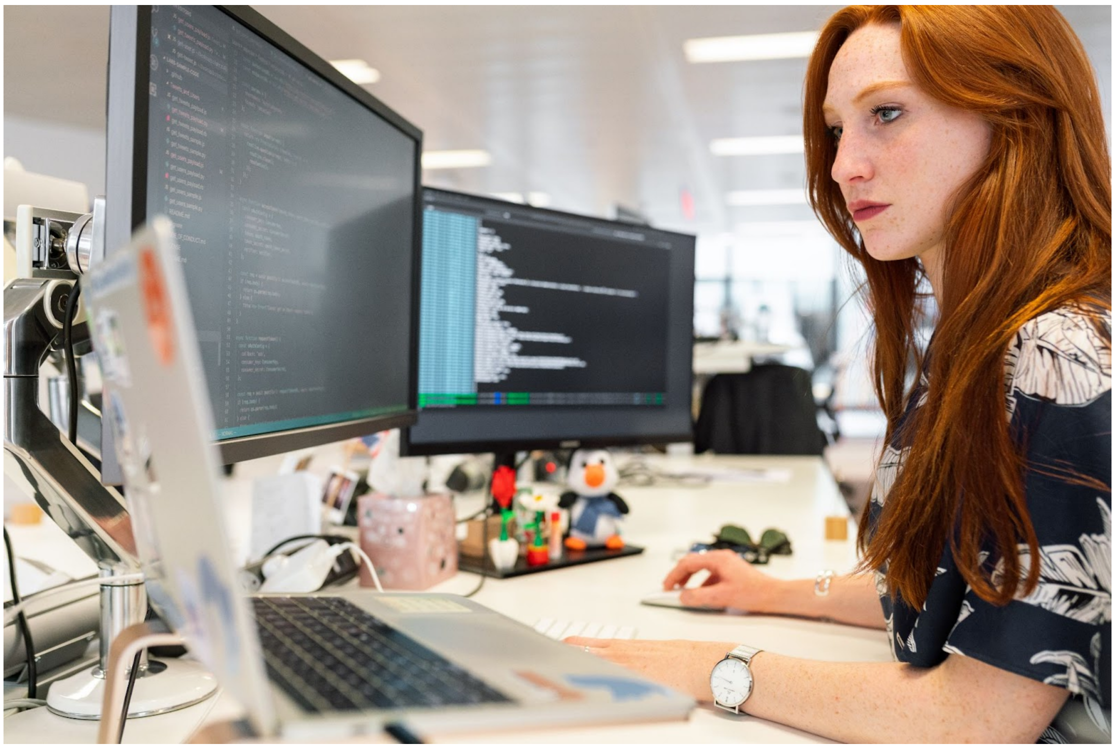


Computing & Informatics

NUMBER SYSTEM & BOOLEAN ALGEBRA

Number System and Boolean Algebra

Number System

There are many systems in which numbers can be expressed. The decimal number system is familiar to us. In this system the base is 10 and the digits are 0,1,2,3,4,5,6,7,8,9.

The following are other number systems which are more popular in use:

1.	Binary system	radix	2	2^1
2.	Octal system	radix	8	2^3
3.	Hexa-decimal system	radix	16	2^4

Most of the digital computers use binary, octal and Hexa-decimal systems.

BINARY ARITHMETIC

In practice, we use both positive and negative numbers as operands. In the binary system, we represent the sign of number using an extra bit at the extreme left of the number. By convention the symbol '0' is used to represent left the '+' sign and '1' to represent the '-' sign. For instance, +6 is represent by 0,110 and -7 is represented by 1,111. This method is known as sign magnitude representation.

When we have to subtract a number B from a number A, we look at the magnitude as well as the sign of these numbers. When A and B have opposite signs, in effect we add their magnitudes and determine the sign of the result. On the other hand, if A and B have the same sign, we always subtract the smaller magnitude from the large and once again, decide the sign independently. similarly, when we have to add, the procedures for which are different.

It would, be more convenient if we could evolve another convention for representing positive and negative numbers which would allow us to use one basic procedure for both addition and subtraction. So we could use a single electronic circuit to implement both addition and subtraction. a convention for representing negative numbers which allows this 'complement representation' of numbers.

OCTAL SYSTEMS

Binary system is very convenient system for the present computers as they deal in terms of 0's and 1's. The representation of binary numbers to external world can be made compact with the help of octal and hexadecimal systems. A group of 3 bits will be able to represent $2^3 = 8$ different possibilities. We can use eight symbols say 0,1,2,3,4,5,6 & 7 to represent these sequences of 3 bits. Such a system of eight symbols is called **octal system**. Similarly, a group of 4 bits will be representing $2^4 = 16$ combinations. To represent 16 sequences of 4 bit each we can use 16 symbols say 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E & F. decimal system equivalent of A is 10, B is 11, c is 12, D is 13, E is 14 and F is 15. A number system with 16 symbols is known as **hexadecimal** are also positional number systems.

Octal to Decimal. To convert as octal number to a decimal number we use the polynomial but this time the radix will be 8. For example, 234 in octal is $2x8^2 + 3x8^1 + 4x8^0 = 128 + 24 + 4 = 156$ in decimal. Octal number 15.24 is $1x8^1 + 5x8^0 + 2x8^{-1} + 4x8^{-2} = 8 + 5 + 2/8 + 4/64 = 13.3125$ in decimal.

Binary to Octal. Since 3 bits are taken at a time is one octal digit, a given binary number can be directly converted in octal by making group of three size bits each starting from octal point. For instance, 101011 in binary is 53 in octal, 1110001 in binary is 161 in octal (add two leading zeros to complete the last group of 3 bits), (1110100.0100111 in binary is 164.234 in octal.

Octal to Binary. An octal number can be converted into its binary equivalent by replacing each octal digit with its three-bit binary equivalent. We take the three-bit equivalent because the base of the octal number system is 8 and it is the third power of the base of the binary number system, i.e. 2. All we have then to remember is the three-bit binary equivalents of the basic digits of the octal number system. (i) Create a look up table with 8 entries and their corresponding 3 bit binary code (ii) separate each octal digit from the given number and using look up table find binary value and print it (iii) repeat step (ii) until all the digits are converted into binary.

HEXADECIMAL NUMBERS.

In principal, Hexadecimal numbers work in the exact same way as decimal numbers do. Nearly all the same rules apply. The main difference is that there are more symbols. In the case of Hexadecem, Latin for 16, there are 16. The first ten are old faithful 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The six new ones are A, B, C, D, E, F. Just like we made a little counting table for Decimal, we can make one for Hexadecimal as well:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F

Apart from the difference in the symbols and the different number of them, the Math rules in Hexadecimal, which we will from now on simply call Hex, are the same as the rules in Decimal. In Hexadecimal math, 4+1=5 and 6 + 3 = 9 and 9 + 3 = C. As in Decimal math, you add an MSD to the left in Hex when you run out of Symbols, example in Hex: C +5 = 11 and Hex 11 + 3C = 4D. Although this sounds simple and logical, our brain has trouble with it since we are trained in decimal!

Following table shows equivalences between hexagonal, binary and decimal digits.

Decimal	**Binary**	**Hexagonal**
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

Hexadecimal Arithmetic. Covert hexadecimal number to binary number and then apply arithmetic operations.

RADIX

The total number of digits applicable to any system is called its *radix*. It is always one more than the highest digit of the system. The digits of various types of number system are as under:

1.	Binary system	0,1
2.	Octal system	0, 1, 2, 3, 4, 5, 6, 7
3.	Decimal system	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
4.	Hexa-Decimal system	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

CONVERSION OF NUMBERS

Conversion of a number from any system to decimal system

Let us study the formation of the number 1970 in decimal system.

$$(1970)_{10} = 1\text{x}10^3 + 9\text{x}10^2 + 7\text{x}10^1 + 0\text{x}10^0$$
$$= 1000 + 900 + 70 + 0 = 1970$$

(i) Convert binary number $(111)_2$ into decimal system.

$$(111)_2 = 1\text{x}2^2 + 1\text{x}2^1 + 1\text{x}2^0 = 4 + 2 + 1 = (7)_{10}$$

(ii) Convert octal number $(1654)_8$ into decimal system

$$(1654)_8 = 1\text{x}8^3 + 6\text{x}8^2 + 5\text{x}8^1 + 4\text{x}8^0 = 512 + 384 + 40 + 4 = (940)_{10}$$

(iii) Convert Hexa-decimal number $(123)_{16}$ to decimal system

$$(123)_{16} = 1\text{x}16^2 + 2\text{x}16^1 + 3\text{x}16^0 = 256 + 32 + 3 = (291)_{10}$$

Conversion of a number from decimal system to other systems

The number of any system, may be converted to decimal system by the following two methods.

1. Remainder method
2. Power method

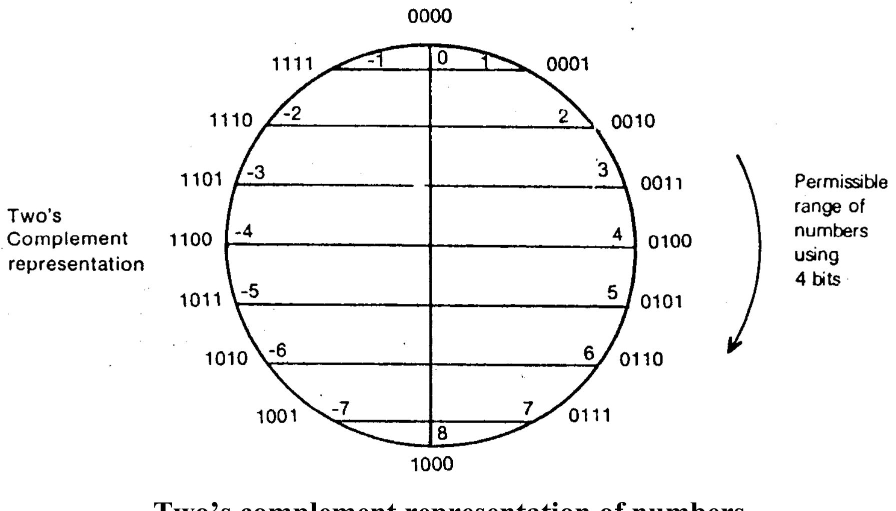
These methods will be discussed in the classified examples.

TWO'S COMPLEMENT REPRESENTATION OF NUMBERS

The 2's complement of a binary number x, which has n bits, is given by (2^n- x). Consider the number + 3 whose binary representation is 011. It has three bits. Hence the 2's complement of + 3 is (2^3- 3), which is 5. The binary representation of 5 is 101. Thus, 101 is the 2's complement of 011.

