Number System

The technique to represent and work with numbers is called **number system**. **Decimal number system** is the most common number system. Other popular number systems include **binary number system, octal number system, hexadecimal number system,** etc.

**Decimal Number System**

Decimal number system is a **base 10** number system having 10 digits from 0 to 9. This means that any numerical quantity can be represented using these 10 digits. Decimal number system is also a **positional value system**. This means that the value of digits will depend on its position. Let us take an example to understand this.

Say we have three numbers – 734, 971 and 207. The value of 7 in all three numbers is different−

* In 734, value of 7 is 7 hundreds or 700 or 7 × 100 or 7 × 102
* In 971, value of 7 is 7 tens or 70 or 7 × 10 or 7 × 101
* In 207, value 0f 7 is 7 units or 7 or 7 × 1 or 7 × 100

The weightage of each position can be represented as follows −

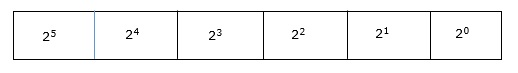


In digital systems, instructions are given through electric signals; variation is done by varying the voltage of the signal. Having 10 different voltages to implement decimal number system in digital equipment is difficult. So, many number systems that are easier to implement digitally have been developed. Let’s look at them in detail.

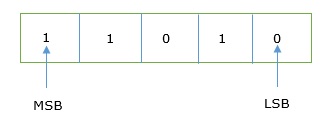
**Binary Number System**

The easiest way to vary instructions through electric signals is two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. The number system having just these two digits – 0 and 1 – is called **binary number system**.

Each binary digit is also called a **bit**. Binary number system is also positional value system, where each digit has a value expressed in powers of 2, as displayed here.



In any binary number, the rightmost digit is called **least significant bit (LSB)** and leftmost digit is called **most significant bit (MSB)**.



And decimal equivalent of this number is sum of product of each digit with its positional value.

110102 = 1×24 + 1×23 + 0×22 + 1×21 + 0×20

= 16 + 8 + 0 + 2 + 0

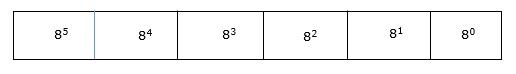
= 2610

Computer memory is measured in terms of how many bits it can store. Here is a chart for memory capacity conversion.

* 1 byte (B) = 8 bits
* 1 Kilobytes (KB) = 1024 bytes
* 1 Megabyte (MB) = 1024 KB
* 1 Gigabyte (GB) = 1024 MB
* 1 Terabyte (TB) = 1024 GB
* 1 Exabyte (EB) = 1024 PB
* 1 Zettabyte = 1024 EB
* 1 Yottabyte (YB) = 1024 ZB

**Octal Number System**

**Octal number system** has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8, as shown here −



Decimal equivalent of any octal number is sum of product of each digit with its positional value.

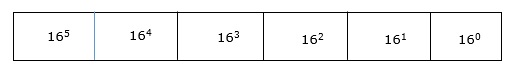
7268 = 7×82 + 2×81 + 6×80

= 448 + 16 + 6

= 47010

**Hexadecimal Number System**

**Octal number system** has 16 symbols – 0 to 9 and A to F where A is equal to 10, B is equal to 11 and so on till F. Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16, as shown here −



Decimal equivalent of any hexadecimal number is sum of product of each digit with its positional value.

27FB16 = 2×163 + 7×162 + 15×161 + 10×160

= 8192 + 1792 + 240 +10

= 1023410

**Number System Relationship**

The following table depicts the relationship between decimal, binary, octal and hexadecimal number systems.

|  |  |  |  |
| --- | --- | --- | --- |
| **HEXADECIMAL** | **DECIMAL** | **OCTAL** | **BINARY** |
| 0 | 0 | 0 | 0000 |
| 1 | 1 | 1 | 0001 |
| 2 | 2 | 2 | 0010 |
| 3 | 3 | 3 | 0011 |
| 4 | 4 | 4 | 0100 |
| 5 | 5 | 5 | 0101 |
| 6 | 6 | 6 | 0110 |
| 7 | 7 | 7 | 0111 |
| 8 | 8 | 10 | 1000 |
| 9 | 9 | 11 | 1001 |
| A | 10 | 12 | 1010 |
| B | 11 | 13 | 1011 |
| C | 12 | 14 | 1100 |
| D | 13 | 15 | 1101 |
| E | 14 | 16 | 1110 |
| F | 15 | 17 | 1111 |

**ASCII**

Besides numerical data, computer must be able to handle alphabets, punctuation marks, mathematical operators, special symbols, etc. that form the complete character set of English language. The complete set of characters or symbols are called alphanumeric codes. The complete alphanumeric code typically includes −

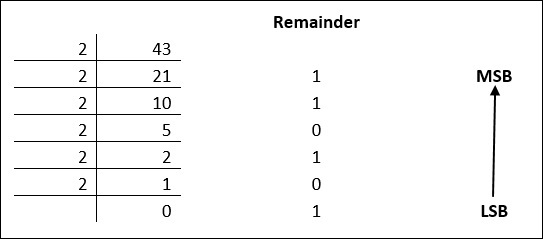
* 26 upper case letters
* 26 lower case letters
* 10 digits
* 7 punctuation marks
* 20 to 40 special characters

Now a computer understands only numeric values, whatever the number system used. So all characters must have a numeric equivalent called the alphanumeric code. The most widely used alphanumeric code is American Standard Code for Information Interchange (ASCII). ASCII is a 7-bit code that has 128 (27) possible codes.

