

Unit 1

BINARY SYSTEMS

Unit 1: Binary Systems (6 Hrs.)

Digital Systems, Binary numbers, Number base conversion, Octal and hexadecimal numbers, compliments, Signed Binary numbers, Decimal codes (BCD, 2 4 2 1, 8 4 -2 -1, Excess 3, Gray Code), Binary Storage and Registers, Binary logic

Digital Systems

Introduction:

The general-purpose digital computer is the best example of digital systems. Other examples include, digital watch, digital calculator, digital display, telephone switching exchanges etc. The most widely used digital system is the digital computer itself. Digital computers have made possible many scientific, industrial and commercial advances that would have been unattainable otherwise. Advance scientific studies, researches and experiments require real-time, continuous computer monitoring, and many businesses enterprises function efficiently only with the aid of automatic data processing. Computers are used in scientific calculations, commercial and business data processing, air traffic control, space guidance, the educational field, and many other areas. Digital computers are largely used in design, manufacture, distribution and sales. Besides, we use digital computers in education, exploration, entertainment and many other fields to accomplish wide variety of data processing tasks.

The major characteristic of digital system is manipulation of discrete elements of information. Such elements of information may be electric impulses, numbers, alphabets, operations etc. mostly the computers are known for numerical computations. In this case digits are the discrete elements and hence the name digital computer has emerged. A more appropriate word for a digital computer would be “discrete information processing system”.

Discrete elements of information are represented in digital systems by physical quantities called signals. Electrical signals such as voltage and currents are the most common. A piece of signal has two possible values and hence is binary in nature. The digital system designer is restricted to use two binary signals. Multivalued discrete systems are found to be less reliable than binary system. A circuit in binary system will be in two states. A system with ten states could be designed with ten states circuit using one discrete voltage value for each state, but it would possess very less reliability. For example, a transistor circuit has two states “on” and “off” and is extremely reliable. Moreover, human logic also tend to be binary.

The two constraints in digital systems are:

- Information representation in discrete format. It means, system can be in either of the fixed, predefined, discrete states. Not continuous.

- The information is stored in binary format. System must have only two states.



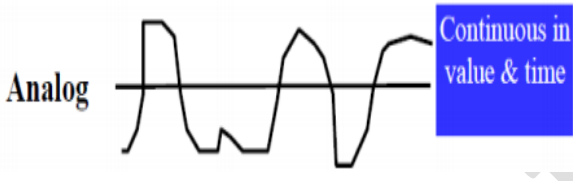
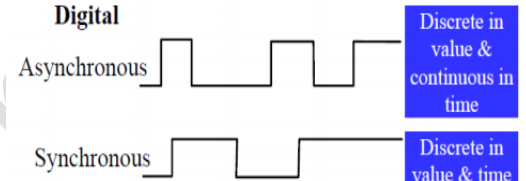
Discrete quantities of information arise either from the nature of the process or may be quantized from a continuous process. For example, number of people is by nature discrete information. The weight of a person is continuous in nature. In a digital weighing machine, the continuous information is needed to be quantized.

Information representation in digital systems:

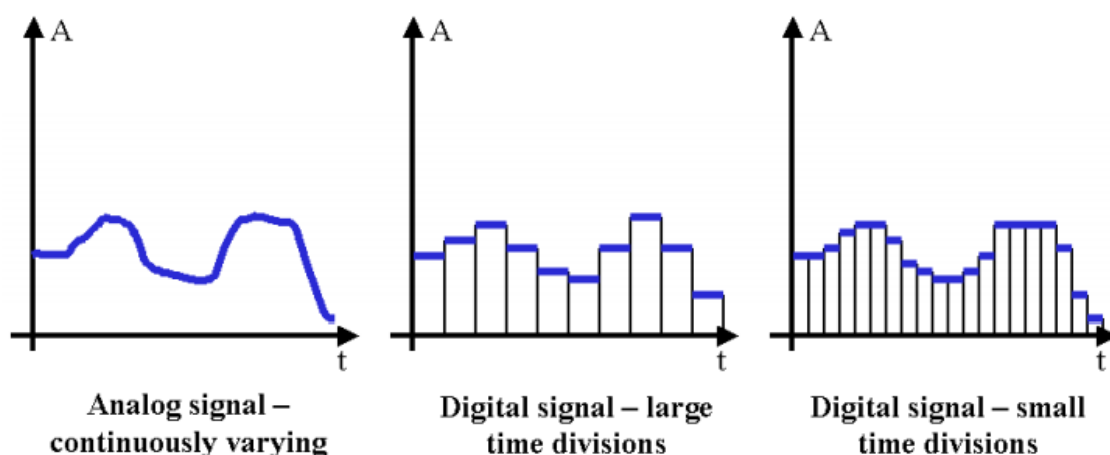
- Process variables (Information) are represented in the form of discrete signals such as voltage.
- The discrete signals (presence and absence of voltage) are represented by two values (for discrete binary systems) 0 and 1.
- Signal values are obtained after quantization of the process variables (information).
- After quantization, a single piece of discrete value may represent a certain range of real-time-value.
- Lower the quantization range, higher is the precision of digital system.

Differences between analog and digital systems:

Analog System	Digital System
Can perform direct simulation of the physical processes. If the variables of the process are discrete in nature, they are converted to continuous signals by using digital to analog converter (modulator).	To simulate a physical process in a digital system, the quantities (process variable) must be quantized. If the variables of the process are real time continuous signals, they are quantized by analog to digital converter (demodulator).
Variables in an analog system are represented by the continuous signals. Usually by the electric voltage that vary with time.	Variables in a digital system are represented by the discrete signals. Usually in binary state.
The process variables and the signals are analogous in behave in the same manner. For example, in an analog voltmeter, the process variable is voltage and same can be used as signal.	The process variables and the signals are not analogous in behave in the same manner. For example, in a digital speedometer, the rotation of wheel (process variable) is represented by the quantized electrical signals.
The analog signals are continuous in nature.	The digital signals are discrete in nature.
An analog signal may represent infinite states.	A digital signal may represent either of the fixed number of states. (two states in binary digital system).
A physical system whose behavior is described by mathematical equations is simulated in analog system by using calculus.	A physical system whose behavior is described by mathematical equations is simulated in digital system by using numerical methods.

There are infinite possible values between any two values, however the ability to read them precisely may differ.	There are only fixed number of values between any two values. They can be read precisely.
 <p>Above speedometer is an analog system representing probably 1.25. It can represent infinite values between 1.25 and 1.26. We can increase the precision of the system by 0</p>	 <p>Above speedometer is a digital system representing exactly 6.25. The next value it can represent is 6.26. It can't represent any value between 6.25 and 6.26. We can increase the precision of the system by using better architecture.</p>
Analog systems use microdevices such as shifts and gates.	Digital systems contain devices such as logic gates, flip-flops, shift registers and counters.
Signal represents continuous values.	Signal represents discrete values.
	