In a computer, all numbers are represented in a uniform fashion using a fixed number of bits. Thus, for an n-bit machine, the range of numbers it can handle is 0 to 2^n- 1. For simplicity, consider a 4 bit machine. Sixteen numbers (0 to 15) can normally be represented using these four bits. Now we devise a new scheme of representing negative numbers as follows. We use the first seven combinations of bits for representing positive numbers one to seven. We reserve seven of the remaining combinations for representing negative number -1 to -7. Thus, we have divided the total range into two parts, 0 and 8 being common to the two halves. Now we restrict ourselves to the use of numbers having a maximum magnitude of seven. In this way we can represent both positive and negative numbers as indicated in Fig. below. With reference to the figure we see that the code 1111, which normally represents 15, is assigned to -1. Similarly the binary equivalent of 14 is assigned to -2. Since 15 = 2^4- 1, 14 = 2^4- 2, etc., this is called the 2's complement representation.

There is a simple procedure to obtain the 2's complement of a binary number. We first complement each bit of the number (i.e. replace '1' by '0' and '0' by '1'). Now we add a '1' to the number. For example, consider the number 5 whose binary representation is 1011. Bit complementation yields 1010. Now adding a '1' to this number gives 1011 which is the 2's complement representation for -5.



Two's complement representation of numbers

Yet another method of obtaining the 2's complement of a binary number is to scan the number from right to left and complement all bits appearing after the first appearance of a '1'. For example the 2's complement of 0010 is 1110.

Example

Convert $(940)_{10}$ to binary system.

Solution

Remainder Method

The radix of binary system is 2.

2	940	
2	470	0
2	235	0
2	117	1
2	58	1
2	29	0
2	14	1
2	7	0
2	3	1
2	1	1

i.e. $(940)_{10} = (1110101100)_2$

Example

Convert $(2040)_{10}$ to octal system.

Solution

Remainder Method

The radix of octal system is 8.

8	2040	
8	255	0
8	31	7
	3	7

i.e. $(2040)_{10} = (3770)_8$.

Example

Convert $(2040)_{10}$ to hexa-decimal system.

Solution

Remainder Method

The radix of hexa-decimal system is 16.

16	2040	
16	127	**8**
	7	**15 = F**

i.e. $(2040)_{10} = (7F8)_{16}$.

Problem (AMIE Winter 2001)

Convert following decimal number to hexadecimal number:

395

Answer: $(18B)_{16}$

Example

Obtain the octal equivalent of $(3964)_{10}$.

Solution

Remainder Method

Radix of the octal system is 8.

8	3964	
8	495	4
8	61	7
	7	5

i.e. $(3964)_{10} = (7574)_8$.

Example

Convert $(300)_{10}$ to the binary system.

Solution

Power Method

Here radix of binary system is 2.

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

We note that 256 is the highest number which is less than 300, the given number

```
 300   2^8  = 1 x 2^8
- 256
 ----
  44
- 32   2^5  = 1 x 2^5
 ----
  12
- 8    2^3  = 1 x 2^3
 ----
  4
- 4    2^2  = 1 x 2^2
 ----
  0
```

As powers of 2^7, 2^6, 2^4, 2^1, 2^0 are not represented, these values may be placed in their respective places after multiplying with zero i.e.

$1 \times 2^8,\ 0 \times 2^7,\ 0 \times 2^6,\ 1 \times 2^5,\ 0 \times 2^4,\ 1 \times 2^3,\ 1 \times 2^2,\ 0 \times 2^1,\ 0 \times 2^0$

i.e. $(300)_{10} = (100101100)$.

Example

Convert $(300)_{10}$ to octal system.

Solution

Power Method

The radix is 8

8^5	8^4	8^3	8^2	8^1	8^0
32768	4096	512	64	8	1

We note that 256 is the highest number i.e., 4×8^2 .

```
 300   4 x 8^2
- 256
 ----
  44
- 40   5 x 8^1
 ----
  4
- 4    4 x 8^0
 ----
  0
```

i.e. $(300)_{10} = (454)_8$.

Example

Convert $(1654)_8$ into binary system.

Solution:

Power Method

First convert the given number to decimal system.

Then, convert the number from decimal system to binary system.

i.e. $(1654)_8 = 1 \text{ x } 8^3 + 6 \text{ x } 8^2 + 5 \text{ x } 8^1 + 4 \text{ x } 8^0$

$= 512 + 348 + 40 + 4$

$= (940)_{10}$

Now, convert $(940)_{10}$ to binary system as in example 1.1.

i.e. $(1654)_{10} = (940)_{10} = (1110101100)_2$

Conversion of fractions of decimal system to other systems

Example

Convert $(0.25)_{10}$ to the binary system.

Solution

$0.25 \text{ x } 2 = 0.50$ $I_0 = 0$

$0.50 \text{ x } 2 = 1.00$ $I_2 = 1$

$(0.25)_{10} = (0.01)_2$

Example

Convert (634.640625) to octal equivalent.

Solution

Integer part is $(634)_{10}$

8	634	
8	79	2
8	9	7
	1	1

$\therefore$ $(634)_{10} = (1172)_8$.

Fraction part is 0.640625

$0.640625 \text{ x } 8 = 5.125$ 5

$0.125 \times 8 = 1.0$ 1

i.e. Octal equivalent = $(1172.51)_8$

Conversion of a fraction of any system with radix R to the decimal system

Example

Convert $(0.135)_8$ to the decimal system.

Solution

Radix is 8.

$$(0.135)8 = 1 \times 8^{-1} + 3 \times 8^{-2} + 5 \times 8^{-3}$$

$$= \frac{1}{8} + \frac{3}{64} + \frac{2}{512} = \frac{93}{512} = (0.1816406)_{10}$$

Example

Convert $(3.75)_8$ to binary system.

Solution

The integer part 3 = $(11)_2$.

The fraction part (0.75) any first be converted to the decimal system.

i.e. $(0.75)_8 = 7 \times 8^{-1} + 5 \times 8^{-2}$

$$= \frac{7}{8} + \frac{5}{64} = \frac{61}{64} = (0.953125)_{10}$$

i.e. $(0.75)_8 = (0.953125)_{10}$.

Now convert $(0.953125)_{10}$ to the binary system as under:

0.953125×2	$= 1.90620$	$I_1 = 1$
0.906250×2	$= 1.18125$	$I_2 = 1$
0.8125×2	$= 1.6250$	$I_3 = 1$
0.625×2	$= 1.250$	$I_4 = 1$
0.25×2	$= 0.50$	$I_5 = 0$
0.50×2	$= 1.00$	$I_6 = 1$

$\therefore$ $(0.953125)_{10} = (0.111101)_2$

$\therefore$ $(3.75)_8 = (11.111101)_2$

Problem (AMIE WINTER 98)

Convert

(i) 01011.101 binary to decimal

(ii) 22.75 decimal to binary

Answer: (i) 11.625 (ii) 1 0110.11

Direct method of conversion the number from one system to the other

For Converting the digits of decimal system into binary system, use Table 1.

Table 1

Decimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

From the above table, it is evident that a decimal digit may have at the most four bits.

For converting the digits of binary system into-decimal system, use Table 2

Table 2

Decimal	Binary	Hexa-decimal
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

For converting the digits of binary system into octal system, use the Table 3

Table 3

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

Example

Convert $(1352)_8$ into binary system.

Solution

Here radix $8 = 2^3$

From pairs of each digit of the octal system with binary system.

$\therefore$ $(1352)_8 = (001,\ 011,\ 101,\ 010)_2$

Example

Convert $(CD578)_{16}$ into binary system.

Solution

Here $16 = 2^4$, power is 4

Form pairs of each digit of the hexa-decimal system into binary equivalents.

C = 12 $(12)_2 = 1100$

D = 13 $(13)_2 = 1101$

$(5)_2 = 0101$

$(7)_2 = 0111$

$(8)_2 = 1000$

$\therefore$ The required number is $(1100, 1101, 0101, 0111, 1000)_2$.

Example

Convert the binary number 1010011001.1011001 into octal system.

Solution

Radix $8 = 2^3$

```
001   010   011   001   101   100   100

 1     2     3     1     5     4     4
```

∴ $(001010011001.101100100)_2 = (1231.544)_8$

Example (AMIE Summer 2008, 8 marks)

Give an algorithm to convert an octal number into a binary number. Using an example, explain how the algorithm works.

Solution

Following steps are followed

(i) Create a look up table with 8 entries and their corresponding 3 bit binary code as given below:

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

(ii) Separate each octal digit from the given number and using look up table, find binary value and print it.

(iii) Repeat (ii) until all the digits are converted into binary.

Example: $(642.71)_8 = (110100010.111001)_2$

Example (AMIE Summer 2014, 5 marks)

Write an algorithm to convert the decimal numbers into binary numbers. Explain your algorithm.

Solution

Following steps describe how to convert decimal to binary

Step 1: Divide the original decimal number by 2

Step 2: Divide the quotient by 2

Step 3: Repeat the step 2 until we get quotient equal to zero.

Equivalent binary number would be remainders of each step in the reverse order.

Decimal to binary conversion with example:

For example we want to convert decimal number 25 in the binary.

Step 1: 25 / 2 Remainder : 1 , Quotient : 12

Step 2: 12 / 2 Remainder : 0 , Quotient : 6

Step 3: 6 / 2 Remainder : 0 , Quotient : 3

Step 4: 3 / 2 Remainder : 1 , Quotient : 1

Step 5: 1 / 2 Remainder : 1 , Quotient : 0

So equivalent binary number is: 11001

That is $(25)_{10} = (11001)_2$

Example (AMIE Summer 2014, 5 marks)

Write an algorithm to convert the binary numbers into octal numbers. Explain your algorithm.

Solution

Binary to octal conversion method:

Step1: Arrange the binary number in the group 3 from right side.

Step 2: Replace the each group with following values:

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

Binary to octal conversion examples:

For example we want to convert binary number 1011010101001101 to octal.

Step 1: 001 011 010 101 001 101

Step 2: 1 3 2 5 1 5

So $(1011010101001101)_2 = (132515)_8$

Problem

Convert $(294.6875)_{10}$ into octal.

Answer: $(446.54)_8$

Problem

Convert $(642.71)_8$ into binary.

Answer: $(110100010.111001)_2$

Problem

Convert the following numbers to their hexadecimal equivalent

(i) $(49.5)_{10}$

(ii) $(972.625)_{10}$

Answer: (i) $(31.8)_{16}$ (ii) $(3CC.A)_{16}$

BINARY CALCULATIONS

The addition, subtraction, multiplication and division of the binary numbers, may be done as in the case of decimal system.

Addition

Example

Add $(110101)_2$ and $(101111)_2$.

Solution

$101111_2 = 47_{10}$

$110101_2 = 53_{10}$

Their sum

$47 + 53 = 100$

Result in binary form

$100_{10} = 1100100_2$

Subtraction

Example

Subtract $(100101100)_2$ from $(1110101010)_2$.