As you know decimal, binary, octal and hexadecimal number systems are positional value number systems. To convert binary, octal and hexadecimal to decimal number, we just need to add the product of each digit with its positional value. Here we are going to learn other conversion among these number systems.

**Decimal to Binary**

Decimal numbers can be converted to binary by repeated division of the number by 2 while recording the remainder. Let’s take an example to see how this happens.

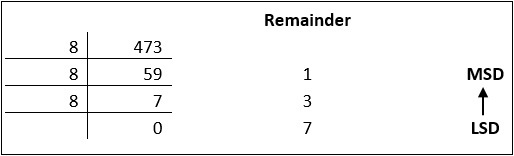


The remainders are to be read from bottom to top to obtain the binary equivalent.

4310 = 1010112

**Decimal to Octal**

Decimal numbers can be converted to octal by repeated division of the number by 8 while recording the remainder. Let’s take an example to see how this happens.

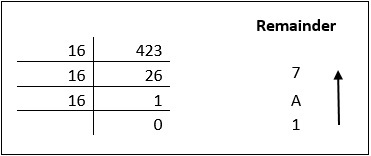


Reading the remainders from bottom to top,

47310 = 7318

**Decimal to Hexadecimal**

Decimal numbers can be converted to octal by repeated division of the number by 16 while recording the remainder. Let’s take an example to see how this happens.



Reading the remainders from bottom to top we get,

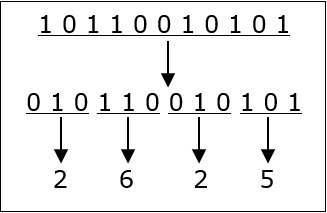
42310 = 1A716

**Binary to Octal and Vice Versa**

To convert a binary number to octal number, these steps are followed −

* Starting from the least significant bit, make groups of three bits.
* If there are one or two bits less in making the groups, 0s can be added after the most significant bit
* Convert each group into its equivalent octal number

Let’s take an example to understand this.



101100101012 = 26258

To convert an octal number to binary, each octal digit is converted to its 3-bit binary equivalent according to this table.

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

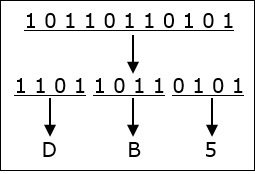
546738 = 1011001101110112

**Binary to Hexadecimal**

To convert a binary number to hexadecimal number, these steps are followed −

* Starting from the least significant bit, make groups of four bits.
* If there are one or two bits less in making the groups, 0s can be added after the most significant bit.
* Convert each group into its equivalent octal number.

Let’s take an example to understand this.



101101101012 = DB516

To convert an octal number to binary, each octal digit is converted to its 3-bit binary equivalent.

Number System

The technique to represent and work with numbers is called **number system**. **Decimal number system** is the most common number system. Other popular number systems include **binary number system, octal number system, hexadecimal number system,** etc.

**Decimal Number System**

Decimal number system is a **base 10** number system having 10 digits from 0 to 9. This means that any numerical quantity can be represented using these 10 digits. Decimal number system is also a **positional value system**. This means that the value of digits will depend on its position. Let us take an example to understand this.

Say we have three numbers – 734, 971 and 207. The value of 7 in all three numbers is different−

* In 734, value of 7 is 7 hundreds or 700 or 7 × 100 or 7 × 102
* In 971, value of 7 is 7 tens or 70 or 7 × 10 or 7 × 101
* In 207, value 0f 7 is 7 units or 7 or 7 × 1 or 7 × 100

The weightage of each position can be represented as follows −

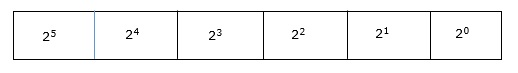


In digital systems, instructions are given through electric signals; variation is done by varying the voltage of the signal. Having 10 different voltages to implement decimal number system in digital equipment is difficult. So, many number systems that are easier to implement digitally have been developed. Let’s look at them in detail.

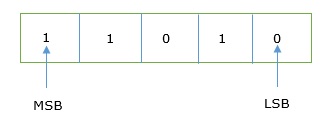
**Binary Number System**

The easiest way to vary instructions through electric signals is two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. The number system having just these two digits – 0 and 1 – is called **binary number system**.

Each binary digit is also called a **bit**. Binary number system is also positional value system, where each digit has a value expressed in powers of 2, as displayed here.



In any binary number, the rightmost digit is called **least significant bit (LSB)** and leftmost digit is called **most significant bit (MSB)**.



And decimal equivalent of this number is sum of product of each digit with its positional value.

110102 = 1×24 + 1×23 + 0×22 + 1×21 + 0×20

= 16 + 8 + 0 + 2 + 0

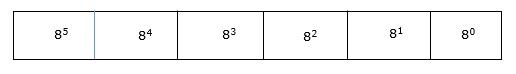
= 2610

Computer memory is measured in terms of how many bits it can store. Here is a chart for memory capacity conversion.

* 1 byte (B) = 8 bits
* 1 Kilobytes (KB) = 1024 bytes
* 1 Megabyte (MB) = 1024 KB
* 1 Gigabyte (GB) = 1024 MB
* 1 Terabyte (TB) = 1024 GB
* 1 Exabyte (EB) = 1024 PB
* 1 Zettabyte = 1024 EB
* 1 Yottabyte (YB) = 1024 ZB

**Octal Number System**

**Octal number system** has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8, as shown here −



Decimal equivalent of any octal number is sum of product of each digit with its positional value.

