



“Binary Number System”

Dr. Cahit Karakuş, February-2019



Data

Data – Information

- Data: Facts or pieces of information that have not gained meaning, have not been associated, have not been assimilated, and have not been processed. They are in forms devoid of any content. They carry no comments but are available for processing. Raw data.
- The computer takes data as input to process it and stores it in memories; processed and produced as output. The produced output is recorded in memory, displayed on the screen or transferred to other systems via I/O devices.
- Information: Giving meaning to data by adding value to it. It is the result of bringing together and arranging interrelated data for a certain purpose and has a meaning and news quality. From an institutional perspective, information is a database with meaning. Data is transformed into information by adding value through content processes. Information arises from data and information turns into knowledge.
- Information: In general, it emerges through the interpretation of data and information.
- Big Data: The concept of "Big Data" has emerged as data has increased greatly in terms of speed, diversity and capacity (volume) and technology has supported this increase and produced new solutions.

Data Science

- Veri Bilimi, bilgiler hakkında sonuçlar çıkarmak amacıyla istatistik ve makine öğrenimi tekniklerini kullanarak ham verileri analiz etme bilimidir.
- Data Science is the science of analyzing raw data using statistics and machine learning techniques to draw conclusions about the information.

Data Science is the science of analyzing raw data.

- Data gathering; Sensors, Telemetry systems, Internet, Data base management
- Data visualization
- Finding statistical measures by evaluating data
- Creating mathematical models and algorithms

Data – Information – Knowledge - Wisdom

- Symbols (Signals, Pictures, Shapes, ...): Numbers and symbols are pieces of information such as numbers, words, images, video and sound that are transferred to the computer's memory during the input phase.
- Data: Facts or pieces of information that have not gained meaning, have not been associated, have not been assimilated, and have not been processed. They are in forms devoid of any content. Sometimes it is a physical event, uninterpreted observations. They carry no comments but are available for processing. They are not effective in decision making.
- Information: What, who, when and where questions need to be answered. Information is processed, organized and meaningful data. Information is organized, meaningful and useful data. During the output phase, the information created is put into presentation form with printed reports, graphics and visuals. The information is stored on the computer for future use.
- Knowledge: Increasing performance in decision-making, estimation and searching for the truth.
- Understand: To become conscious by understanding, comprehending and feeling.
- Wisdom: It is evaluated understanding. It is about making decisions and interpreting by questioning and guessing.

Data Types

Data comes in different sizes and also types.

Signals: Sound, heat, electricity, electromagnetic, smell, vibration, picture

texts

numbers

Clickstreams

Graphics

Tables

images

Transactions

Videos

Data Types

1. Which of the following is not a data type?

- a) Symbolic Data
- b) Alphanumeric Data
- c) Numerical Data
- d) Alphabetical Data

• Answer: a

- Description: Data types are of three basic types: Numeric, Alphabetic, and Alphanumeric. Numeric Data consists of numbers only.
- Alphabetic Data consists of only letters and a space character, while alphanumeric data consists of symbols.

Symbols

- A symbol is a mark, sign, or word that indicates, signifies, or is understood as representing an idea, object, or relationship.
- Symbols allow people to go beyond what is known or seen by creating linkages between otherwise very different concepts and experiences.
- All communication (and data processing) is achieved through the use of symbols.
- Symbols take the form of words, sounds, gestures, ideas, or visual images and are used to convey other ideas and beliefs.
- For example, a red octagon is a common symbol for "STOP"; on maps, blue lines often represent rivers; and a red rose often symbolizes love and compassion.
- Numerals are symbols for numbers; letters of an alphabet may be symbols for certain phonemes; and personal names are symbols representing individuals.

Different types of data

- Types of data
 - Text: ASCII code, unicode
 - Numbers: Integers, Reals, Floating point numbers, Fractional number, ...
- Text
- Numbers
 - Binary number representation of integers
 - If we save one bit to signify positive (+) or negative (-), then an n-bit binary word can represent integers in the range: $-2^{(n-1)} - 1 .. +2^{(n-1)}$
- Audio
- Image and graphics
- Video

ASCII code (American Standard Committee on Information Interchange)

- A unique 8-bit binary code for each character:
 - A-Z, a-z, 1-9, ., - ! " £ \$ % ^ & * () _ +
 - Special unprintable characters such as the
 - ENTER key (CR for carriage return)

Types of data

- Data Types are used to define the type of a variable. It defines what type of data we are going to store in a variable. The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters.
- Numeric - int, float, complex
- String - str
- Sequence - list, tuple, range
- Binary - bytes, bytearray, memoryview
- Mapping - dict
- Boolean - bool
- Set - set, frozenset
- None - NoneType

Data Quality

- Data quality issues:
 - Noise
 - Outliers (Aykırı değerler)
 - Wrong data
 - Fake data
 - Missing Values
 - Duplicate Data (Yinelenen veriler)
 - Similarity and Dissimilarity Measures (Benzerlik ve Farklılık Ölçüleri)

Data Structures

- Lists
- Arrays
- Queue
- Trees
- Stacks
- Storage – Hash structure

Algebraic Expressions

Common Powers

Prefix	Symbol	Power of 10	Power of 2	Prefix	Symbol	Power of 10
Kilo	K	1 thousand = 10^3	$2^{10} = 1024$	Milli	m	1 thousandth = 10^{-3}
Mega	M	1 million = 10^6	2^{20}	Micro	μ	1 millionth = 10^{-6}
Giga	G	1 billion = 10^9	2^{30}	Nano	n	1 billionth = 10^{-9}
Tera	T	1 trillion = 10^{12}	2^{40}	Pico	p	1 trillionth = 10^{-12}
Peta	P	1 quadrillion = 10^{15}	2^{50}	Femto	f	1 quadrillionth = 10^{-15}
Exa	E	1 quintillion = 10^{18}	2^{60}	Atto	a	1 quintillionth = 10^{-18}
Zetta	Z	1 sextillion = 10^{21}	2^{70}	Zepto	z	1 sextillionth = 10^{-21}
Yotta	Y	1 septillion = 10^{24}	2^{80}	Yocto	y	1 septillionth = 10^{-24}

2^n : Byte, Bellek kapasitesi

10^n : Bit, veri işleme hızı veya veri transfer hızı

Laws of Exponents

- Let a and b be positive numbers and let x and y be real numbers. Then,

1. $b^x \cdot b^y = b^{x+y}$

2. $\frac{b^x}{b^y} = b^{x-y}$

3. $(b^x)^y = b^{xy}$

4. $(ab)^x = a^x b^x$

5. $\left(\frac{a}{b}\right)^x = \frac{a^x}{b^x}$

What is the difference between a numeric expression and an algebraic or variable expression?

Numeric Expression

$$-3 + 2 + 4 - 5$$

Algebraic Expression

$$-3x + 2y - 4z - 5$$

An algebraic or variable expression consists of three parts

- Variable
- Coefficient
- Constant

In the expression $12 + B$, the letter “B” is a variable.

The **variable** is a symbol or letter that represents a number.

What are the variables in the expression below?

$$f(x,y,z) = -4x + 3y - 8z + 9$$

x, y, z : Independent variable

$f(x, y, z)$: Dependent variable

What are coefficients?

A **coefficient** is the number multiplied by the variable in an algebraic expression.

Algebraic Expression

Coefficient

$$6m + 5$$

6

$$8r + 7m + 4$$

8, 7

$$14b - 8$$

14

The coefficient is the number that multiplies the variable. More plainly, it is the number in front of a variable.

What are the coefficients in the expression below?

$$f(x,y,z) = -4x + 3y - 8z + 9$$

$$f(x,y,z) = ax + by + cz + d$$

What is a term?

A **term** is the name given to a number, a variable, or a number and a variable combined by multiplication or division.

Algebraic Expressions

$$a + 2$$

$$3m + 6n - 6$$

Terms

$$a, 2$$

$$3m, 6n, - 6$$

The constant is any term that does not have a variable.

What is the constant in the expression below?

$$f(x,y) = -8 + 5y + 3x$$

What are constants?

- A constant is a number that cannot change its value.

In the expression: $5x + 7y - 2$

the constant is -2 .

An algebraic expression can also be classified as a polynomial which is an expression that consists of more than one term. Two types of polynomials are **binomials and trinomials**.

A binomial has two terms. Ex: $3x + 4$

A trinomial has three terms. Ex: $4a^2 + 3a + 7$

Evaluating expressions examples

Example 1

Evaluate $4y - 15$ for $y = -9$

- $4y - 15$
- $4(-9) - 15$
- $-36 - 15 = -\mathbf{51}$

Example 2

Evaluate $3ab + c^2$ for $a = 2$, $b = -10$, and $c = 5$.

- $3ab + c^2$
- $3(2)(-10) + 5^2$
- $-60 + 25 = -\mathbf{45}$

Example 3

Evaluate $6(g + h)$ for $g = 8$ and $h = 7$.

- $6(8 + 7)$
- $6(15) = \mathbf{90}$



Numbers

Number Systems

- **The decimal, binary, octal, and hexadecimal numbering systems and be able to convert from one numbering or coding system to another**
- **The terms bit, byte, word, least significant bit (LSB), and most significant bit (MSB) as they apply to binary memory locations**
- **Add, subtract, multiply, and divide binary number**

Key Vocabulary

Expression – a math sentence *without* equal signs

Equation – a math sentence *with* equal signs

Variable – a letter that is used to represent a number

Terms – the part of an expression that is being added together

Coefficient – the number part of the term that has a variable with it

Constant Term – a term that has a number but no variable

Like Terms – terms that have identical variables

Inverse Operation – the opposite operation used to solve an equation

Expressions vs Equations

Expressions – written as phrases

Example – the sum of six and a number

$$x + 6$$

Equations – written as sentences

Example – the sum of six and a number ***IS*** ten

$$x + 6 = 10$$

- Addition:
 - Add
 - Plus
 - Sum
 - Total
 - Increased by
 - More than
- Subtraction:
 - Minus
 - Difference
 - Subtract
 - Less than
 - Decreased by
 - less
- Multiplication:
 - Product
 - Times
 - multiply
- Division:
 - Quotient
 - divide

Add	Subtract	Multiply	Divide
Plus	Minus	Times	Divided by
The sum of	The difference of	The product of	The quotient of
Increased by	Decreased by	of	

Translating Verbal Phrases

- A. A number increased by 5
- B. 7 less than a number
- C. 3 more than twice a number
- D. 4 decreased by the quotient of a number and 7
- E. 16 increased by a number is 27
- F. The difference of twice a number and 3 equals -4
- G. The product of $\frac{1}{2}$ and a number is 36
- H. -3 is equal to twice the sum of a number and 2

Number Systems

The Number System

When learning about 'data representation' we understand the importance and versatility of binary numbers (İkili sayıların önemi ve çok yönlülüğü) and how binary is used to store data in computer systems.

- Numbers
- Text
- Images
- Sound
- Instructions
- ...are all stored in binary and we will understand how this is done!

It is fairly easy to understand how numbers are stored in binary format in computer systems. But what about text, images and sound? The key thing to remember is that text, images and sound

Numbers

Natural Numbers: Zero and any number obtained by repeatedly adding one to it.

Examples: 100, 0, 45645, 32

Negative Numbers: A value less than 0, with a – sign

Examples: -24, -1, -45645, -32

Integers: A natural number, a negative number, zero

Examples: 249, 0, - 45645, - 32

Rational Numbers: An integer or the quotient of two integers

Examples: -249, -1, 0, $\frac{3}{7}$, $-\frac{2}{5}$

Representing Numbers

- Positive whole numbers
 - We already know one way to represent them: i.e., just use base 2 number system
- All integers, i.e., including negative integers
 - Set aside a bit for storing the sign
 - 1 for +, 0 for –
- Decimal numbers, e.g., 3.1415936, 100.34
 - Floating point representation:
 - $\text{sign} * \text{mantissa} * 2^{\text{exp}}$
 - 64 bits: one for sign, some for mantissa, some for exp.

Decimal System

Knowledge of different number systems and digital codes is quite useful when working with almost any type of **digital computer**. This is true because a basic requirement of these devices is to **represent, store, and operate on numbers**. In general, computer work on binary numbers in one form or another; these are used to represent various codes or quantities.

The **decimal system**, which is most common to us, has a **base of 10**. The radix or base of a number system determines the total number of different symbols or digits used by that system. For instance, in the decimal system, 10 unique numbers or digits—i.e., **the digits 0 through 9**—are used: the total number of symbols is the same as the base, and the symbol with the largest value is 1 less than the base.

The value of a decimal number depends on the digits that make up the number and the place value of each digit. A place (**weight**) value is assigned to each position that a digit would hold from right to left. In the decimal system the first position, starting from the **rightmost position, is 0**; the **second is 1**; the **third is 2**; and **so on up to the last position**. The weighted value of each position can be expressed as the base (10 in this case) raised to the power of the position. For the decimal system then, the position weights are 1, 10, 100, 1000, and so on. **Figure 6.1 illustrates how the value of a decimal number can be calculated by multiplying each digit by the weight of its position and summing the results.**

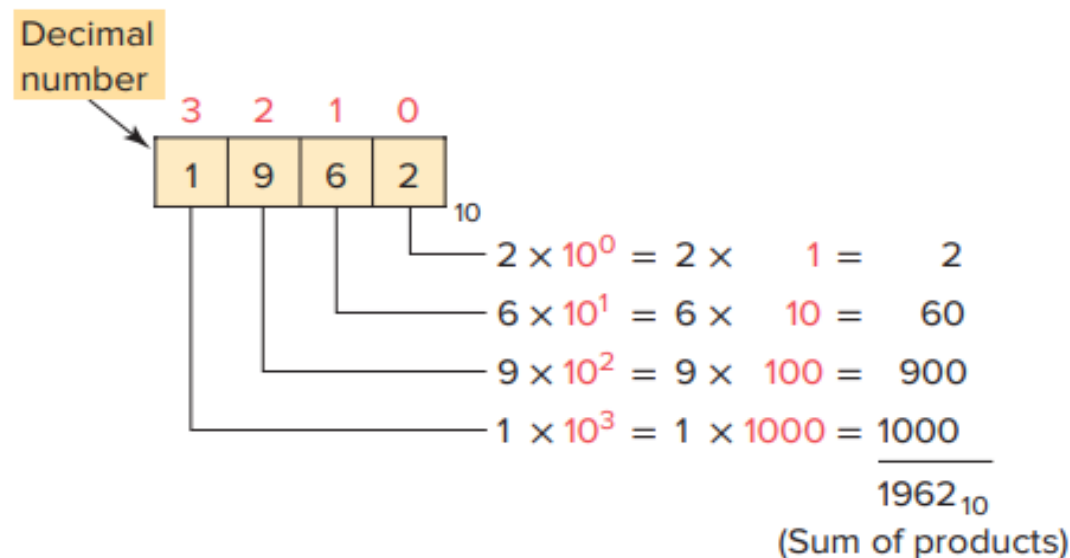


Figure 6.1 Weighted value in the decimal system.

Natural Numbers

How many ones are there in 642?

$$600 + 40 + 2 ?$$

Or is it

$$384 + 32 + 2 ?$$

Or maybe...

$$1536 + 64 + 2 ?$$

Natural Numbers

642 is $600 + 40 + 2$ in BASE 10

The **base** of a number determines the number of digits and the value of digit positions

Positional Notation

Continuing with our example...

642 in base 10 *positional notation* is:

$$\begin{aligned} 6 \times 10^2 &= 6 \times 100 = 600 \\ + 4 \times 10^1 &= 4 \times 10 = 40 \\ + 2 \times 10^0 &= 2 \times 1 = 2 \end{aligned} = 642 \text{ in base 10}$$

This number is in
base 10

The power indicates
the position of
the number

Positional Notation

Example:

$$642 \text{ is } 6 * 10^2 + 4 * 10 + 2$$

R is the base
of the number

As a formula:

$$d_n * R^{n-1} + d_{n-1} * R^{n-2} + \dots + d_2 * R + d_1$$

n is the number of
digits in the number

d is the digit in the
 i^{th} position
in the number

Number-Base Conversion

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

- The six letters A, B, C, D, E, and F in hexadecimal represent: 10, 11, 12, 13, 14, and 15, respectively.

- Number systems
 - Decimal
 - Binary (iki sayı sistemi 0/1)
 - Bits: 0 ya da 1
 - Bytes: 8bit
 - Hexadecimal: 4bit
 - Numbers conversion among different systems
- The on and off states of the capacitors in RAM can be thought of as the values 1 and 0, respectively.
- Therefore, thinking about how information is stored in RAM requires knowledge of the **binary (base 2) number system**.

Numbering System

- The decimal number system consists of symbols 0 through 9. It is multiplied by 10 to create a two-digit decimal number system. It is multiplied by 100 to create a three-digit number system. The steps are continued in multiples of 10.
- The binary number system consists of 0 and 1. It is expressed by the 0 or 1 output of the transistor. It exists as an electrical signal. Digits are expressed as multiples of 2.
- Example: $(109)_{10} = 1 * 10^2 + 0 * 10^1 + 9 * 10^0$
- Example: $(101)_2 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = (5)_{10}$
- In the computer, everything is kept as 1 or 0.
- $(100\ 0001)_b$ is the symbol, which corresponds to the value 65 in the decimal number system, or the character A.
- In the binary world, everything can be expressed as 1 or 0. Its equivalent in computer memory is called a bit.
- Bits come together to identify more bits. For example, in a 2-bit system, four different states or symbols 00, 01, 10, 10 can be created. When it is desired to be displayed numerically, it can be expressed as 0, 1, 2, 3.

Binary

Decimal is base 10 and has 10 digits:

0,1,2,3,4,5,6,7,8,9

Binary is base 2 and has 2 digits:

0,1

For a number to exist in a given base, it can only contain the digits in that base, which range from 0 up to (but not including) the base.

What bases can these numbers be in? 122, 198, 178

The Decimal Number System

- The decimal number system is also known as **base 10**. The values of the positions are calculated by taking 10 to some power.
- Why is the base 10 for decimal numbers?
 - Because we use 10 digits, the digits 0 through 9.
- The decimal number system is a positional number system.
- Example:

$$\begin{array}{cccc} 5 & 6 & 2 & 1 \\ 10^3 & 10^2 & 10^1 & 10^0 \end{array}$$

$$1 \times 10^0 = 1$$

$$2 \times 10^1 = 20$$

$$6 \times 10^2 = 600$$

$$5 \times 10^3 = 5000$$

Decimal Numbering System

- Ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Represent larger numbers as a sequence of digits
- Each digit is one of the available symbols
- Example: 7061 in decimal (base 10)
- $(7061)_{10} = (7 \times 10^3) + (0 \times 10^2) + (6 \times 10^1) + (1 \times 10^0)$

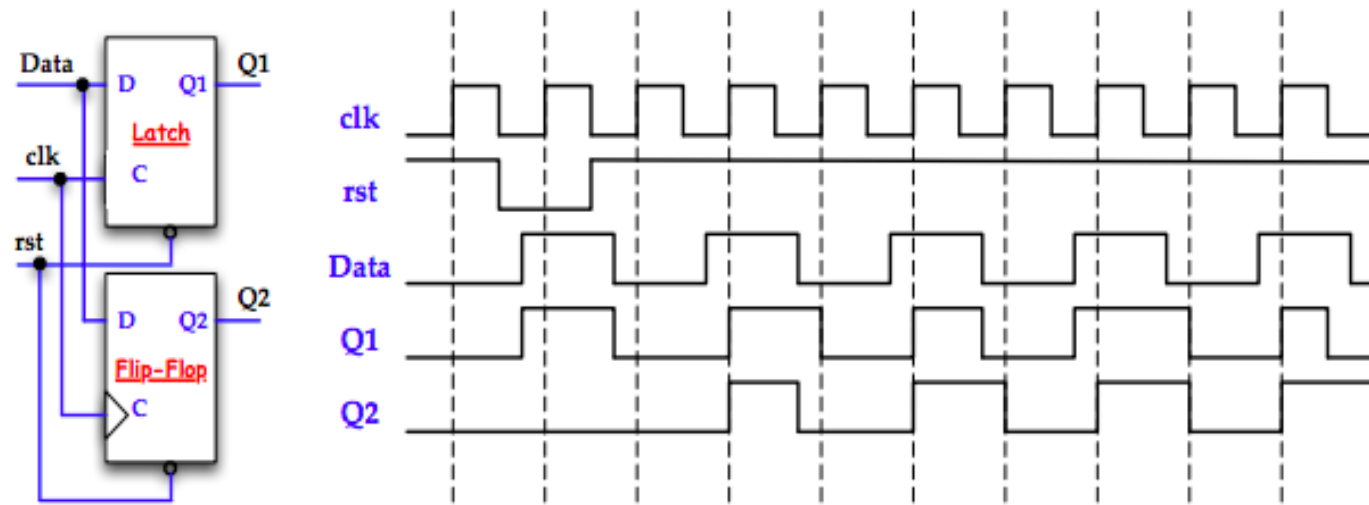
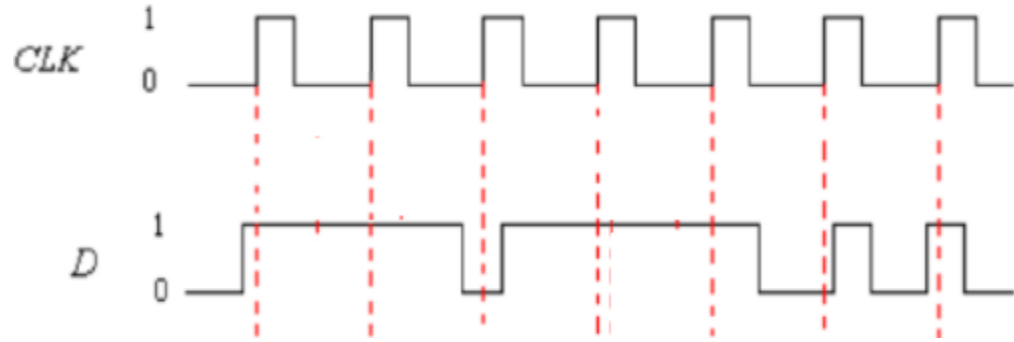
Major Computer Components

Major Computer Components

- Central Processing Unit (CPU)
- Bus: Address, Data, Control
- Main Memory: RAM (W/R), ROM(R), CMOS(R-Battery)
- Secondary Storage Media: HD
- I / O Devices
- Timing & Clock

Data and Clock

- Data and Clock signal work together all computer systems.