Solution

$(1110101010)_2 - (100101100)_2 = 938_{10} - 300_{10} = 638_{10}$

$638_{10} = 1001111110_2$

Multiplication

Example

Multiplying $(111)_2$ by $(101)_2$.

Solution

$111_2 = 7_{10}$

$101_2 = 5_{10}$

Their product

7 x 5 = 35

Result in binary form

$35_{10} = 100011_2$

Problem

Multiply 1101 by 1010

Answer: 10000010

Division

Example

Dividing $(100011)_2$ by $(101)_2$

Solution

$100011_2 = 35_{10}$

$101_2 = 5_{10}$

35/5 = 7

Result in binary form

$7_{10} = 111_2$

Problem

Divide 1011011 by 111

Answer: 1101

Example (AMIE W2001)

Write following decimal number to BCD form

1782

Solution

To find BCD form of a decimal number convert each digit of the decimal number into its 4 bit binary form

$(1782)_{10} = (0001 \quad 0111 \quad 1000 \quad 0010)_{BCD}$

Example (AMIE Summer 2001)

$(F9A.BC3)_{16} = (?)_{10} = (?)_2$

Solution

$$(F9A.BC3)_{16} = F \times 16^2 + 9 \times 16^1 + A \times 16^0 + B \times 16^{-1} + C \times 16^{-2} + 3 \times 16^{-3}$$

$$= (15 \times 56) + (9 \times 16) + (10 \times 1) + (11/16) + (12/256) + (3/4096)$$

$$= 3840 + 144 + 10 + (11/16) + (12/256) + (3/4096)$$

$$= (3994.7351074)_{10}$$

(F9A.BC3)=	F	9	A	.	B	C	3
	1111	1001	1010		1011	1100	0011

$= (1111 \; 1001 \; 1010.1011 \; 1100 \; 0011)_2$

FLOATING POINT NUMBERS

Floating-point notation can be used conveniently to represent both large as well as small fractional or mixed numbers. This makes the process of arithmetic operations on these numbers relatively much easier. Floating-point representation greatly increases the range of numbers, from the smallest to the largest, that can be represented using a given number of digits. Floating-point numbers are in general expressed in the form

$$N = m \, x \, b^e \qquad (1)$$

where m is the fractional part, called the *significand* or *mantissa*, e is the integer part, called the exponent, and b is the base of the number system or numeration. Fractional part m is a p-digit number of the form (±d.dddd......dd), with each digit d being an integer between 0 and b – 1 inclusive. If the leading digit of m is nonzero, then the number is said to be normalized.

Equation (1) in the case of decimal, hexadecimal and binary number systems will be written as follows:

Decimal system

$$N = m\,x 10^e$$

Hexadecimal system

$$N = m\,x 16^e$$

Binary system

$$N = m\,x 2^e$$

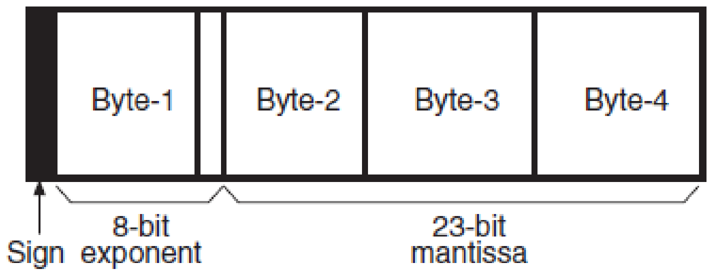
For example, decimal numbers 0.0003754 and 3754 will be represented in floating-point notation as 3.754×10^{-4} and 3.754×10^3 respectively. A hex number 257.ABF will be represented as $2.57ABF \times 16^2$. In the case of normalized binary numbers, the leading digit, which is the most significant bit, is always '1' and thus does not need to be stored explicitly.

Also, while expressing a given mixed binary number as a floating-point number, the radix point is so shifted as to have the most significant bit immediately to the right of the radix point as a '1'. Both the mantissa and the exponent can have a positive or a negative value.

The mixed binary number $(110.1011)_2$ will be represented in floating-point notation as $.1101011 \times 2^3$ = .1101011e + 0011. Here, .1101011 is the *mantissa* and e+0011 implies that the *exponent* is +3.

The IEEE-754 floating point is the most commonly used representation for real numbers on computers including Intel-based personal computers, Macintoshes and most of the UNIX platforms. It specifies four formats for representing floating-point numbers. These include single-precision, double-precision, single-extended precision and double-extended precision formats. Of the four formats mentioned, the single-precision and double-precision formats are the most commonly used ones. The single-extended and double-extended precision formats are not common.

Following figure shows the basic constituent parts of the single precision format. As shown in the figure, the floating-point numbers, as represented using these formats, have three basic components including the sign, the exponent and the mantissa.



Single precision

For example, if we want to represent +10.25 in IEEE 754 format, the following steps will be used:

Step 1: Convert 10.25 in binary.

$(10.25)_{10} = 1010.01$

Step 2: Normalise by bringing decimal point in front of first non zero bit

$1010.01 = 1.01001 x 2^3$

Step 3: Now mantissa is 01001 and exponent is 3

Step 4: Add 127 to exponent to get exponent in excess 127 format

$127 + 3 = 30 = (10000010)_2$

Now the representation is

1	8	23
0	10000010	01001000000000000000000
Sign ↑	Exponent	Mantissa

Example

What is a sign bit ? If sign bit contains the binary digit 0 then the integer represents is either positive or negative. Give reason.

Answer

Sign bit is used to represent the sign (+ or -) of a binary number inside computer memory. It normally precedes the binary sequence of the number. e.g., 0, 1000000 represents + 64 in decimal. e.g., 1,1100 can represent -12 in decimal when 0 represents positive sign, it is known a positive logic. It can also represent negative sign in negative logic. In 2's complementation notation, it can be act both positive and negative inorder to remove the ambiguity of zero magnitude represent (when preceded by 0) in signed magnitude numbers.

Example

Why do suppose people adopted the decimal the decimal number system for everyday use ? If you have to propose another number system so as to facilitate arithmetic computations, which radix would choose ? Justify your answer. What radix would you recommend for a computer ?

Answer

The decimal number system must have been adopted due to ten figures (including two thumbs) in two hands of humans. This led to convenience in counting also. It is sure that the decimal number system is the most appropriate owing to ease in arithmetic computation by people. However, one can use any of 2, 8 or 16 as radix to facilitate the arithmetic computations. Justification goes to these radices towards the inter-convertibility in a convenient way.

For a computer based switching circuits, the number system with radix 2 is the most suitable and is to be recommended. However, it depends upon the potential technology which might allow other radix systems.

Example

Explain briefly the following term

(i)Bit (ii) Byte

Answer

Bit : Short form of Binary digit. A binary digit is one of the two digits, represented by 0 and 1, that are used in the binary number system.

Byte : A group of eight bits, which forms the smallest portion of memory that an 8-bit CPU can recall from, or store in memory.

Example (AMIE SUMMER 2001)

Explain ASCII code? How many bits per character dose ASCII code use ?

Answer

The American standards Institution has evolved a standard code to represent characters to be stored and processed by computers. This code, called ASCII, uses 7 bits to represent each character. The ASCII code(*A*merican *S*tandard *C*ode for *I*nformation *I*nterchange)defines codes for English letters (capital and small), decimal digits, 32 special characters and codes for a number of symbols used to control the operation of a computer. The symbols used for control are non-printable.)

Example (AMIE SUMMER 2001)

Explain EBCDIC ?

Answer

In addition to the ASCII, another code known as *E*xtended *B*inary *C*oded *D*ecimal *I*nterchange *C*ode (EBCDIC) is used in computers manufactured by International Business machines Corporation (IBM). This code uses 8 bits per character and is thus capable of representing 256 characters. If data coded in ASCII is to be used in a computer which codes data in EBCDIC, it is necessary to transform ASCII code to EBCDIC code. A special ready made electronic circuit is available for carrying out this transformation.

Example

Write algorithm for converting binary to decimal.

Solution

Convert from binary to decimal algorithm:

For this we multiply each digit separately from right side by 1, 2, 4, 8, 16 … respectively then add them.

Binary number to decimal conversion with example:

For example we want to convert binary number 101111 to decimal:

Step1:	1 * 1 = 1
Step2:	1 * 2 = 2
Step3:	1 * 4 = 4
Step4:	1 * 8 = 8
Step5:	0 * 16 = 0
Step6:	1 * 32 = 32

Its decimal value: 1 + 2 + 4+ 8+ 0+ 32 = 47

That is $(101111)_2 = (47)_{10}$

Boolean Algebra

IMPORTANT THEOREMS AND EXPRESSIONS

1. $A + 0 = A$
2. $A.1 = A$
3. $A + 1 = 1$
4. $A.0 = 0$
5. $A + A = A$
6. $A.A = A$
7. $A + \overline{A} = 1$
8. $A.\overline{A} = 0$

9. $A.(B+C) = A.B + A.C$
10. $A + B.C = (A+B).(A+C)$
11. $A + A.B = A$
12. $A.(A+B) = A$
13. $A + \overline{A}B = A + B$
14. $A.(\overline{A}+B) = A.B$
15. $A.B + A.\overline{B} = A$
16. $(A+B).(A+\overline{B}) = A$
17. $\overline{A.B.C} = \overline{A} + \overline{B} + \overline{C} + ...$ (De Morgan’s Law)
18. $\overline{A + B + C + ...} = \overline{A}.\overline{B}.\overline{C}.....$ (De Morgan’s Law)
19. $A.B + \overline{A}.C = (A+C).(\overline{A}+B)$
20. $(A+B).(\overline{A}+C) = A.C + \overline{A}.B$
21. $A.B + \overline{A}.C + B.C = A.B + \overline{A}.C$
22. $(A+B)(\overline{A}+C).(B+C) = (A+B).(\overline{A}+C)$

Example

Prove that

(i) $A.(A+B) = A$

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

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

Solution

(i) $A.(A+B) = A.A + A.B = A + A.B = A.(1+B) = A$ [1 + B= 1]

(ii) $A + \overline{A}.B = (A+\overline{A}).(A+B)$ (By theorem10) $= A + B$

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

$= ABC + AB\overline{C} + \overline{A}BC + \overline{A}\overline{B}C$

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

$= BC + AB\overline{C} + \overline{A}\overline{B}C$

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

$= C(B + \overline{A}) + AB\overline{C}$ [Theorem 13]

$= CB + C\overline{A} + AB\overline{C}$

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

$= B(C + A) + \overline{A}C$ [Theorem 13]

$= BC + AB + \overline{A}C + A\overline{A}$

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

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

Example

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

Solution

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

$= (B + AC)(\overline{A} + C)$ (Theorem 10)

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

$= \overline{A}B + BC + AC$

$= \overline{A}B + BC + AC + A\overline{A}$

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

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

Example

Factorize following Boolean equations

(a) $Y = A\overline{B} + AB$

(b) $Y = AB + AC + BD + CD$

(c) $Y = (B + CA)(C + \overline{A}B)$

(d) $Y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D$

Solution

(a) $Y = A\overline{B} + AB = A(\overline{B} + B) = A.1 = A$

(b) $Y = AB + AC + BD + CD$

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

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

(c) $Y = (B + CA)(C + \overline{A}B)$

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

$= BC + B\overline{A}B + CAC + CA\overline{A}B;\ A\overline{A} = 0, BB = B, CC = C$

$= BC + \overline{A}B + AC$

(d) $Y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D$

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

$= \overline{B}\overline{C}[\overline{A}(\overline{D} + D) + A(\overline{D} + D)]$

$= \overline{B}\overline{C}[\overline{A}.1 + A.1]$

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

$= \overline{B}\overline{C}.1 = \overline{B}\overline{C}$

Problem

Prove the following identities using Boolean theorems.