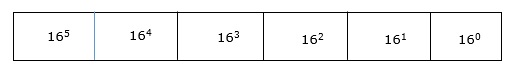
7268 = 7×82 + 2×81 + 6×80

= 448 + 16 + 6

= 47010

**Hexadecimal Number System**

**Octal number system** has 16 symbols – 0 to 9 and A to F where A is equal to 10, B is equal to 11 and so on till F. Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16, as shown here −



Decimal equivalent of any hexadecimal number is sum of product of each digit with its positional value.

27FB16 = 2×163 + 7×162 + 15×161 + 10×160

= 8192 + 1792 + 240 +10

= 1023410

**Number System Relationship**

The following table depicts the relationship between decimal, binary, octal and hexadecimal number systems.

|  |  |  |  |
| --- | --- | --- | --- |
| **HEXADECIMAL** | **DECIMAL** | **OCTAL** | **BINARY** |
| 0 | 0 | 0 | 0000 |
| 1 | 1 | 1 | 0001 |
| 2 | 2 | 2 | 0010 |
| 3 | 3 | 3 | 0011 |
| 4 | 4 | 4 | 0100 |
| 5 | 5 | 5 | 0101 |
| 6 | 6 | 6 | 0110 |
| 7 | 7 | 7 | 0111 |
| 8 | 8 | 10 | 1000 |
| 9 | 9 | 11 | 1001 |
| A | 10 | 12 | 1010 |
| B | 11 | 13 | 1011 |
| C | 12 | 14 | 1100 |
| D | 13 | 15 | 1101 |
| E | 14 | 16 | 1110 |
| F | 15 | 17 | 1111 |

**ASCII**

Besides numerical data, computer must be able to handle alphabets, punctuation marks, mathematical operators, special symbols, etc. that form the complete character set of English language. The complete set of characters or symbols are called alphanumeric codes. The complete alphanumeric code typically includes −

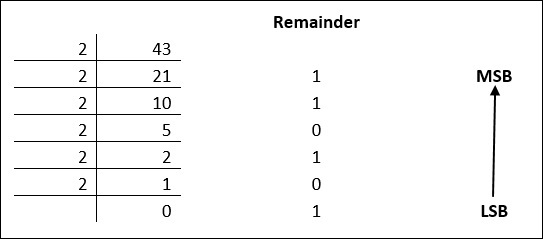
* 26 upper case letters
* 26 lower case letters
* 10 digits
* 7 punctuation marks
* 20 to 40 special characters

Now a computer understands only numeric values, whatever the number system used. So all characters must have a numeric equivalent called the alphanumeric code. The most widely used alphanumeric code is American Standard Code for Information Interchange (ASCII). ASCII is a 7-bit code that has 128 (27) possible codes.

As you know decimal, binary, octal and hexadecimal number systems are positional value number systems. To convert binary, octal and hexadecimal to decimal number, we just need to add the product of each digit with its positional value. Here we are going to learn other conversion among these number systems.

**Decimal to Binary**

Decimal numbers can be converted to binary by repeated division of the number by 2 while recording the remainder. Let’s take an example to see how this happens.

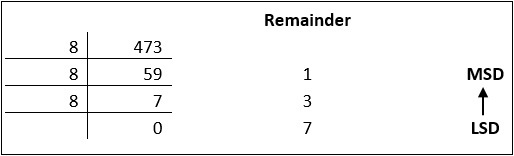


The remainders are to be read from bottom to top to obtain the binary equivalent.

4310 = 1010112

**Decimal to Octal**

Decimal numbers can be converted to octal by repeated division of the number by 8 while recording the remainder. Let’s take an example to see how this happens.

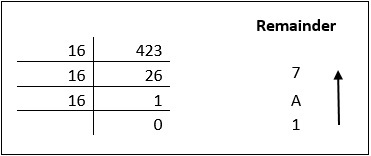


Reading the remainders from bottom to top,

47310 = 7318

**Decimal to Hexadecimal**

Decimal numbers can be converted to octal by repeated division of the number by 16 while recording the remainder. Let’s take an example to see how this happens.



Reading the remainders from bottom to top we get,

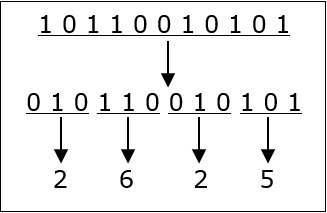
42310 = 1A716

**Binary to Octal and Vice Versa**

To convert a binary number to octal number, these steps are followed −

* Starting from the least significant bit, make groups of three bits.
* If there are one or two bits less in making the groups, 0s can be added after the most significant bit
* Convert each group into its equivalent octal number

Let’s take an example to understand this.



101100101012 = 26258

To convert an octal number to binary, each octal digit is converted to its 3-bit binary equivalent according to this table.

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

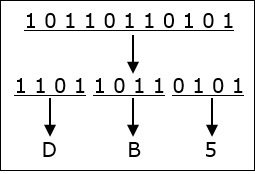
546738 = 1011001101110112

**Binary to Hexadecimal**

To convert a binary number to hexadecimal number, these steps are followed −

* Starting from the least significant bit, make groups of four bits.
* If there are one or two bits less in making the groups, 0s can be added after the most significant bit.
* Convert each group into its equivalent octal number.

Let’s take an example to understand this.



101101101012 = DB516

To convert an octal number to binary, each octal digit is converted to its 3-bit binary equivalent.