The CPU

- **Central Processing Unit**
- The “brain” of the computer
- Controls all other computer functions
- In **PCs (personal computers)** also called the **microprocessor** or simply **processor**.

The Bus

- Computer components are connected by a **bus**.
- A bus is a group of parallel wires that carry address, **control** and **data** signals between CPU and components.

Main Memory

- Main memory holds information such as computer programs, numeric data, or documents created by a **word processor**.
- Main memory is made up of transistors and **capacitors**.
- If a capacitor is charged, then its state is said to be **1**, or **ON**.
- We could also say the **bit is set**.
- If a capacitor does not have a charge, then its state is said to be **0**, or **OFF**.
- We could also say that **the bit is reset** or **cleared**.
- Memory is divided into cells, where each cell contains 8 bits (a 1 or a 0). Eight bits is called a byte.
- Each of these cells is uniquely numbered. The number associated with a cell is known as its address. Memory capacity= 2^n byte.
- Main memory, Ram is volatile storage. That is, if power is lost, the information in main memory is lost. Ram is R/W memory.
- Main memory, Rom is not volatile storage. That is, if power is lost, the information in main memory is not lost. Rom is only read memory.

Main Memory (con't)

- Other computer components can
 - get the information held at a particular address in memory, known as a **READ**,
 - or store information at a particular address in memory, known as a **WRITE**.
- Writing to a memory location alters its contents.
- Reading from a memory location does not alter its contents.
- All addresses in memory can be accessed in the same amount of time.
- We do not have to start at address 0 and read everything until we get to the address we really want (sequential access).
- We can go directly to the address we want and access the data (direct or random access).
- That is why we call main memory RAM (Random Access Memory).

Memory: Binary Storage and Registers

■ Registers

- ♣ The binary information in a digital computer must have a physical existence in some medium for storing individual bits.
- ♣ A *binary cell* is a device that possesses two stable states and is capable of storing one bit of information (0 or 1).
- ♣ A *register* is a group of binary cells.
- ♣ A register with n cells can store any discrete quantity of information that contains n bits.

n cells  2^n possible states

- ♣ The content of a register (the information stored in it) is interpreted differently depending on the application of CPU.

Secondary Storage Media

- Disks -- floppy, hard, USB memory (Flash), removable (random access)
- Tapes (sequential access)
- CDs (random access)
- DVDs (random access)
- **Secondary storage media** store **files** that contain
 - computer programs
 - data
 - other types of information
- This type of storage is called **persistent (permanent) storage** because it is **non-volatile**.

I/O (Input/Output) Devices

- Information input and output is handled by **I/O (input/output) devices**.
- More generally, these devices are known as **peripheral devices**.
- Examples:
 - monitor
 - keyboard
 - mouse
 - disk drive (floppy, hard, removable)
 - CD or DVD drive
 - printer
 - scanner



Bit (0/1)

Data Representation in Computer

- In all computers, all information is represented by using binary values.
- Each storage location (cell - Transistor): has two states
 - low-voltage electrical signal => 0
 - High-voltage electrical signal => 1
 - it can store a binary digit, **bit**
- Eight bits grouped together to form a **byte**
- Two bytes grouped together to form a **word**
 - Word length of a computer, e.g., 32 bits computer, 64 bits computer, 128 bits.
Data bus length is 32 or 64 or 128

Basic Units of Digital systems

- The basic unit of information in computer systems is bit: (1/0)
- Bit: 0/1; bits are represented by electrical signals. Electronic circuit element Transistor. Data Bus, Memory compartment contents, Registers
- Byte: Represents 8 bits of data. Or it points to the 1 byte memory slot. It represents base-2 operations in memory. Address Bus; memory and memory choose the eye.
- Bit/sec: Represents the amount of data to be transferred or processed per second. It is represented by exponential operations in base 10.
- Qubit: Represents the smallest data in quantum calculations.
- Electron: Qubits are represented by electrons.

Binary Signals- Bit (0/1)

Binary signal, binary state signal: Two-state data (0/1).

- off & on

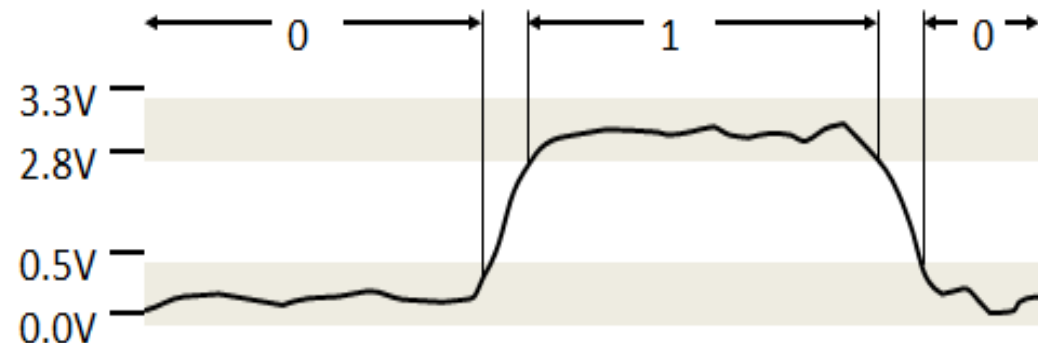
- It is carried and stored by electrical signals.

low voltage & high voltage; 0v & 5v

Bit: It is not only a mathematical concept, but also has its counterpart in the physical world.

- The binary number system has the value 0 or 1 and nothing else.

- A bit is the smallest unit of information in a computer



Binary Number System

- Computers use the binary number system, which consists of only the digits 0 and 1.
- 1 represents presence, 0 represents absence. 1 means there is voltage (5Volts), 0 means there is no voltage.
- All numbers, text, picture, video, symbols in the binary system consist of 1 and 0.
- In computer systems we focus on the binary number system (base 2), Why? Because the digital components that make up the computer can be in one of two states: on or off.
- The bit name binary digit is derived from the binary number system.
- Numbers (positive, negative, integer, float, fraction), strings, booleans, images, sounds, programming instructions represent as bits.
- While the numbers and letters typed on the keyboard are translated into a form that the computer can understand, the binary number system comes into play.
- For unsigned integer values, we can store them directly using binary.
- The numbers we use in our daily lives are in base ten, also known as decimal. As we know, the numbers that make up the decimal system are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Binary Number System

- For example, if an m -bit system is to be created, 2^m states can be obtained.
- When 8 bits come together, 1 byte is obtained. It represents $2^8=256$ different symbols.
- It can be represented with the number system from 0 to 255, it can display negative and positive numbers up to -128 ... 0 ...127; or we can assign characters.
- More characters or number systems can be obtained by placing bytes side by side. For example, 32 bits are obtained by placing 4 bytes sideways.

Bit

- Bit is the basic concept, the cornerstone of information used in computer systems. Bits consist of 0s or 1s. They are mathematical objects, and they correspond to physical situations.
- A bit (0/1) is the basic unit of information. Bits are used to represent information in computers. Regardless of its physical implementation, a bit is always understood to be either a 0 or a 1. An analogy to this is a light switch with 0 representing the off position and 1 representing the on position.
- In normal or "classical" computing, the basic unit of information is the bit. A bit can be in one of two states, 0 or 1.
- The building block of classical computing devices is represented by the two-state system (0/1).
- In fact, any system that has a finite set of discrete, steady states with controlled transitions between them will do.

Groups of bits

- Groups of bits can represent data or information
 - 1 bit - 2 alternatives
 - 2 bits - 4 alternatives
 - 3 bits - 8 alternatives
 - 4 bits - 16 alternatives
 - n bits - 2^n alternatives
 - 8 bits - $2^8 = 256$ alternatives
 - a group of 8 bits is called a byte

4 Bits

- | | |
|--------|--------|
| • 0000 | • 1000 |
| • 0001 | • 1001 |
| • 0010 | • 1010 |
| • 0011 | • 1011 |
| • 0100 | • 1100 |
| • 0101 | • 1101 |
| • 0110 | • 1110 |
| • 0111 | • 1111 |

Bit Grouping (Memory)

- The bit grouping created by the address lines, CPU selects the memory, memory slot and I/O unit by using address lines. If the number of address lines is n bits, the memory capacity is 2^n bytes.
- How many bytes in 64kbytes? $64 * 2^{10}$ bytes = $2^6 * 2^{10} = 2^{16}$ bytes = 2^n . Here n is number of the address lines.
- Then, if the number of address lines is 19, how much bytes is the memory capacity? How many Kbytes is it? 2^{19} bytes = 2^9 Kbytes = 512 Kbytes
- If the memory capacity is 128 Kbytes, what is the number of address lines? $128 \text{ Kbyte} = (2^7) * (2^{10}) = 2^{17}$ bytes, number of address lines = 17.

Question: How many address lines? 16 pieces. From where? Number of bit groupings, $n=16$

- Description: Since it defines the memory capacity of 64 Kbytes, there are 16 address bus lines. $2^{16} = 2^6 * 2^{10} = 64 \text{ Kbytes}$

Bit

- The values a bit can take are 0 or 1. Number of states= $2^1=2$, State: 0,1
- If we have two bits of information, how many states are there? There are $2^2=4$ situations. Status: 00, 01, 10,11.
- There are 8 states that three bits can take: 000, 001, 010, 011, 100, 101, 110, 111. Number of states= $2^3=8$
- The number of states that n bits can take = 2^n . It is defined as a state space.

For example, the temperature of the room is 25 degrees, this is information.

- $(25)_d = 2^4 + 2^3 + 2^0$; Indexing = 43210
- $(25)_d = (11001)_b$, $[(25)]_{10} = [(11001)]_2$
- How to convert a binary number to decimal? It is indexed starting with 0 from the right. Index values from 1 to 2 are added together. $[(11001)]_2 = 2^4 + 2^3 + 2^0 = 25$.
- $(11001)_b$: this is information stored in the computer. It represents data in the binary number system.
- In the computer, all physical systems and signals are represented by bits.

Bit (0/1)

- BIT is a unit of information equivalent to the result of a choice between only 2 possible alternatives in the binary number system.
- Bits consist of 0s or 1s. They are mathematical objects and correspond to physical situations.
- It is information that is asked to a system and is one of two answers given as true or false.
- Bit is the basic concept, the cornerstone of information used in computer systems.
- Bits are used to represent information by computers.
- Regardless of its physical implementation, a bit is always understood to be 0 or 1.
- An analogy to this is a light switch with 0 representing the off position and 1 representing the on position.
- There are two types of bits, a classical cbit, bit (0,1) and a quantum bit, qubit ($|0\rangle$, $|0\rangle$).

Bit

Bit: In digital electronics and the binary number system, there are only values 0 and 1. All operations are performed on these two values. Each piece of information 0 or 1 is called a bit. Information consisting of bit → 0/1

- Bits are the units used to describe an amount of data in a network
 - 1 kilobit (Kbit) = 1×10^3 bits = 1,000 bits
 - 1 megabit (Mbit) = 1×10^6 bits = 1,000,000 bits
 - 1 gigabit (Gbit) = 1×10^9 bits = 1,000,000,000 bits

Bit/Sec

Bit/Second: *Bit/sec* → 1 sec. or information transmitted from one point to another. Or it is the bit, that is, the amount of information processed in one second. BPS (Bit Per Second); The number of bits transmitted per second is called bps.

- Seconds are the units used to measure time
 - 1 millisecond (msec) = 1×10^{-3} seconds = 0.001 seconds
 - 1 microsecond (msec) = 1×10^{-6} seconds = 0.000001 seconds
 - 1 nanosecond (nsec) = 1×10^{-9} seconds = 0.000000001 seconds
- Bits per second are the units used to measure channel capacity/bandwidth and throughput
 - Bit per second (Bps)
 - Kilobits per second (Kbps)
 - Megabits per second (Mbps)

Bits and Bytes are Slightly Different

“Kilo” or “Mega” have slightly different values when used with bits per second or with bytes.

- When Referring to Bytes (as in computer memory)
 - Kilobyte (KB) $2^{10} = 1,024$ bytes
 - Megabyte (MB) $2^{20} = 1,048,576$ bytes
 - Gigabyte (GB) $2^{30} = 1,073,741,824$ bytes
 - Terabyte (TB) $2^{40} = 1,099,511,627,776$ bytes
- When Referring to Bits Per Second (as in transmission rates): It is the data processing or data transfer rate.
 - Kilobit per second (Kbps) = 1000 bps (thousand)
 - Megabit per second (Mbps) = 1,000,000 bps (million)
 - Gigabit per second (Gbps) = 1,000,000,000 bps (billion)
 - Terabit per second (Tbps) = 1,000,000,000,000 bps (trillion)

Byte

Byte: Gives the memory size. In electronics and computer science, it is a memory measurement unit that generally incorporates 1 or 0 values along an 8-bit sequence and is independent of the type of information recorded.

The term bit is defined as the smallest unit of memory that marks or describes an 8-bit value.

Byte→memory defines the 8-bit address box or memory size.

- Kilobyte (K) $2^{10} = 1,024$ bytes
- Megabyte (M) $2^{20} = 1,048,576$ bytes
- Gigabyte (G) $2^{30} = 1,073,741,824$ bytes
- Terabytes (T) $2^{40} = 1,099,511,627,776$ bytes
- Petabytes (P) $2^{50} = 1,125,899,906,842,624$ bytes
- Exabytes (E) $2^{60} = 1,152,921,504,606,846,976$ bytes
- Zettabytes (Z) $2^{70} = 1,180,591,620,717,411,303,424$ bytes
- Yottabytes (Y) $2^{80} = 1,208,925,819,614,629,174,706,176$ bytes

Baud Rate

Baud Rate is the unit for symbol rate or [modulation](#) rate in **symbols per second** or **pulses per second**.

Example: Let the transmission speed of a data transmission line be 4800 baud/sec. If this transmission contains information encoded with 4 bits per baud (symbol), our speed in bps is $4800 * 4 = 19200$ bps. The purpose of using Baud Rate is to use bandwidth more efficiently.

- The symbol duration time T_s can be calculated as: $T_s = 1/f_s$, where f_s is the symbol rate.
- Example: Communication at the baud rate *1000 Bd* means communication by means of sending *1000 symbols per second*. The symbol duration time is *1/1000 second* (that is, *1 millisecond*). In [digital](#) systems with [binary code](#), 1 Bd = 1 bit/s. By contrast, non-digital (or [analog](#)) systems use a continuous range of values to represent information and in these systems the exact informational size of 1 Baud varies.

Örnekler

How many Bytes is 12Gbyte?

- Instead 12 we take 16 alınır. (2^n olmalıdır.)
- $16\text{GByte} = 2^4 \cdot 2^{30} = 2^{34}$ Byte

How much Bytes is 16Gbyte?

How much Bytes is 128Mbyte?

- $128\text{MByte} = 2^7 \cdot 2^{20} = 2^{27}$ Byte

How much Bytes is 256Gbyte?

- $256\text{MByte} = 2^8 \cdot 2^{20} = 2^{28}$ Byte

How much Bytes is 4Kbyte?

- $4\text{KByte} = 2^2 \cdot 2^{10} = 2^{12}$ Byte

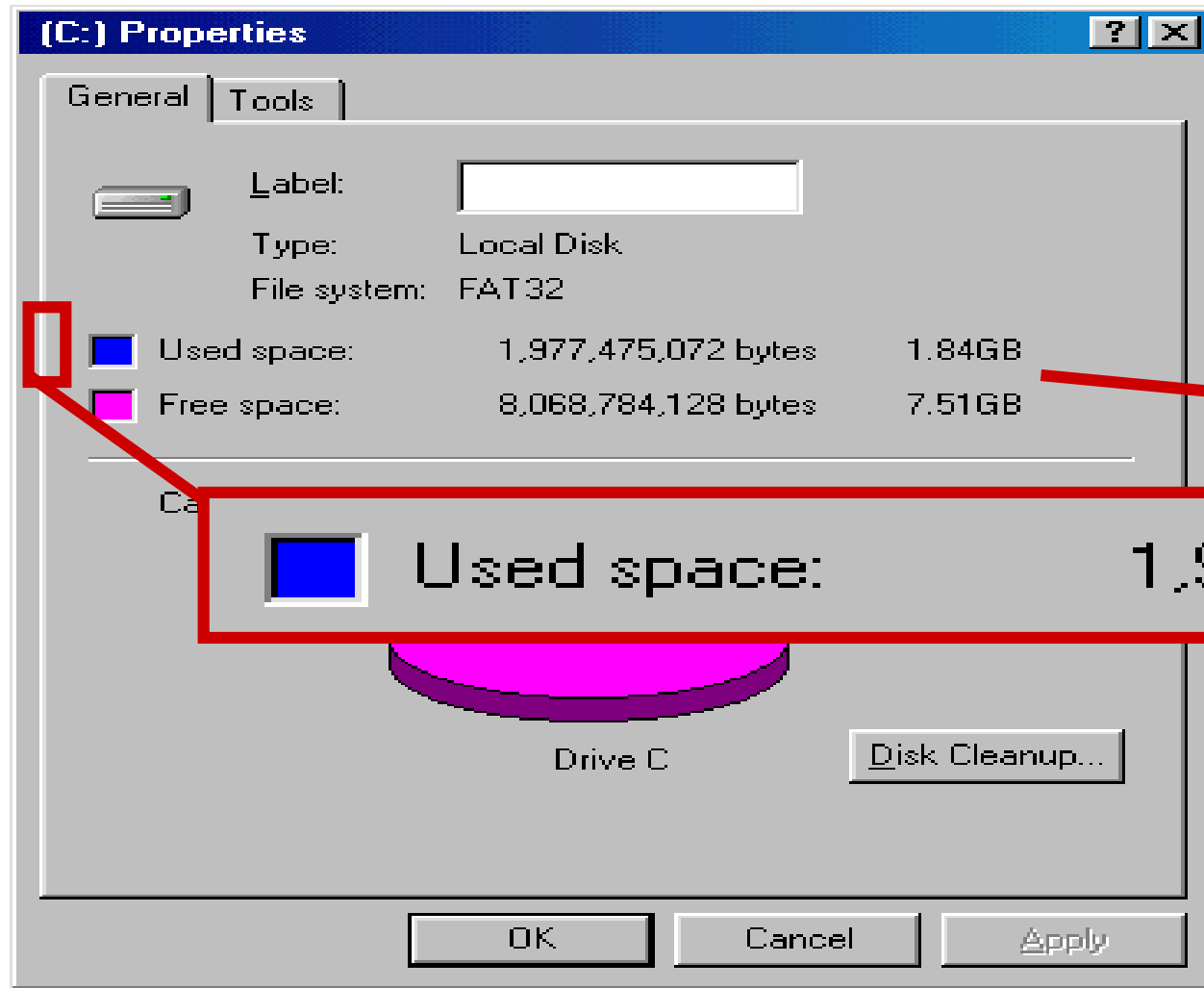
How much Bytes is 32Tbyte?

- $32\text{TByte} = 2^5 \cdot 2^{40} = 2^{45}$ Byte
- $32\text{TByte} = 2^5 \cdot 2^{40} = 2^{45} / 2^{10} = 2^{35}$ KByte
- $32\text{TByte} = 2^5 \cdot 2^{40} = 2^{45} / 2^{20} = 2^{25}$ MByte
- $32\text{TByte} = 2^5 \cdot 2^{40} = 2^{45} / 2^{30} = 2^{15}$ GByte
- $32\text{TByte} = 2^5 \cdot 2^{40} = 2^{45} / 2^{40} = 2^5$ TByte

How much Bytes is 8Pbyte?

- $8\text{PByte} = 2^3 \cdot 2^{50} = 2^{53}$ Byte

Example



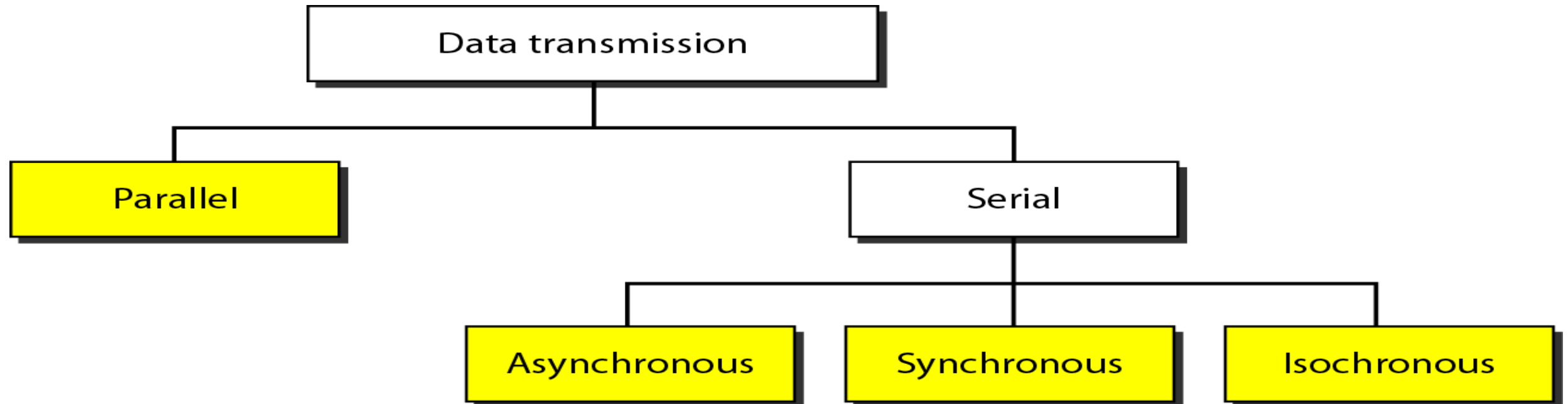
In the lab...

