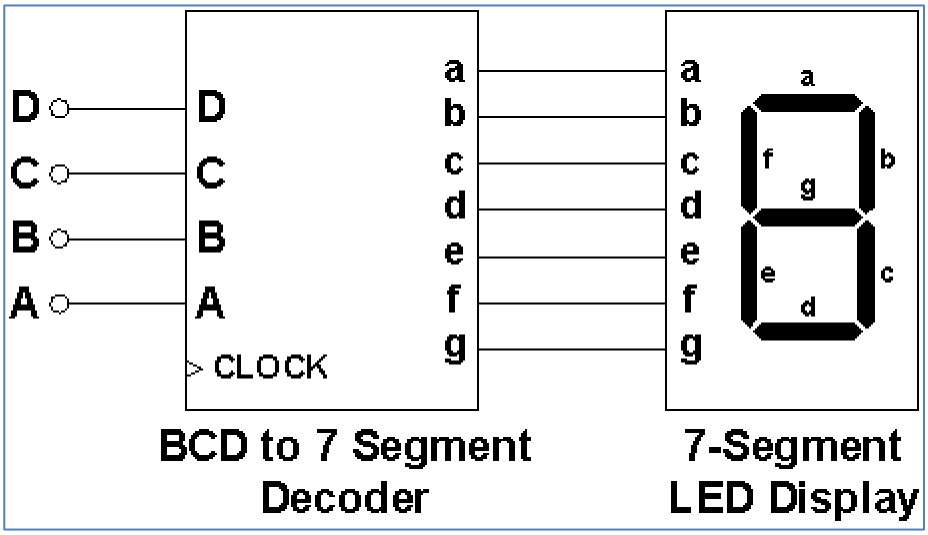


FIGURE 3: Seven segment display format showing arrangement of segments.

FIGURE 4 : Display of decimal digits with a 7-segment device.



BINARY ARITHMETIC

Binary Addition جمع الثنائي

The four basic rules for adding binary digits (bits) are as follows:

0 + 0 = 0	Sum of 0 with a carry of 0
0 + 1 = 1	Sum of 1 with a carry of 0
1 + 0 = 1	Sum of 1 with a carry of 0
1 + 1 = 10	Sum of 0 with a carry of 1

When there is a carry of 1, you have a situation in which three bits are being added (bit in each of the two numbers and a carry bit). This situation has illustrated as follows:

Carry bits

1 + 0 + 0 = 01	Sum of 1 with a carry of 0
1 + 1 + 0 = 10	Sum of 0 with a carry of 1
1 + 0 + 1 = 10	Sum of 0 with a carry of 1
1 + 1 + 1 = 11	Sum of 1 with a carry of 1

Example:

Add the following binary numbers:

(a) 11 + 11 (b) 100 + 10 (c) 111 + 11 (d) 110 + 100

Solution The equivalent decimal addition is also shown for reference.

(a)	11	3	(b)	100	4	(c)	111	7	(d)	110	6
	+11	+3		+10	+2		+11	+3		+100	+4
	110	6		**110**	6		**1010**	10		**1010**	10

Addition in octal جمع الثمانى

When we add two octal number if greater than 7 we subtract 8 from result digit

Example:

$$\begin{array}{r} 71 \\ +47 \\ \hline 140 \end{array}$$

1+7 = 8 >7 than 8-8=0 with carry 1

7+4+1 =12 >7 than 12-8 = 4 with carry 1

Hexadecimal Addition : جمع السادس عشر use the following rules:

1. In any given column of an addition problem, think of the two hexadecimal digits in terms of their decimal values. For instance, $5_{16} = 5_{10}$ and $C_{16} = 12_{10}$.
2. If the sum of these two digits is 15_{10} or less, bring down the corresponding hexadecimal digit.
3. If the sum of these two digits is greater than 15_{10}, bring down the amount of the sum that exceeds 16_{10} and carry a 1 to the next column.