**Binary Codes**

In the coding, when numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or word is being encoded. The group of symbols is called as a code. The digital data is represented, stored and transmitted as group of binary bits. This group is also called as **binary code**. The binary code is represented by the number as well as alphanumeric letter.

**Advantages of Binary Code**

Following is the list of advantages that binary code offers.

* Binary codes are suitable for the computer applications.
* Binary codes are suitable for the digital communications.
* Binary codes make the analysis and designing of digital circuits if we use the binary codes.
* Since only 0 & 1 are being used, implementation becomes easy.

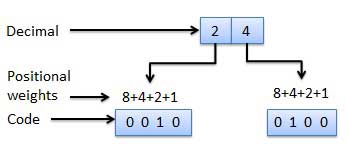
**Classification of binary codes**

The codes are broadly categorized into following four categories.

* Weighted Codes
* Non-Weighted Codes
* Binary Coded Decimal Code
* Alphanumeric Codes
* Error Detecting Codes
* Error Correcting Codes

**Weighted Codes**

Weighted binary codes are those binary codes which obey the positional weight principle. Each position of the number represents a specific weight. Several systems of the codes are used to express the decimal digits 0 through 9. In these codes each decimal digit is represented by a group of four bits.

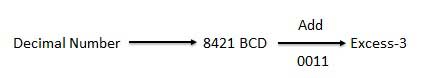


**Non-Weighted Codes**

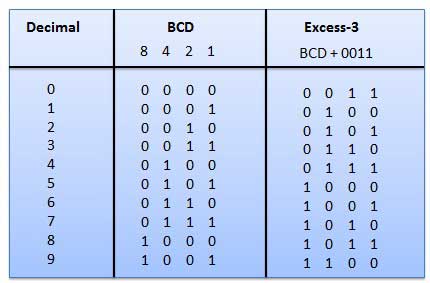
In this type of binary codes, the positional weights are not assigned. The examples of non-weighted codes are Excess-3 code and Gray code.

**Excess-3 code**

The Excess-3 code is also called as XS-3 code. It is non-weighted code used to express decimal numbers. The Excess-3 code words are derived from the 8421 BCD code words adding (0011)2 or (3)10 to each code word in 8421. The excess-3 codes are obtained as follows −

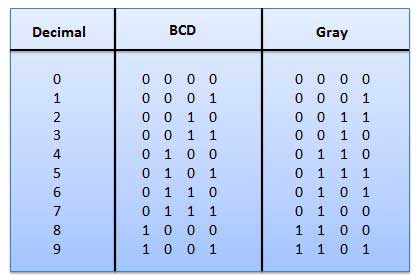


**Example**



**Gray Code**

It is the non-weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position. It has a very special feature that, only one bit will change each time the decimal number is incremented as shown in fig. As only one bit changes at a time, the gray code is called as a unit distance code. The gray code is a cyclic code. Gray code cannot be used for arithmetic operation.

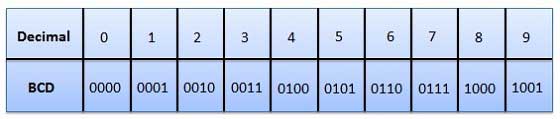


**Application of Gray code**

* Gray code is popularly used in the shaft position encoders.
* A shaft position encoder produces a code word which represents the angular position of the shaft.

**Binary Coded Decimal (BCD) code**

In this code each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code. In the BCD, with four bits we can represent sixteen numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.



**Advantages of BCD Codes**

* It is very similar to decimal system.
* We need to remember binary equivalent of decimal numbers 0 to 9 only.

**Disadvantages of BCD Codes**

* The addition and subtraction of BCD have different rules.
* The BCD arithmetic is little more complicated.
* BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

**Alphanumeric codes**

A binary digit or bit can represent only two symbols as it has only two states '0' or '1'. But this is not enough for communication between two computers because there we need many more symbols for communication. These symbols are required to represent 26 alphabets with capital and small letters, numbers from 0 to 9, punctuation marks and other symbols.

The alphanumeric codes are the codes that represent numbers and alphabetic characters. Mostly such codes also represent other characters such as symbol and various instructions necessary for conveying information. An alphanumeric code should at least represent 10 digits and 26 letters of alphabet i.e. total 36 items. The following three alphanumeric codes are very commonly used for the data representation.

* American Standard Code for Information Interchange (ASCII).
* Extended Binary Coded Decimal Interchange Code (EBCDIC).
* Five bit Baudot Code.

ASCII code is a 7-bit code whereas EBCDIC is an 8-bit code. ASCII code is more commonly used worldwide while EBCDIC is used primarily in large IBM computers.

**Error Codes**

There are binary code techniques available to detect and correct data during data transmission.

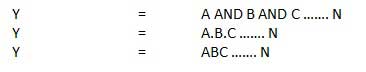
|  |  |
| --- | --- |
| **Error Code** | **Description** |

**LOGIC GATES AND COMBINATIONAL LOGIC CIRCUIT**

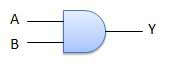
Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on a **certain logic**. Based on this, logic gates are named as AND gate, OR gate, NOT gate etc.

**AND Gate**

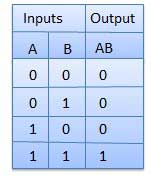
A circuit which performs an AND operation is shown in figure. It has n input (n >= 2) and one output.



**Logic diagram**



**Truth Table**



**OR Gate**

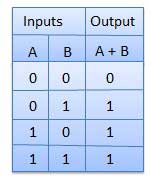
A circuit which performs an OR operation is shown in figure. It has n input (n >= 2) and one output.