(a) $A + \overline{A}B = A + B$

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

(c) $(A + C)(A + D)(B + C)(B + D) = AB + CD$

SUM OF PRODUCTS/ PRODUCTS OF SUM METHOD

The switching function (Boolean function) implied by a truth table may be stated as sum of products (sop) or a product of sums (pos). For example for an EXOR truth table

output = $\overline{A}B + A\overline{B}$ (sop form)

output = $(A + B)(\overline{A} + \overline{B})$ (pos form)

Here each term contains all the inputs i.e. individual Boolean variables in complemented or uncomplemented form called literals. This is the canonical or standard form. Each term in standard sop form is called a *minterm* and that in standard pos form is called a *maxterms*. For n inputs these are 2^n maxterms. Two logically different do not contain the same set of minterms (maxterms).

Consider the following example of a two OR with truth table shown below:

Truth table

A	**B**	**Y**	**Minterm**	**Maxterm**
0	0	0	$\bar{A}\bar{B}$	$A+B$
0	1	1	$\bar{A}B$	$A+\bar{B}$
1	0	1	$A\bar{B}$	$\bar{A}+B$
1	1	1	AB	$\bar{A}+\bar{B}$

Output is given by the sum of minterm corresponding to 1 output.

$$Y = \bar{A}B + A\bar{B} + AB; \text{(sop form)}$$

Output can be written as the product of those maxterms which correspond to 0 output.

$$Y = A + B; \text{(pos form)}$$

The equivalence of these expressions is established below:

$$\bar{A}B + A\bar{B} + AB = \bar{A}B + A(\bar{B} + B)$$

$$= \bar{A}B + A = A + \bar{B}C$$

$$= (A + \bar{A})(A + B)\ \text{Distributive}$$

$$= (A + B)$$

Example

Obtain sop and pos expressions for the truth table of table.

A	***B***	***C***	***Decimal Eq.***	***Y***	***Minterm***	***Maxterm***
0	*0*	*0*	0	*0*	$\bar{A}\bar{B}\bar{C}$	$A+B+C$
0	*0*	*1*	1	*1*	$\bar{A}\bar{B}C$	$A+B+\bar{C}$
0	*1*	*0*	2	*0*	$\bar{A}B\bar{C}$	$A+\bar{B}+C$
0	*1*	*1*	3	*1*	$\bar{A}BC$	$A+\bar{B}+\bar{C}$
1	*0*	*0*	4	*0*	$A\bar{B}\bar{C}$	$\bar{A}+B+C$
1	*0*	*1*	5	*1*	$A\bar{B}C$	$\bar{A}+B+\bar{C}$

1	*1*	*0*	6	*0*	$AB\overline{C}$	$\overline{A}+\overline{B}+C$
1	*1*	*1*	7	*1*	ABC	$\overline{A}+\overline{B}+\overline{C}$

From above table picking up the minterms corresponding to Y = 1 in the truth table, we can write

$$Y = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + ABC \text{ ; sop form}$$

In term of maxterms which correspond to Y = 0, we can write

$$Y = (A+B+C)(A+\overline{B}+C)(\overline{A}+B+C)(\overline{A}+\overline{B}+C)\text{; pos form}$$

We observed that

$$Y + \overline{Y} = 1$$

So while that minterms which correspond to 1 outputs constitute sop, the complimentary minterms which correspond to 0 outputs, their sop would then constitute $\overline{Y}$ i.e.

$$\overline{Y} = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + AB\overline{C}$$

similarly for maxterms the outputs corresponding to 1 outputs would constitute $\overline{Y}$ i.e.

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

Example

Find the equivalent product-of-sums expression of the sum-of products (SOP) expression

$$A.B + \overline{A}.\overline{B}$$

Solution

The dual of the given expression $A.B + \overline{A}.\overline{B}$

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

$$= 0 + A.\overline{B} + B.\overline{A} + 0$$

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

The dual of $(A.\overline{B} + \overline{A}.B) = (A+\overline{B}).(\overline{A}+B)$

Therefore $\qquad A.B + \overline{A}.\overline{B} = (A+\overline{B}).(\overline{A}+B)$

This is POS expression.

Example (AMIE Summer 2008, 5 marks)

Draw the truth table for the boolean function $\overline{A}BC + A\overline{B}C + AB$.

Solution

Given boolean expression

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

$$= \overline{A}BC + A\overline{B}C + ABC + AB\overline{C} \leftarrow \text{SOP canonical form}$$

Truth table can easily be obtained by making a 1 at the output column for each min term in above expression.

Minterms

$\overline{A}BC \rightarrow 011$

$A\overline{B}C \rightarrow 101$

$AB\overline{C} \rightarrow 110$

$ABC \rightarrow 111$

Truth table is

A	B	C	Y (output)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

LOGIC GATES

A logic gate is an electronic circuit, which accepts a binary input and produces a binary output namely 0 and 1. The inverter (NOT) logic gate has one input and one output, but a logic gate in general accepts one or more inputs and produces one output. Apart from the NOT gate there are six other types of logic gates.

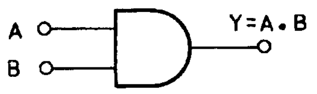
Input to a gate will be designated by binary variables A, B, C etc. and the output will be indicated by binary variable Y. As stated earlier, a binary variable can take on values 0 and 1 which are electronically represented by LOW and HIGH voltage levels. In terms of boolean algebra the function of a logic gate will be represented by a binary expression.

AND Gate

In this gate, output will be HIGH only if both the inputs are HIGH and so *both* diodes are OFF. Such a logic operation is called AND and is represented by the Boolean expression.

$$Y = A \text{ AND } B = A.B$$

wherein dot indicates ANDing.



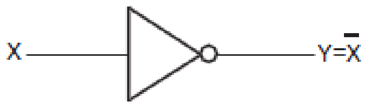
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(a) Symbol **(b) Truth table**

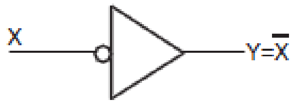
The logic symbol for an AND gate is drawn in given figure (a) with its truth table given in figure (b).

NOT Gate

A NOT gate is a one-input, one-output logic circuit whose output is always the complement of the input. That is, a LOW input produces a HIGH output, and vice versa. When interpreted for a positive logic system, a logic '0' at the input produces a logic '1' at the output, and vice versa. It is also known as a 'complementing circuit' or an 'inverting circuit'. Following figure shows the circuit symbol and the truth table.



(a)



X	Y
0	1
1	0

(b)

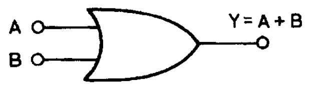
The NOT operation on a logic variable X is denoted as $\overline{X}$ or X' . That is, if X is the input to a NOT circuit, then its output Y is given by $\overline{X}$ or X' and reads as Y equals NOT X. Thus, if X = 0, Y = 1 and if X = 1, Y = 0.

OR Gate

The output will be HIGH, if A *or* B (or both) are HIGH. Such a logic operation is called OR and is expressed by the Boolean expression

$$Y = A \text{ OR } B = A + B$$

The symbolic representation of OR and its truth table are given in Figs. (a) and (b).



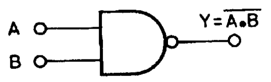
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(a) Symbol **(b) Truth table**

NAND Gate

It is an AND gate followed by a NOT gate and is represented by a symbol as shown in figure (a). Its truth table is given in figure (b). A small circle in its symbol shows NOT operation. Its Boolean expression is

$$Y = \text{NOT}(\text{A AND B}) = \overline{A.B}; \text{complemet of AND}$$



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

(a) Symbol **(b) Truth table**

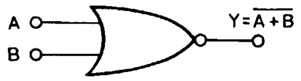
NAND gate may have more than two inputs.

NOR Gate

Logically a NOR gate is expressed as

$$Y = \text{NOT}(\text{A OR B}) = \overline{A + B}; \text{ complement of Or}$$

Its symbol and truth table are shown in figures (a) and (b).



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

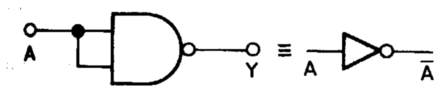
(a) Symbol **(b) Truth table**

There can be more than two inputs.

Universality of NAND/NOR Gates

NAND and NOR gates are called universal logic gates, because any logic operation can be realized by using these gates.

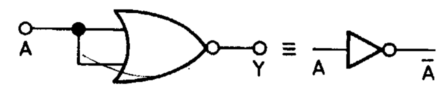
Consider the NAND gate of figure (a) with input A at both ports.



Then

$$Y = \overline{A.A} = \overline{A} = \text{NOT A}$$

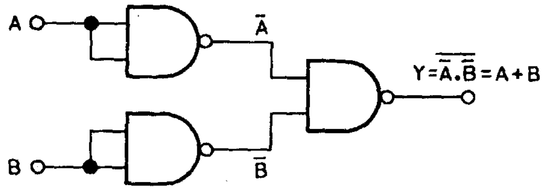
Similarly for NOR gate



$$Y = \overline{A + A} = \overline{A} = \text{NOT A}$$

OR Realization by NAND/NOR

Consider following circuit which shows OR realization by NAND gates.



(a) OR realization by NAND gates

Its truth table is given by figure (b).

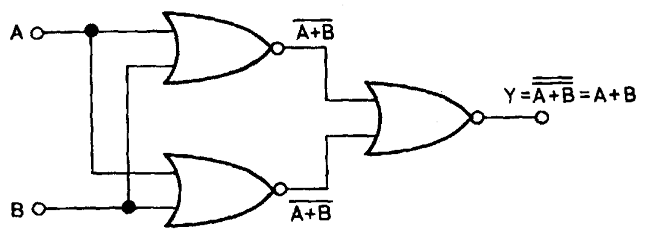
A	$\overline{A}$	B	$\overline{B}$	$\overline{A}.\overline{B}$	$\overline{\overline{A}.\overline{B}}$	A+B
0	1	0	1	1	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1

(b) Truth table

It follows that

$$Y = \overline{\overline{A}.\overline{B}} = A + B = \text{A OR B}$$

OR realization by NOR gate is given in figure (c).