Example:

Add the following hexadecimal numbers:

(a) $23_{16} + 16_{16}$ (b) $58_{16} + 22_{16}$ (c) $2B_{16} + 84_{16}$ (d) $DF_{16} + AC_{16}$

Solution

(a)
$$\begin{array}{r} 23_{16} \\ +16_{16} \\ \hline 39_{16} \end{array}$$
right column: $3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$
left column: $2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$

(b)
$$\begin{array}{r} 58_{16} \\ +22_{16} \\ \hline 7A_{16} \end{array}$$
right column: $8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16}$
left column: $5_{16} + 2_{16} = 5_{10} + 2_{10} = 7_{10} = 7_{16}$

(c)
$$\begin{array}{r} 2B_{16} \\ +84_{16} \\ \hline AF_{16} \end{array}$$
right column: $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$
left column: $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$

(d)
$$\begin{array}{r} DF_{16} \\ +AC_{16} \\ \hline 18B_{16} \end{array}$$
right column: $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$
$27_{10} - 16_{10} = 11_{10} = B_{16}$ with a 1 carry
left column: $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$
$24_{10} - 16_{10} = 8_{10} = 8_{16}$ with a 1 carry

1-3-2 : COMPLEMENTS:

1'S AND 2'S COMPLEMENTS OF BINARY NUMBERS

The l's complement and the 2's complement of a binary number are important because they permit the representation of negative numbers. The method of 2's complement arithmetic has commonly used in computers to handle negative numbers.

Finding the 1's Complement

The 1's complement of a binary number were found by changing all 1s to 0s and all 0s to 1s,

1 0 1 1 0 0 1 0	Binary number
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	
0 1 0 0 1 1 0 1	1's complement

As illustrated:

The 2' s Complement

2's complement = (1's complement) + 1

Example:

Find the 2's complement of 10110010.

Solution

10110010	Binary number
01001101	1's complement
+ 1	Add 1
01001110	2's complement

An alternative method of finding the 2's complement of a binary number is as follows:

1. Start at the right with the LSB and write the bits as they are up to and including the first 1.

2. Take the 1's complements of the remaining bits.

Find the 2's complement of 10111000 using the alternative method.

Solution

	10111000	Binary number
1's complements of original bits →	**01001000**	2's complement
	↑ These bits stay the same.	

1st And 2nd complement in decimal

Express the decimal number −39 as an 8-bit number in the sign-magnitude, 1's complement, and 2's complement forms.

Solution First, write the 8-bit number for +39.

00100111

In the *sign-magnitude form*, −39 is produced by changing the sign bit to a 1 and leaving the magnitude bits as they are. The number is

10100111

In the *1's complement form*, −39 is produced by taking the 1's complement of +39 (00100111).

11011000

In the *2's complement form*, −39 is produced by taking the 2's complement of +39 (00100111) as follows:

11011000	1's complement
+ 1	
11011001	2's complement

1st And 2nd complement in hexadecimal number

Method 1. Convert the hexadecimal number to binary. Take the 2's complement of the binary number. Convert the result to hexadecimal. This is illustrated in Figure 2–4.

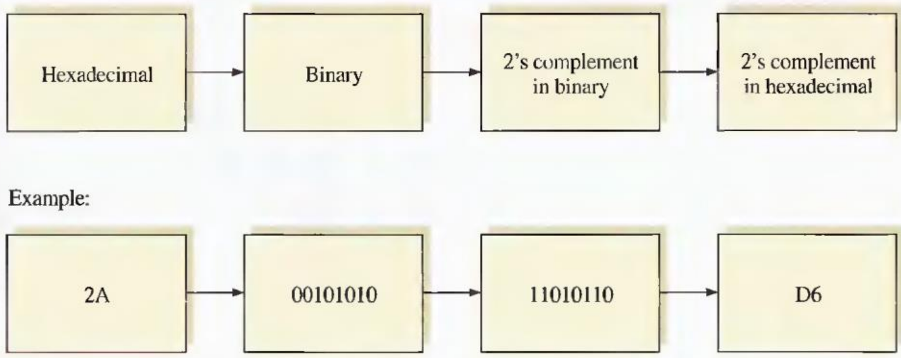


FIGURE 2–4

Getting the 2's complement of a hexadecimal number, Method 1.

Method 2. Subtract the hexadecimal number from the maximum hexadecimal number and add 1. This is illustrated in Figure 2–5.

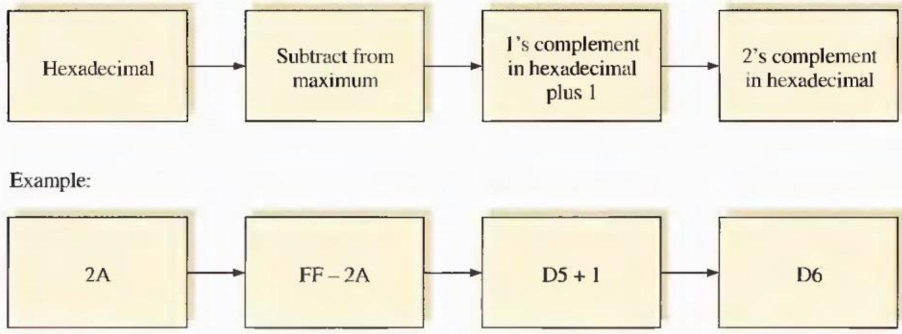


FIGURE 2–5

Getting the 2's complement of a hexadecimal number, Method 2.