1. Double click on My Computer
2. Right click on C:
3. Click on Properties

$$/ 2^{30} =$$

**Data Transmission
on
Computer Systems**

Data transmission and modes



Serial Transmission

Data is transmitted, on a single channel, one bit at a time one after another
- Much faster than parallel



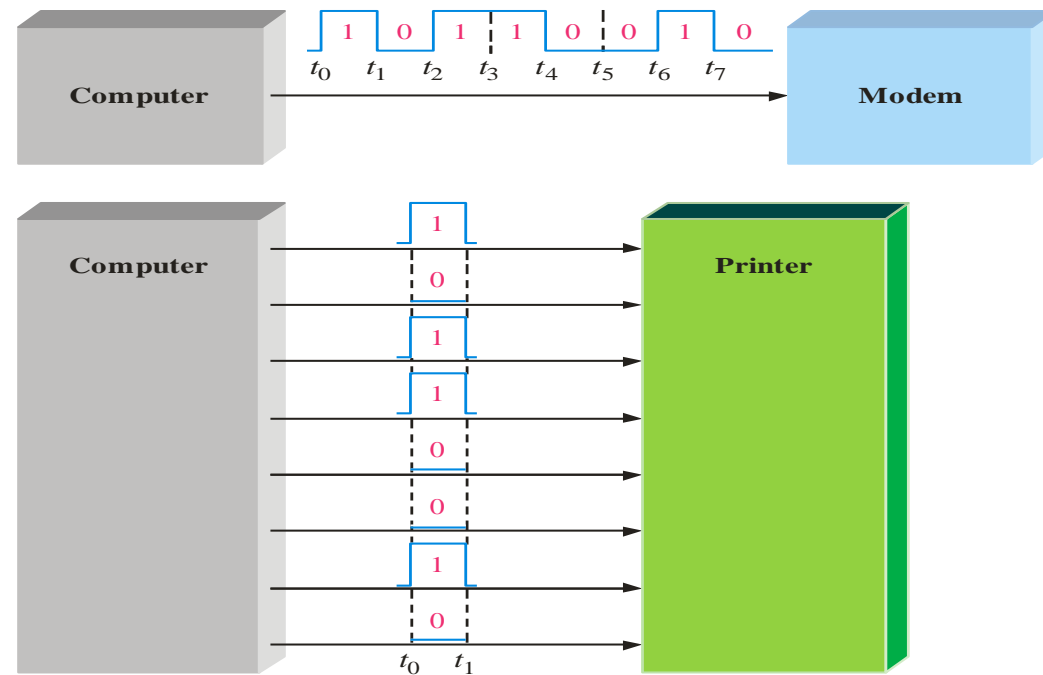
Parallel Transmission

- each bit has it's own piece of wire along which it travels
- often used to send data to a printer

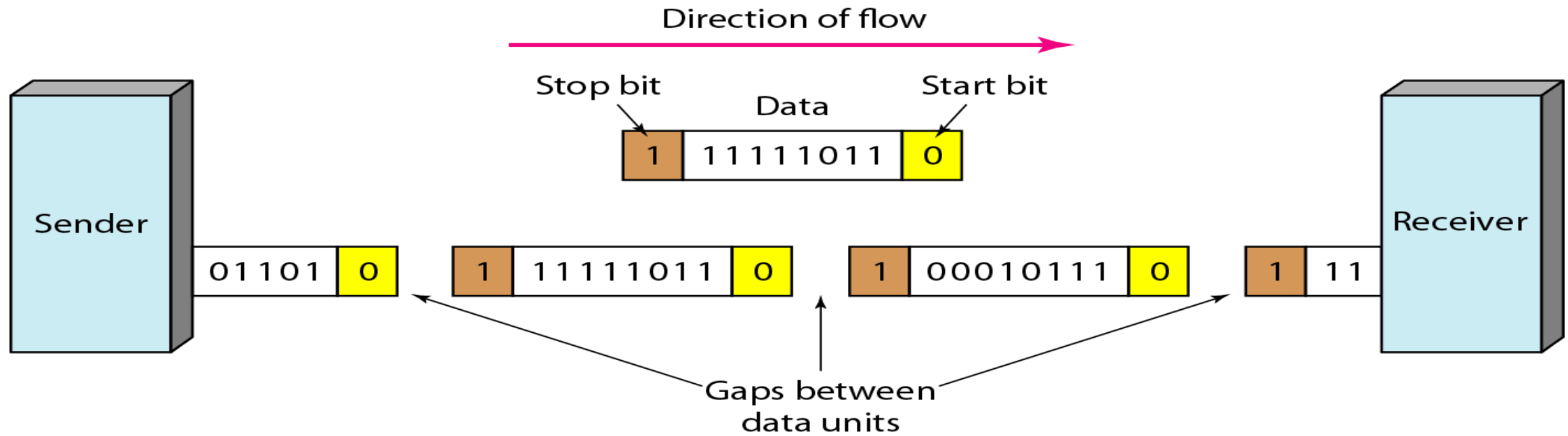


Serial and Parallel Data

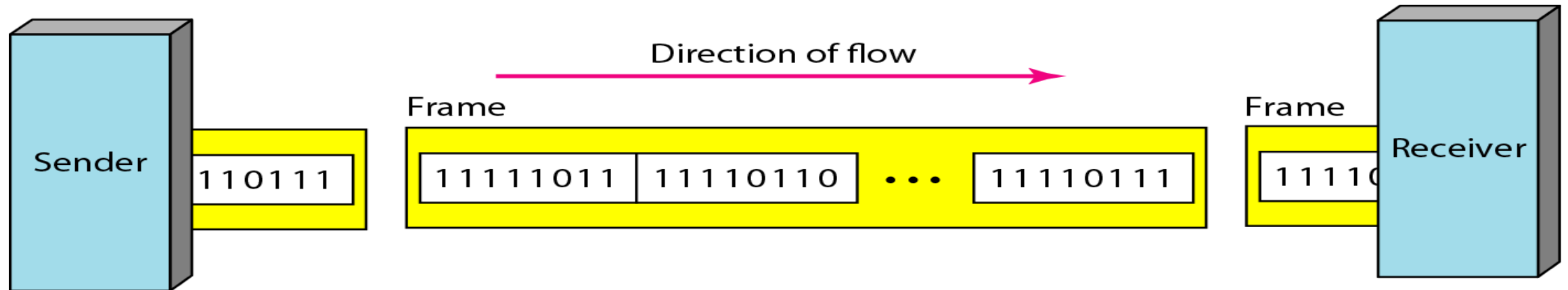
Data can be transmitted by either serial transfer or parallel transfer.



Asynchronous transmission



Synchronous transmission





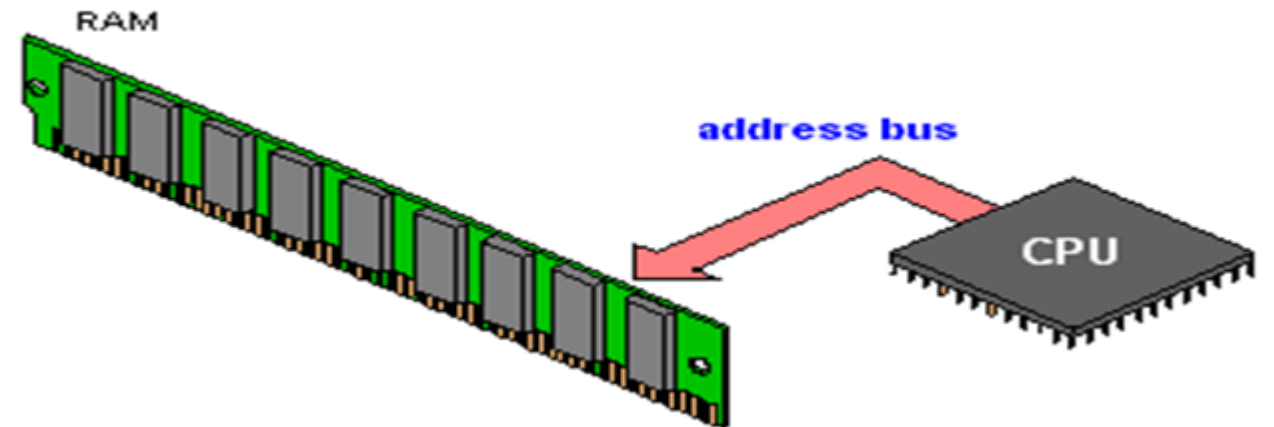
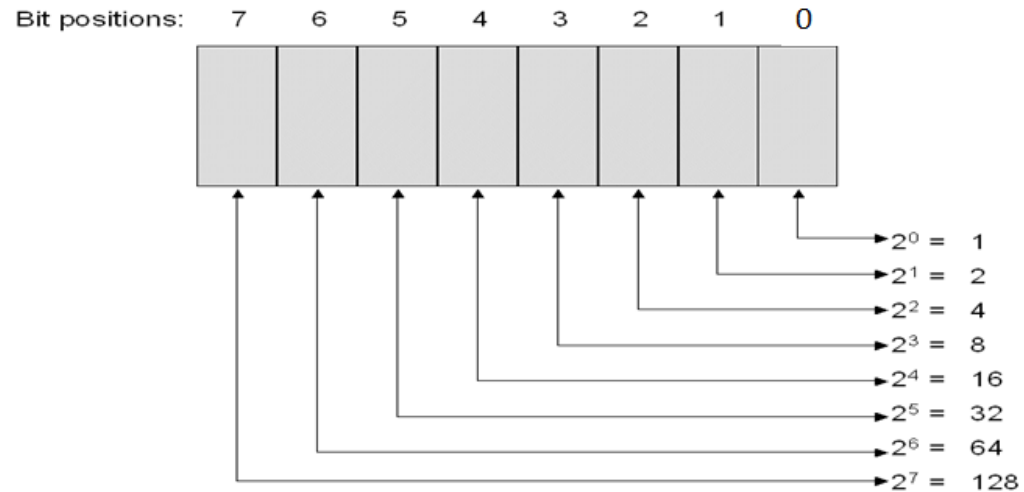
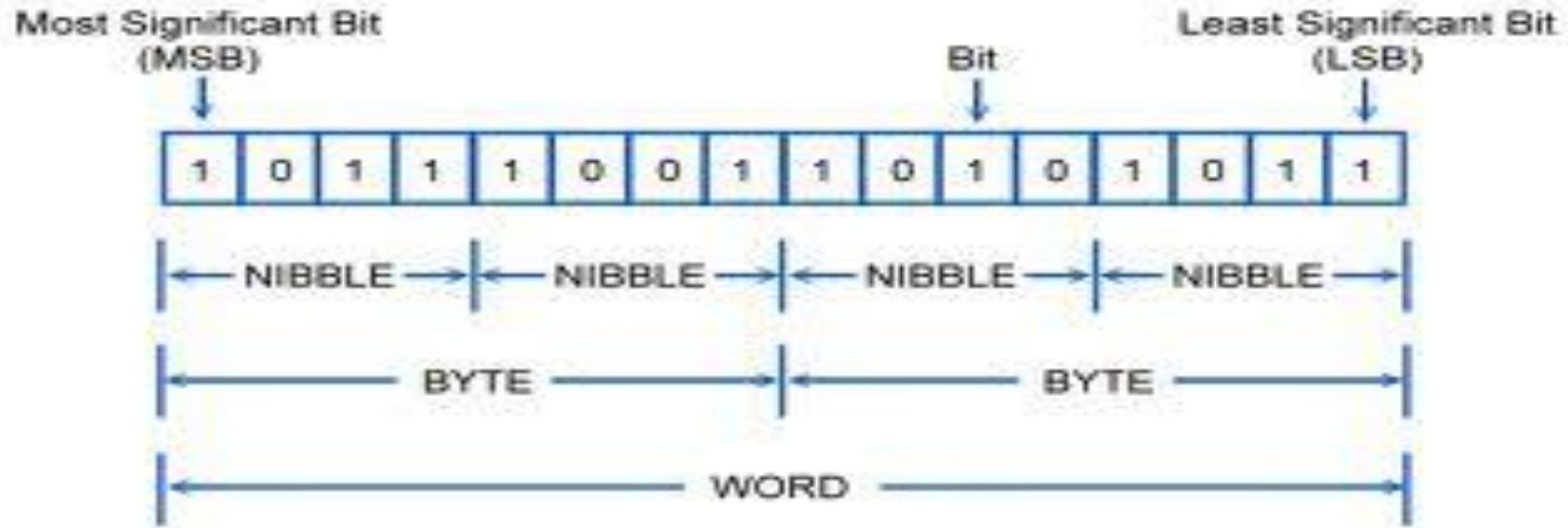
Data Representation in Computer

Data Types and Size



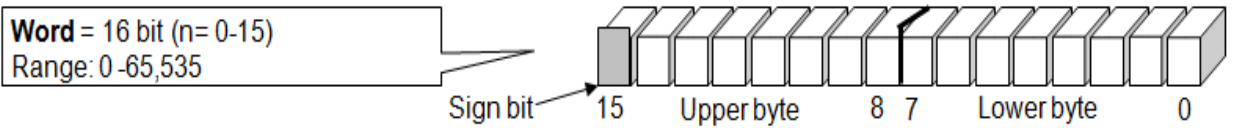
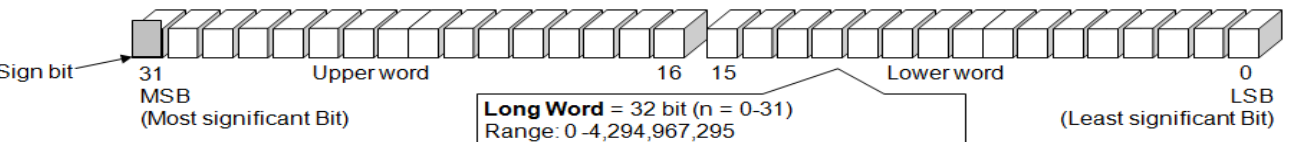
- A **bit** is a single **binary digit** (a 1 or 0).
- A **byte** is 8 bits
- A **word** is 16 bits or 2 bytes
- **Double word** is 32 bits or 4 bytes
- **Long word** = 8 bytes = 64 bits
- **Quad word** = 16 bytes = 128 bits
- Programming languages use these standard number of bits when organizing data storage and access.
- What do you call 4 bits? (hint: it is a small byte) (Nibble)

Question: 10^9 bit/sec (Data processing or data transfer speed)

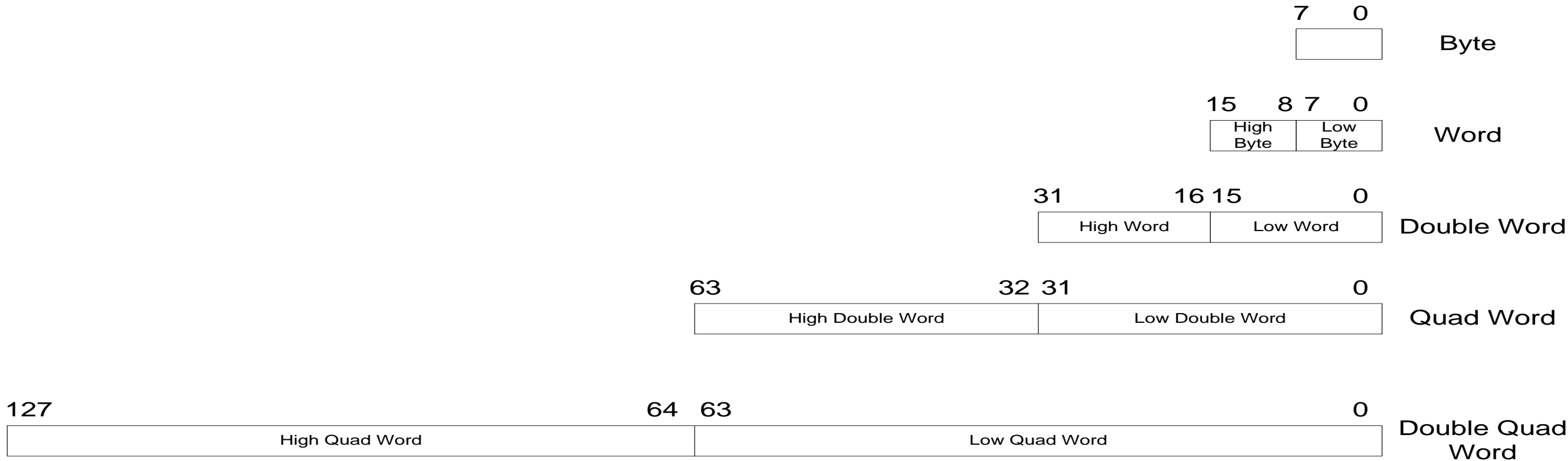
Bits and Bytes



DATA SIZE

<p>Nibble</p>	<p>4 bit</p>	<p>Nibble = 4 bit (n= 0-3) Range: 0 -15</p> 
<p>Byte</p>	<p>8 bit</p>	<p>Byte = 8 bit (n = 0-7) Range: 0 -255</p> 
<p>Word</p>	<p>16 bit</p>	<p>Word = 16 bit (n= 0-15) Range: 0 -65,535</p> 
<p>Long word</p>	<p>32 bit</p>	<p>Long Word = 32 bit (n = 0-31) Range: 0 -4,294,967,295</p> 

Data Types - size



Always take care of the type of data an instruction accesses!!!!

Data Representations

- Sizes (in Bytes)

– Data Type	Compaq Alpha	Typical 32-bit	Intel IA32
• int	4	4	4
• long int	8	4	4
• char	1	1	1
• short	2	2	2
• float	4	4	4
• double	8	8	8
• long double	8	8	10/12
• char *	8	4	4

Digital Number systems

- Digital Number systems
 - Binary, Bits (0/1)
 - Byte: 8 bit
 - Hexadecimal: 4 bit
 - Octal: 3 bit
 - Numbers conversion among different systems
- Ascii code (Each key on the keyboard is represented by 8 bits, that is, 1 byte). For example, what appears in memory when the A key on the keyboard is pressed? 8 bit: $(41)_h = (0100\ 0001)_b$

Example: Digital Number Systems

- **Binary** numbers
 - Binary Digits = {0, 1}
 - Example: $(11010)_2 = ?$
 - Indexing: It is indexed from right to left.
 - Indexing: 43210;
 - When converting to the decimal number system, when adding expressions to the power of 2, multiply by 1 if the coefficient is 1 and by 0 if it is 0.
 - Decimal value: $1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (26)_d$

Example: Float Point Digital Number Systems

- **Binary** numbers
 - Digits = {0, 1}
 - $(11010.11)_2 = ?$
 - Indexing: It is indexed separately from the point to the left and right.
 - Indexing: 43210.-1-2
 - Decimal value: $1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$
 $= (26.75)_{10}$

Example.....

- If a family has 3 children, find the probability that all 3 children are girls.
- ing for all 3 girls.....

- Sample Space:

BBB

BBG

BGB

GBB

→ **GGG**

GGB

GBG

BGG

$$P(\text{All 3 Girls}) = \frac{1}{8}$$

(2³) 2: number of people, 3: number of lineups (Number of children)

Analogue and Digital Signals

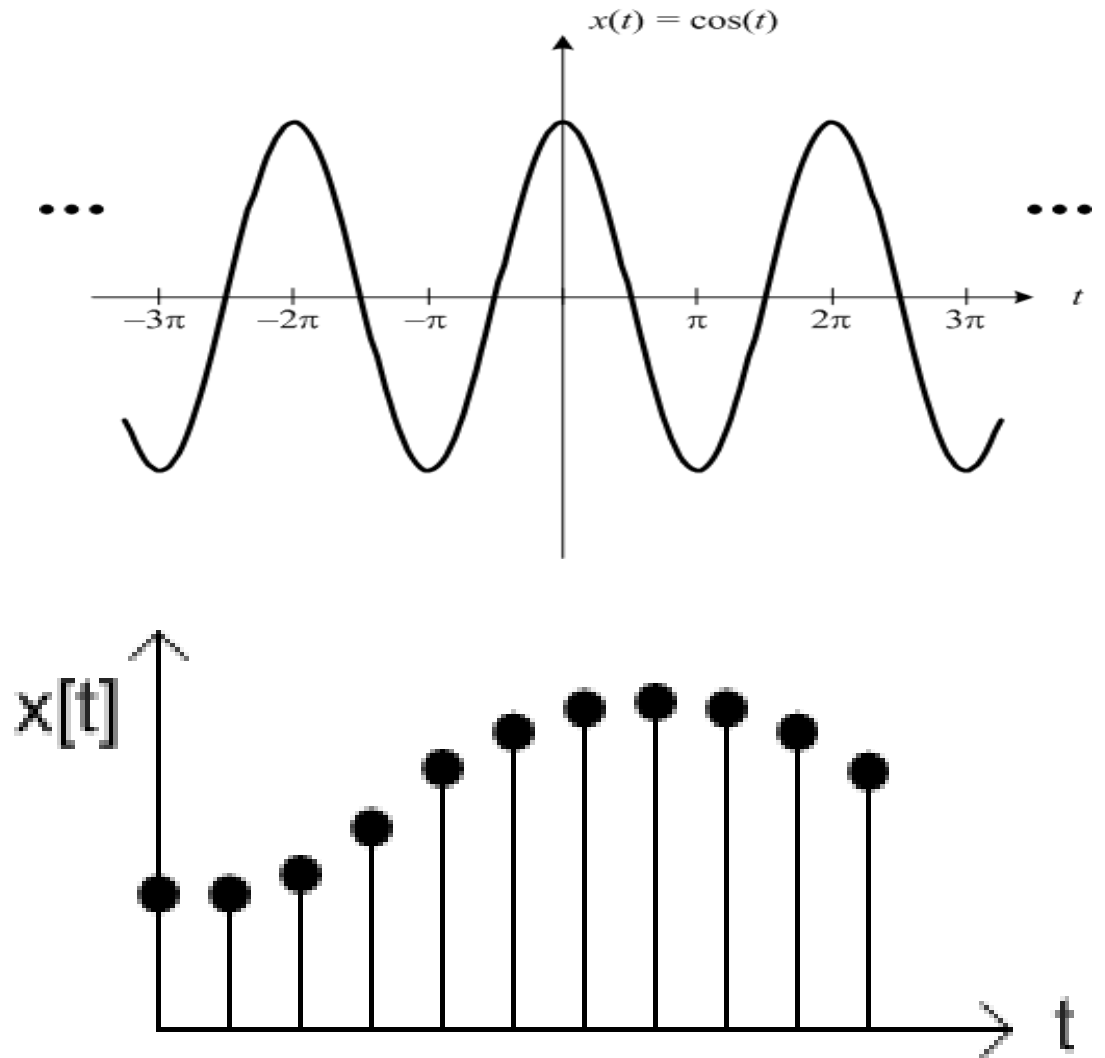
How is data represented inside the computer?