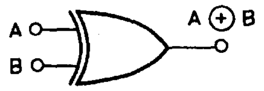


(c) NOR realization of OR

Exclusive OR (EXOR) Gate

The output of this gate is high only when either of the inputs is high but not when both inputs are high i.e. either, but not both.

Symbol of XOR gate and its truth table are given in given figures (a) and (b).



(a)

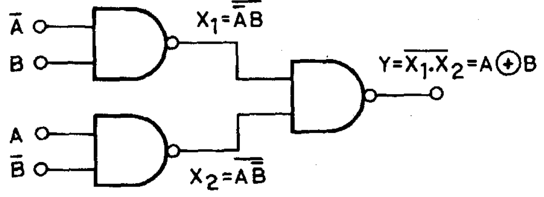
A	B	$A\oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

(b) Truth table

Generally a symbol ⊕ is used in XOR equations. It immediately follows that

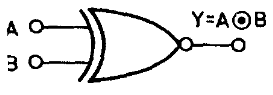
$$Y = A\ \text{XOR}\ B = A \oplus B$$

NAND implementation of Exclusive OR is shown in figure below.


$\bar{A}$
B
A
$\bar{B}$
$X_1=\overline{\bar{A}B}$
$X_2=\overline{A\bar{B}}$
$Y=\overline{X_1.X_2}=A\oplus B$

Exclusive NOR Gate

It is gate complementary of EXOR gate. Its symbol and truth table are given in figures in (a) and (b).



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

(a) Symbol **(b) Truth table**

FLIP FLOPS

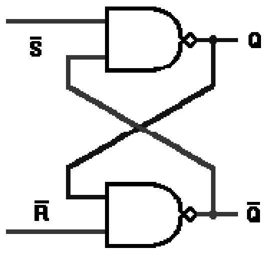
Basic Memory Cell

The basic memory cell is a circuit that stores one bit of information. This one bit memory element is called a **flip-flop** or **latch**, since it latches(or locks) data in it.

A flip flop has two outputs, Q and $\bar{Q}$, that are always complements of each other. It can exist in two stable states; **set** and **reset**. In set state, Q is HIGH(logic 1) and $\bar{Q}$ is LOW(logic 0). In reset state $\bar{Q}$ is HIGH(logic 1) and Q is LOW(logic 0). For a flip flop to act as a memory device, it should retain the information stored in it. Thus, if it is in set state it should remain set and if it is in reset state, it should remain reset.

NAND Latch

The circuit shown below is a basic NAND latch. The inputs are generally designated "S" and "R" for "Set" and "Reset" respectively. Because the NAND inputs must normally be logic 1 to avoid affecting the latching action, the inputs are considered to be inverted in this circuit.



The outputs of any single-bit latch or memory are traditionally designated Q and Q'. In a commercial latch circuit, either or both of these may be available for use by other circuits. In any case, the circuit itself is:

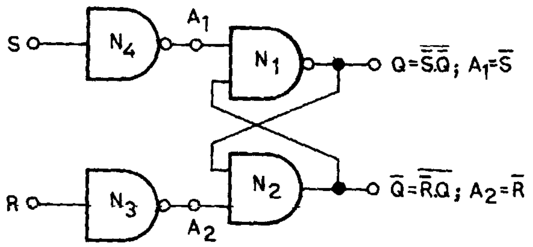
For the NAND latch circuit, both inputs should normally be at a logic 1 level. Changing an input to a logic 0 level will force that output to a logic 1. The same logic 1 will also be applied to the second input of the other NAND gate, allowing that output to fall to a logic 0 level. This in turn feeds back to the second input of the original gate, forcing its output to remain at logic 1.

Applying another logic 0 input to the same gate will have no further effect on this circuit. However, applying a logic 0 to the other gate will cause the same reaction in the other direction, thus changing the state of the latch circuit the other way.

Note that it is forbidden to have both inputs at a logic 0 level at the same time. That state will force both outputs to a logic 1, overriding the feedback latching action. In this condition, whichever input goes to logic 1 first will lose control, while the other input (still at logic 0) controls the resulting state of the latch. If both inputs go to logic 1 simultaneously, the result is a "race" condition, and the final state of the latch cannot be determined ahead of time.

SR Flip Flop(Latch)

To serve the purpose of storing desired bits in flip flops, two input NAND or NOR gates are used as shown in following figure. This way we get an S-R flip flop.


S
R
N4
N3
A1
A2
N1
N2
$Q=\overline{\bar{S}.\bar{Q}}$; $A_1=\bar{S}$
$\bar{Q}=\overline{\bar{R}.Q}$; $A_2=\bar{R}$

S-R Flip Flop

Truth table is given below.

Truth table for SR flip flop

S	R	A_1	A_2	State
0	0	1	1	No change
0	1	1	0	Reset
1	0	0	1	Set
1	1	0	0	Race

Set state: When S = 1 and R = 0, A_1 becomes 0, making Q = 1. So, $\bar{Q}=\overline{1.1}=0$. This input condition sets the flip flop($Q=1,\bar{Q}=0$); so it is called set state.

Reset state: When S = 0 and R = 1,A_2 becomes 0, making $\bar{Q}=1$ and $Q=\overline{1.1}=0$. This input condition resets the flip flop($Q=0,\bar{Q}=1$); so it is called reset state.

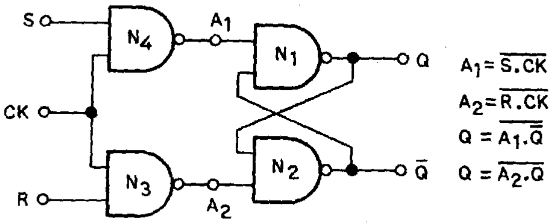
No change: When $S=R=0, Q=\overline{1.\bar{Q}}=Q$ and $\bar{Q}=1.\bar{Q}=\bar{Q}$. The flip flop remains in whatever state it is in.

Race: When S = R = 1, A_1 and A_2 both become 0. So $Q=\bar{Q}=1$. This is an undesired output state because if S and R now change to 0, A_1 and A_2 will be 1, both Q and $\bar{Q}$ will try to become 0. The actual state of the flip flop depends on the relative delays of the two gates. If N_2 is faster, $\bar{Q}$ will become $\overline{1.1}=0$ first and will make Q =1. Similarly if N_1 is faster, Q will become 0 first and will make $\bar{Q}=1$. This is called **race condition**. Here the state of the flip flop is uncertain, so this condition is not allowed.

This circuit is called an SR flip flop. It is an asynchronous circuit because the output changes with changes as and when the S, R input change. If we want synchronous operation, a clock pulse can be introduced at the inputs as discussed below.

Clocked SR Flip Flop(Latch)

The SR flip flop modified to include clock(CK) pulses is drawn in following figure.



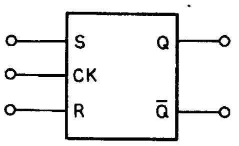
Clocked SR flip flop

When $CK = 1, A_1 = \bar{S}$ and $A_2 = \bar{R}$, the circuit functions like the SR flip flop already discussed. When $CK = 0, A_1 = A_2 = 1$, the circuit reduces to that of a basic memory cell and outputs remain unchanged(latched). Here N_1 and N_2 form the basic latch, whereas N_3 and N_4 control the state of the flip flop.

The truth table and logic symbol of SR flip flop are shown below.

Truth table for clocked SR flip flop

CK	S	R	Q_{n+1}	Comments
1	0	0	Q_n	SR enabled
1	0	1	0	SR enabled
1	1	0	1	SR enabled
1	1	1	?	Not allowed
0	x	x	Q_n	SR disabled



Symbol clocked SR

JK FLIP FLOP

As discussed earlier, in SR flip flop, the input combination S = R =1 is not allowed because its output is uncertain. This uncertainty may cause problems in a digital system. So, we need a one bit memory cell which has its outputs well defined for all possible input combinations.

One such device is JK flip flop with inputs J and K. Also, let this flip flop have the same truth table as the SR flip flop except for the condition when J = K = 1. Since $Q_{n+1} = Q_n, 1, 0$ are already occurring in the truth table of the SR flip flop, let $Q_{n+1} = \overline{Q}_n$ for the new condition (J = K = 1). Following table gives the combined truth table of SR/JK flip flops.

Combined truth table of SR/JK flip flops

J	K	Q_n	Q_{n+1}	S	R	Comment
0	0	0	0	0	0/1	Inactive state
0	0	1	1	1/0	0	$Q_{n+1} = Q_n$
0	1	0	0	0	0/1	Reset state
0	1	1	0	0	1	$Q_{n+1} = 0$
1	0	0	1	1	0	Set state
1	0	1	1	1/0	0	$Q_{n+1} = 1$
1	1	0	1	1	0	Toggle state
1	1	1	0	0	1	$Q_{n+1} = Q_n$

The K- maps with S and R as outputs, and taking J, K and Q_n as inputs are shown below.

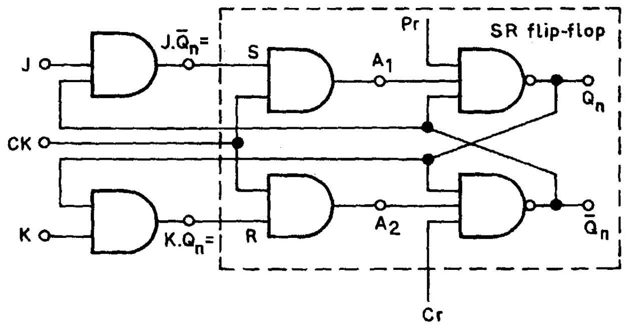
$$S = J.\overline{Q}_n$$

JK → Q_n ↓	00	01	11	10
0	0	0	1	1
1	x	0	0	x

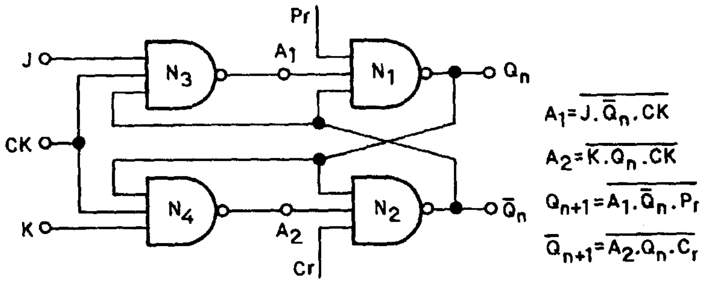
$$R = K.\overline{Q}_n$$

JK → Q_n ↓	00	01	11	10
0	x	x	0	0
1	0	1	1	0

Thus we have a simple way of designing a JK flip flop from the SR flip flop. The circuit corresponding to K maps given earlier is drawn below.