Method three :

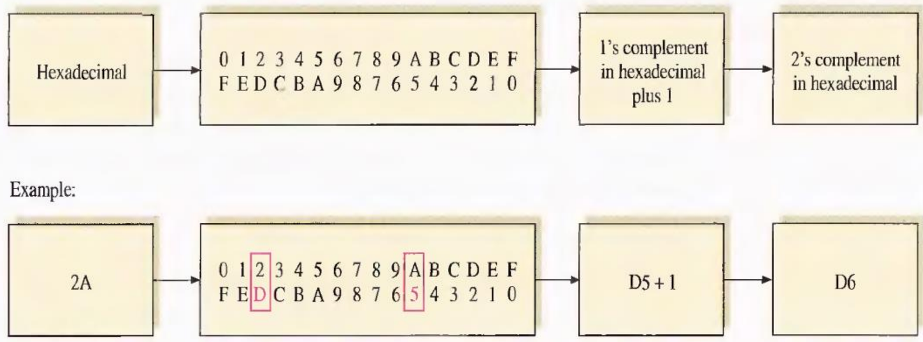


FIGURE 2–6

Getting the 2's complement of a hexadecimal number, Method 3.

Binary Subtraction طرح اعداد الثنائى

Subtraction is addition with the sign of the subtrahend changed, and adds it to the minuend. The result of a subtraction has called the difference.

To subtract two signed numbers, take the 2's complement of the subtrahend and add. Discard any final carry bit.

Example:

Perform each of the following subtractions of the signed numbers:

(a) 00001000 − 00000011 **(b)** 00001100 − 11110111

(c) 11100111 − 00010011 **(d)** 10001000 − 11100010

Solution Like in other examples, the equivalent decimal subtractions are given for reference.

(a) In this case, $8 - 3 = 8 + (-3) = 5$.

	00001000	Minuend (+8)
	+ 11111101	2's complement of subtrahend (−3)
Discard carry →	1 **00000101**	Difference (+5)

(b) In this case, $12 - (-9) = 12 + 9 = 21$.

00001100	Minuend (+12)
+ 00001001	2's complement of subtrahend (+9)
00010101	Difference (+21)

(c) In this case, $-25 - (+19) = -25 + (-19) = -44$.

	11100111	Minuend (−25)
	+ 11101101	2's complement of subtrahend (−19)
Discard carry →	1 **11010100**	Difference (−44)

(d) In this case, $-120 - (-30) = -120 + 30 = -90$.

10001000	Minuend (−120)
+ 00011110	2's complement of subtrahend (+30)
10100110	Difference (−90)

Multiplication الضرب

The sign of the product of a multiplication depends on the signs of the multiplicand and the multiplier according to the following two rules:
If the signs are the same, the product is positive.
If the signs are different, the product is negative.
The basic steps in the partial products method of binary multiplication are as follows:
Step 1. Determine if the signs of the multiplicand and multiplier are the same or different. This determines what the sign of the product will be.
Step 2. Change any negative number to true (uncomplemented) form. Because most computers store negative numbers in 2's complement, a 2's complement operation is required to get the negative number into true form.
Step 3. Starting with the least significant multiplier bit, generate the partial products. When the multiplier bit is 1, the partial product is the same as the multiplicand. When the multiplier bit is 0, the partial product is zero. Shift each successive partial product one bit to the left.
Step 4. Add each successive partial product to the sum of the previous partial products to get the final product.
Step 5. if the sign bit that was determined in step 1 is negative. Take the 2's complement of the product. if positive. Leave the product in true form. Attach the sign bit to the product.

Example

Multiply the signed binary numbers: 01010011 (multiplicand) and 11000101 (multiplier).

Solution **Step 1:** The sign bit of the multiplicand is 0 and the sign bit of the multiplier is 1. The sign bit of the product will be 1 (negative).

Step 2: Take the 2's complement of the multiplier to put it in true form.

11000101 ⟶ 00111011

Steps 3 and 4: The multiplication proceeds as follows. Notice that only the magnitude bits are used in these steps.