- Inside the computer, strings are represented as a sequence of 1s and 0s, just like numbers.
- Signals spread far away in the form of waves. Waves carry messages in signals. Signals: Acoustic signals, Seismic Signals, Electrical signals, Electromagnetic signals, Heat, Vibration, ...
- Messages/Symbols: Sound, Numbers (positive, negative, integer, float, fraction), Characters, Texts, Picture, Image, Keyboard keys; messages, symbols
- Analog signals: These are signals whose amplitude, frequency and phase change over time.
 $X(t)=A(t)*\sin(\omega t+\phi(t));$
- $\omega=2\pi f$, f : frequency (Hz=1/sec), ϕ : phase (degrees or radians), A : Amplitude (unit, volt, ampere)
- Since data in the computer system is made as bit (1/0) while arithmetic and logical operations are performed, stored in memories, and transferred between relevant units, analog signals must be converted into digital signals.

Classification of signals

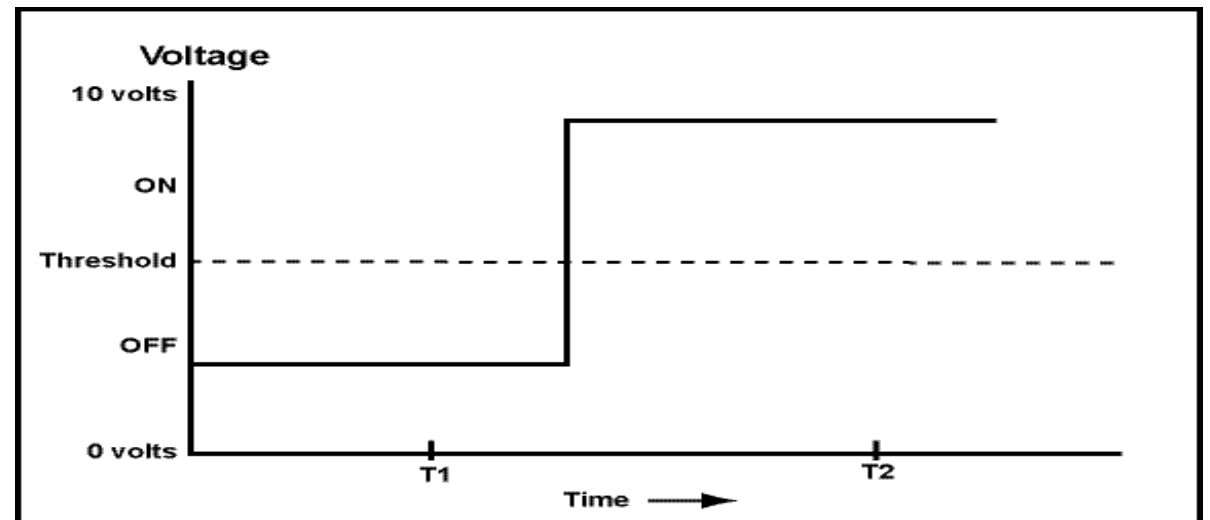
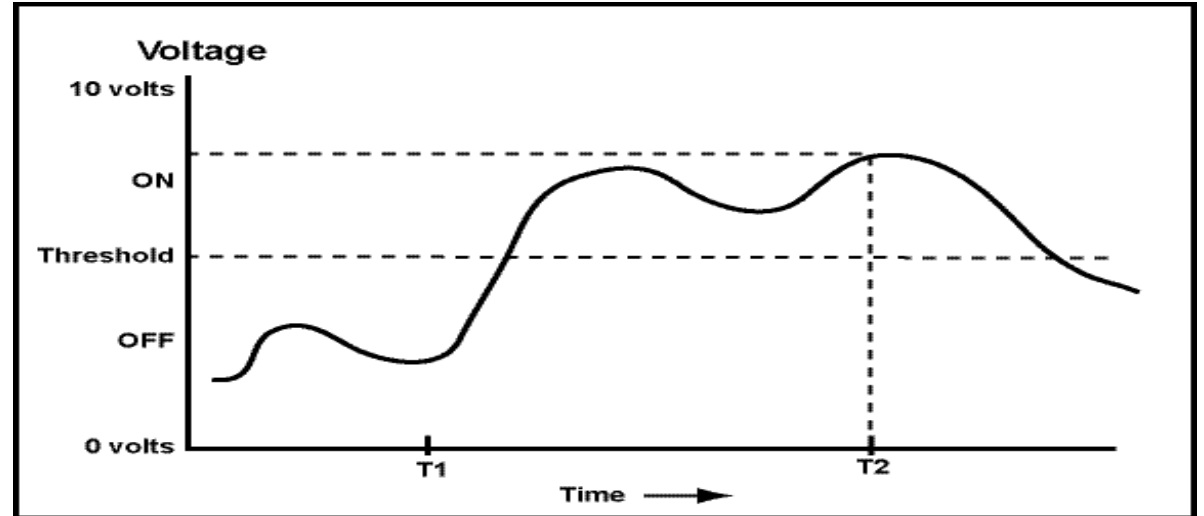
Signals are basically classified into two different types as follows.

- Continuous time signals, Analog signals
- Discrete (Ayrık) time signals: It is the name given to the output signal obtained by measuring the input value at certain intervals or levels. The discrete-time signal is derived from the input signal by the sampling process. Samples taken from the discrete-time signal are converted into a digital signal by quantization.



Analog to Binary Signal

- A voltage below the *threshold*
 - Off (0)
- A voltage above the *threshold*
 - On (1)



Analogue and Digital Signals

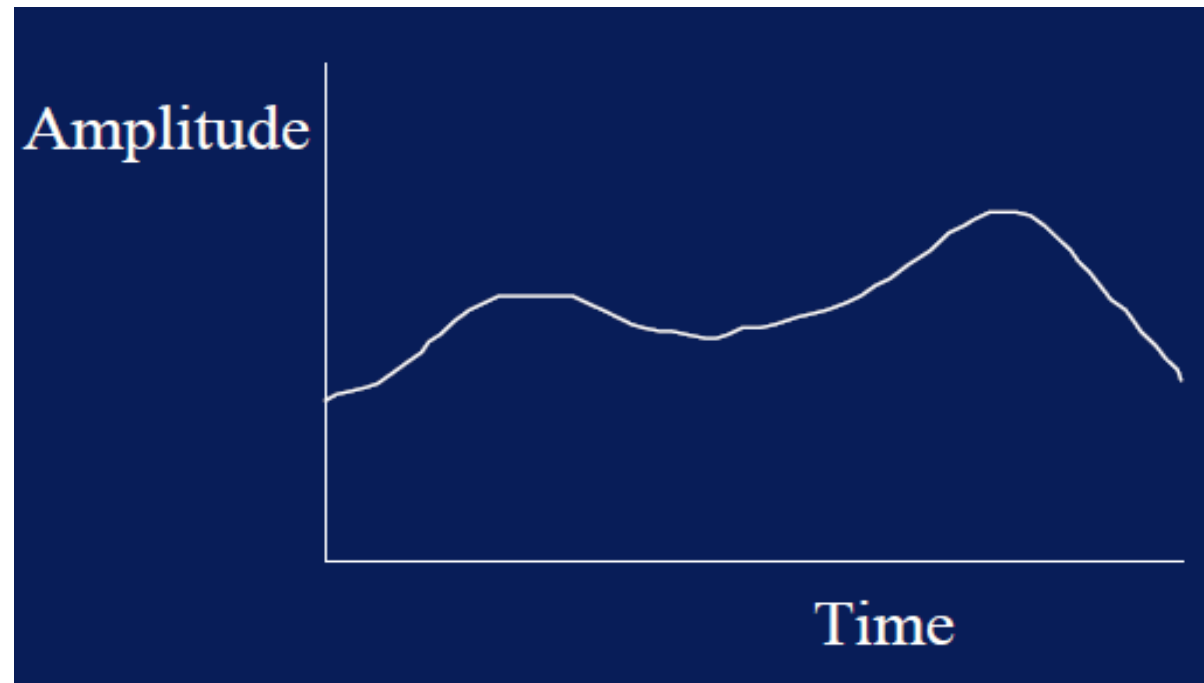
- Calculation with numbers is usually done in base 10 arithmetic
- Easier to effect machine computation in base 2 or binary notation
- We can also use base 2 or binary notation to represent logic values: TRUE and FALSE
- Manipulation of these (digital) logic values is subject to the laws of logic as set out in the formal rules of Boolean algebra

Analogue and Digital Signals

- An analogue signal can have any value within certain operating limits
- For example, in a (common emitter) amplifier, the output (O/P) can have any value between 0v and 10v.
- A digital signal can only have a fixed number of values within certain tolerances

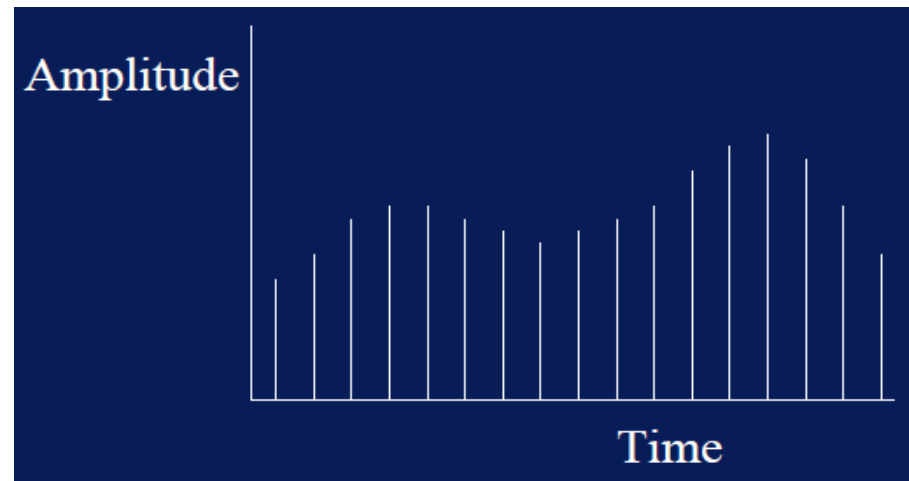
Analogue Signals

- The amplitude is defined at all moments in time



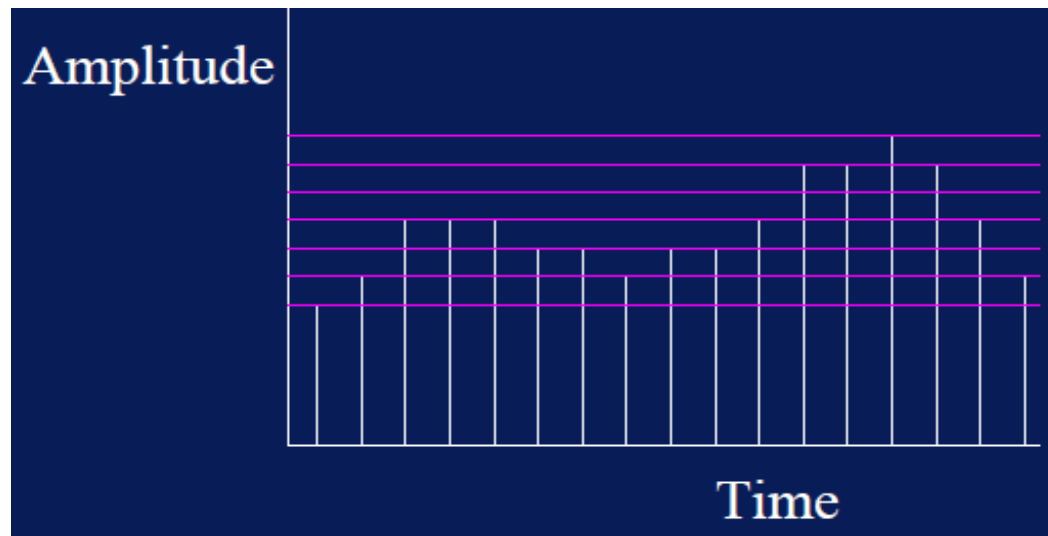
A digital signal

- It is a sampled version of the analogue signal
- Only defined at certain discrete times
- DISCRETE TIME SIGNAL
- A digital signal is a sampled version of the analogue signal
- Analog signal sampling interval= $1/f_{\max}$, [sec]. Here f_{\max} =maximum frequency of the analog signal.



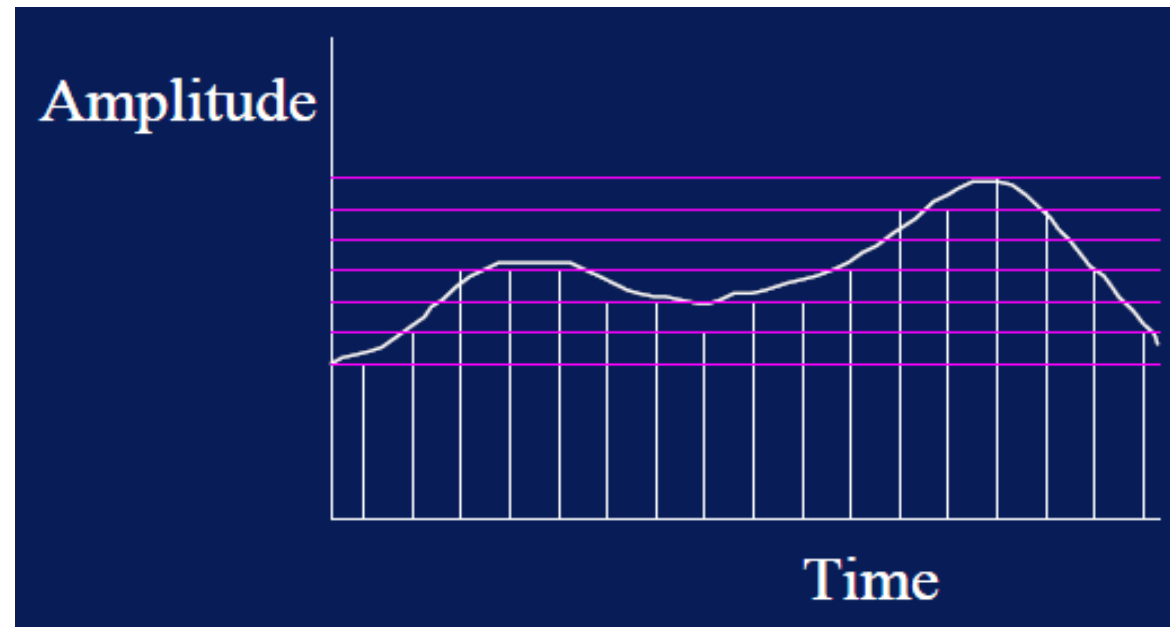
Analogue and Digital Signals

- The amplitude may also be restricted to take on discrete values only
- • In which case it is said to be quantized



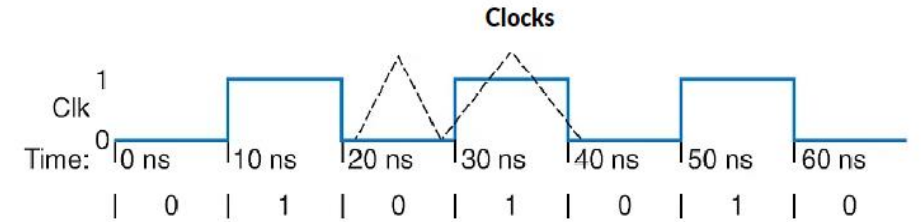
Analogue and Digital Signals

- Quantization introduces errors which depend on the step size or the resolution
- Signals (voltages or currents) which are samples and quantized are said to be DIGITAL
- They can be represented by a sequence of numbers



Digital Signal Definitions

- Digital signal: A signal consisting of a series of digital discrete values. The environments where information is processed are digital. Unit Pulse Signal: It is the signal that changes from one level to another for a period of time, called pulse width, after which it returns to the original level. Clock Signal: It is an evenly spaced continuous electrical pulse train signal. The bits (0/1) representing the data gain meaning by being triggered by the rising or falling edge of the clock pulse train. The length of the data must be the same as the period of the clock signal. It is the unit length of the signal that repeats at equal intervals. Bit: The smallest possible unit of information in a computer (0/1). It is transported and processed as an electrical signal. Binary number system: It is the number system represented by bit (0/1). Byte: Represents 8 bits of data or points to an 8-bit memory unit.



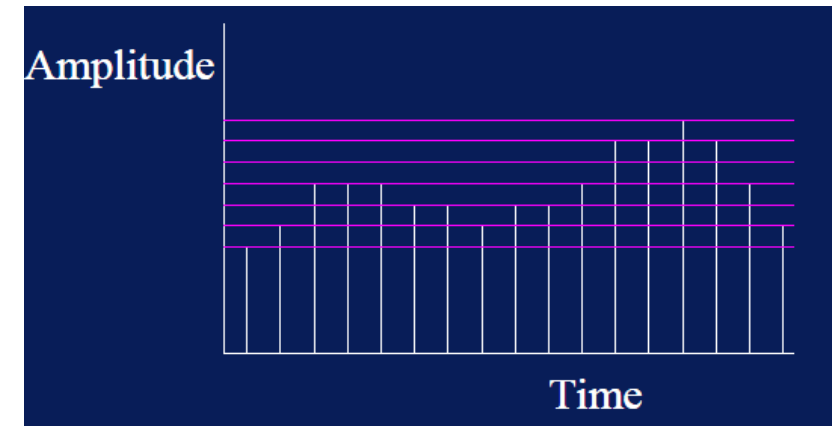
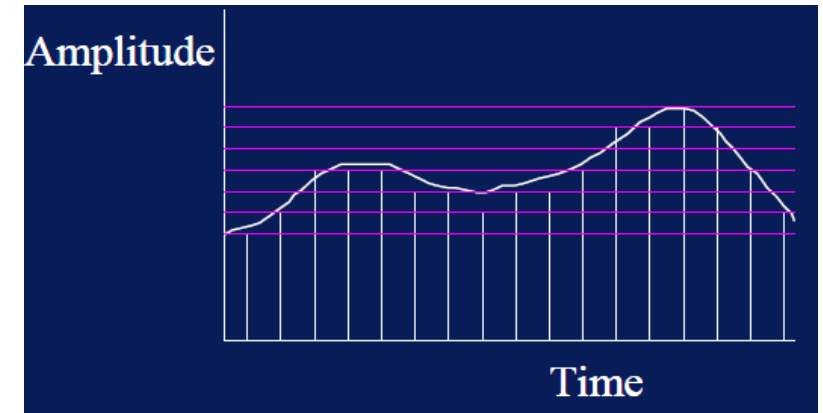
- **Clock period:** time interval between pulses
 - Above signal: period = 20 ns
- **Clock cycle:** one such time interval
 - Above signal shows 3.5 clock cycles
- **Clock frequency:** $1/\text{period}$
 - Above signal: frequency = $1 / 20 \text{ ns} = 50 \text{ MHz}$
 - $1 \text{ Hz} = 1/\text{s}$

Freq	Period
100 GHz	0.01 ns
10 GHz	0.1 ns
1 GHz	1 ns
100 MHz	10 ns
10 MHz	100 ns

Digital Signals – Sampling Theorem

Analog ve Digital Signals

- The messages produced by all the components that make up the universe are transmitted via analog signals; In this way, they are in interactive communication with each other.
- While the analog signal is converted into a digital signal, samples are taken from the amplitude and phase values at certain time intervals (sampling frequency). This process is called the sampling function. When sampling, the sampling frequency must be greater than or equal to twice the maximum frequency of the analog signal.
- The values of the samples taken from the analog signal are assigned to discrete values within the amplitude scaling range. This process is called quantization. During this assignment process, quantization errors occur depending on the sampling time interval, quantization values, translation and resolution.
- Representing discrete values with a certain number of binary number systems is called encoding in the binary number system. Each discrete amplitude value is represented by a certain number of binary (0/1) numbers. Thus, digital signals are obtained.