This circuit can be simplified to following circuit.



Truth table of above JK flip flop is given below:

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

Race Around Condition

The solution that we found in the JK flip flop, for the condition when both the inputs of the flip flop are *high*, is not perfect. For the level triggered JK flip flop of above figure, consider the case when J = K = 1, $Q_n = 0$ and a clock pulse occurs. After a propagation delay of $\Delta t(<<t_p$, the pulse width) equal to the delay time of two NAND gates, the output will *toggle* to $Q_n = 1$. Since this is feedback to the inputs, the output can toggle back to $Q_n = 0$ after another delay of Δt, if the clock is still high.

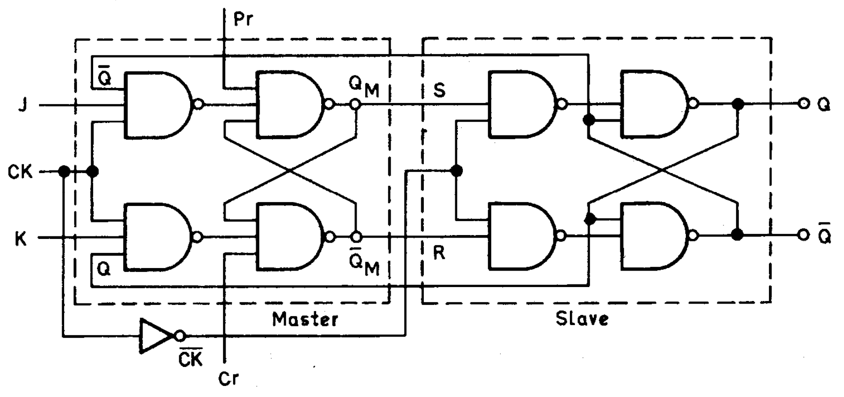
Thus, as long as the clock pulse is present, the output will toggle at every Δt seconds. This is called race around condition.

Removing race around condition

- Making propagation delay $\Delta t > t_p$
- Passing outputs Q_n and $\overline{Q}_n$ through delay lines before feeding them back to input
- Use a JK master slave flip flop

JK MASTER SLAVE FLIP FLOP

This flip flop is made of two SR flip flops. The output of the first, called the *master* flip flop, is given to the inputs of the second, called the *slave* flip flop. The output of the slave is feedback to the inputs of the master. The master is triggered by a positive clock pulse and the slave is triggered by a negative clock pulse, obtained by inverting the clock pulse applied to the master. The circuit of MS flip flop is shown below.



JK master-slave flip flop

When the clock is HIGH(CK = 1, Pr = 1, Cr = 1), the master is enables, while the slave is disabled as $\overline{CK} = 0$). So, the master functions like a JK flip flop and its output appears at the input (S,R) of the slave. Since CK is LOW, the slave is inactive. Thus Q remains unchanged for the duration of the clock pulse t_p.

When the clock goes LOW, the slave functions like an SR flip flop and its output changes to Q_M, which also appears at the input of the master. Since the clock is LOW, the master is inactive so that Q_M(and so S and R) remains unchanged as long as the clock remains LOW.

Thus, when the clock is HIGH, Q_M changes according to JK flip flop logic and it is transferred to Q when the clock goes LOW(negative edge transition). This eliminates the race around condition as the output remains constant for the duration of one clock pulse. The truth table of this flip flop is shown below.

Truth table of MS JK flip flop

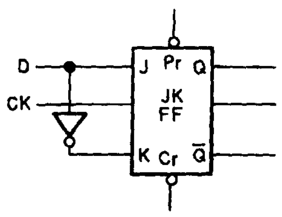
Pr	**Cr**	**CK**	**J**	**K**	**Q_{n+1}**	**States**
0	0	x	x	x	?	Not desired
0	1	x	x	x	1	Preset
1	0	x	x	x	0	Clear
1	1	x	0	0	Q_n	Inactive
1	1	⎍	0	1	0	Zero
1	1	⎍	1	0	1	One
1	1	⎍	1	1	Q_n	Toggle

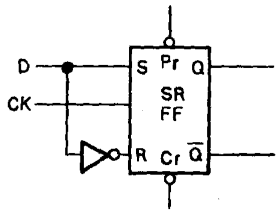
D FLIP FLOP

In a D flip flop the output follows the input whenever the flip flop is triggered. This can be achieved by means of a JK/SR flip flop. A D flip using JK/SR flip flop along with its truth table is shown below.

Truth table of D flip flop

D	**Q_{n+1}**
0	0
1	1



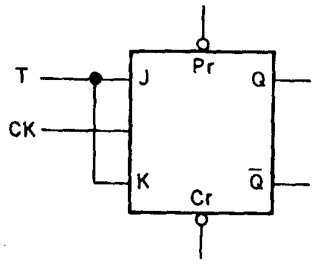


D flip flop from JK flip flop **D flip flop from SR flip flop**

Here the output follows the input, when the flip flop is triggered. If the flip flop is edge triggered the output is the same as the input, but is delayed by one clock pulse. Owing to this property, the D flip flop is used as a delay device. Its function is illustrated by following waveform.

T FLIP FLOP

A T flip flop can be realized by JK flip flop. T flip flop using JK flip flop along with truth table is given below.



T flip flop

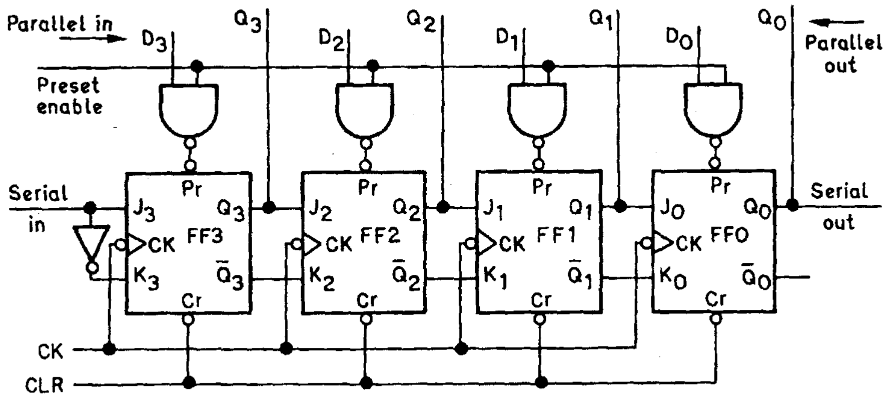
Truth table of T flip flop

T	J	K	Q_{n+1}
0	0	0	Q_n
1	1	1	$\overline{Q}_n$

T flip flops are widely used as toggle switches.

SHIFT REGISTER

Shift registers store as well as manipulate data. Following figure shows a 4-bit shift register.



4-bit shift register

Four JK(or SR) flip flops are converted into D flip flops by giving mutually complementary inputs. All the four flip flops are triggered by a common clock and can be cleared by giving a 0 at the CLR input. During normal operation, this input is held at 1.

Arithmetic Circuits

A logic gate is an electronic circuit, which accepts a binary input and produces a binary output namely 0 and 1. The inverter (NOT) logic gate has one input and one output, but a logic gate in general accepts one or more inputs and produces one output. Apart from the NOT gate there are six other types of logic gates.

Input to a gate will be designated by binary variables A, B, C etc. and the output will be indicated by binary variable Y. As stated earlier, a binary variable can take on values 0 and 1 which are electronically represented by LOW and HIGH voltage levels. In terms of boolean algebra the function of a logic gate will be represented by a binary expression.

HALF ADDER

This circuit adds two binary variables, yields a carry but does not accept carry from another circuit(adder). The truth table of half adder is given in below.

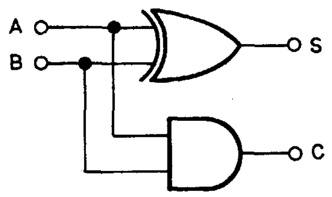
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From this table

$$S = \bar{A}B + A\bar{B} = A \oplus B$$

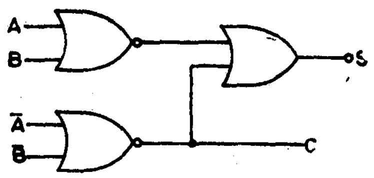
$$C = AB$$

Half adder logic circuit is shown in given figure.



Circuit using XOR

Half adder circuit using NOR gates is shown in given figure.

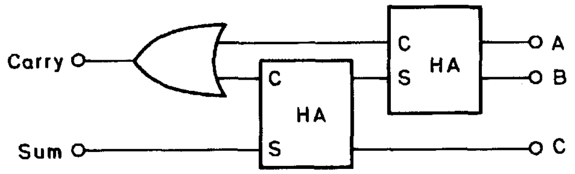


Half adder circuit using NOR gates

Here $S = \overline{\overline{S}} = \overline{\overline{AB} + \overline{\overline{A}\,\overline{B}}} = \overline{(\overline{A} + \overline{B}) + (\overline{\overline{A} + \overline{B}})}$

FULL ADDER

This circuit can add two binary numbers, accept a carry and yield a carry. Such a circuit can easily be visualized by means of two half adders(HA) and an OR as in given figure.



Full adder using two half adders

To synthesize the full adder circuit, we should proceed from the truth table as in given table.

A	B	C_i	S	C_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

It can be immediately written from this table that

$$S = \overline{A}\,\overline{B}C_i + \overline{A}B\overline{C}_i + A\overline{B}\,\overline{C}_i + ABC_i \quad (1)$$

and $$C_0 = \overline{A}BC_i + A\overline{B}C_i + AB\overline{C}_i + ABC_i \quad (2)$$

Recognizing that $Y + Y + Y + ... + Y = Y$ and adding ABC_i twice to right hand side of eq. (2), we can write

$$C_0 = (\overline{A}BC_i + ABC_i) + (A\overline{B}C_i + ABC_i) + (AB\overline{C}_i + ABC_i)$$

Observing that $Y + \overline{Y} = 1$, we get

$$C_0 = BC_i + AC_i + AB \quad (3)$$

The following is easily established

$$S = [\overline{(AC_i + BC_i + AB)} + ABC_i](A + B + C_i) \quad (4)$$

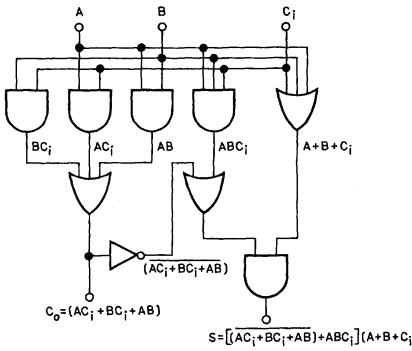
$$= [(\overline{AC_i}.\overline{BC_i}.\overline{AB}) + ABC_i](A + B + C_i)$$

$$= (\overline{AC_i}.\overline{BC_i}.\overline{AB})(A + B + C_i) + ABC_i$$

$$= (\overline{A} + C_i)(\overline{B} + \overline{C}_i)(\overline{A} + \overline{B})(A + B + C_i) + ABC_i$$

$$= \overline{A}\,\overline{B}C_i + \overline{A}B\overline{C}_i + A\overline{B}\,\overline{C}_i + ABC_i = S$$

Using AND OR INVERT(AOI) gates, equations (3) and (4) are implemented in figure given, which gives outputs S and C_0.