Here is an example waveform of a quantized signal. In case of the digital signal, notice how the magnitude of the wave can only take certain values, and that creates a step-like appearance. This image is discrete in magnitude, but is continuous in time (asynchronous):



Note that, in digital systems, the signal represent two states (normally indicated by 0 and 1) of the corresponding physical quantity. The two states are the quantized representation of corresponding physical variable. The physical variable being:

- High and low voltage in CPU.
- Magnetized and non-magnetized spot in magnetic disk.
- Pits and lands in CD.

- Electrical charge + and – in RAM etc.

Working principle of digital computers:

The best example of a digital system is a digital computer system. Following diagram represent the basic concept of digital computer system.

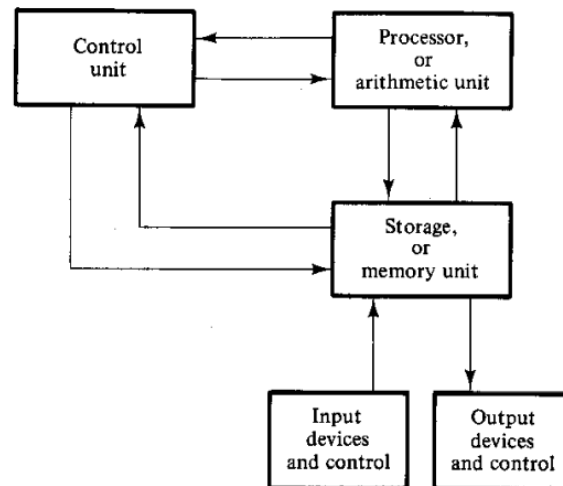


Fig: Block diagram of a digital computer system

The memory unit stores data regarding input, output or intermediate. It also stores the program to be executed. The processing unit performs the arithmetic and logic operations as instructed by the program. The control unit generates and gives control signals to various units to implement any instruction mentioned in the program. The program and data prepared by the user are transferred into the memory unit by means of an input device such as a keyboard. And output device such as printer receives the result of a computation and provides the result to the user. The input and output devices are special digital systems driven by electromechanical parts and controlled by electronic digital circuits.

Advantages of digital systems:

- It made many scientific, industrial, and commercial advances possible that would have been unattainable otherwise.
- It is less expensive and more reliable.
- Provides higher flexibility and compatibility
- Information storage can be easier in digital computer systems than in analog ones.
- Digital systems are easier to add new features and upgrade.

Disadvantages of digital systems:

- It is highly fragile system. Misinterpretation of tiny bit results in large error.
- Physical systems cannot be directly represented, rather quantization is needed which may result in quantization error.
- Poor architecture may result into higher data limitation.

The digital computer manipulates discrete elements of information represented in binary form. Data needed for calculation are represented in binary number system. Decimal digits are stored in binary codes. Data storage is done in binary storage elements. The instructions

are represented in binary systems. It is the most important concept to have the idea of binary numbers to better understand the digital systems.

Number System:

Number system is a system of representing values by using a set of specified symbols following the predefined rules.

The two types of number systems are:

- Non-positional number system
- Positional number system

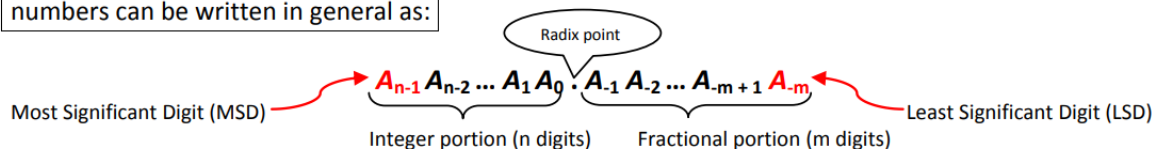
In the non-positional number systems, the digits do not have their place values. Each symbol has a fixed value, irrespective of the position it is placed. For example, roman number system.

In the positional number system, numbers are expressed in a sequence of digits. Each digit has its face value and place value. Face value of the digit is fixed and the place value of the digits depends in its position in the sequence of digits.

Representation of positional number system:

Here we discuss positional number systems with Positive radix (or base) r . A number with radix r is represented by a string of digits as:

numbers can be written in general as:



Value of each A_i is such that $0 \leq A_i \leq r$. Where;

- $A_i \Rightarrow$ symbols used in the number system (Eg: 0-9 for decimal & 0,1 for binary number system)
- $r \Rightarrow$ radix of the number system. It is equal to the total number of symbols used in the system. For example, $r=10$ for decimal number system & $r=2$ for binary number system.
- $i \Rightarrow$ Position of the digit/symbol being used in the number.

In general, a number in base r contains r digits, 0, 1, 2... $r-1$, and is expressed as a power series in r with the general form:

$$(\text{Number})_r = A_{n-1} r^{n-1} + A_{n-2} r^{n-2} + \dots + A_1 r^1 + A_0 r^0 + A_{-1} r^{-1} + A_{-2} r^{-2} + \dots + A_{-m+1} r^{-m+1} + A_{-m} r^{-m}$$

$$(\text{Number})_r = \left(\sum_{i=0}^{n-1} A_i \cdot r^i \right) + \left(\sum_{j=-m}^{-1} A_j \cdot r^j \right)$$

(Integer Portion) + (Fraction Portion)

Number Systems Overview:**1. Decimal number system:**

Radix/Base (r)	10
Symbols	0 to r-1 => 0,1,2,3,4,5,6,7,8,9
Example: 426.56 ₁₀	$(4 \times 10^2 + 2 \times 10^1 + 6 \times 10^0) + (5 \times 10^{-1} + 6 \times 10^{-2})$ Whole part fractional part
Use	Daily arithmetic calculation in real life.

It is used vastly in everyday arithmetic besides computers to represent numbers by strings of digits or symbols defined above, possibly with a decimal point. Depending on its position in the string, each digit has an associated value of an integer raised to the power of 10.

2. Binary Number System:

Radix/Base (r)	2
Symbols	0 to r-1 => 0,1
Example: 11010.01 ₂ is written in decimal as	$(1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + (1 \times 2^{-1} + 0 \times 2^{-2})) = (26.25)_{10}$ Whole part fractional
Use	Used to represent values in binary digital systems. The binary number system is very useful in computer technology and computer programming languages also uses binary number system that is helpful in digital encoding. The binary number system can also be used in Boolean algebra.