OR gate

**Logic diagram**

OR Logical Diagram

**Truth Table**

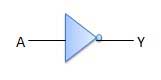


**NOT Gate**

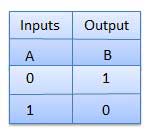
NOT gate is also known as **Inverter**. It has one input A and one output Y.

NOT gate

**Logic diagram**



**Truth Table**

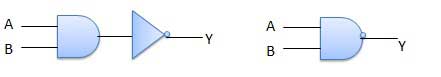


**NAND Gate**

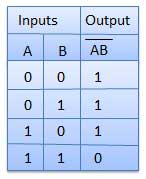
A NOT-AND operation is known as NAND operation. It has n input (n >= 2) and one output.

NAND gate

**Logic diagram**



**Truth Table**

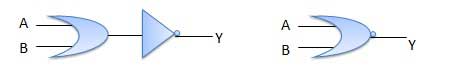


**NOR Gate**

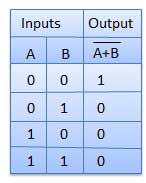
A NOT-OR operation is known as NOR operation. It has n input (n >= 2) and one output.

NOR gate

**Logic diagram**

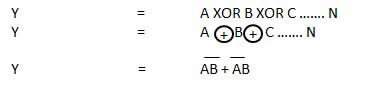


**Truth Table**

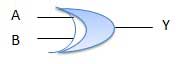


**XOR Gate**

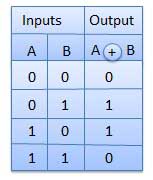
XOR or Ex-OR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-OR gate is abbreviated as EX-OR gate or sometime as X-OR gate. It has n input (n >= 2) and one output.



**Logic diagram**

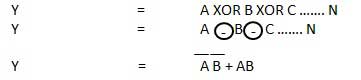


**Truth Table**

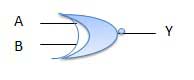


**XNOR Gate**

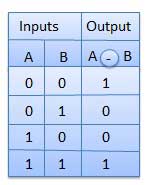
XNOR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-NOR gate is abbreviated as EX-NOR gate or sometime as X-NOR gate. It has n input (n >= 2) and one output.



**Logic diagram**



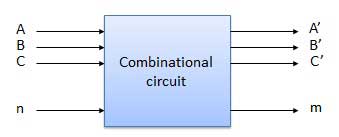
**Truth Table**



Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer. Some of the characteristics of combinational circuits are following −

* The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
* The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
* A combinational circuit can have an n number of inputs and m number of outputs.

### Block diagram

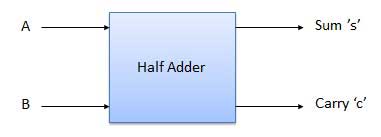


We're going to elaborate few important combinational circuits as follows.

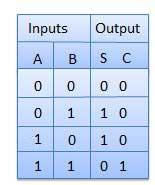
## Half Adder

Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**.

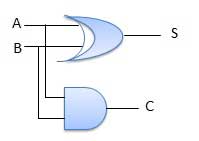
### Block diagram



### Truth Table



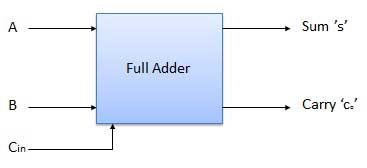
### Circuit Diagram



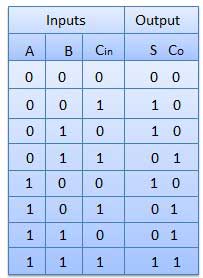
## Full Adder

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

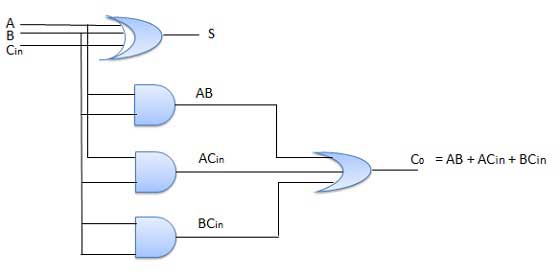
### Block diagram



### Truth Table



### Circuit Diagram



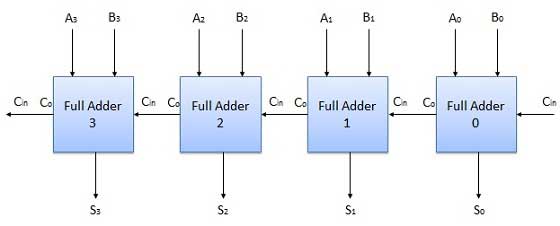
## N-Bit Parallel Adder

The Full Adder is capable of adding only two single digit binary number along with a carry input. But in practical we need to add binary numbers which are much longer than just one bit. To add two n-bit binary numbers we need to use the n-bit parallel adder. It uses a number of full adders in cascade. The carry output of the previous full adder is connected to carry input of the next full adder.

### 4 Bit Parallel Adder

In the block diagram, A0 and B0 represent the LSB of the four bit words A and B. Hence Full Adder-0 is the lowest stage. Hence its Cin has been permanently made 0. The rest of the connections are exactly same as those of n-bit parallel adder is shown in fig. The four bit parallel adder is a very common logic circuit.