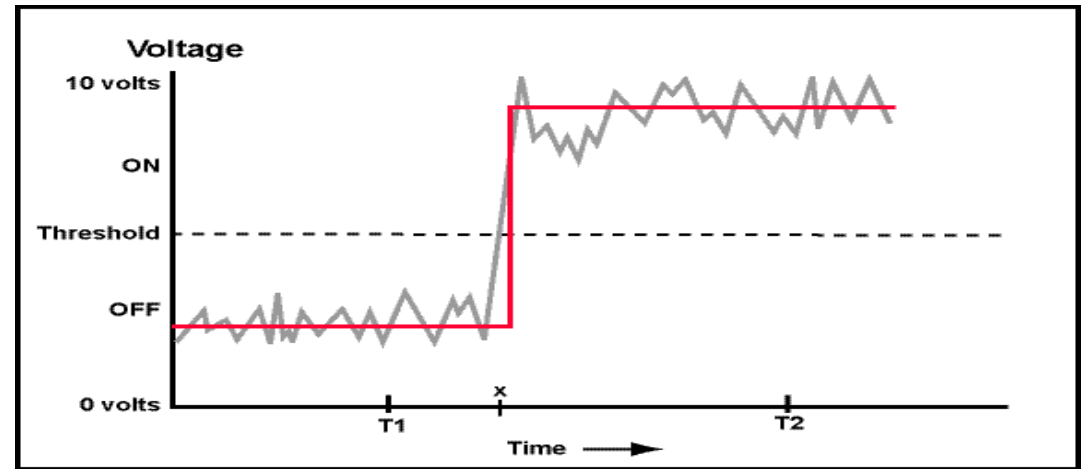
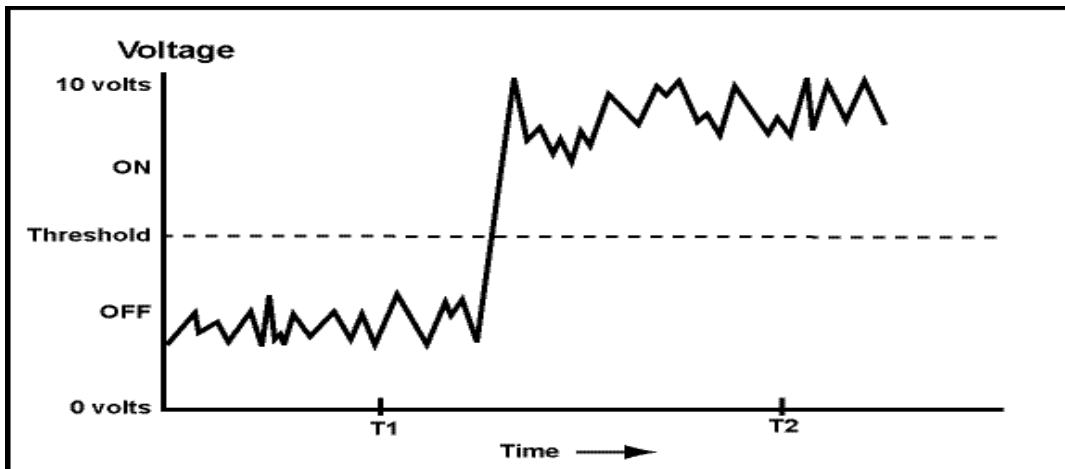
ANALOG-TO-DIGITAL CONVERSION

A digital signal is superior to an analog signal because it is economical, more resistant to noise, and can be easily recovered, rectified, and amplified. It can be stored temporarily or permanently on the transistor. Therefore, the current trend is to change an analog signal to digital data. Computer or digital systems are mentioned. Digital signal is not suitable in the transmission medium. While digital signals are transmitted through communication channels, they are converted into analog signals.

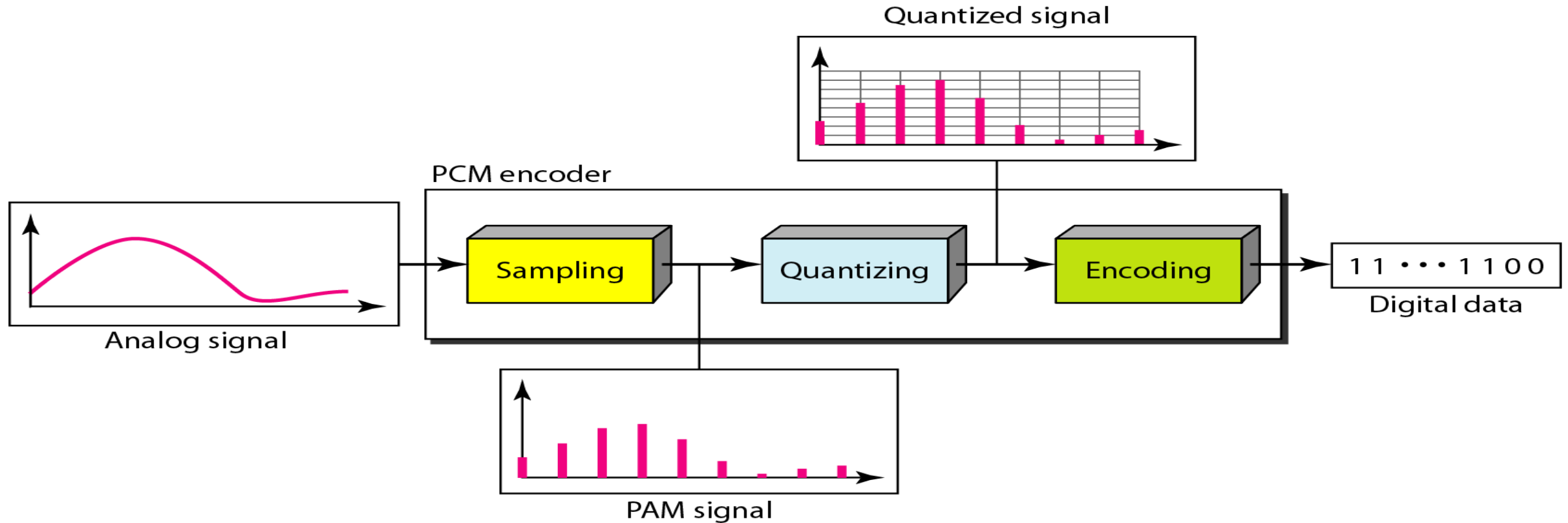
Dijital bir sinyal analog sinyale göre daha üstündür çünkü ekonomiktir, gürültüye karşı daha dayanıklıdır ve kolayca geri kazanılabilir, düzeltilebilir ve yükseltilebilir. Transistör üzerinde geçici ya da sürekli saklanabilmektedir. Bu sebeple, günümüzdeki eğilim bir analog sinyali dijital verilere değiştirmektir. Bilgisayar ya da sayısal sistemlerden bahsedilmektedir. İletim ortamında sayısal sinyal elverişli değildir. Sayısal sinyaller haberleşme kanallarında iletilirken analog sinyallere dönüştürülür.

Noise on Transmission

- When the signal is transferred it will pick up noise from the environment
- Even when the noise is present the binary values are transmitted without error
- Recovery - Filtering

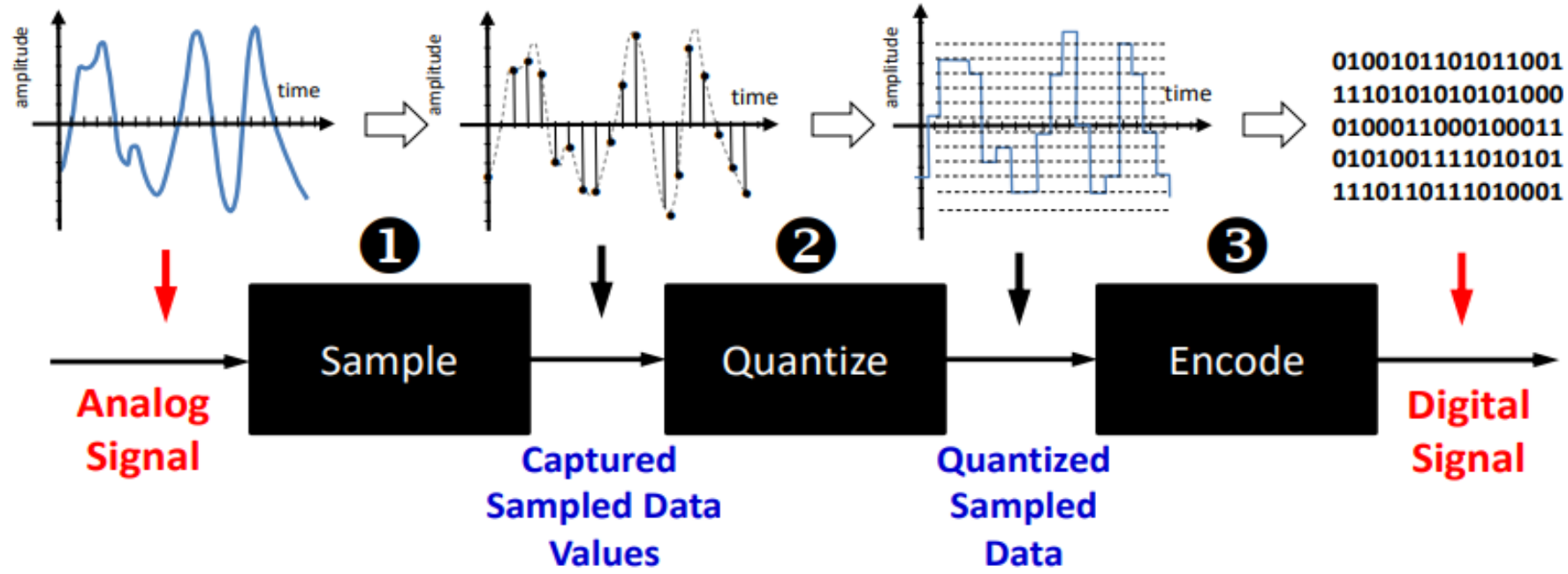


Digitization of analog signal



- The values of the samples taken from the analog signal are assigned to discrete values in the amplitude scaling range. This process is called quantization. During this assignment process, quantization errors occur depending on the sampling time interval, quantization values, translation and resolution. Representing discrete values with a certain number of binary number systems is called encoding in the binary number system. Each discrete amplitude value is represented by a certain number of binary (0/1) numbers. Thus, digital signals are obtained.

Analog to Digital Conversion Process (ADC)



Quantization: While the samples taken from the analog signal are converted into a bit sequence (converted to digital), the values between the layers are shifted either to the upper layer or to the lower layer. This process is called quantization.

Nyquist Sampling Theorem

When an analog signal is sampled and converted to digital and then converted back to an analog signal, the sampling frequency must be equal to or greater than twice the bandwidth of the signal in order to obtain the same signal.

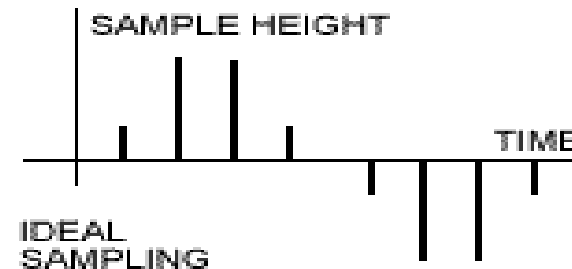
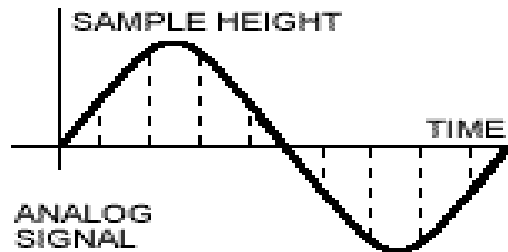
$$f_s \geq 2 * f_m$$

$$f_s \geq 2 * B$$

- Here f_s is the sampling frequency and f_m is the maximum frequency in the signal; B is the bandwidth. Frequency is the number of periods in one second. $f=1/T$. T : period [second], f : frequency [Hz=1/second] In telephone communication, bandwidth is $B= 4\text{KHz}$ (Understanding, Recognition, Feeling) and $f_s=2*B$. Then the sampling interval is $T=1/8\text{KHz}=125\text{microseconds}$.

The Sampling

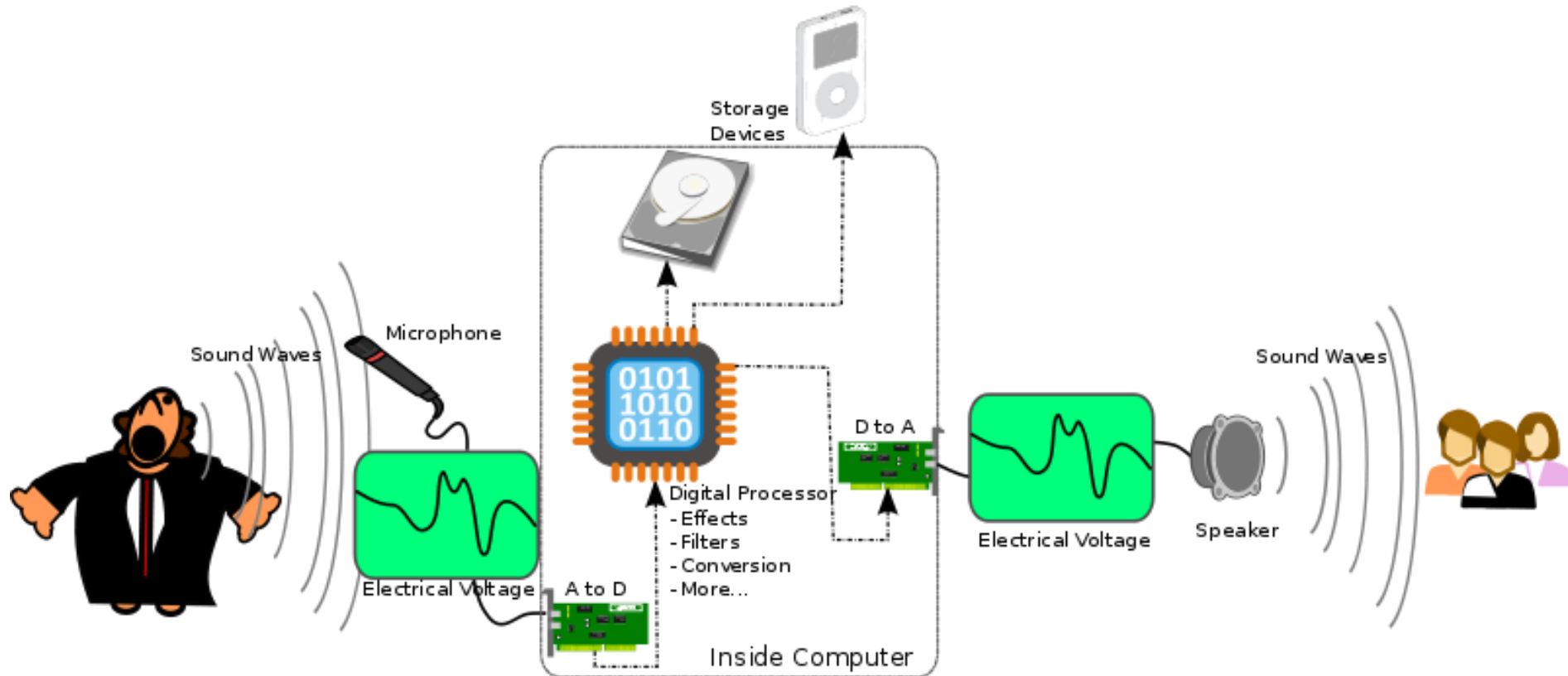
- Sampling is converting a continuous time signal into a discrete time signal
- Sampling is usually done at equal time intervals; This interval is called the sampling interval. The reciprocal of the sampling interval is called sampling frequency or sampling rate. The unit of sampling rate is Hz.
- In accordance with the sampling theorem, telephone voice signals frequency ranges from 300 Hz to 3400 Hz. It is taken as 4KHz, the sampling frequency is taken as greater than or equal to 8000 Hz.
- If 8000 samples are taken from an analog signal at equal intervals per second and one sample is represented by 8 bits, how many bits are taken per second? $8000 \times 8 = 64\,000 \text{ bit/sec} = 64\text{Kbit/sec}$.
- Minimum capacity of a channel in data communication = 64Kbit/s.
- If each sample is represented by 8 bits, the number of sampling intervals (Quantization) is $2^8 = 256$.
- 256 quantization ranges (layers) are obtained.
- After the sound waves are converted to an analog signal, a sample is taken at $125 \mu\text{sec}$. Sampling interval, $T = 1/8000 = 0.000125 \text{ sec} = 125 \mu\text{s}$, where T is the sampling interval.



Ideal Sampling and Aliasing

- Sampled signal is discrete in time domain with spacing T_s
- Spectrum will repeat for every f_s Hz
- Aliasing (spectral overlapping – Loss of an analog signal within another analog signal) if f_s is too small ($f_s < 2f_m$)
- Nyquist sampling rate $f_s = 2f_m$
- Generally oversampling is done $\rightarrow f_s > 2f_m$

Lifecycle from Sound to Digital to Sound



Source: http://en.wikipedia.org/wiki/Digital_audio

Digital Design and Boolean Algebra

Fundamentals of Digital Logic

- A binary number system is used to represent digital logic values: 1/0 (True/False, Good/Bad, Day/Night, 0V/5V)
- Mathematical operations of digital logic values are governed by the laws specified in the rules of Boolean algebra.
- The mathematical inputs and outputs of the laws specified in the rules of Boolean algebra are represented by the binary number (1/0) system.
- Logical gates that make up the hardware of computer systems
 - AND, OR, NOT, NAND, NOR, XOR, ...
- Logic gates are created using transistors.
 - NOT gate can be implemented by a single transistor
 - AND gate requires 3 transistors
- Transistors are the basic circuit elements of computer systems.
 - Pentium consists of 3 million transistors
 - Compaq Alpha consists of 9 million transistors
 - Now we can build chips with more than 100 million transistors

Design Hierarchy

Many digital systems can be divided into three design levels that form a well-defined hierarchy:

- The Architecture Level: High-level concerned with overall system management
- The Logic Level: Intermediate level concerned with the technical details of the system
- The Physical Level: Low level concerned with the details needed to manufacture or assemble the system
- We have already studied the architecture level
- Now we will address the logic level
- At the logic level, there are two classes of digital system
 - Combinational - digital systems without memory
 - Sequential - digital systems with memory

Boolean Algebra (Mantıksal devre)

A logic variable x can have only one of two possible values or states

- $x = \text{TRUE}$
- $x = \text{FALSE}$

In binary notation, we can say

- $x = \text{TRUE} = 1$
- $x = \text{FALSE} = 0$
- This is called positive logic or high-true logic

Electrically, 1 is represented by a more positive voltage than zero and 0 is represented by zero volts

- $x = \text{TRUE} = 1 = 5 \text{ volts}$
- $x = \text{FALSE} = 0 = 0 \text{ volts}$

Boolean Algebra Theorems

1. a) $a+b=b+a$ Commutativity
b) $a \cdot b = b \cdot a$
2. a) $a+b+c=a+(b+c)$ Merger Feature (Birleşme)
b) $a \cdot b \cdot c = a \cdot (b \cdot c)$
3. a) $a+b \cdot c = (a+b) \cdot (a+c)$ Dispersion Feature (Dağılma)
b) $a \cdot (b+c) = a \cdot b + a \cdot c$
c) $a(b+c) = ab+ac$
4. a) $a+a=a$ Idempotency (Değişkende Fazlalık Özelliği)
b) $a \cdot a = a$
5. a) $a+a \cdot b = a$ Swallowing Feature (Yutma)
b) $a \cdot (a+b) = a$
6. a) $(a)^n = a$ Redundancy Feature in transaction (işlemede Fazlalık Özelliği)
b) $(a \times n) = a$
7. a) $\overline{(a + b)} = \bar{a} \cdot \bar{b}$ De Morgan Rule
b) $\overline{(a \cdot b)} = \bar{a} + \bar{b}$
8. a) $0+a=a$ Ineffectiveness Feature (Etkisizlik Özelliği)
b) $1 \cdot a = a$
9. a) $a+\bar{a} = 1$ Fixed Feature (Sabit Özelliği)
b) $a \cdot \bar{a} = 0$
10. a) $1+a=1$ Devourer Fixed Feature (Yutan Sabit Özelliği)
b) $0 \cdot a = 0$
11. a) $(a+b) \cdot b = a \cdot b$
b) $a \cdot b + b = a+b$
12. a) $a+b \cdot a + c \cdot b + c = a+b \cdot (a + c)$
b) $a \cdot b + a \cdot c + b \cdot c = a \cdot b + a \cdot c$
13. a) $a+b \cdot a + c = a \cdot c + a \cdot b$
b) $a \cdot b + a \cdot c = a + c \cdot (a + b)$
14. a) $f a, b, c, d, \dots = [a + f(0, b, c, d, \dots)] \cdot [a + f(1, b, c, d, \dots)]$
Shannon Teoremi
b) $f a, b, c, d, \dots = a \cdot f 1, b, c, d, \dots + [a \cdot f(0, b, c, d, \dots)]$

- **Boolean Algebra:** rules for rewriting Boolean functions

- Useful for simplifying Boolean functions
 - Simplifying = reducing gate count, reducing gate "levels"
- Rules: similar to logic (0/1 = F/T)
 - **Identity:** $A1 = A, A+0 = A$
 - **0/1:** $A0 = 0, A+1 = 1$
 - **Inverses:** $(A')' = A$
 - **Idempotency:** $AA = A, A+A = A$
 - **Tautology:** $AA' = 0, A+A' = 1$
 - **Commutativity:** $AB = BA, A+B = B+A$
 - **Associativity:** $A(BC) = (AB)C, A+(B+C) = (A+B)+C$
 - **Distributivity:** $A(B+C) = AB+AC, A+(BC) = (A+B)(A+C)$
 - **DeMorgan's:** $(AB)' = A'+B', (A+B)' = A'B'$

- The 12 Rules of Boolean Algebra

- $A + 0 = A$
- $A + 1 = 1$
- $A \cdot 0 = 0$
- $A \cdot 1 = A$
- $A + A = A$
- $A + \bar{A} = 1$
- $A \cdot A = A$
- $A \cdot \bar{A} = 0$
- $\overline{\bar{A}} = A$
- $A + AB = A$
- $A + \bar{A}B = A + B$
- $(A + B)(A + C) = A + BC$

Rules and Laws of Boolean Algebra

- Operations on Boolean variables are defined by rules and laws, the most important of which are presented here
- Commutative Law
$$A \cdot B = B \cdot A$$
$$A + B = B + A$$
- This states that the order of the variables is unimportant
- Associative Law
$$A \cdot (B \cdot C) = A \cdot (B \cdot C)$$
$$A + (B + C) = A + (B + C)$$
- This states that the grouping of the variables is unimportant
- Distributive Law: $A \cdot (B + C) = A \cdot B + A \cdot C$
- This states that we can remove the parenthesis by 'multiplying through'
- The above laws are the same as in ordinary algebra, where '+' and '.' are interpreted as addition and multiplication

Rules and Laws of Boolean Algebra

- Basic rules involving one variable:

$$A + 0 = A \quad A \cdot 0 = 0$$

$$A + 1 = 1 \quad A \cdot 1 = A$$

$$A + A = A \quad A \cdot A = A$$

$$A + A' = 1 \quad A \cdot A' = 0$$

- It should be noted that $A'' = A$
- An informal proof of each of these rules is easily accomplished by taking advantage of the fact that the variable can have only two possible values
- For example, rule 2: $A + 1 = 1$
 - If $A = 0$ then $0 + 1 = 1$
 - If $A = 1$ then $1 + 1 = 1$

Rules and Laws of Boolean Algebra

Basic rules of single variable

- A proof of each of the rules and laws of Boolean algebra can be easily proved by taking advantage of the fact that a variable can only have two bits (0/1) of value.
- Note: $A=0$ or 1 .
- $A + A + A + A + A \dots + A + 1 = 1$; In the OR gate, if any of the inputs is 1, the output is one. The other method of proof is to search for accuracy by giving values of 1 and 0.
- $A + A + A + \dots + A = A$ (Why? Bivariate 0 or 1 inputs are available)
- $AAA \dots A = A$
 - If $A = 0$ then $0 + 1 = 1$
 - If $A = 1$ then $1 + 1 = 1$

Rules and Laws of Boolean Algebra

$$\begin{array}{ll} A + 0 = A & A \cdot 0 = 0 \\ A + 1 = 1 & A \cdot 1 = A \\ A + A = A & A \cdot A = A \\ A + \bar{A} = 1 & A \cdot \bar{A} = 0 \end{array}$$

It should be noted that $\overline{\bar{A}} = A$

In logic circuits and mathematics, there are two numbers: 0 and 1.