Digits in a binary number are called bits (Binary digITs). When a bit is equal to 0, it does not contribute to the sum during the conversion. Therefore, the conversion to decimal can be obtained by adding the numbers with powers of 2 corresponding to the bits that are equal to 1. Looking at above example,

$$(11010.01)_2 = 16 + 8 + 2 + 0.25 = (26.25)_{10}$$

Note: Please revise the binary arithmetic (addition, subtraction and multiplication and division of binary numbers).

3. Octal Number System:

Radix/Base (r)	8
Symbols	0 to r-1 => 0,1,2,3,4,5,6,7
Example: (426.56) ₈ Written in decimal as	$(4 \times 8^2 + 2 \times 8^1 + 6 \times 8^0) + (5 \times 8^{-1} + 6 \times 8^{-2}) = (278.72)_8$ Whole fractional
Use	The octal numbers are not as common as they used to be. However, Octal is used when the number of bits in one word is a multiple of 3. It is also used as a shorthand for

	representing file permissions on UNIX systems and representation of UTF8 numbers, etc.
--	--

Octal number are expressed with a string of symbols defined above, possibly, an octal point within it. The decimal equivalent of a octal number can be found by expanding the number into a power series with a base of 8 as shown above.

4. Hexadecimal Number System:

Radix/Base (r)	16
Symbols	0 to r-1 => 0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F Where; A = 10, B=11, C=12, D=13, E=14, F=15
Example: (4B6.56) ₁₆	$(4 \times 16^2 + 11 \times 16^1 + 6 \times 16^0) + (5 \times 16^{-1} + 6 \times 16^{-2}) = (1206.34)_{16}$ Whole part fractional part
Use	Hexadecimal Number System is commonly used in Computer programming and Microprocessors. It is also helpful to describe colors on web pages. Each of the three primary colors (i.e., red, green and blue) is represented by two hexadecimal digits to create 255 possible values, thus resulting in more than 16 million possible colors. Hexadecimal number system is used to describe locations in memory for every byte. These hexadecimal numbers are also easier to read and write than binary or decimal numbers for Computer Professionals.

A hexadecimal number is expressed with a string of symbols defined above (0-F), possibly, a hexadecimal point with in it. The decimal equivalent of a hexadecimal number can be found by expanding the number into a power series with a base of 16.

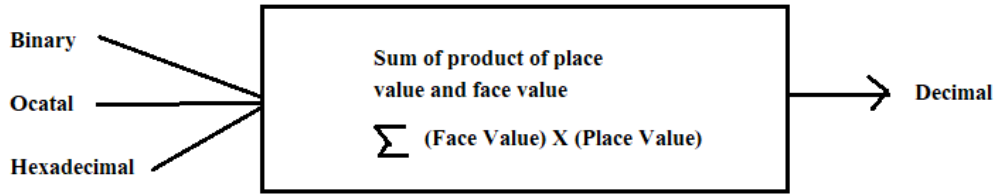
Following figure shows the representation of first 16 decimal numbers in different number systems.

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Fig: First 16 numbers in different number system

Number Base Conversion:

1. Conversion from any number system to decimal:



Example: $(100110)_2 = (?)_{10}$

Face Value	1	0	0	1	1	0
Place Value	32	16	8	4	2	1

$$\text{Answer} = (1 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1)_{10}$$

$$= (38)_{10}$$

Example: $(245)_8 = (?)_{10}$

Face Value	2	4	5
Place Value	64	8	1

$$\text{Answer} = (2 \times 64 + 4 \times 8 + 5 \times 1)_{10}$$

$$= (165)$$

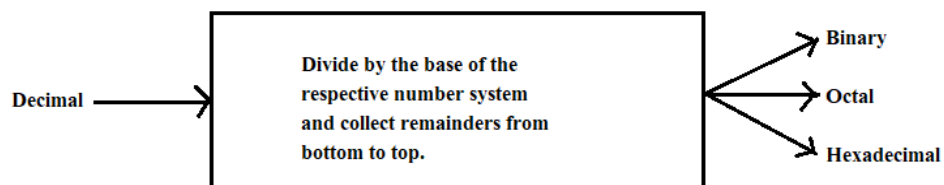
Example: $(4B6)_{16} = (?)_{10}$

Face Value	4	11	6
Place Value	256	16	1

$$\text{Answer} = (4 \times 256 + 11 \times 16 + 6 \times 1)_{10}$$

$$= (1206)_{10}$$

2. Conversion from decimal to any other number system:



Example: $(120)_{10} = (?)_2$

2	120		
2	60	-	0
2	30	-	0
2	15	-	0
2	7	-	1
2	3	-	1
2	1	-	1
	0	-	1

Answer = $(1111000)_2$

Example: $(120)_{10} = (?)_8$

8	120		
8	15	-	0
8	1	-	7
	0	-	1

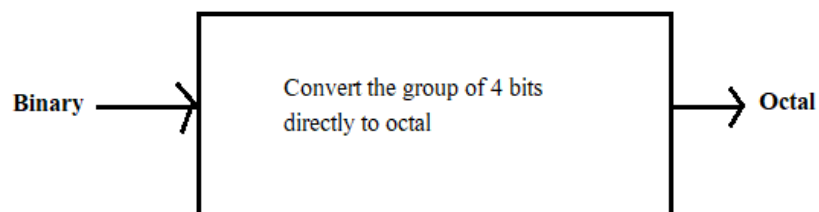
Answer = $(170)_8$

Example: $(120)_{10} = (?)_{16}$

16	120		
16	7	-	8
	0	-	7

Answer = $(78)_{16}$

3. Binary to octal conversion:

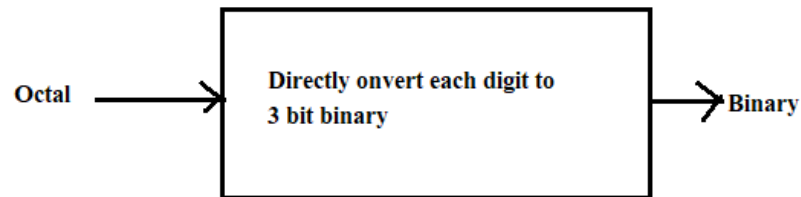


Example: $(1001101010110)_2 = (?)_8$

1001010110
 ↓ ↓ ↓ ↓
 1 1 2 6

Answer = $(1126)_8$

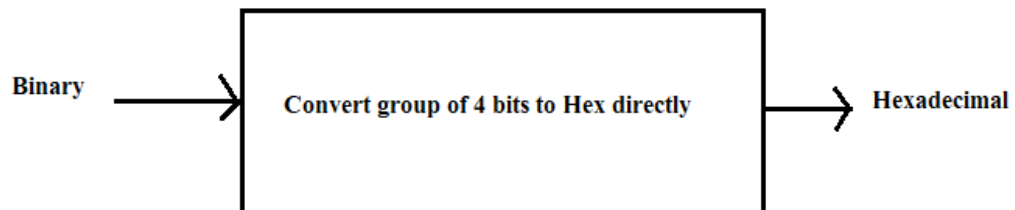
4. Octal to binary conversion:



Example: $(452)_8 = (?)_2$

4 5 2
 ↓ ↓ ↓
 100 101 010
 Ans = $(100101010)_2$

5. Binary to hexadecimal conversion:

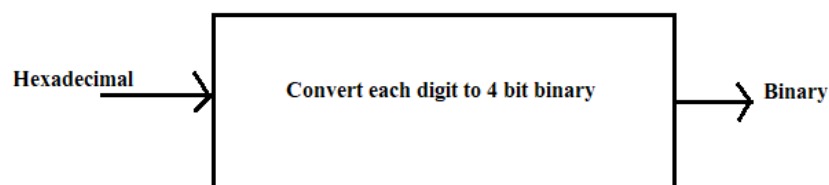


Example: $(110010101110)_2 = (?)$

110010101110
 ↓ ↓ ↓
 B A E

Ans = $(BAE)_{16}$

6. Hexadecimal to binary conversion:

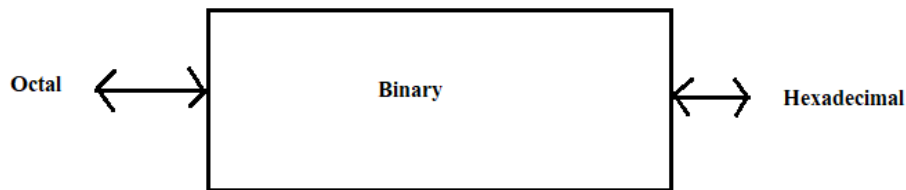


Example: $(CAB8)_{16} = (?)_2$

$\begin{array}{cccc} C & A & B & 8 \\ \swarrow & \downarrow & \downarrow & \searrow \\ 1100 & 1010 & 1011 & 1000 \end{array}$

Ans = $(1100101010111000)_2$

7. Octal to hexadecimal conversion (And vice-versa):



To convert from octal to hexadecimal, first we convert octal to binary and then binary to hexadecimal. Similarly, for hexadecimal to octal.

Note that, computer understands only binary numbers. If the human needs to communicate, use of binary system is hectic. Use of binary numbers in human communication is too complex because it needs large numbers of digits to represent a simple value. For example, to represent $(B5C)_{16}$ 3 digits, we need $(101101011100)_2$ 12 digits.

Complements:

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulations. There are two types of complements for base- r system.

- The r 's complement (Example: 2's complement for binary, 10's complement for decimal)
- The $(r-1)$'s complement (Example: 1's complement for binary, 9's complement for decimal)

a. The r 's complement (radix complement):

Given a positive number N in base r with an integer part of n digits, the r 's complement of N is defined as $r^n - N$ for $N > 0$ and 0 for $N = 0$.

Example: Find 10's complement of $(256)_{10}$.

Answer : Here,

$$N = 256, n = 3 \text{ and } r = 10$$

$$r\text{'s complement of } N = 10\text{'s complement of } 256_{10}$$

$$= r^n - N$$

$$= 10^3 - 256$$

$$= (744)_{10}$$

Note: We can easily find the 10's complement of a decimal number by following procedure:

- Leave all the least significant zeros unchanged.
- Subtract first non-zero least significant digit from 10.
- Subtract all the other higher significant digits from 9.
- Eg: 10's complement of 256 is $(9-2)(9-5)(10-6) = 744$.
- For fraction also, same rule works. For example, 10's complement of 25.639 is 74.361.

Example: Find 2's complement of $(10110)_2$.

Answer: Here,

$$N = 10110, n = 5 \text{ and } r = 2$$

$$r\text{'s complement of } N = 2\text{'s complement of } 10110$$

$$= r^n - N$$

$$= 2^5 - 10110_2$$

$$= 32 - 10110_2$$

$$= 100000_2 - 10110_2$$

$$= 01010_2$$

Note: We can easily find the 2's complement of a binary number by following procedure:

- Leave all the least significant zeros unchanged.
- Leave the first least significant non-zero digit.
- Replace all 0 by 1 and 1 by 0 in the higher significant digits.
- Eg: 2's complement of 10110 is 01010.

b. The $(r-1)$'s complement (Diminished radix complement):

Given a positive number N in base r with an integer part of n digits and a fraction part of m digits, the $(r-1)$'s complement of N is defined as $r^n - r^{-m} - N$.

Note: In case of a whole number, the $(r-1)$'s complement is defined as $r^n - 1 - N$.

Example: Find 9's complement of 256.

Ans: Here, $N = 256, n = 3, m = 0, r = 10$.

$$9\text{'s complement of } 256 = (r-1)\text{'s complement of } N$$

$$= r^n - r^{-m} - N$$

$$= 10^3 - 10^0 - 256$$

$$= 1000 - 1 - 256$$

$$= 743$$

Note: To find 9's complement of a decimal number, we can simply subtract each digit from 9. Whether it is whole number or fraction.

Try yourself:

- 9's complement of 52520 is 47479.
- 9's complement of 0.32670.6732

Example: Find 1's complement of 10110.

Ans: Here, $N = 10110$, $n=5$, $m=0$, $r=2$

$$\begin{aligned} \text{1's complement of } 10110 &= (r-1)\text{'s complement of } N \\ &= r^n - r^m - N \\ &= 2^5 - 2^0 - 10110 \\ &= 32 - 1 - 10110 \\ &= 31 - 10110 \\ &= 11111 - 10110 \\ &= 01001 \end{aligned}$$

Note: To find 1's complement of a binary number, we can simply replace 1's by 0's and 0's by 1's.

Try yourself:

- 1's complement of 101100 is 010011
- 1's complement of 0.0110 is 0.1001.

Note: If we add 1 to $(r-1)$'s complement of a number, we will get r 's complement.

Subtraction with r 's complement:

The general method of subtraction uses a borrow method when a larger digit is to be subtracted from smaller one. This seems to be hard to implement in digital components. Moreover, computers are more efficient on adding rather than subtracting. We can apply the r 's complement to perform subtraction in more efficient way.

Let, A and B be two positive numbers in base-r system. We may evaluate $A-B$ by following the procedure given below.

- Add the minuend A to the r 's complement of subtrahend B.
- Inspect the result obtained in above step for the end carry:
 - If an end carry occurs, discard the end carry and remaining value gives the desired result.

- If an end carry doesn't occur, take the r 's complement of the obtained value and place a negative sign to get the final result.