1010011	Multiplicand
× 0111011	Multiplier
1010011	1st partial product
+ 1010011	2nd partial product
11111001	Sum of 1st and 2nd
+ 0000000	3rd partial product
011111001	Sum
+ 1010011	4th partial product
1110010001	Sum
+ 1010011	5th partial product
100011000001	Sum
+ 1010011	6th partial product
1001100100001	Sum
+ 0000000	7th partial product
1001100100001	Final product

Step 5: Since the sign of the product is a 1 as determined in step 1, take the 2's complement of the product.

1001100100001 ⟶ 0110011011111

Attach the sign bit ⟶ **1 0110011011111**

Division القسمة

The numbers in a division are the **dividend,** the **divisor,** and the **quotient.** These are illustrated in the following standard division format.

$$\frac{\text{dividend}}{\text{divisor}} = \text{quotient}$$

The sign of the quotient depends on the signs of the dividend and the divisor according to the following two rules:

If the signs are the same, the quotient is positive.

If the signs are different, the quotient is negative.

When two binary numbers are divided, both numbers must be in true (uncomplememed) form. The basic steps in a division process are as follows:

Step 1. Determine if the signs of the dividend and divisor are the same or different. This determines what the sign of the quotient will be. Initialize the quotient to zero.

Step 2. Subtract the divisor from the dividend using 2's complement addition to get the first partial remainder and add 1 to the quotient. If this partial remainder is positive, go to step 3. If the partial remainder is zero or negative, the division is complete.

Step 3. Subtract the divisor from the partial remainder and add 1 to the quotient. If the result is positive, repeat for the next partial remainder. If the result is zero or negative, the division is complete.

Example

Divide 01100100 by 00011001.

Solution **Step 1:** The signs of both numbers are positive, so the quotient will be positive. The quotient is initially zero: 00000000.

Step 2: Subtract the divisor from the dividend using 2's complement addition (remember that final carries are discarded).

01100100	Dividend
+ 11100111	2's complement of divisor
01001011	Positive 1st partial remainder

Add 1 to quotient: 00000000 + 00000001 = 00000001.

Step 3: Subtract the divisor from the 1st partial remainder using 2's complement addition.

01001011	1st partial remainder
+ 11100111	2's complement of divisor
00110010	Positive 2nd partial remainder

Step 4: Subtract the divisor from the 2nd partial remainder using 2's complement addition.

00110010	2nd partial remainder
+ 11100111	2's complement of divisor
00011001	Positive 3rd partial remainder

Add 1 to quotient: 00000010 + 00000001 = 00000011.

Step 5: Subtract the divisor from the 3rd partial remainder using 2's complement addition.

00011001	3rd partial remainder
+ 11100111	2's complement of divisor
00000000	Zero remainder

Add 1 to quotient: 00000011 + 00000001 = **00000100** (final quotient). The process is complete.

Chapter two

- **BINARY CODED DECIMAL (BCD)**: نظام عشري مشفر ثنائي

The 8421 code is a type of BCD (binary coded decimal) code. Binary coded decimal means that each decimal digit, 0 through 9, is represented by a binary code of four bits.

DECIMAL DIGIT	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Example:

Convert each of the following decimal numbers to BCD:

(a) 35 (b) 98 (c) 170 (d) 2469

Solution

(a) 3 5 → $\underbrace{0011}\underbrace{0101}$

(b) 9 8 → $\underbrace{1001}\underbrace{1000}$

(c) 1 7 0 → $\underbrace{0001}\underbrace{0111}\underbrace{0000}$

(d) 2 4 6 9 → $\underbrace{0010}\underbrace{0100}\underbrace{0110}\underbrace{1001}$

جمع الاعدادBCD Addition

Step 1. Add the two BCD numbers, using the rules for binary addition.

Step 2. If a 4-bit sum is equal to or less than 9, it is a valid BCD number.

Step 3. If a 4-bit sum is greater than 9, or if a carry out of the 4-bit group is generated, it is an invalid result. Add 6 (0110) to the 4-bit sum in order to skip the six invalid states and return the code to 8421. If a carry results when 6 is added. simply add the carry to the next 4-bit group.

Examples

(a)		1001		9
		+ 0100		+4
		1101	Invalid BCD number (>9)	13
		+ 0110	Add 6	
	0001	0011	Valid BCD number	
	↓	↓		
	1	3		

(d)	0100	0101	0000	450
	+ 0100	0001	0111	+ 417
	1000	0110	0111	867

Exess 3: اكسيس 3 كود

The *Excess–3* code also uses 4 bits to represent the decimal numbers 0 through 9 and these are shown in the Table 4.2.