### Block diagram



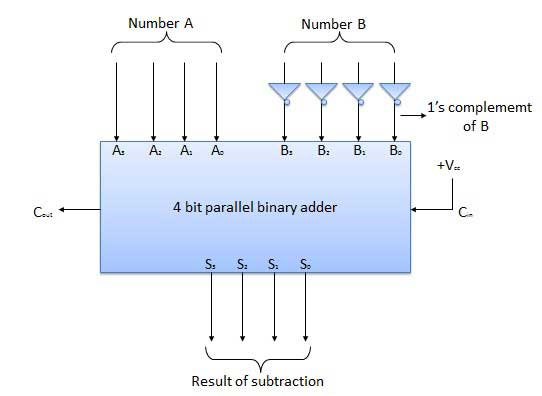
## N-Bit Parallel Subtractor

The subtraction can be carried out by taking the 1's or 2's complement of the number to be subtracted. For example we can perform the subtraction (A-B) by adding either 1's or 2's complement of B to A. That means we can use a binary adder to perform the binary subtraction.

### 4 Bit Parallel Subtractor

The number to be subtracted (B) is first passed through inverters to obtain its 1's complement. The 4-bit adder then adds A and 2's complement of B to produce the subtraction. S3 S2 S1 S0 represents the result of binary subtraction (A-B) and carry output Cout represents the polarity of the result. If A > B then Cout = 0 and the result of binary form (A-B) then Cout = 1 and the result is in the 2's complement form.

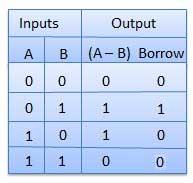
### Block diagram



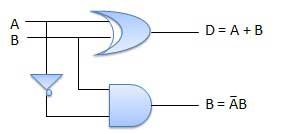
## Half Subtractors

Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit.

### Truth Table



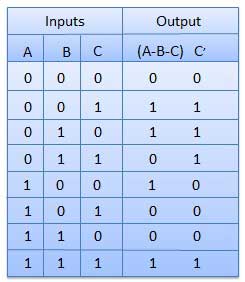
### Circuit Diagram



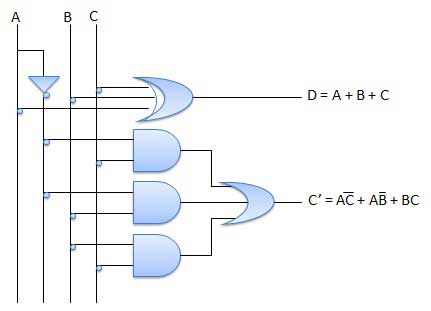
## Full Subtractors

The disadvantage of a half subtractor is overcome by full subtractor. The full subtractor is a combinational circuit with three inputs A,B,C and two output D and C'. A is the 'minuend', B is 'subtrahend', C is the 'borrow' produced by the previous stage, D is the difference output and C' is the borrow output.

### Truth Table



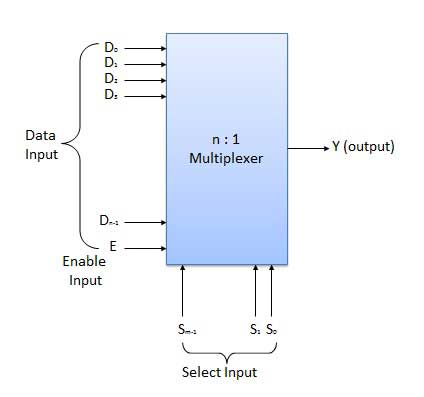
### Circuit Diagram



## Multiplexers

Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with 2m = n. It is a digital circuit which selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the selected inputs. Depending on the digital code applied at the selected inputs, one out of n data sources is selected and transmitted to the single output Y. E is called the strobe or enable input which is useful for the cascading. It is generally an active low terminal that means it will perform the required operation when it is low.

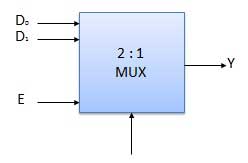
### Block diagram



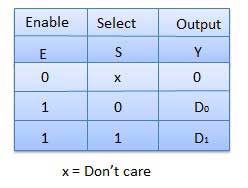
Multiplexers come in multiple variations

* 2 : 1 multiplexer
* 4 : 1 multiplexer
* 16 : 1 multiplexer
* 32 : 1 multiplexer

### Block Diagram



### Truth Table



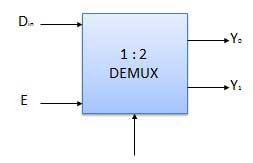
## Demultiplexers

A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs. It has only one input, n outputs, m select input. At a time only one output line is selected by the select lines and the input is transmitted to the selected output line. A de-multiplexer is equivalent to a single pole multiple way switch as shown in fig.

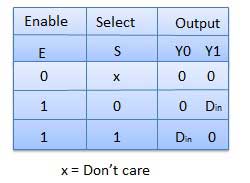
Demultiplexers comes in multiple variations.

* 1 : 2 demultiplexer
* 1 : 4 demultiplexer
* 1 : 16 demultiplexer
* 1 : 32 demultiplexer

### Block diagram



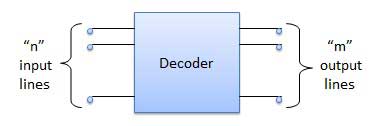
### Truth Table



## Decoder

A decoder is a combinational circuit. It has n input and to a maximum m = 2n outputs. Decoder is identical to a demultiplexer without any data input. It performs operations which are exactly opposite to those of an encoder.

### Block diagram



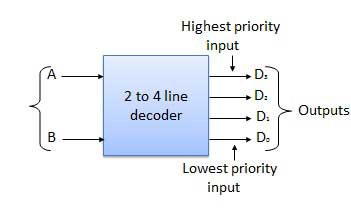
Examples of Decoders are following.