Proof: Let A and B be two positive numbers in base- r system. Let the numbers be represented in N digits. Then,

$$A + r^n - B \geq r^n \quad \text{if } A > B \quad \text{----- (a)}$$

$$A + r^n - B < r^n \quad \text{if } A < B \quad \text{----- (b)}$$

Our desired result is to obtain $A-B$.

In case (a), we can obtain positive $A-B$ simply by removing r^n which is equivalent to discarding end carry.

In case (b), the result should be $-(B-A)$, which is obtained by taking r 's complement once again and giving a minus sign in front as;

$$\text{Complement once again} \Rightarrow r^n - (A + r^n - B)$$

$$\text{Attach minus sign} \Rightarrow -[r^n - (A + r^n - B)] = -(B-A)$$

Example: Using 10's complement method, subtract 752-435.

Here, Minuend (A) = 752
 Subtrahend (B) = 435
 10's Complement of Subtrahend = 565

Now,

$$\begin{array}{r} 752 \\ + 565 \\ \hline \text{end carry} \leftarrow 1317 \\ \hline \therefore \text{Ans} = 317 \end{array}$$

Example: Using 10's complement method, subtract 245-275.

Here,

Minuend (A) = 245
 Subtrahend (B) = 275
 10's Complement of Subtrahend = 725

Now,

$$\begin{array}{r} 245 \\ + 725 \\ \hline \text{No end carry} \leftarrow 970 \\ \hline \end{array}$$

Answer = -10's Complement of 970
 = -030
 = -30 ✓

Example: Using 2's complement method, subtract $10110 - 10011$.

Here,

$$\begin{aligned}\text{Minuend (A)} &= 10110 \\ \text{Subtrahend (B)} &= 10011 \\ \text{2's Complement of B} &= 01101\end{aligned}$$

Now,

$$\begin{array}{r} 10110 \\ + 01101 \\ \hline \text{End Carry} \leftarrow 1100011 \end{array}$$

$\therefore \text{Ans} = 11$

Example: Subtract $101001 - 110110$ using 2's complement method.

Here,

$$\begin{aligned}\text{Minuend (A)} &= 101001 \\ \text{Subtrahend (B)} &= 110110 \\ \text{2's Complement of B} &= 001010\end{aligned}$$

Now,

$$\begin{array}{r} 101001 \\ + 001010 \\ \hline \text{No end carry} \leftarrow 110011 \end{array}$$

Ans = - 2's Complement of the result
 $= - 001101$
 $= - 1101$

Subtraction with (r-1)'s complement:

Let A and B be two positive numbers in base-r system. We may evaluate A-B as:

- Add the minuend A with (r-1)'s complement of subtrahend B.
- Inspect the result obtained in above step for end carry as:
 - If an end carry occurs, drop the end carry and add 1 to the least significant digit to get the final result.
 - If end carry doesn't occur, take (r-1)'s complement of the value to obtain final result.

Example: Subtract $536 - 345$ using 9's complement.

Here,

$$\text{Minuend (A)} = 536$$

$$\text{Subtrahend (B)} = 345$$

$$9\text{'s Complement of B} = 654$$

Now,

$$\begin{array}{r} 536 \\ + 654 \\ \hline \boxed{1}190 \end{array}$$

End carry ←

$$\begin{aligned} \therefore \text{Ans} &= 190 + 1 \\ &= 191 \end{aligned}$$

Example: Subtract 345-432 using 9's complement.

Here,

$$\text{Minuend (A)} = 345$$

$$\text{Subtrahend (B)} = 432$$

$$9\text{'s Complement of B} = 567$$

Now,

$$\begin{array}{r} 345 \\ + 567 \\ \hline 912 \end{array}$$

No end carry! ←

$$\text{Ans} = - 9\text{'s Complement of } 912$$

$$= - 087$$

$$= -87 \checkmark$$

Example: Subtract 10110-101 using 1's complement.

Here,

$$\text{Minuend (A)} = 10110$$

$$\text{Subtrahend (B)} = 101 = 00101$$

$$1\text{'s Complement of B} = 11010$$

Now,

$$\begin{array}{r} 10110 \\ + 11010 \\ \hline \boxed{1}10000 \end{array}$$

End carry ←

$$\text{Ans} = 10000 + 1$$

$$= 10001 \checkmark$$

Example: Subtract 1101-10110 using 1's complement method.

Here,

$$\begin{aligned} \text{Minuend (A)} &= 1101 = 01101 \\ \text{Subtrahend (B)} &= 10110 \\ \text{1's Complement of B} &= 01001 \end{aligned}$$

Now,

$$\begin{array}{r} 01101 \\ + 01001 \\ \hline 10110 \end{array}$$

No End Carry \leftarrow

$$\begin{aligned} \text{Ans} &= - \text{1's Complement of } 10110 \\ &= - 01001 \\ &= -1001 \end{aligned}$$

Note: The number of digits must match while doing any subtraction. If the minuend and subtrahend are not having same number of digits, we fix this by adding additional zeros in the most significant digit position (that means at the beginning of the number).

Comparison of 1's complement and 2's complement methods for subtraction:

2's complement method seems to be more appropriate than 1's complement method in computers due to the following reasons:

- In case of 1's complement method, if an end carry occurs, two addition operations are to be performed. Which leads to operational overhead.
- Zero in 2's complement has only one notation but in 1's complement, zero has two notations (+0 = 0000 and -0 = 1111).

Unsigned and Signed Binary Numbers:

Binary numbers can be represented in signed and unsigned way. Unsigned binary numbers do not have sign bit, whereas signed binary numbers uses signed bit as well or these can be distinguishable between positive and negative numbers.

1. **Unsigned Binary Numbers:** Unsigned numbers don't have any sign, these can contain only magnitude of the number. So, representation of unsigned binary numbers are all positive numbers only. For example, representation of positive decimal numbers are positive by default. We always assume that there is a positive sign symbol in front of every number.

Since there is no sign bit in this unsigned binary number, so N bit binary number represent its magnitude only. Zero (0) is also unsigned number. This representation has only one zero (0), which is always positive. Every number in unsigned number representation has only one unique binary equivalent form, so this is unambiguous representation technique.

The range of unsigned binary number is from 0 to $(2^n - 1)$.

Example: Represent 43_{10} in unsigned binary number.

Ans: Here, we simply convert 43_{10} into binary form. The unsigned representation considers only magnitude. Hence, $43_{10} = 101011_2$. All the bits are magnitude bits.

Example: Determine the range of 8-bits unsigned binary number. Also evaluate the minimum and maximum values in this range.