Question: Perform the following operation using Boolean algebra. $A=9$, $A+1=?$

a)0 b)1 c)10 d) 8 e)none

Question: What values does A take in Boolean algebra?

a) 0 b)1 c) 0/1 d) 0,1,2, ..., 9 d)Any value e)No value

Question: If $A=1$ in Boolean algebra, $A+A+A+A+A=?$

A)1 B)0 C)A D)5 D)5A E) None

Question: In Boolean Algebra, $A*A*A*A=?$ A)A B) A^4

DeMorgan's Laws

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

A	B	A + B	$\overline{A + B}$	\bar{A}	\bar{B}	$\bar{A} \cdot \bar{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

A	B	A · B	$\overline{A \cdot B}$	\bar{A}	\bar{B}	$\bar{A} + \bar{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

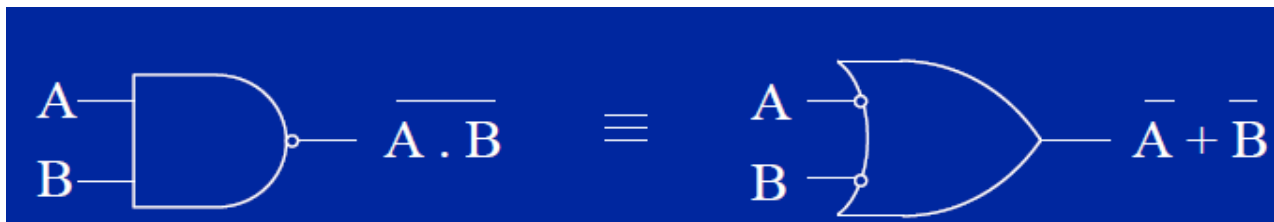
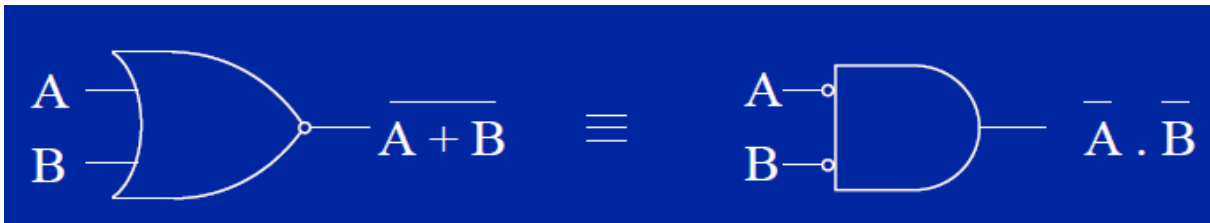


Rules and Laws of Boolean Algebra

De Morgan's Laws are particularly useful when dealing with NAND and NOR logic.

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$



Some useful theorems

- $A+A.B=A.(1+B)=A$, $1+B=1$
- $A+A'.B=(A+A').(A+B)=A+B$, $A+A'=1$
- $A.B+AB'=A(B+B')=A$, $B+B'=1$
- $A.(A+B)=A.A+AB=A+AB=A(1+B)=A$, $A.A=A$, $1+B=1$
- $A(A'+B)=A.A'+AB=AB$; $A.A'=0$
- $(A+B)(A+B')=AA+AB'+AB+BB'=A+AB'+AB=A(1+B+B')=A$; $A.A=1$, $B.B'=0$, $1+B+B'=1$
- $A + A'.B = A+B$

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

The result of logical operations is always 1 or 0.

- $a + a + a + a + \dots + a = a$
- $a * a * a * \dots * a = a$
- $1 + a + b + c + \dots + z = 1$
- $ab'c + ab'c = ab'c$; The sum of similar expressions always equals a similar one.

Boolean Algebra

- Developed by George Boole in 19th Century
 - Algebraic representation of logic
 - Encode “True” as 1 and “False” as 0

And

- $A \& B = 1$ when both $A=1$ and $B=1$

$\&$	0	1
0	0	0
1	0	1

Not

- $\sim A = 1$ when $A=0$

\sim	
0	1
1	0

Or

- $A | B = 1$ when either $A=1$ or $B=1$

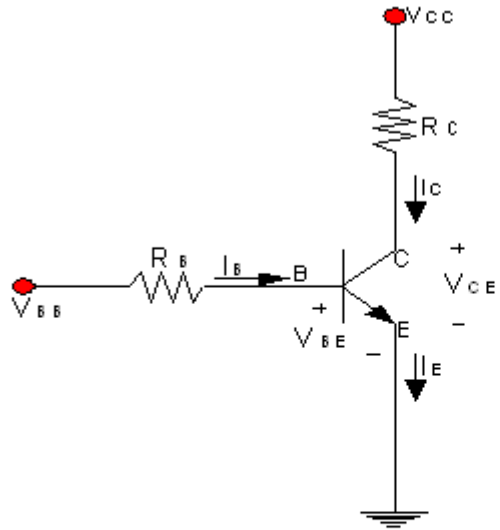
	0	1
0	0	1
1	1	1

Exclusive-Or (Xor)

- $A \wedge B = 1$ when either $A=1$ or $B=1$, but not both

\wedge	0	1
0	0	1
1	1	0

Gates and Transistors



$$V_{CC} = R_C * I_C + V_{CE}$$

$$V_{BB} = R_B * I_B + V_{BE}$$

$$I_C = \beta * I_B$$

$$I_{C\ SAT} = \frac{V_{CC}}{R_C}$$

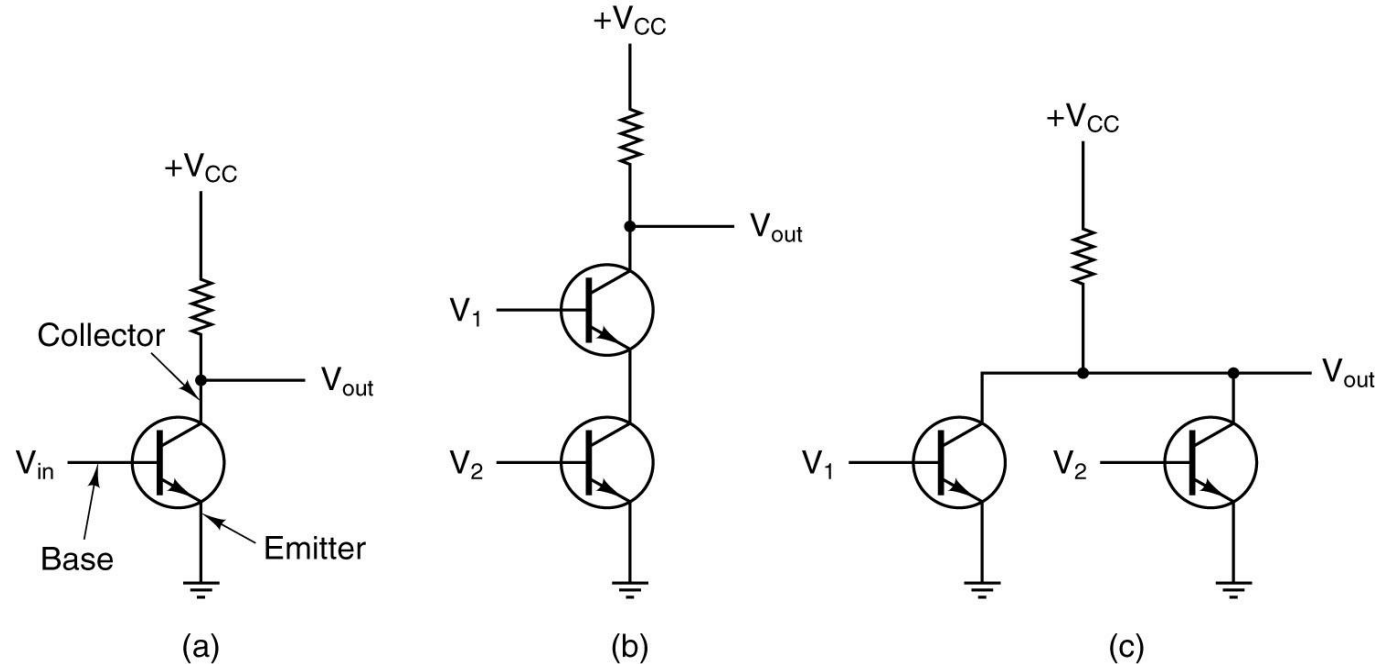
$$I_B = \frac{V_{BB} - V_{BE}}{R_B}$$

$V_{CE} \leq 0V$ ya da $I_c \geq I_{c\ SAT}$; Saturasyon $V_{CE} = 0V$ Olur

$I_B \leq 0A$ ise ; Kesmede $I_B = I_C = 0A$ Olur

VCE=VCC kesme durumunda

Transistor is a circuit element produced in semiconductor technology that controls the flow of electrons.



(a) A transistor inverter.

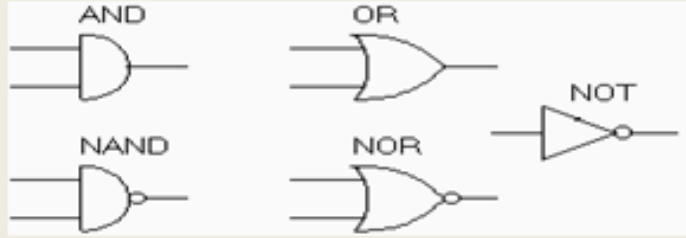
(b) A NAND gate.

(c) A NOR gate.

Transistor

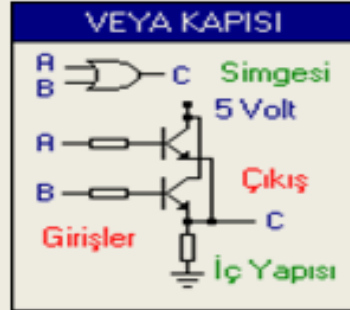
- Semiconductor circuit element that controls the flow of electrons.
- Subatomic particles (Quantum Mechanics): Proton, Neutron, Electron, Photon
- Current is created from the flow of electrons.
- The transistor memory element stores the bit (0/1) state on it. Performs Switching. Or it strengthens the signal.
- Transistor is the most used electronic circuit element in the world.
- The smallest basic electronic circuit element of a microprocessor is the transistor.
- The CPU's basic function cycles occur at a dizzying speed as transistors turn on and off millions or even billions of times per second..

Gates



Doğruluk tablosu:

A	B	OR $A+B$	AND $A*B$	NOT A'	NOR $(A+B)'$	NAND $(A*B)'$	EXOR $(A')*B+A*(B')$
0	0	0	0	1	1	1	0
0	1	1	0	1	0	1	1
1	0	1	0	0	0	1	1
1	1	1	1	0	0	0	0



Formüller	0 Değeri Verildiğinde	1 Değeri Verildiğinde
$A \cdot 0 = 0$	A = 0 ise, $0 \cdot 0 = 0$	A = 1 ise, $1 \cdot 0 = 0$
$A \cdot 1 = A$	A = 0 ise, $0 \cdot 1 = 0$	A = 1 ise, $1 \cdot 1 = 1$
$A + 0 = A$	A = 0 ise, $0 + 0 = 0$	A = 1 ise, $1 + 0 = 1$
$A + 1 = 1$	A = 0 ise, $0 + 1 = 1$	A = 1 ise, $1 + 1 = 1$
$A \cdot A = A$	A = 0 ise, $0 \cdot 0 = 0$	A = 1 ise, $1 \cdot 1 = 1$
$A + A = A$	A = 0 ise, $0 + 0 = 0$	A = 1 ise, $1 + 1 = 1$
$A \cdot A' = 0$	A = 0 ise, $0 \cdot 1 = 0$	A = 1 ise, $1 \cdot 0 = 0$
$A + A' = 1$	A = 0 ise, $0 + 1 = 1$	A = 1 ise, $1 + 0 = 1$
$(A')' = A$	A = 0 ise, A' = 1, $(A')' = 0$	A = 1 ise, A' = 0, $(A')' = 1$

Sadeleştirmeler

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

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

$$(A \cdot B) \cdot C = A \cdot (B \cdot C) = A \cdot B \cdot C$$

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

$$(A' \cdot B) + (A \cdot B') = A \oplus B$$

$$(A + B)' = A' \cdot B'$$

$$(A \cdot B) = (B \cdot A)$$

$$(A' \cdot B') + (A \cdot B) = (A \oplus B)'$$

$$(A \cdot B)' = A' + B'$$

Binary Logic

- ♣ Binary logic consists of binary variables and a set of logical operations.
- ♣ The variables are designated by letters of the alphabet, such as A, B, C, x, y, z , etc, with each variable having two and only two distinct possible values: 1 and 0.
- ♣ There are three basic logical operations: AND, OR, and NOT.

AND: represented by a dot or by the absence of an operator.

- $x \cdot y = z$ or $xy = z$ is read “ x AND y is equal to z .”
- $z = 1$ if and only if $x = 1$ and $y = 1$; otherwise $z = 0$. (Remember that x, y , and z are binary variables and can be equal either to 1 or 0, and nothing else.)

OR: represented by a plus sign.

- $x + y = z$ is read “ x OR y is equal to z ,” meaning that $z = 1$ if $x = 1$ or if $y = 1$ or if both $x = 1$ and $y = 1$.
- If both $x = 0$ and $y = 0$, then $z = 0$.

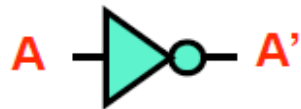
NOT: represented by a prime (sometimes by an overbar).

- $x' = z$ (or $x = \bar{z}$) is read “not x is equal to z ,” meaning that z is what x is not.
- If $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$.
- also referred to as the *complement* operation, since it changes a 1 to 0 and a 0 to 1.

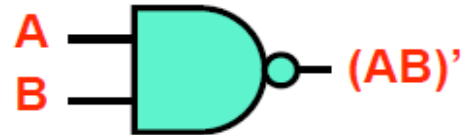
Logic Gates

- **Logic gates:** implement Boolean functions
 - Basic gates: NOT, NAND, NOR
 - Underlying CMOS transistors are naturally inverting (● = NOT)

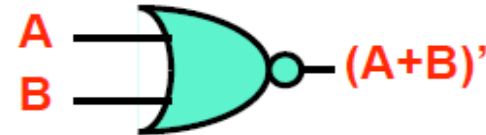
NOT (INV)



NAND

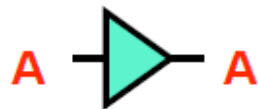


NOR



- NAND, NOR are "Boolean complete"

BUF



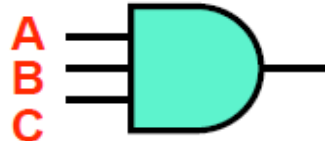
AND



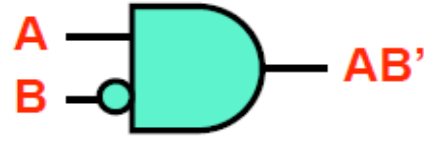
OR



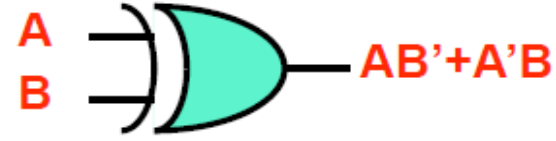
AND3



ANDNOT



XOR



Basic Logic Gates

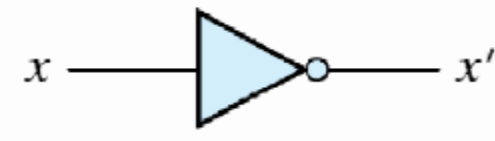
♣ Graphic Symbols and Input-Output Signals for Logic gates:



(a) Two-input **AND** gate

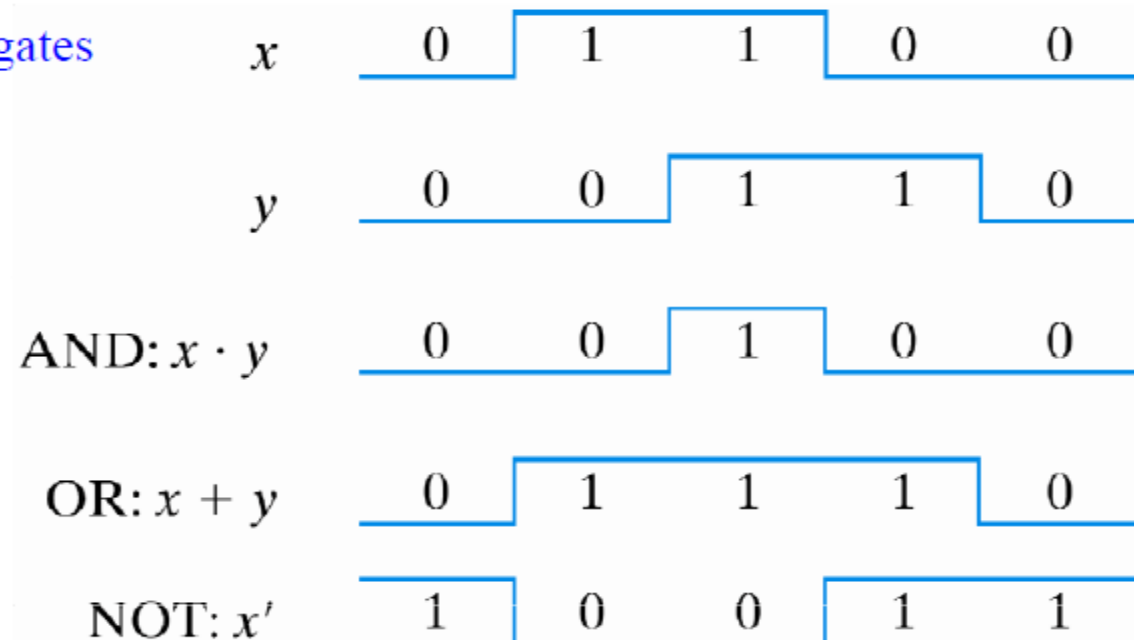
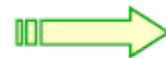


(b) Two-input **OR** gate



(c) **NOT** gate or inverter

➤ Input-Output signals for logic gates



Temel Lojik Kapılar -1

- Simple gates
 - AND
 - OR
 - NOT
- Functionality can be expressed by a truth table
 - A truth table lists output for each possible input combination
- Precedence
 - NOT > AND > OR
 - $F = A B + A \bar{B}$
 $= (A (B)) + ((A) \bar{B})$



Gate

Symbol

Truth-Table

Expression

NAND



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

$$Z = \overline{X \cdot Y}$$

AND



X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

$$Z = X \cdot Y$$

NOR



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

$$Z = \overline{X + Y}$$

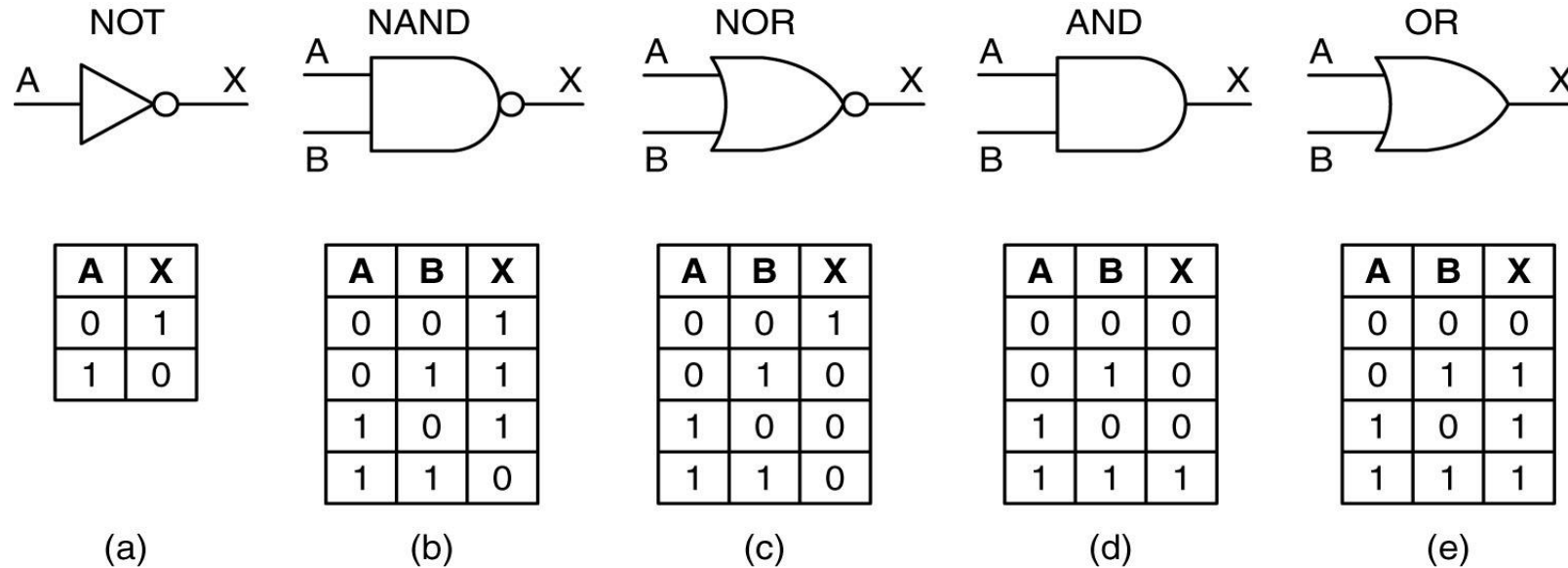
OR



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

$$Z = X + Y$$

Symbols and functional behavior for Logic Gates



AND Gate: If any of the inputs is 0, the output is 0. If all inputs are 1, the output is 1.