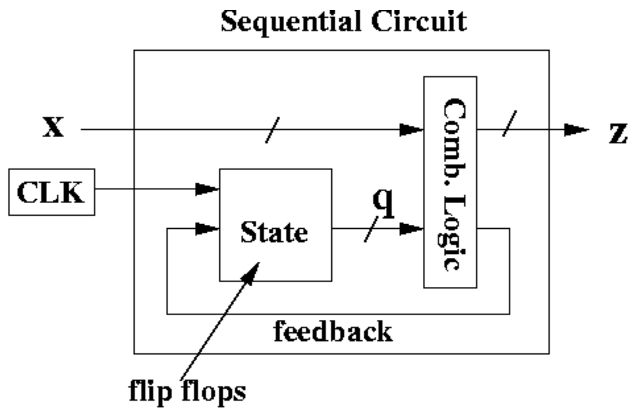

A
B
C_i
BC_i
AC_i
AB
ABC_i
$A+B+C_i$
$\overline{(AC_i+BC_i+AB)}$
$C_o=(AC_i+BC_i+AB)$
$S=[\overline{(AC_i+BC_i+AB)}+ABC_i](A+B+C_i)$

Full Adder

4 -bit parallel binary adder(cascaded)

COMBINATIONS CIRCUITS VS. SEQUENTIAL CIRCUITS

The main difference between sequential circuits and combinational circuits is that sequential circuits compute their output based on input and state, and that the state is updated based on a clock. Combinational logic circuits implement Boolean functions, so they are functions only of their inputs, and are not based on clocks. A sequential circuit uses flip flops. Unlike combinational logic, sequential circuits have state, which means basically, sequential circuits have memory. Here are some of the key things to notice:



- Like combinational logic circuits, a sequential logic circuit has inputs (labelled with **x** with subscripts) and outputs (labelled with **z** with subscripts).
- Unlike combinational logic circuits, a sequential logic circuit uses a *clock*.
- Also, there is a box inside the circuit called **State**.
- This box contains flip flops. Assume it has **k** flip flops. The flip flops basically store a k-bit number representing the current state.
- The output **z** is computed based on the inputs (**x** with subscripts) and the state coming out of the state box (**q** with subscripts).
- The state may be updated at each positive clock edge. When there's not a positive clock edge, the state remains unchanged.
- The information needed to update to the state (called the *next state*) comes from the current state (the current value of **q**) and the input, which is fed through combinational logic, and fed back into the state box, telling the state box how to update itself.

For example, suppose you are currently in state 00, and see an input of 1. This may produce an output of, say, 10, and then produce feedback that tells the state box to update to state 01 by the next clock edge.

ASSIGNMENT

NUMBER SYSTEM

Q.1. (AMIE, S08, 8 marks): Give an algorithm to convert an octal number into a binary number. Using an example, explain how the algorithm works.

Q.2. (AMIE, W01, 5 marks): Perform following operation using 2's complement representation

-53 + 25 [Given numbers are decimal]

Answer: $(-28)_{10}$

Q.3. (AMIE, W01, 5 marks): Convert following hexadecimal number to octal number C005A

Answer: $(3000132)_8$

Q.4. (AMIE, S01): Convert the following numbers as indicated against each:

$(?)_{16} = (285.48)_{10} = (?)_8 = (?)_2$

Answer: $(11D.7AE147)_{16} = (285.48)_{10} = (435.365605)_8 = (100011101.111101)_2$

Hint: $(285.48)_{10} = (?)_{16}$

Integer part $(285)_{10} = (11D)_{16}$

$$\begin{array}{r|l l}
16 & 285 & 13=D \\
16 & 17 & 1 \\
16 & 1 & 1
\end{array}$$

Fractional part = $(0.48)_{10}$

Fractional part x 16	Remainder	Integer
0.48 x 16	7.*68*	7
0.68 x 16	10.*88*	10 = A
0.88 x 16	14.*08*	14 = E
0.08 x 16	1.*28*	1
0.28 x 16	4.*48*	4
0.48 x 16	7.*68*	7

and so on.

$(0.48)_{10} = (0.7AE147)_{16}$ approx.

$\therefore$ $(285.48)_{10} = (11D.7AE147)_{16}$ approx

Q.5. (AMIE S05, 7 marks): Convert 123789 decimal into its equivalent Binary number. Clearly depict all steps.

Answer: $(11110000111000110 1)_2$

Q.6. (AMIE W05, 10 marks): Convert the following from one number system to another:

(i) $(1267.3125)_{10} = (\ )_2$

(ii) $(10110.101)_2 = (\ )_{10}$

(iii) $(1234)_8 = (\ )_{16}$

(iv) $(B2C)_{16} = (\ )_2$

(v) $(10110111.1)_2 = (\ )_8$

Answer: (i) $(10011110011.010)_2$ (ii) $(22.625)_{10}$ (iii) $(29C)_{16}$ (iv) $(101100101100)_2$ (v) $(263.4)_8$

Q.7. (AMIE 07, 10 marks): Convert the following from one number system to another:

(i) $(1357)_{10} = (\ )_2$

(ii) $(1463)_{10} = (\ )_8$

(iii) $(1010010110)_2 = (\ )_{16}$

(iv) $(573)_8 = ( )_{16}$

(v) $(1100100110)_2 = ( )_8$

Answer: (i) $(10101001101)_2$ (ii) $(2667)_8$ (iii) $(296)_{16}$ (iv) $(17B)_{16}$ (v) $(1446)_8$

Q.8. (AMIE W07, 12 marks): Convert the following integers into its equivalent form as specified. Specify all the steps.

(i) 2456_{10} to its equivalent octal

(ii) 1267_8 to its equivalent binary

(i) $AE29_{16}$ to its equivalent decimal

(ii) 11011100101_2 to its equivalent hexadecimal number

Answer: (i) $(4739)_8$ (ii) $(1010110111)_2$ (iii) $(44585)_{10}$ (iv) $(6E5)_{16}$

Q.9. (AMIE S09, 6 marks): Convert the following two hexadecimal numbers into binary and decimal numbers (i) 9F (ii) E7

Answer: (i) $(159)_{10}$ (ii) $(231)_{10}$

Q.10. (AMIE S09, 6 marks): Perform the following hexagonal operations (i) 5F + AB (ii) CD + BE

Answer: (i) 1 OC (ii) 18 B

Q.11. (AMIE W09, 8 marks): (i) Convert the binary real number 1101.1010 to a equivalent decimal number.

(ii) Convert the decimal fraction 0.62 to its equivalent hexagonal fraction

(i) Convert the octal number 365 to its equivalent decimal number

(ii) (iv) Convert the octal number 536 to its equivalent hexadecimal number.

Answer: (i) 13.625 (ii) $(0.9EB851...)_{16}$ (iii) 240 (iv) $(15\ E)_{16}$

Q.12. (AMIE W09, 4 marks): (i) Perform following addition 1010111 + 1011010

(ii) Perform following subtraction 1101011 – 1010110

Answer: (i) 0110001 (ii) 0010101

Q.13. (AMIE S10, 6 marks): Convert the following binary numbers into hexadecimal numbers: (i) 101101.0101 (ii) 1010.0111

Answer: (i) 2D.5 (ii) A.7

Q.14. (AMIE W10, 6 marks): Convert the following binary numbers to their equivalent hexadecimal numbers:

(i) $(111100110.101011)_2$ (ii) $(111010100011.01010110)_2$

Answer: 1E6.AC, EA3.56

Q.15. (AMIE W10, 8 marks): Find the values of the following binary arithmetic operations:

(i) Divide $(100101100)_2$ by $(1010)_2$

(ii) Multiply $(11101)_2$ by $(11011)_2$

Answer: (i) 11110 (ii) 1100001111

Q.16. (AMIE W10, 6 marks): How many bits are required to represent the following decimal numbers as unsigned/binary integers: (i) 384 (ii) 147

Answer: (i) 384 = 110000000 (i.e. 9 bits) (ii) 147 = 10010011 (i.e. 8 bits)

Q.17. (AMIE W11, 5 marks): (a) Convert 211.25 in decimal to binary (b) Convert 211.25 in decimal to octal

Answer: (i) $(11010011.01)_2$ (b) 10001001.010101

Q.18. (AMIE S12, 6 marks): Convert the following octal number into its binary equivalent: 735

Answer: 111011101

Q.19. (AMIE S12, 6 marks): Convert the following hexadecimal number into its octal equivalent: AFB8

Answer: 127670

Q.20. (AMIE S12, 8 marks): How a floating point number represented in a computer?

Q.21. (AMIE W12, 4 marks): Find the hexadecimal equivalent of $(0.3)_{10}$.

Answer: $(4CC)_H$

Q.22. (AMIE W12, 4 marks): Find the octal equivalent of the decimal fraction 0.789.

Answer: $(0.6237)_8$

Q.23. (AMIE S13, 10 marks): Convert the following two binary numbers into Hex and Octal numbers: 01101010 and 01011011

Answer: $(6A)_{16}$, $(152)_8$; $(5B)_{16}$, $(133)_8$

Q.24. (AMIE S15, 10 marks): Convert the following numbers:

(i) $(723)_8 = (?)_2$

(ii) $(10001010101)_{16}$

(iii) $(285.48)_{10} = (?)_{16}$

(iv) $(0.8125)_{10} = (?)_2$

Answer: (i) 111010011_2 (ii) 456_{16} (iii) $(11D.7AE1)_{16}$ (iv) $0.110011\ldots_2$

Q.25. (AMIE W15, 6 marks): Convert the following numbers as directed:

(i) $(253.65)_{10} = (?)_2$

(ii) $(325.62)_8 = (?)_2$

Answer: (i) (1111101.101001…..) (ii) 011010101110010

Q.26. (AMIE S16, 6 marks): Convert the following numbers into appropriate base as directed:

(i) $(475.66)_{10} = (?)_2$

(ii) $(638.12)_{10} = (?)_8$

(iii) $(10110101011)_2 = (?)_{16}$

Answer: (i) 111011011.101010001 (ii) $(1176.0753412)_8$ (iii) $(5AB)_{16}$

Q.27. (AMIE W16, 20 marks): Answer the following:

(i) Find the hexadecimal equivalent of $(41819.5625)_{10}$

(ii) Find the octal equivalent of $(D6C1)_{16}$

(iii) Find the binary equivalent of $(37.8125)_{10}$

(iv) Find the binary equivalent of $(727)_8$

(v) What do you understand by the acronym MOSFET and list the purpose of logic gate in MOSFET.