* Code converters
* BCD to seven segment decoders
* Nixie tube decoders
* Relay actuator

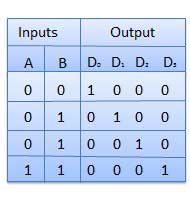
## 2 to 4 Line Decoder

The block diagram of 2 to 4 line decoder is shown in the fig. A and B are the two inputs where D through D are the four outputs. Truth table explains the operations of a decoder. It shows that each output is 1 for only a specific combination of inputs.

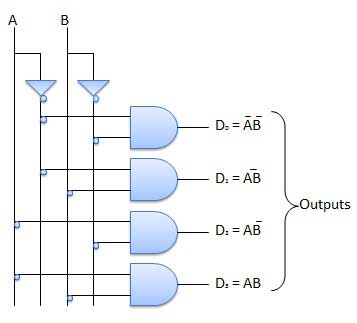
### Block diagram



### Truth Table



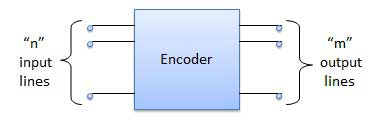
### Logic Circuit



## Encoder

Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder. An encoder has n number of input lines and m number of output lines. An encoder produces an m bit binary code corresponding to the digital input number. The encoder accepts an n input digital word and converts it into an m bit another digital word.

### Block diagram



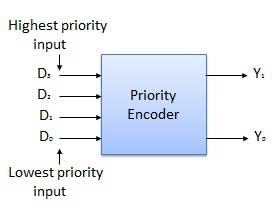
Examples of Encoders are following.

* Priority encoders
* Decimal to BCD encoder
* Octal to binary encoder
* Hexadecimal to binary encoder

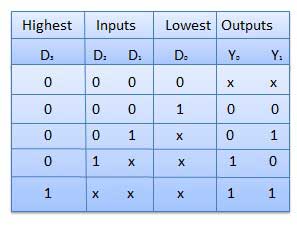
## Priority Encoder

This is a special type of encoder. Priority is given to the input lines. If two or more input line are 1 at the same time, then the input line with highest priority will be considered. There are four input D0, D1, D2, D3 and two output Y0, Y1. Out of the four input D3 has the highest priority and D0 has the lowest priority. That means if D3 = 1 then Y1 Y1 = 11 irrespective of the other inputs. Similarly if D3 = 0 and D2 = 1 then Y1 Y0 = 10 irrespective of the other inputs.

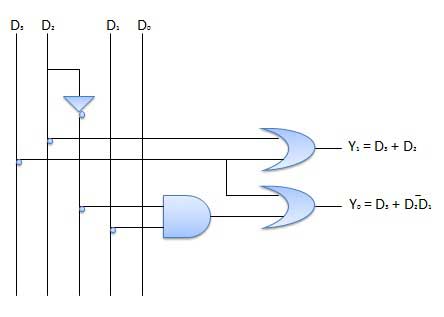
### Block diagram



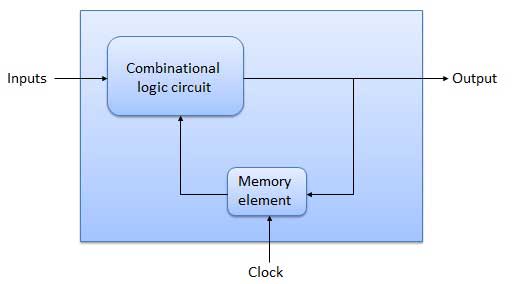
### Truth Table



### Logic Circuit



## Block diagram



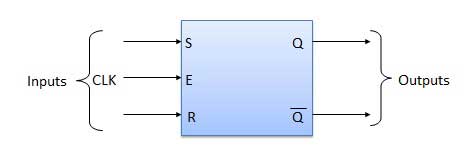
## Flip Flop

Flip flop is a sequential circuit which generally samples its inputs and changes its outputs only at particular instants of time and not continuously. Flip flop is said to be edge sensitive or edge triggered rather than being level triggered like latches.

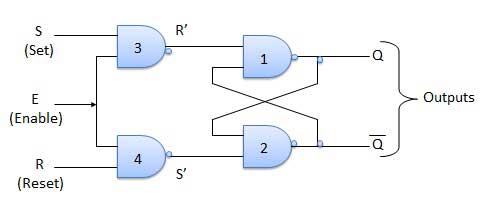
## S-R Flip Flop

It is basically S-R latch using NAND gates with an additional **enable** input. It is also called as level triggered SR-FF. For this, circuit in output will take place if and only if the enable input (E) is made active. In short this circuit will operate as an S-R latch if E = 1 but there is no change in the output if E = 0.