Ans: The range of unsigned binary numbers is $0 - (2^n - 1)$. Hence the range of 8-bits unsigned binary numbers is from 0 to $2^8 - 1 = 256_{10}$ which is 11111111_2 .

2. **Signed binary numbers:** Signed numbers contain sign flag. This representation distinguishes positive and negative numbers. This technique contains both sign bit and magnitude of a number.

There are three representations for the signed binary numbers.

- a. **Signed magnitude form:** For n bit binary number, 1 bit is reserved for sign symbol. If the value of sign bit is 0, then the given number will be positive, else if the value of sign bit is 1, then the given number will be negative. Remaining (n-1) bits represent magnitude of the number.

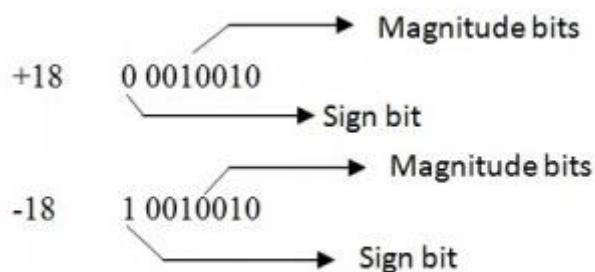
The range of n-bit number in this representation is from $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$.

The limitation of this representation is that, the number 0 has two representations.

$$+0 = 0\ 0000$$

$$-0 = 1\ 0000$$

Example:



Q: Represent 45_{10} and -62_{10} in signed magnitude form.

Ans:

Here, $45_{10} = 101101_2$

In sign magnitude form, $+45_{10}$ can be represented as;

0	101101
Sign	magnitude

Again, $62_{10} = 111110_2$

In sign magnitude form, -62_{10} can be represented as;

1	111110
Sign	magnitude

- b. 1's complement form:** we represent positive numbers in signed magnitude form and negative numbers in 1's complement form. There is extra bit for sign representation. If value of sign bit is 0, then number is positive and directly represents it in simple binary form, but if value of sign bit is 1, then number is negative and you have to take 1's complement to get the actual magnitude.

The range of n-bit number in this representation is from $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$.

1's complement method is also ambiguous since 0 has two representations.

$$+0 = 0\ 0000$$

$$-0 = 1\ 1111$$

Example: Represent 57_{10} and -38_{10} in 1's complement form.

Ans: Here, 57_{10} is a positive number so we simply represent it in signed magnitude form as;

0	111001
Sign	magnitude

Again, to represent -38_{10} , we use 1 as sign bit and 1's complement of 38 in remaining bits as;

1	011001
Sign	Magnitude in 1's complement

- c. 2's complement form:** we represent positive numbers in signed magnitude form and negative numbers in 2's complement form. There is extra bit for sign representation. If value of sign bit is 0, then number is positive and directly represents it in simple binary form, but if value of sign bit is 1, then number is negative and you have to take 2's complement to get the actual magnitude.

The range of n-bit number in this system is from $-(2^{n-1})$ to $+(2^{n-1}-1)$.

This method of representing negative numbers is most efficient for computers. Usually computers use 2's complement method for representing negative numbers.

Example: Represent 57_{10} and -38_{10} in 2's complement form.

Ans: Here, 57_{10} is a positive number so we simply represent it in signed magnitude form as;

0	111001
Sign	magnitude

Again, to represent -38_{10} , we use 1 as sign bit and 2's complement of 38 in remaining bits as;

1	011010
Sign	Magnitude in 1's complement

Decimal Codes

Electronic digital systems use signals that have two distinct values and circuit elements that have two stable states. There is a direct analogy among binary signals, binary circuit elements, and binary digits. A binary number of n digits, for example, may be represented by n binary circuit elements, each having an output signal equivalent to a 0 or a 1. Digital systems represent and manipulate not only binary numbers, but also many other discrete elements of information. Any discrete element of information distinct among a group of quantities can be represented by a binary code. Binary codes play an important role in digital computers. The codes must be in binary because computers can only hold 1's and 0's.

n -bit binary number can be used to code maximum of 2^n distinct items. For example, 3 bit binary numbers can be used to code $2^3 = 8$ distinct values. Binary

Binary codes for decimal digits require at least 4 bits. Different codes can be obtained by arranging four or more bits in ten distinct possible combinations. Some common binary codes for decimal digits are discussed below.

1. Binary Coded Decimal (BCD)

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.

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

BCD codes are also called 8421 code as the four bits from left to right have the positional weight 8, 4, 2 and 1 respectively.

While converting BCD code to equivalent decimal, we convert each group of 4-bits separately. For example,

BCD value of 45_{10} is $(0100\ 0101)_{BCD}$

and

$(100100100011)_{BCD} = 1001\ 0010\ 1011_{BCD} = 923_{10}$

Though the BCD system is easy for humans, it is quite complicated to be applied in digital electronics. Moreover, it needs a greater number of bits to represent a simple number. It needs complex architecture.

Note: If we take weight of the 4-bits as 2, 4, 2, 1 respectively from left to right, the corresponding code is called 2421 code.

2. Error Detection Codes

Binary information can be transmitted from one location to another by electric wires or other communication medium. Any external noise introduced into the physical communication medium may change some of the bits from 0 to 1 or vice versa.

The purpose of an error-detection code is to detect such bit-reversal errors. One of the most common ways to achieve error detection is by means of a parity bit. A parity bit is the extra bit included to make the total number of 1's in the resulting code word either even or odd. A message of 4-bits and a parity bit P are shown in the table below:

Message	0000	0101	1010	1011	1111
Even Parity	0				
Odd Parity	1				

In odd parity, 1 represents even number of 1's in the message. In even parity, 0 represents an even number of 1's in the message.

During the message transmission, a parity bit along with the message is sent to the destination. If the parity of the message at sending and receiving end doesn't match, at least one bit in the message has changed during transmission.

This method has following limitations:

- Error detection becomes erroneous if the parity bit itself gets affected during transmission.
- It can only detect the change in odd number of bits during transmission. If even number of bits get affected, parity is not affected.

3. Reflected Code (Also called Gray Code or Unit Distance 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. It is also called Gray code named after Frank Gray. 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. 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	Gray
0.	0000	0 000
1.	0001	0 001
2.	0010	0 011
3.	0011	0 010
4.	0100	0 110
5.	0101	0 111
6.	0110	0 101
7.	0111	0 100
MIRROR (last 3 bits of 7 & 8 are same, similarly for 6&9, 5&10 till 0&15)		
8.	1000	1 100
9.	1001	1 101
10.	1010	1 111
11.	1011	1 110
12.	1100	1 010
13.	1101	1 011
14.	1110	1 001
15.	1111	1 000

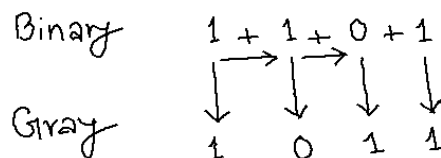
Gray code is also called reflected binary code due to mirror values as shown in above figure. The Gray code is used in applications where the normal sequence of binary numbers may produce an error or ambiguity during the transition from one number to the next.