BOOLEAN ALGEBRA/ARITHMETIC CIRCUITS

Q.28. (AMIE S03): Simplify the following Boolean algebraic expression using rules of Boolean algebra

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

Answer: $A+BC$

Q.29. (AMIE W02, 12 marks): Prove the following Boolean identities using Boolean algebra

(i) $\overline{\overline{A}+B}+\overline{\overline{A}+\overline{B}} = A$

(ii) $AB+AC+B\overline{C} = AC+B\overline{C}$

(iii) $(A+B)(B+C)(C+A) = AB+BC+CA$

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

Q.30. (AMIE S02, 8 marks): Simplify the following Boolean equations using rules of Boolean algebra.

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

(ii) $X = ABC+\overline{A}BC+AB\overline{C}$

Answer: (i) $\overline{B}\overline{C}$ (ii) $AC+BC+AB$

Q.31. (AMIE W09, 3 marks): Simplify following Boolean expression: X + XY' + Y' + (X + Y')X'Y

Answer: 1

Q.32. (AMIE W08, 4 marks): Prove that the following Boolean expression reduces to 0:

R = (A + B).(A'.B')

Q.33. (AMIE S11, 6 marks): Show that $A+\overline{A}B = A+B$

Q.34. (AMIE W06, 12 marks): Show truth table for Boolean expression E = A'C' + BC'. Draw a logic circuit to implement the this relation.

Q.35. (AMIE S12, 6 marks): Convert the following sum of product (SOP) expression into product of sum (POS) expression: $\overline{A}\overline{B}+\overline{C}\overline{D}$.

Q.36. (AMIE W12, 6 marks): Simplify the following Boolean function in both sum of products (SOP) and product of sums (POS) forms: $F(A,B,C,D) = \sum(0,1,2,5,8,9,10)$

Answer: $F = \overline{B}\overline{D}+\overline{B}\overline{C}+\overline{A}\overline{C}D$, $F = (\overline{B}+D)(\overline{A}+\overline{B})(\overline{C}+\overline{D})$

Q.37. (AMIE W14, 8 marks): Using theorems of Boolean algebra, prove the following:

(i) $X.Y+X.Z+Y.Z = X.Y+\overline{X}.Y.Z+X.Z$

(ii) $(X.Y).(\overline{X}.\overline{Z}+Z).(X.\overline{Z+Y}) = 0$

Q.38. Describe various logic gates.

Q.39. What do you understand by Flip flop. describe SR flip flop and clocked SR flip flop.

Q.40. (AMIE W14, 7 marks): Explain with the truth table, the working of RS flip flops.

Q.41. (AMIE S13, 10 marks): What is flip flop? For what purpose it is used for? Explain how a flip flop can be realised using NAND gates.

Q.42. (AMIE S05, 14 marks): Show that only 2 input NAND gates can be used to implement each of the following logic gates:

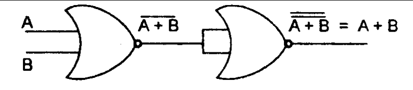
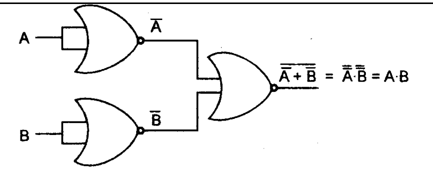
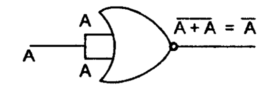
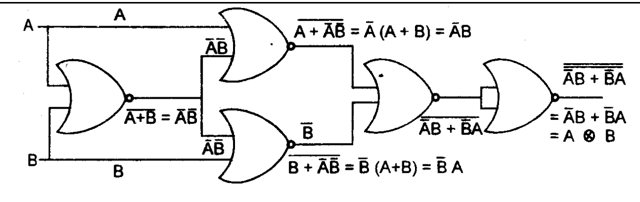
(i) 2 input OR gates
(ii) 2 input AND gates
(iii) 2 input Ex-OR gate
(iv) NOT gate

Q.43. (AMIE W07, 8 marks): Draw logic diagram that use only 2 input NOR gates to implement each of the following logic gates:

(i) 2 input OR
(ii) 2 input AND
(iii) NOT

(iv) 2 input EX-OR

Answer:

(i)	 A B $\overline{A+B}$ $\overline{\overline{A+B}} = A + B$
(ii)	 A $\overline{A}$ B $\overline{B}$ $\overline{\overline{A}+\overline{B}} = \overline{\overline{A}}\cdot\overline{\overline{B}} = A\cdot B$
(iii)	
(iv)	 A A $\overline{A+\overline{AB}} = \overline{A}\,(A+B) = \overline{A}B$ $\overline{AB}$ $\overline{A+B} = \overline{A}\overline{B}$ $\overline{AB}$ $\overline{\overline{A}B+\overline{B}A}$ $\overline{A}B + \overline{B}A$ $= \overline{A}B + \overline{B}A$ $= A \otimes B$ $\overline{B}$ B B $\overline{B+\overline{AB}} = \overline{B}\,(A+B) = \overline{B}A$

Q.44. (AMIE W08, 4 marks): Draw the logic circuit for the following Boolean expression:

P = AB + BC + AC

Q.45. (AMIE S10, 10 marks): Write the truth table of Boolean expression $Z = \overline{(a+b).c}$. Also draw logic circuit for the Boolean function part.

Q.46. (AMIE S11, 6 marks): Draw truth table for the Boolean function

$$f(A,B,C) = A \otimes B \otimes C$$

Q.47. (AMIE W11, 5 marks): Draw the truth table for the Boolean expression $\overline{a}b + \overline{b}c + a\overline{c}$

Q.48. (AMIE S07, 6 marks): How an EX-NOR gate works? What is its truth table.

Q.49. (AMIE S12, 7 marks): Briefly explain by using suitable diagrams, how various basic logic gates can be realised using NOT gate.

Q.50. (AMIE S05, W10, 6 marks): Construct an R-S flip flop using NOR gates only.

Q.51. (AMIE W05, 10 marks): Why is NAND gate considered as a universal gate? Implement an EX-OR (2 input) logic using NAND gates.

Q.52. (AMIE W09, 3 marks): Why are NAND and NOR gates are called universal gates?

Q.53. (AMIE W14, 6 marks): Construct the truth table for NAND and NOR gates.

Q.54. (AMIE S14, 8 marks): What do you mean by universal logic gates? Design a full adder using a universal gate.

Q.55. (AMIE S11, 8 marks): Explain the working of NAND latch with a diagram.

Q.56. (AMIE S09, 7 marks): What is a flip flop? Draw the logic gate representation of a flip flop. How is flip flop useful?

Q.57. (AMIE W15, 7 marks): What is a flip flop? Write the truth tables for different types of flip-flops.

Q.58. (AMIE W05, 6 marks): What is a D-type flip flop? Construct a D type latch using RS inputs.

Q.59. (AMIE S12, 7 marks): What is D flip flop? By using an approximate diagram, briefly explain how a shift register can be realised using D flip flop?

Q.60. (AMIE S06, 4 marks): Specify the structure of JK flip flop using RS flip flop.

Q.61. (AMIE S16, 10 marks): What are disadvantages of using S-R flip flop? Explain how J-K flip flop overcomes these issues. Draw the basic block structure and the truth table of a J-K flip flop.

Q.62. (AMIE W15, 7 marks): What do you mean by a master-slave flip flop? Explain your answer with a proper diagram.

Q.63. (AMIE S06, 6 marks): What are the advantages of master slave JK flip flop? How can it be built using ordinary JK flip flop and associated logic gates?

Q.64. (AMIE S07, 6 marks): What is a JK flip flop? Write the truth table and show how it can be converted into a T Flip flop.

Q.65. (AMIE S10, 5 marks): Draw the logic circuit of a J-K flip flop.

Q.66. (AMIE W07, 10 marks): Specify the structure of a bidirectional 4 bit shift register built using JK flip flop and logic gates. Depending on user input, one should be able to shift left or right the stores data.

Q.67. (AMIE W15, 6 marks): Write the truth table, Boolean expression and the logic circuit diagram of a half adder.

Q.68. (AMIE S06, 10 marks): Construct a 1 bit half adder that accepts two one bit operand a_i and b_i and produces the carry out c_i and the summation s_i. Use optimum number of gates.

Q.69. (AMIE S07, 16, W12, 6 marks): Draw a schematic diagram of a full adder and show the truth table.

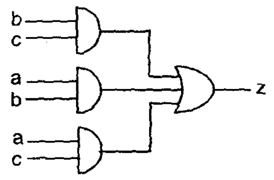
Q.70. (AMIE S10, 14, 5 marks): Distinguish between a combinational logic circuit and a sequential circuit.

Q.71. (AMIE W10, 6 marks): Realize the digital circuit for the boolean function

$$z = b.c + a.b + a.c \text{ z}$$

by using AND and OR gates.

Answer:



Q.72. (AMIE S15, 6 marks): Design a logic circuit to add two positive numbers that are each 2 bits long.

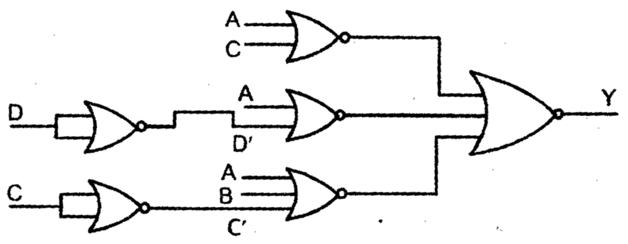
Q.73. (AMIE W14, 14 marks): Implement the following:

(i) $Y = (A + C)(A + D')(A + B + C')$ using NOR gates.

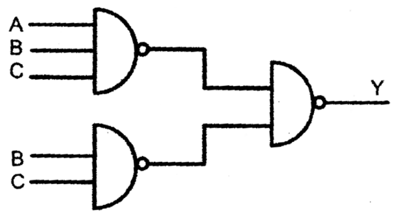
(ii) $Y = (AB + BC)C$ using NAND gates.

Answer:

(i)



(ii)



Q.74. (AMIE S16, 6 marks): Draw logic diagrams for the following: (i) EX-OR (ii) NOR (iii) AND.

*(For online support such as eBooks, video lectures, audio lectures, unsolved papers, online objective questions, test series and course updates, visit **www.amiestudycircle.com**)*