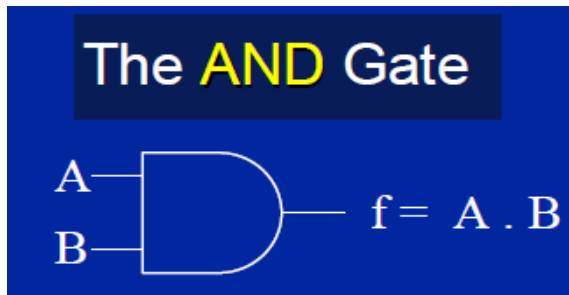
OR Gate: If any of the inputs is 1, the output is 1. If all inputs are 0, the output is 0.

NOT Gate: transposes the input.

A logic gate is an idealized or physical circuit that implements a Boolean function, that is, it performs a logical operation on one or more logic inputs and produces a single logic output.

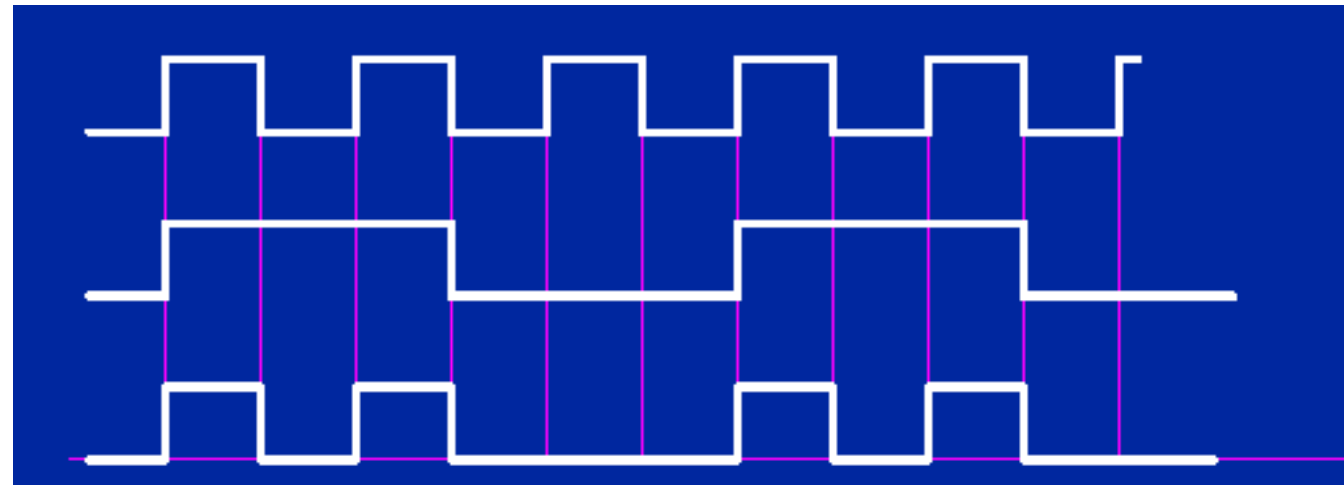
Logic AND Gates

- Logic gates are switching circuits that perform certain simple operations on binary signals
- These operations are chosen to facilitate the implementation of useful functions
 - The AND Gate - Determine the output waveform when the input waveforms A and B are applied to the two inputs of an AND gate
 - A and B are variables and note the use of the . to denote AND
 - Giriş dalga formları A ve B bir mantık kapısının iki girişine uygulandığında çıkış dalga formu belli ise bu kapının türünü belirleyiniz. (AND Kapısı)



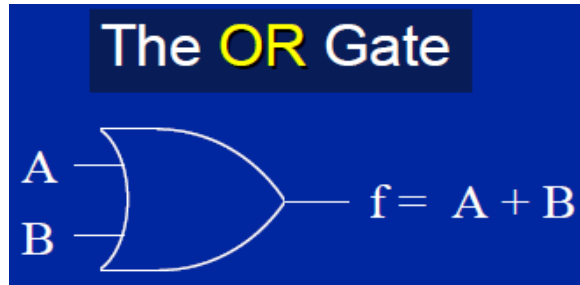
The **AND** Truth Table

A	B	f = A AND B
0	0	0
0	1	0
1	0	0
1	1	1



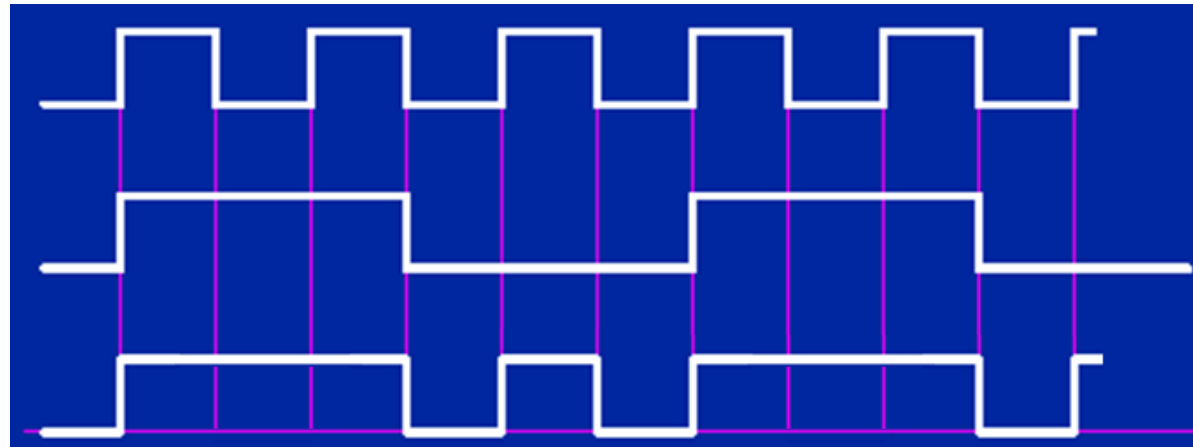
Logic OR Gates

- A and B are variables and note the use of the + to denote OR



The **OR** Truth Table

A	B	f = A OR B
0	0	0
0	1	1
1	0	1
1	1	1



Logic NOT Gates

- Note the use of the bar over the A to denote NOT

The NOT Gate

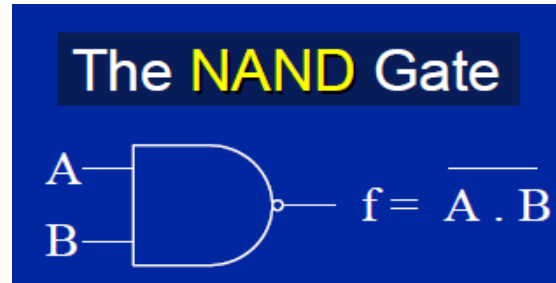
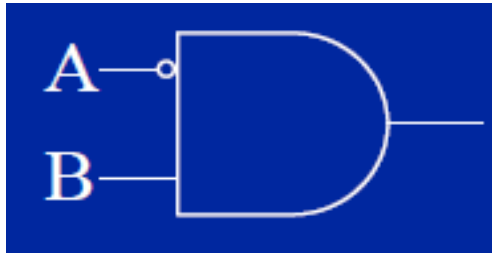


The NOT Truth Table

A	f = NOT A
0	1
1	0

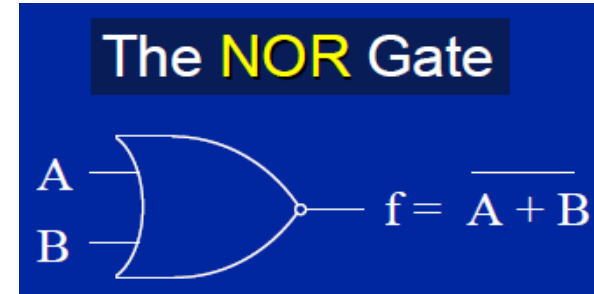
Logic Gates

- Sometimes a 'bubble' is used to indicate Inversion
- In fact it is simpler to manufacture the combination NOT AND and NOT OR than it is to deal with AND and OR
- NOT AND becomes NAND
- NOT OR becomes NOR



The **NAND** Truth Table

A	B	$f = A \text{ NAND } B$
0	0	1
0	1	1
1	0	1
1	1	0



The **NOR** Truth Table

A	B	$f = A \text{ NOR } B$
0	0	1
0	1	0
1	0	0
1	1	0

Logic Gates

Toplama

The **EXCLUSIVE OR** Truth Table

A	B	f = A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Karşılaştırma

The **EXCLUSIVE NOR** Truth Table

A	B	f = A XOR B
0	0	1
0	1	0
1	0	0
1	1	1

The **XOR** Gate



EXCLUSIVE NOR Gate



This is called the equivalence gate

XOR gates are used in comparison and arithmetic addition operations. If all inputs are equal (0 or 1) and the output is zero, it is an XOR gate; if the output is 1, it is an XNOR gate.

The XOR – XNOR Gates

- The **EXCLUSIVE OR** Truth Table

$$f = A \text{ XOR } B$$

$$= A \oplus B$$

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

A	B	f = A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

- The **EXCLUSIVE NOR** Truth Table

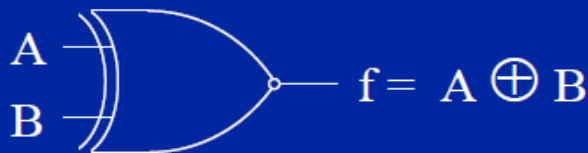
$$f = \text{NOT } (A \text{ XOR } B)$$

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

$$= \overline{\bar{A}B + AB}$$

A	B	f = A XOR B
0	0	1
0	1	0
1	0	0
1	1	1

The **XOR** Gate



The **EXCLUSIVE NOR** Gate



Örnek:

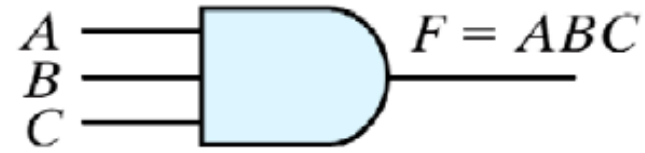
- Sound the alarm when A=1 and B=1 or C=1 and D=1.

$$F=AB+CD$$

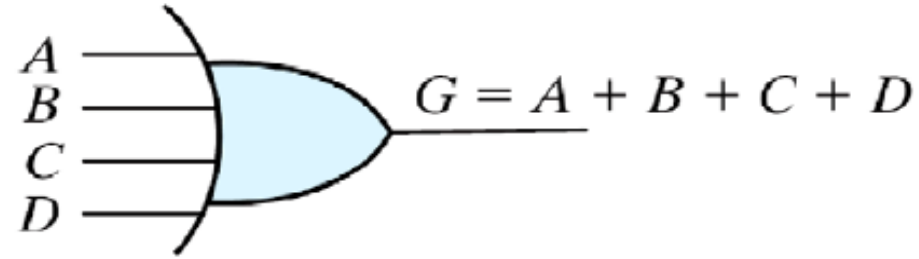
In case of AB: m12, m13, m14, m15 In case of CD: m3, m7, m11, m15

Row	A	B	C	D		Minterm
0	0	0	0	0	$\bar{A}\bar{B}\bar{C}\bar{D}$	m0
1	0	0	0	1	$\bar{A}\bar{B}\bar{C}D$	m1
2	0	0	1	0	$\bar{A}\bar{B}C\bar{D}$	m2
3	0	0	1	1	$\bar{A}\bar{B}CD$	m3
4	0	1	0	0	$\bar{A}B\bar{C}\bar{D}$	m4
5	0	1	0	1	$\bar{A}B\bar{C}D$	m5
6	0	1	1	0	$\bar{A}BC\bar{D}$	m6
7	0	1	1	1	$\bar{A}BCD$	m7
8	1	0	0	0	$A\bar{B}\bar{C}\bar{D}$	m8
9	1	0	0	1	$A\bar{B}\bar{C}D$	m9
10	1	0	1	0	$A\bar{B}C\bar{D}$	m10
11	1	0	1	1	$A\bar{B}CD$	m11
12	1	1	0	0	$AB\bar{C}\bar{D}$	m12
13	1	1	0	1	$AB\bar{C}D$	m13
14	1	1	1	0	$ABC\bar{D}$	m14
15	1	1	1	1	$ABCD$	m15

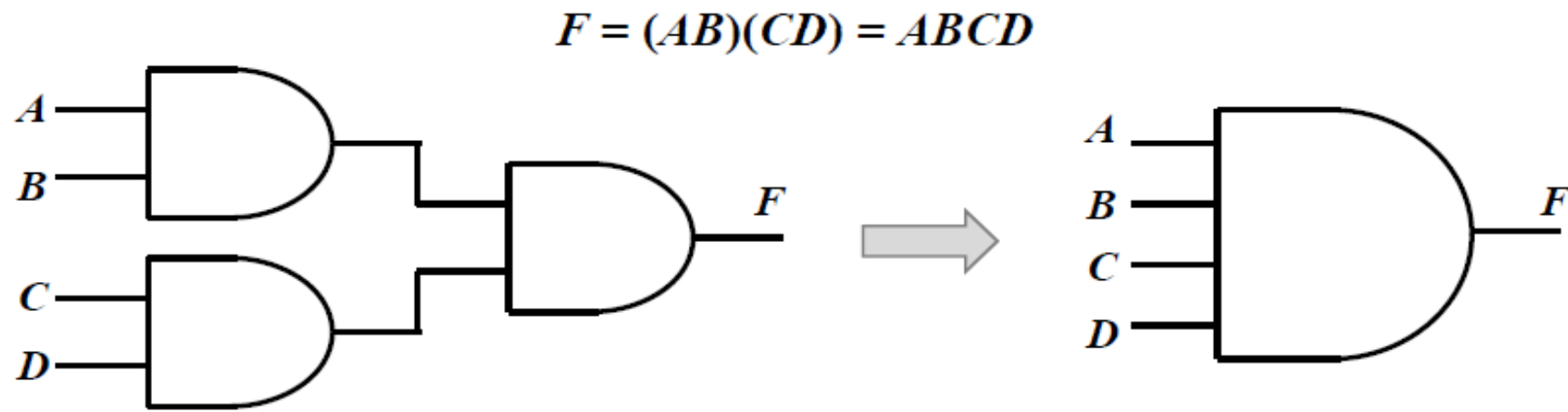
Multiple-Input Gates



(a) Three-input AND gate

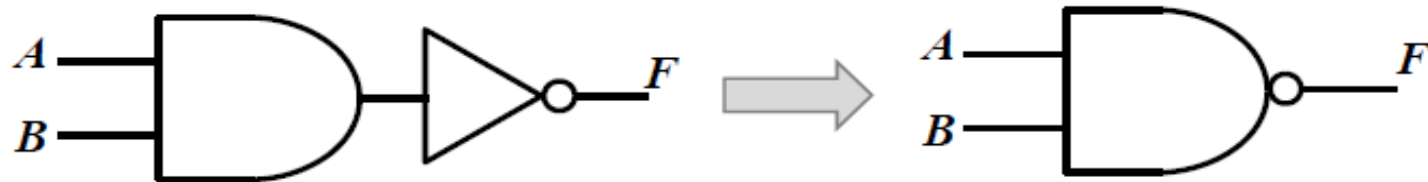


(b) Four-input OR gate



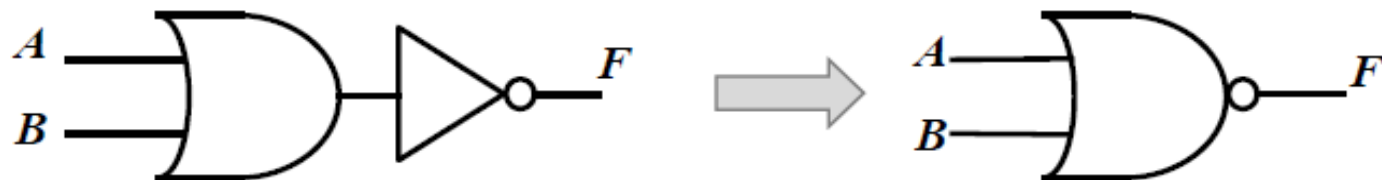
Inverting Gates

NOT + AND = NAND



A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

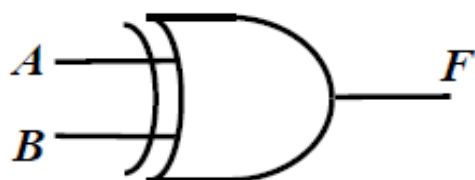
NOT + OR = NOR



A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

Exclusive OR/NOR Gates

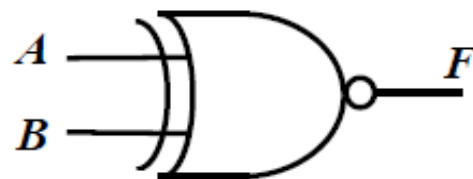
XOR



$$F = A \oplus B$$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

XNOR



$$F = A \odot B$$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Truth Table of Logic Operation

Truth Tables of Logical Operations

AND			OR			NOT	
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

A logic variable is always either 1 or 0.

Logic Functions

- **Logical functions can be expressed in several ways:**

- Truth Table
- Logical Expression
- Impression

- **Logical expression form**

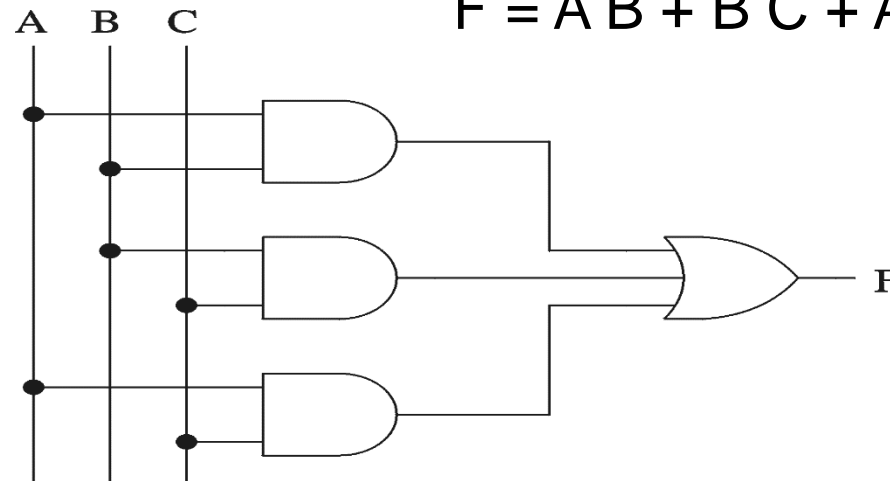
$$F = A'BC + AB'C + ABC' + ABC$$

$$F = AB + BC + AC$$

- The sum of the variables in the given equations gives the number of inputs.
- In logic gates, the result of a logical equation is 1 or 0.