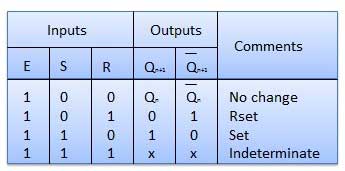
### Block Diagram



### Circuit Diagram



### Truth Table



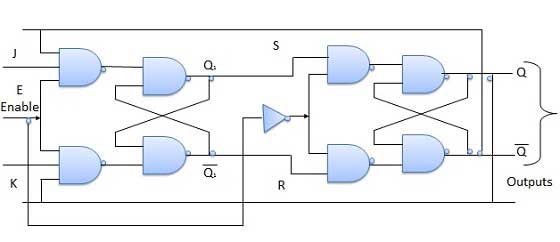
### Operation

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Condition** | **Operation** |
| 1 | **S = R = 0 : No change** | If S = R = 0 then output of NAND gates 3 and 4 are forced to become 1.  Hence R' and S' both will be equal to 1. Since S' and R' are the input of the basic S-R latch using NAND gates, there will be no change in the state of outputs. |
| 2 | **S = 0, R = 1, E = 1** | Since S = 0, output of NAND-3 i.e. R' = 1 and E = 1 the output of NAND-4 i.e. S' = 0.  Hence Qn+1 = 0 and Qn+1 bar = 1. This is reset condition. |
| 3 | **S = 1, R = 0, E = 1** | Output of NAND-3 i.e. R' = 0 and output of NAND-4 i.e. S' = 1.  Hence output of S-R NAND latch is Qn+1 = 1 and Qn+1 bar = 0. This is the reset condition. |
| 4 | **S = 1, R = 1, E = 1** | As S = 1, R = 1 and E = 1, the output of NAND gates 3 and 4 both are 0 i.e. S' = R' = 0.  Hence the **Race** condition will occur in the basic NAND latch. |

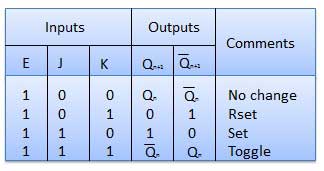
## Master Slave JK Flip Flop

Master slave JK FF is a cascade of two S-R FF with feedback from the output of second to input of first. Master is a positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level. Hence when the clock = 1 (positive level) the master is active and the slave is inactive. Whereas when clock = 0 (low level) the slave is active and master is inactive.

### Circuit Diagram



### Truth Table



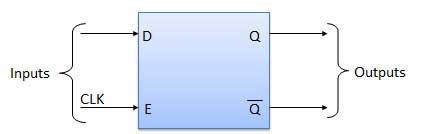
### Operation

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Condition** | **Operation** |
| 1 | **J = K = 0 (No change)** | When clock = 0, the slave becomes active and master is inactive. But since the S and R inputs have not changed, the slave outputs will also remain unchanged. Therefore outputs will not change if J = K =0. |
| 2 | **J = 0 and K = 1 (Reset)** | Clock = 1 − Master active, slave inactive. Therefore outputs of the master become Q1 = 0 and Q1 bar = 1. That means S = 0 and R =1.  Clock = 0 − Slave active, master inactive. Therefore outputs of the slave become Q = 0 and Q bar = 1.  Again clock = 1 − Master active, slave inactive. Therefore even with the changed outputs Q = 0 and Q bar = 1 fed back to master, its output will be Q1 = 0 and Q1 bar = 1. That means S = 0 and R = 1.  Hence with clock = 0 and slave becoming active the outputs of slave will remain Q = 0 and Q bar = 1. Thus we get a stable output from the Master slave. |
| 3 | **J = 1 and K = 0 (Set)** | Clock = 1 − Master active, slave inactive. Therefore outputs of the master become Q1 = 1 and Q1 bar = 0. That means S = 1 and R =0.  Clock = 0 − Slave active, master inactive. Therefore outputs of the slave become Q = 1 and Q bar = 0.  Again clock = 1 − then it can be shown that the outputs of the slave are stabilized to Q = 1 and Q bar = 0. |
| 4 | **J = K = 1 (Toggle)** | Clock = 1 − Master active, slave inactive. Outputs of master will toggle. So S and R also will be inverted.  Clock = 0 − Slave active, master inactive. Outputs of slave will toggle.  These changed output are returned back to the master inputs. But since clock = 0, the master is still inactive. So it does not respond to these changed outputs. This avoids the multiple toggling which leads to the race around condition. The master slave flip flop will avoid the race around condition. |

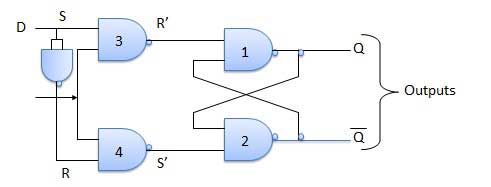
## Delay Flip Flop / D Flip Flop

Delay Flip Flop or D Flip Flop is the simple gated S-R latch with a NAND inverter connected between S and R inputs. It has only one input. The input data is appearing at the output after some time. Due to this data delay between i/p and o/p, it is called delay flip flop. S and R will be the complements of each other due to NAND inverter. Hence S = R = 0 or S = R = 1, these input condition will never appear. This problem is avoid by SR = 00 and SR = 1 conditions.

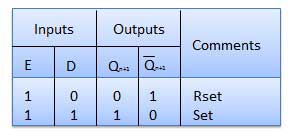
### Block Diagram



### Circuit Diagram



### Truth Table



### Operation

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Condition** | **Operation** |
| 1 | **E = 0** | Latch is disabled. Hence no change in output. |
| 2 | **E = 1 and D = 0** | If E = 1 and D = 0 then S = 0 and R = 1. Hence irrespective of the present state, the next state is Qn+1 = 0 and Qn+1 bar = 1. This is the reset condition. |
| 3 | **E = 1 and D = 1** | If E = 1 and D = 1, then S = 1 and R = 0. This will set the latch and Qn+1 = 1 and Qn+1 bar = 0 irrespective of the present state. |

## Toggle Flip Flop / T Flip Flop

Toggle flip flop is basically a JK flip flop with J and K terminals permanently connected together. It has only input denoted by **T** as shown in the Symbol Diagram. The symbol for positive edge triggered T flip flop is shown in the Block Diagram.

### Symbol Diagram