TABLE 4.2 The Excess–3 code

Decimal	Excess–3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

The Excess–3 Code derives its name from the fact that each decimal representation in Excess–3 code is larger than the BCD code by three. The advantage of the Excess–3 code over the BCD code is that the Excess–3 code is a *self–complementing code* as illustrated below.

- **The Gray Code شفره منعكسه كود**

The Gray code is un weighted and is not an arithmetic code; that is, there are no specific weights assigned to the bit positions.

DECIMAL	BINARY	GRAY CODE	DECIMAL	BINARY	GRAY CODE
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Binary-to-Gray Code Conversion تحويل الثنائي الى كود الشفرة المنعكسة

1. The most significant bit (left-most) in the Gray code is the same as the corresponding MSB in the binary number.
2. Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit. Discard carries.

For example, the conversion of the binary number 10110 to Gray code is as follows:

$$\begin{array}{ccccccccccc} 1 & - + \rightarrow & 0 & - + \rightarrow & 1 & - + \rightarrow & 1 & - + \rightarrow & 0 & & \text{Binary} \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\ 1 & & 1 & & 1 & & 0 & & 1 & & \text{Gray} \end{array}$$

The Gray code is 11101.

Gray-to-Binary Conversion تحويل الثنائي الى كود الشفرة المنعكسة

1. The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Gray code.
2. Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.

For example, the conversion of the Gray code word 11011 to binary is as follows:

$$\begin{array}{ccccccccccc} 1 & & 1 & & 0 & & 1 & & 1 & & \text{Gray} \\ \downarrow & + \nearrow & \downarrow & + \nearrow & \downarrow & + \nearrow & \downarrow & + \nearrow & \downarrow & & \\ 1 & & 0 & & 0 & & 1 & & 0 & & \text{Binary} \end{array}$$

The binary number is 10010.

Parity code البت المكافئ

A given system operates with even or odd **parity,** but not both. For instance, if a system operates with even parity, a check is made on each group of bits received to make sure the total number of 1s in that group is even. If there is an odd number of 1s, an error has occurred.

As an illustration of how parity bits are attached to a code, Table 2–10 lists the parity bits for each BCD number for both even and odd parity. The parity bit for each BCD number is in the *P* column.

EVEN PARITY		ODD PARITY	
P	BCD	*P*	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

TABLE 2–10

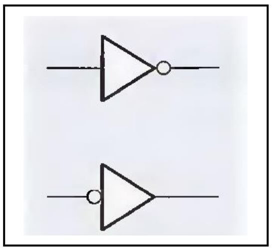
The BCD code with parity bits.

The parity bit can be attached to the code at either the beginning or the end, depending on system design. Notice that the total number of 1s, including the parity bit, is always even for even parity and always odd for odd parity.

Chapter three
Boolean algebra الجبر البولينى

- **INVERTER**

Standard logic symbols for the inverter are shown in Figure below:



Inverter Truth Table جدول الحقيقة بوابه العاكس

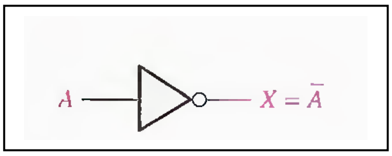
When a HIGH level is applied to an inverter input, a LOW level will appear on its output. When a LOW level is applied to its input, a HIGH will appear on its output.

INPUT	OUTPUT
LOW (0)	HIGH (1)
HIGH (1)	LOW (0)

logic Expression for an Inverter

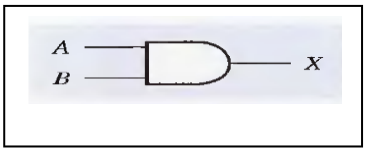
In Boolean algebra, which is the mathematics of logic circuits the operation of an inverter (NOT circuit) can be expressed as follows: If the input variable is called, **A** and the output variable is called X, then

$$X = \overline{A}$$



- **AND GATE بوابه**

An AND gate produces a HIGH output only when all of the inputs are HIGH, otherwise any of the inputs is LOW or both low, the output is LOW.