Binary to Gray conversion:

Starting from left to right:

- If it is MSB, place it as it is in gray code.
- Otherwise get the gray code bit by adding current bit with previous bit and ignore any carry.
- Repeat step 2 till end.

Example: Convert 1101_2 to Gray code.

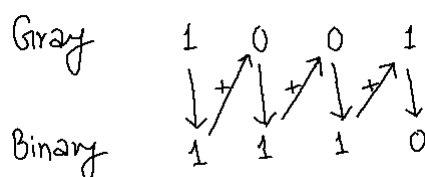


Gray to Binary Conversion:

Starting from left to left:

- If it is MSB, place it as it is in Binary code.
- Otherwise get the Binary code bit by adding current Binary bit with previous Gray code bit and ignore any carry.
- Repeat step 2 till end.

Example: Convert the Gray code 1001 to Binary code.



4. Excess 3 (XS3) Codes:

XS3 Code is the unweighted binary code for decimal numbers. It is an extension to the BCD code. It is the self-complementary binary coded decimal code and numeral system. Excess-3 code was used on some older computers as well as in cash registers and hand-held portable electronic calculators of the 1970s, among other uses.

We obtain XS3 code by adding 3_{10} or 0011_2 to each digit in BCD code.

Decimal	BCD 8421	Excess-3 BCD+8421(0110)
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Excess-3 code is self-complementary code; that means, 1's complement of XS3 code also points to 9's complement of corresponding decimal number.

Example: Find excess-3 code of 45_{10} .

Ans:

Decimal	BCD	XS3 = BCD+3 (each digit)
45	0100 0101	$0100+0011 \quad 0101+0011 = 0111 \quad 1000$

5. 2 4 2 1 Code: The Aiken code (also known as 2421 code) is a complementary binary-coded decimal (BCD) code. A group of four bits is assigned to the decimal digits from 0 to 9 according to the following table. The code was developed by Howard Hathaway Aiken and is still used today in digital clocks, pocket calculators and similar devices.

The Aiken code differs from the standard 8421 BCD code in that the Aiken code does not weight the fourth digit as 8 as with the standard BCD code but with 2. It is a four bit code having positional value 2, 4, 2 and 1 from MSB to LSB.

Decimal	0	1	2	3	4	5	6	7	8	9
2421	0000	0001	0010	0011	0100	1011	1100	1101	1110	1111

The code is self-complementary. 1's complement of 0 is 9 and that of 1 is 8 and so on.

6. Alphanumeric Codes:

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.

Two such codes are ASCII code and EBCDIC codes.

- a. **ASCII Code:** ASCII stands for "American Standard for Information Interchange". It is a globally accepted character encoding standard. ASCII is a 7-bit character set containing 128 characters. It contains the numbers from 0-9, the upper and lower case English letters from A to Z, and some special characters. The character sets used in modern computers, in HTML, and on the Internet, are all based on ASCII.

ASCII codes for the common characters are:

For decimal digits 0 to 9	48 – 57
For lowercase letters a to z	65 – 90
For uppercase letters A to Z	97 - 122

The Extended Binary Coded Decimal (EBCDIC) was used earlier to ASCII as character encoding system specially for IBM mainframe computers. It was the extended form of BCD code to encode alphabets and other characters as well. Though it was used By IBM in the past, todays modern standard for character encoding is either ASCII itself or it is based on ASCII.

Binary Storage and Registers:

Binary Storage:

Binary storage is made up of binary cells. A binary cell is a device that possess two stable states and is capable of storing one bit of information (0 or 1). The excitation signals trigger the cell to generate the output signal representing one of the two states of the binary cell. The output signal represents 0 or 1 based on the interpretation of the two stable states of the output signal. Following diagram represents a typical binary storage system.



Fig: Binary storage System

For example, in a CD, the binary cell is represented by the surface element with two stable states pits or lands. Excitation signal is the laser beam and the output signal is the light detected by the sensor after reflection from pits or lands. Similarly we can consider a punched card as binary cell as well. In a digital computer, a single binary cell is well represented by a flip flop.

Register:

A register is a group of binary cells. A n bit register contains n different binary cells (flipflops) each storing a single bit. It can represent 2^n different states but only one at a time.

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Fig: A 8-bit register

The content of the register is a string of binary digits. The interpretation of the content gives the complete information about the content. For example, above register contains 01011011, which may represent different values depending on the interpretation. If it is storing unsigned binary number, the content is +51. If it is a ASCII character, the content is the character 3 and so on.

Register Transfer:

A digital computer is characterized by its registers. The memory unit is the collection of thousands of registers for storing digital information. Processor unit also contains various registers to hold the operands on which operation are performed. The control unit also uses registers to store probably the commands. Input and output devices are also associated with I/O registers to store the information temporarily to transfer information to or from the device.

Register transfer operation is the operation that involves information transfer from one register to another. Following figure illustrates register transfer operation to illustrate the input from keyboard to memory unit.

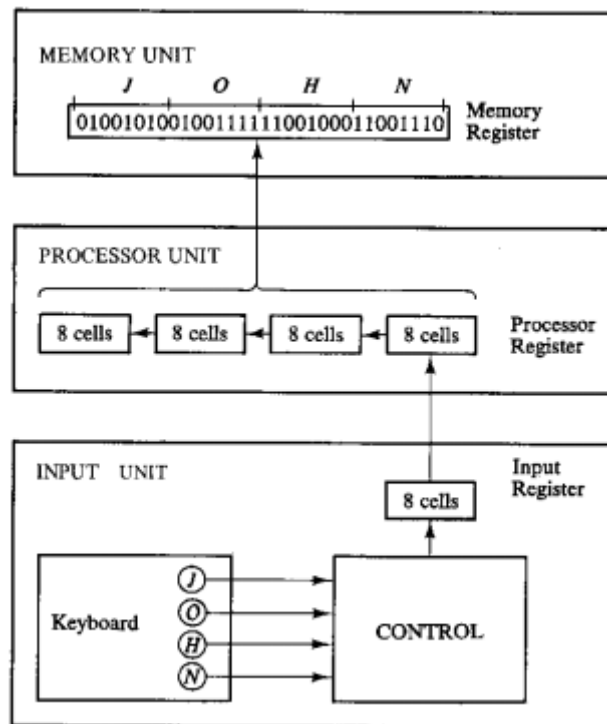


Fig: Transfer of information with register

Four characters “J”, “O”, “H”, “N” are entered from keyboard and are transferred from input register to processor register (register transfer). Capacity of processor register is four times the word length. The content of processor register is ultimately transferred to the memory register associated with memory unit.