AND Gate Truth Table جدول الحقيقة

The total number of possible combinations of binary inputs to a gate is determined by the following formula:

$$N= 2^n$$

The operation of a 2-input AND gate can be expressed in equation form as follows: If one input variable is A, the other input variable is B, and the output variable is X, then the Boolean expression is

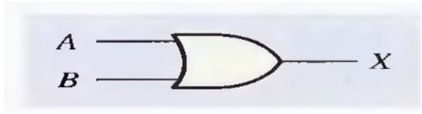
$$X=A.B$$

INPUTS		OUTPUT
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

1 = HIGH, 0 = LOW

- **OR GATE بوابه الاور**

An OR gate symbol as shown in figure



OR Gate Truth Table جدول الحقيقه بوابه الاور

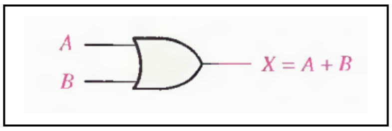
For a 2-input OR gate, output X is HIGH when either input A or input B is HIGH, or when both A and B are HIGH; X is LOW only when both A and B are LOW.

INPUTS		OUTPUT
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

1 = HIGH, 0 = LOW

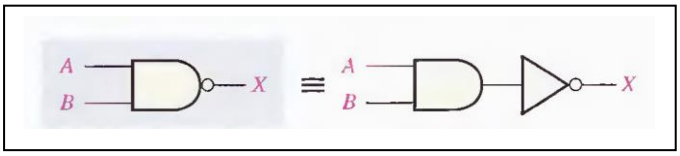
Logic Expressions for an OR Gate الرسم المنطقي لبوابه الاور

The logical OR function of two variables is represented mathematically by a (+) between the two variables, for example, A + B.



NAND GATE بوابه الاند

The term NAND is a contraction of NOT-AND and implies an AND function with a complemented (inverted) output.



Operation of a NAND Gate

For a 2-input NAND gate, output X is LOW only when inputs A and B are HIGH; X is HIGH when either A or B is LOW, or when both A and B are LOW.

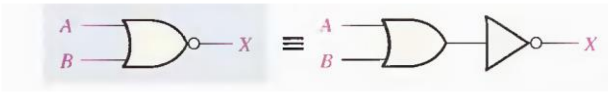
Logic Expressions for a NAND Gate

INPUTS		OUTPUT
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

1 = HIGH, 0 = LOW.

NOR GATE

The NOR is the same as the OR except the output is inverted.



For a 2-input NOR gate, output X is LOW when either input A or input B is HIGH, or when both A and B are HIGH; X is HIGH only when both A and B are LOW.

Logic Expressions for a NOR Gate

The Boolean expression for the output of a 2-input NOR gate can be written as

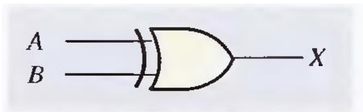
$$X = \overline{A + B}$$

INPUTS A	INPUTS B	OUTPUT X
0	0	1
0	1	0
1	0	0
1	1	0

1 = HIGH, 0 = LOW.

- **Exclusive-OR Gate**

Standard symbols for an exclusive-OR (XOR) gate is shown in Figure

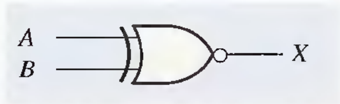


Output X is HIGH when input A is LOW and input B is HIGH, or when input A is HIGH and input B is LOW: X is LOW when A and B are both HIGH or both LOW.

INPUTS A	INPUTS B	OUTPUT X
0	0	0
0	1	1
1	0	1
1	1	0

- **Exclusive-NOR Gate**

Standard symbols for an exclusive-NOR (XNOR) gate are shown in Figure
Output X is LOW when input A is LOW and input B is HIGH, or when A is HIGH and B is LOW; X is HIGH when A and B are both HIGH or both LOW.



INPUTS		OUTPUT
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

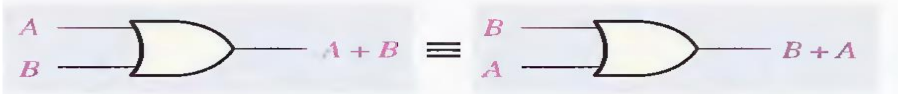
- **LAWS AND RULES OF BOOLEAN ALGEBRA**

Laws of Boolean Algebra

Equation 1

Commutative Laws The *commutative law of addition* for two variables is written as

$$A + B = B + A$$



Equation 2

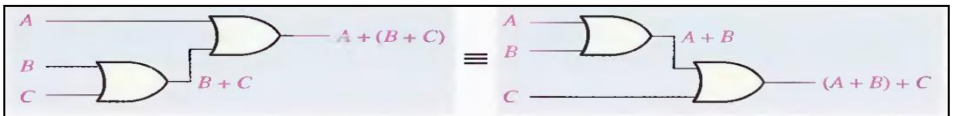
The *commutative law of multiplication* for two variables is

$$AB = BA$$

Equation 3

Associative Laws The *associative law of addition* is written as follows for three variables:

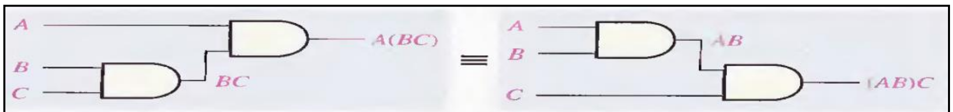
$$A + (B + C) = (A + B) + C$$



Equation 4

The *associative law of multiplication* is written as follows for three variables:

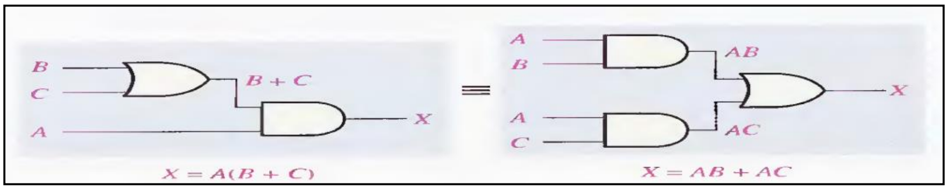
$$A(BC) = (AB)C$$



Equation 5

Distributive Law The distributive law is written for three variables as follows

$A(B + C) = AB + AC$



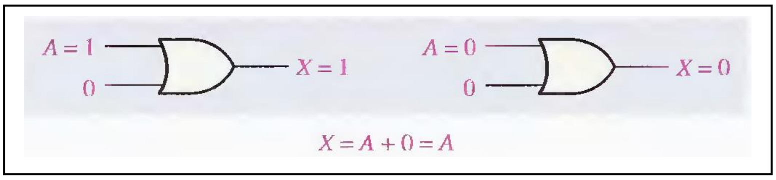
Rules of Boolean algebra

Table below lists 12 basic rules that are useful in manipulating and simplifying Boolean expressions.

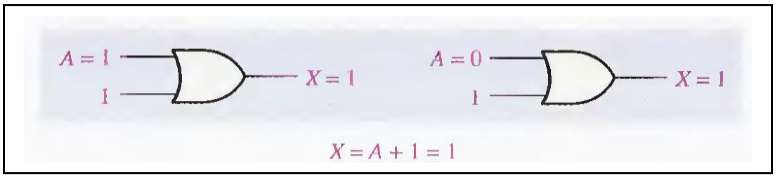
1. $A + 0 = A$	7. $A \cdot A = A$
2. $A + 1 = 1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A + AB = A$
5. $A + A = A$	11. $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	12. $(A + B)(A + C) = A + BC$

A, B, or C can represent a single variable or a combination of variables.

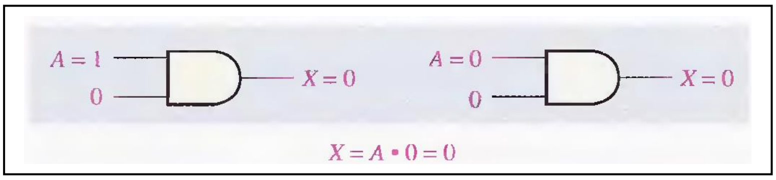
Rule 1. <u>A + 0 = A</u> : A variable OR with 0 is always equal to the variable.



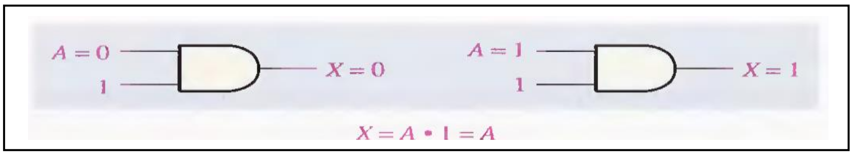
Rule 2. <u>A + 1 = 1</u> : A variable OR with 1 is always equal to 1.



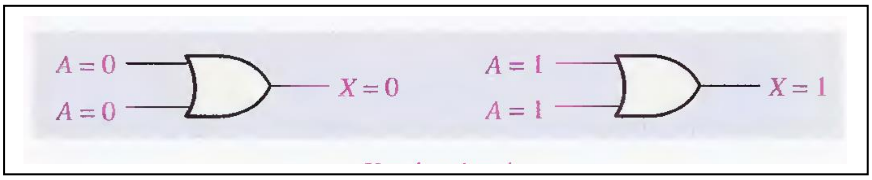
Rule 3. <u>A **.** 0 = 0</u> A variable AND with 0 is always equal to 0.