For nay binary information processing, we need two fundamental components.

- Devices that can hold the data to be processed.
- Circuit that can manipulate individual bit of information.

The device commonly used for holding data is register and the bits manipulation circuit includes logic circuits. Following diagram shows the process of adding two 10-bit binary numbers.

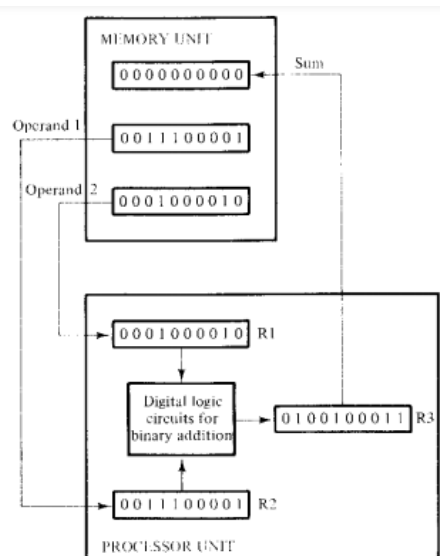


Fig: Example of binary information processing

The operands are fetched from memory unit to the registers R1 and R2 in the processor unit. the digital logic circuit performs binary addition and stores the result in register R3 which is then transferred to the memory unit. This process clearly illustrates the use of two components (registers and circuitry) of binary information processing system.

Binary Logic:

Binary logic deals with manipulation of binary information and generate logical output. Binary logic uses manipulation of binary variables. The binary variable may have any one of two possible states; (true or false, on or off, 1 or 0). For our convenient, we assume 0 or 1 as two possible states of a binary variable (also called as Boolean variable). Binary logic describes manipulation and processing of binary information. Binary logic helps to describe the working of a digital circuits. Binary logic can be explained by using Boolean algebra.

In binary logic, there are three basic logic operations: AND, OR and NOT.

1. **AND Operation:** This operation is represented by a dot operator or by the absence of any operator. It is a binary operator. The result of expression xy is 1 if and only if both $x=1$ and $y=1$. In other cases, 0. Following truth table shows various possible results of AND logic.

AND		
x	y	$Z=xy$
0	0	0
0	1	0
1	0	0
1	1	1

Fig: Truth table of AND logic

The AND logic may be demonstrated by the following switching circuit.

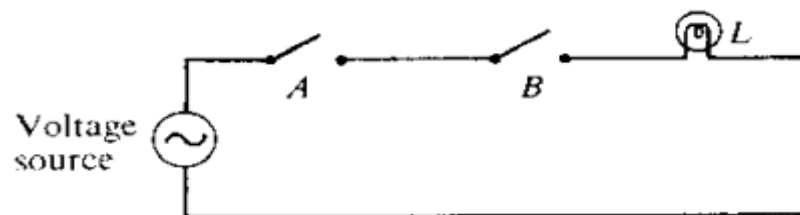


Fig: Switches in series- logic AND

2. **OR Operation:** This operation is represented by + operator. It is a binary operator. The result of expression $x+y$ is 1 if any one of x or y is 1, otherwise 0. Following truth table shows various possible results of OR logic.

OR		
x	y	$Z=x+y$
0	0	0
0	1	1
1	0	1
1	1	1

Fig: Truth table of OR logic

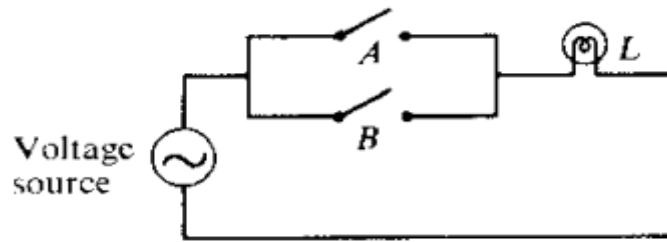


Fig: Switches in Parallel- Logic AND

- 3. NOT Operation:** This operation is represented by a prime or a bar symbol above the Boolean variable. It is unary operator. The result of expression x' is 0 if $x = 1$ and $x' = 1$ if $x = 0$. That means, the result of x' is opposite of x . following truth table shows the various possible results of NOT logic.

NOT	
x	x'
0	1
1	0

Fig: Truth table of NOT logic

Note: One should not be confused with binary arithmetic and binary logic. You can understand the difference by the fact that, In binary arithmetic, $1+1 = 10$ (read as 1 plus 1 is 2, but in binary logic, $1+1 = 1$ (read as 1 OR 1 is 1).

Logic Gates:

Electronic digital circuits are also called logic circuits because, with the proper input, they establish logical manipulation paths. Any particular path in the circuit can pass binary information (a single bit of information) with respect to presence or absence of the signal. The digital electronic circuits for AND, OR and NOT logic are called logic gates. Logic gates are the hardware blocks that produce a logic-1 or logic-0 output based on the input logic information. The NOT gate is often called as inverter due to its behavior.

Following are the symbols used for the fundamental logic gates.

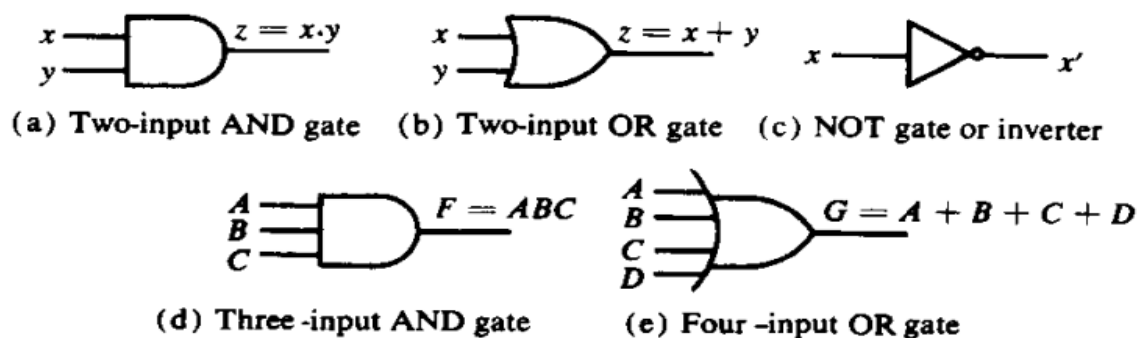


Fig: Fundamental Logic gates

From the above figure, it is clear that, the AND and OR logic gates have two or more inputs and the NOT gate has only one input. All logic gates have only one output.