Rule 4. <u>A . 1 = A</u> :A variable AND with 1 is always equal to the variable.

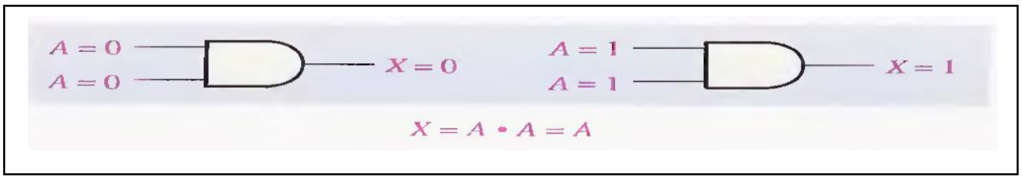


Rule 5. A + A = A : A variable OR with itself is always equal to the variable.

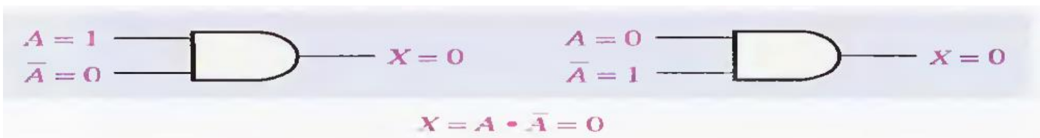


Rule 7. **<u>A . A = A</u>**: A variable AND with itself is always equal to the variable.

Rule 6. $A+\bar{A}=1$ A variable ORed with its complement is always equal to 1



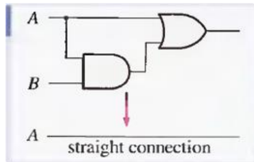
Rule 8. $A\cdot\bar{A}=0$ A variable ANDed with its complement is always equal to 0.

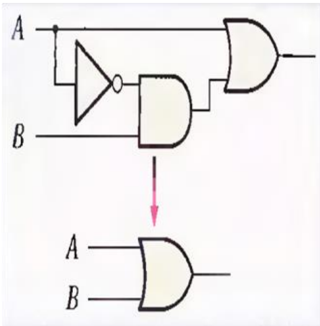


Rule 9. $\bar{\bar{A}}=A$ The double complement of a variable is always equal to the variable.

Rule 10. $A + AB = A$ This rule can be proved by applying the distributive law, rule 2, and rule 4 as follows:

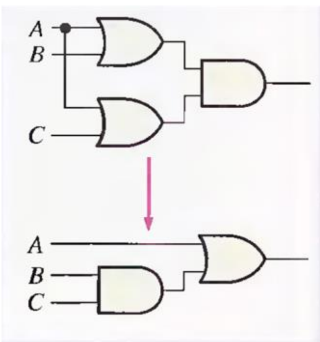
$$
\begin{aligned}
A + AB &= A(1 + B) && \text{Factoring (distributive law)} \\
&= A \cdot 1 && \text{Rule 2: } (1 + B) = 1 \\
&= A && \text{Rule 4: } A \cdot 1 = A
\end{aligned}
$$





Rule 11. $A + \bar{A}B = A + B$ This rule can be proved as follows:

$$
\begin{aligned}
A + \bar{A}B &= (A + AB) + \bar{A}B && \text{Rule 10: } A = A + AB \\
&= (AA + AB) + \bar{A}B && \text{Rule 7: } A = AA \\
&= AA + AB + A\bar{A} + \bar{A}B && \text{Rule 8: adding } A\bar{A} = 0 \\
&= (A + \bar{A})(A + B) && \text{Factoring} \\
&= 1 \cdot (A + B) && \text{Rule 6: } A + \bar{A} = 1 \\
&= A + B && \text{Rule 4: drop the 1}
\end{aligned}
$$



Rule 12. $(A + B)(A + C) = A + BC$ This rule can be proved as follows:

$$
\begin{aligned}
(A + B)(A + C) &= AA + AC + AB + BC && \text{Distributive law} \\
&= A + AC + AB + BC && \text{Rule 7: } AA = A \\
&= A(1 + C) + AB + BC && \text{Factoring (distributive law)} \\
&= A \cdot 1 + AB + BC && \text{Rule 2: } 1 + C = 1 \\
&= A(1 + B) + BC && \text{Factoring (distributive law)} \\
&= A \cdot 1 + BC && \text{Rule 2: } 1 + B = 1 \\
&= A + BC && \text{Rule 4: } A \cdot 1 = A
\end{aligned}
$$