Truth Table: 3-input majority function

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

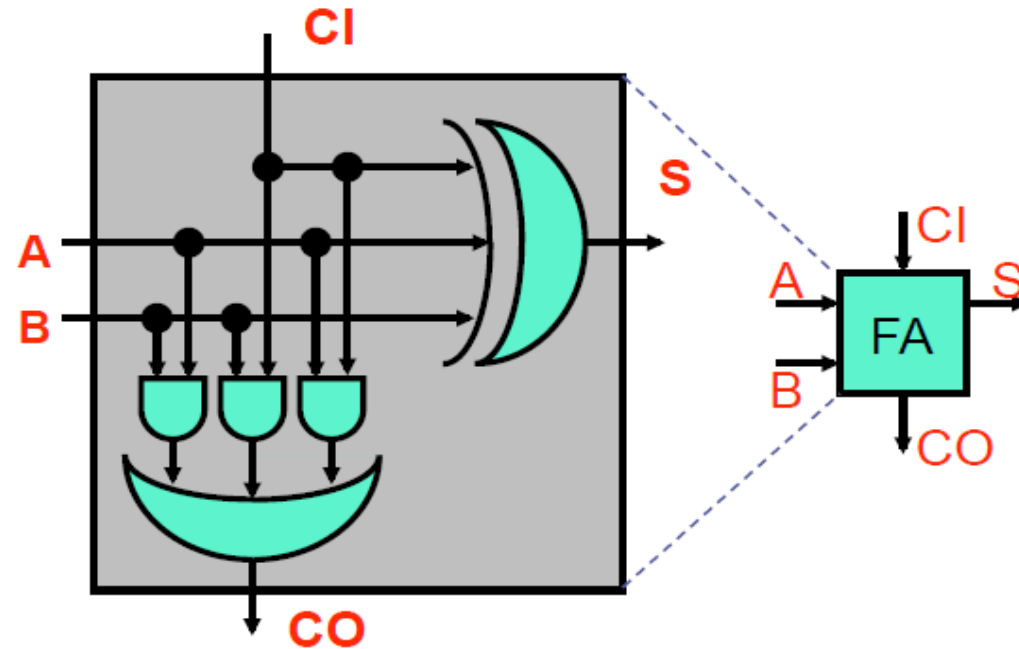


If the number of entries is m, how many different states are there? There are $S=2^m$ states. The reason why it has base 2 is due to the binary number system: 0 or 1, bit

Full Adder

- What is the logic for a full adder?
 - Look at truth table

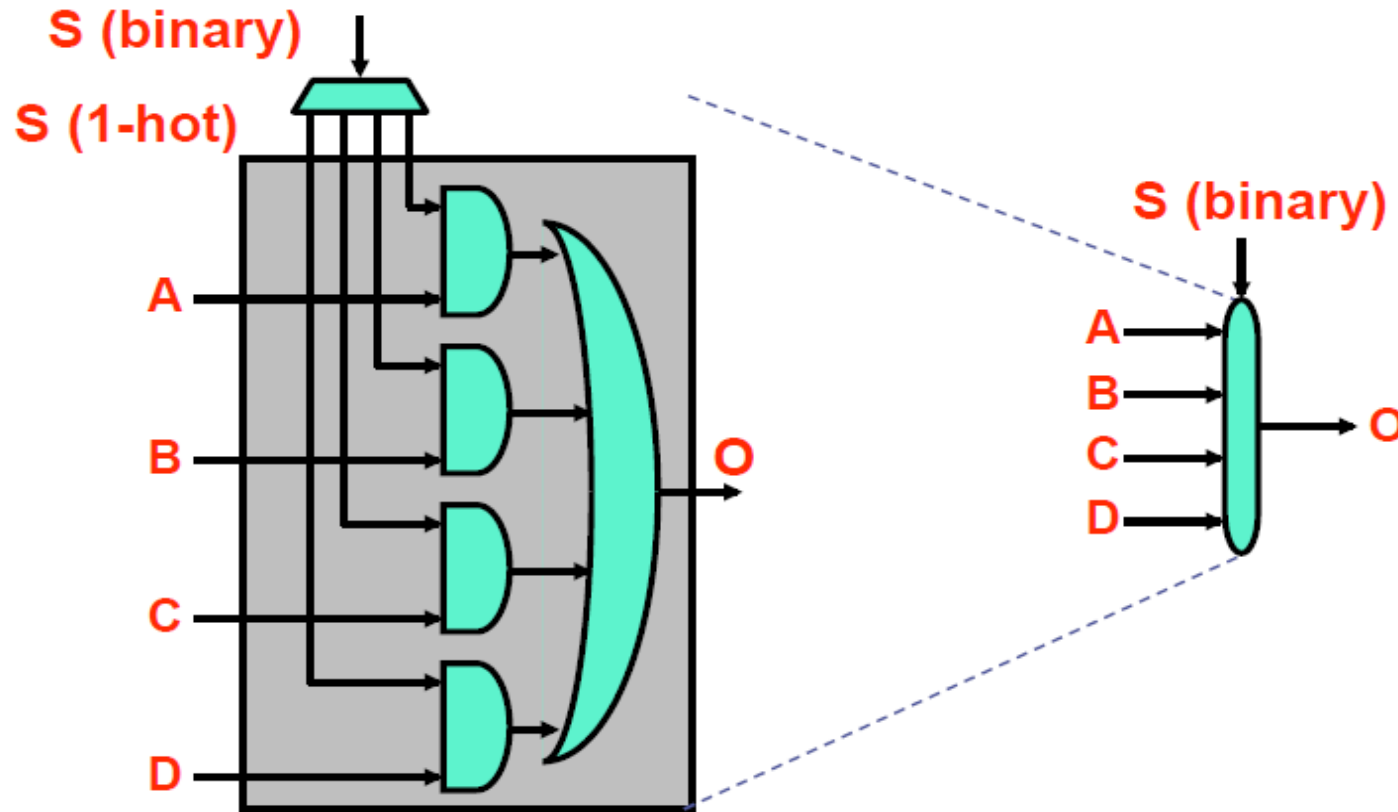
CI	A	B	→	CO	S
0	0	0	→	0	0
0	0	1	→	0	1
0	1	0	→	0	1
0	1	1	→	1	0
1	0	0	→	0	1
1	0	1	→	1	0
1	1	0	→	1	0
1	1	1	→	1	1



- **$S = C'A'B + C'AB' + CA'B' + CAB = C \wedge A \wedge B$**
- **$CO = C'AB + CA'B + CAB' + CAB = CA + CB + AB$**

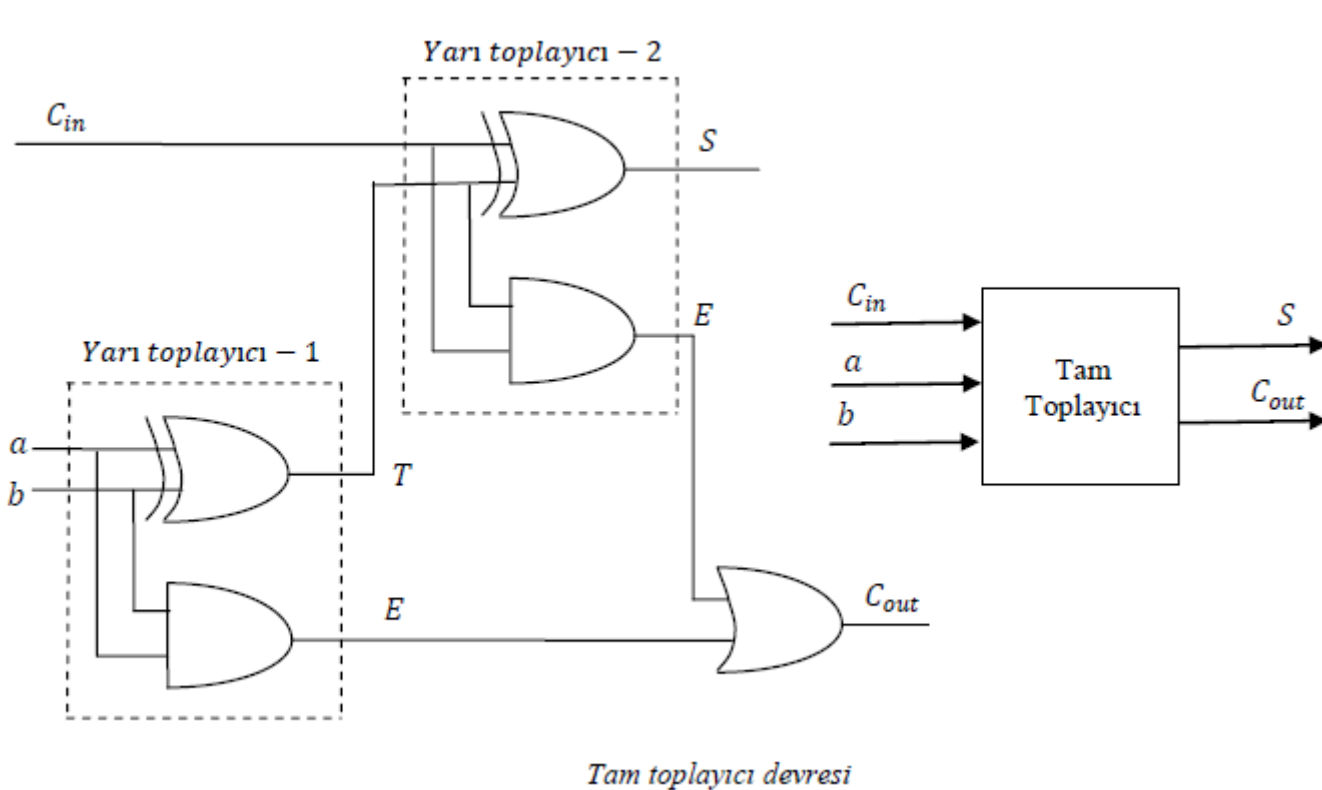
Multiplexer (mux): selects output from N inputs

- Example: 1-bit 4-to-1 mux
- Not shown: N-bit 4-to-1 mux = N 1-bit 4-to-1 muxes + 1 decoder



Tam toplayıcı (Full Adder)

- It is a combinational circuit where the carry bit at the input is added together with two one-bit numbers.



C_{in}	a	b	Toplam S	Elde C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

ab C_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

ab C_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

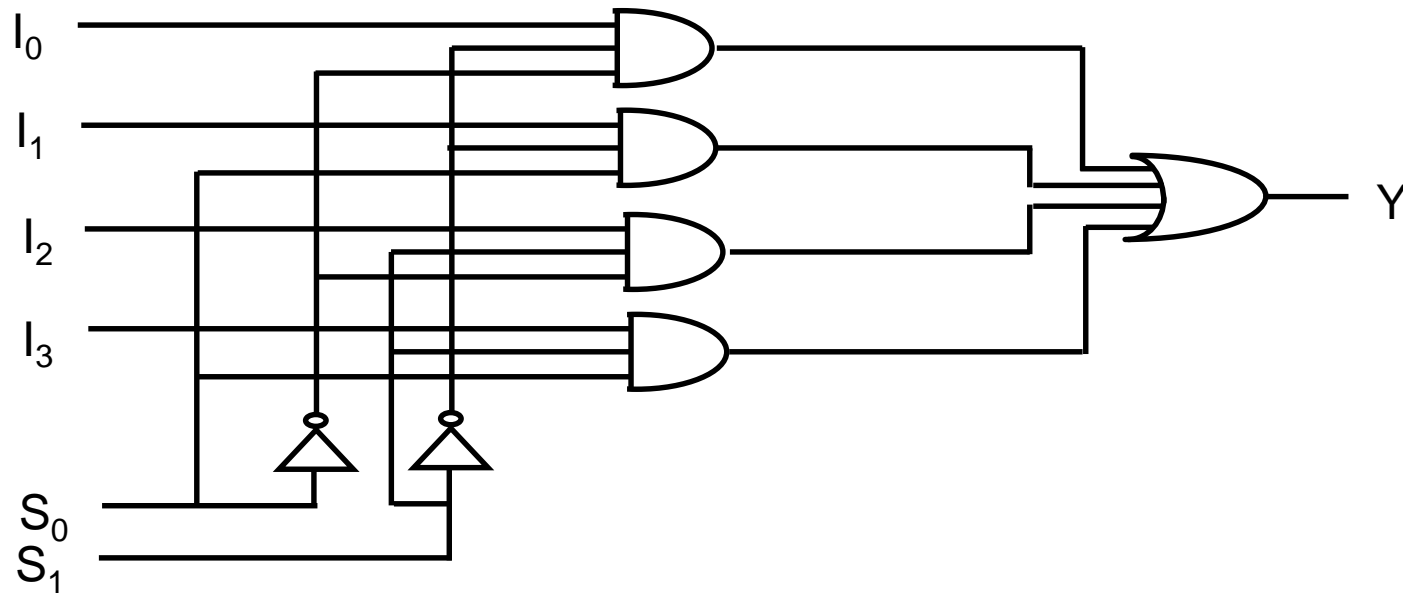
$$T = C_{in}\bar{a}\bar{b} + \bar{C}_{in}\bar{a}b + C_{in}ab + \bar{C}_{in}a\bar{b} = \bar{C}_{in}(\bar{a}b + a\bar{b}) + C_{in}(ab + \bar{a}\bar{b}) \Rightarrow T = a \oplus b \oplus C_{in}$$

$$C_{out} = C_{in}b + C_{in}a + ab$$

MULTIPLEXER

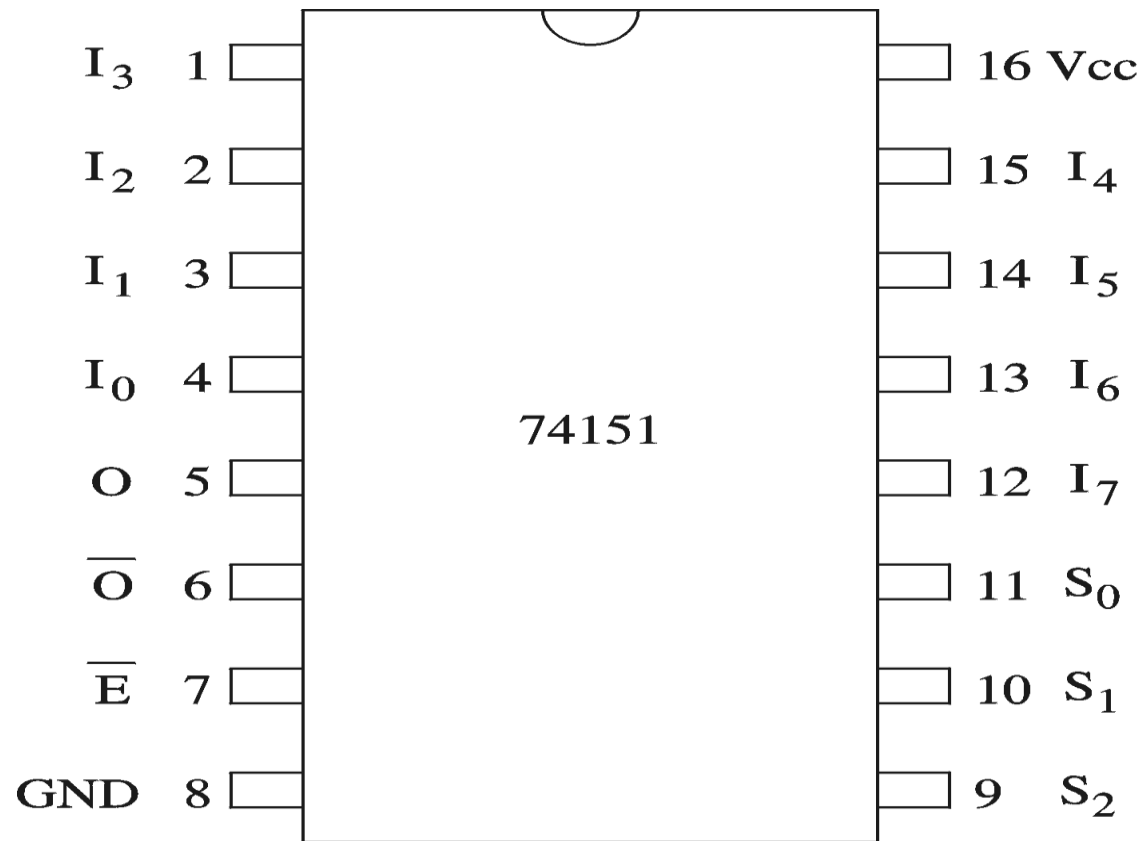
4-to-1 Multiplexer

Select		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

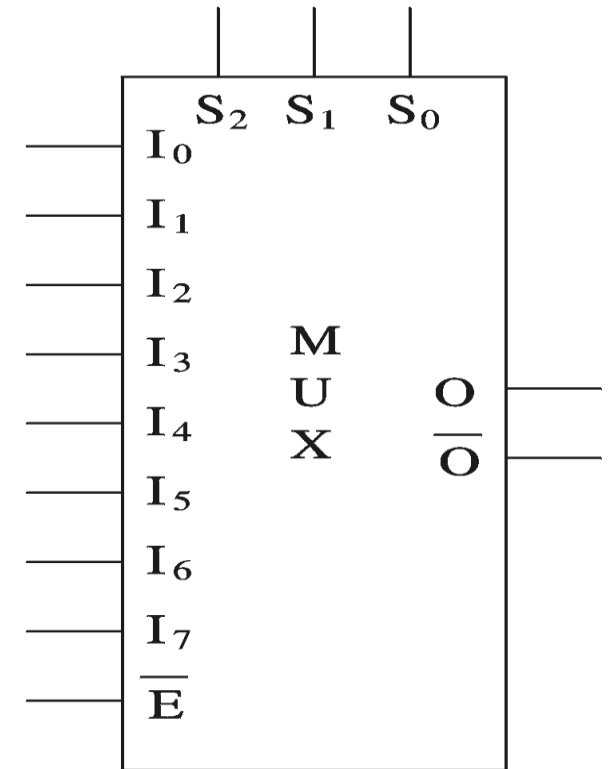


Multiplexers

Example chip: 8-to-1 MUX



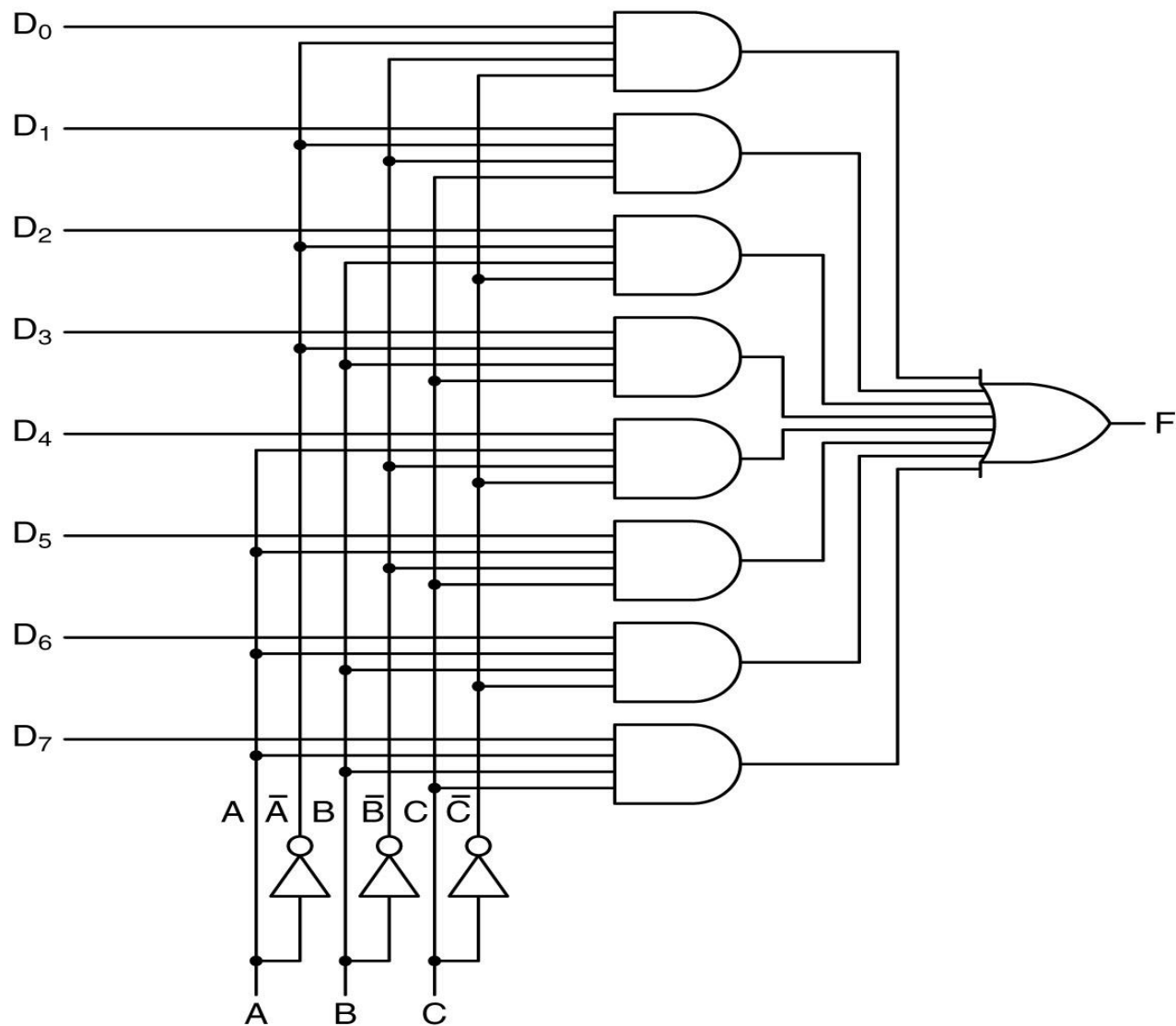
(a) Connection diagram



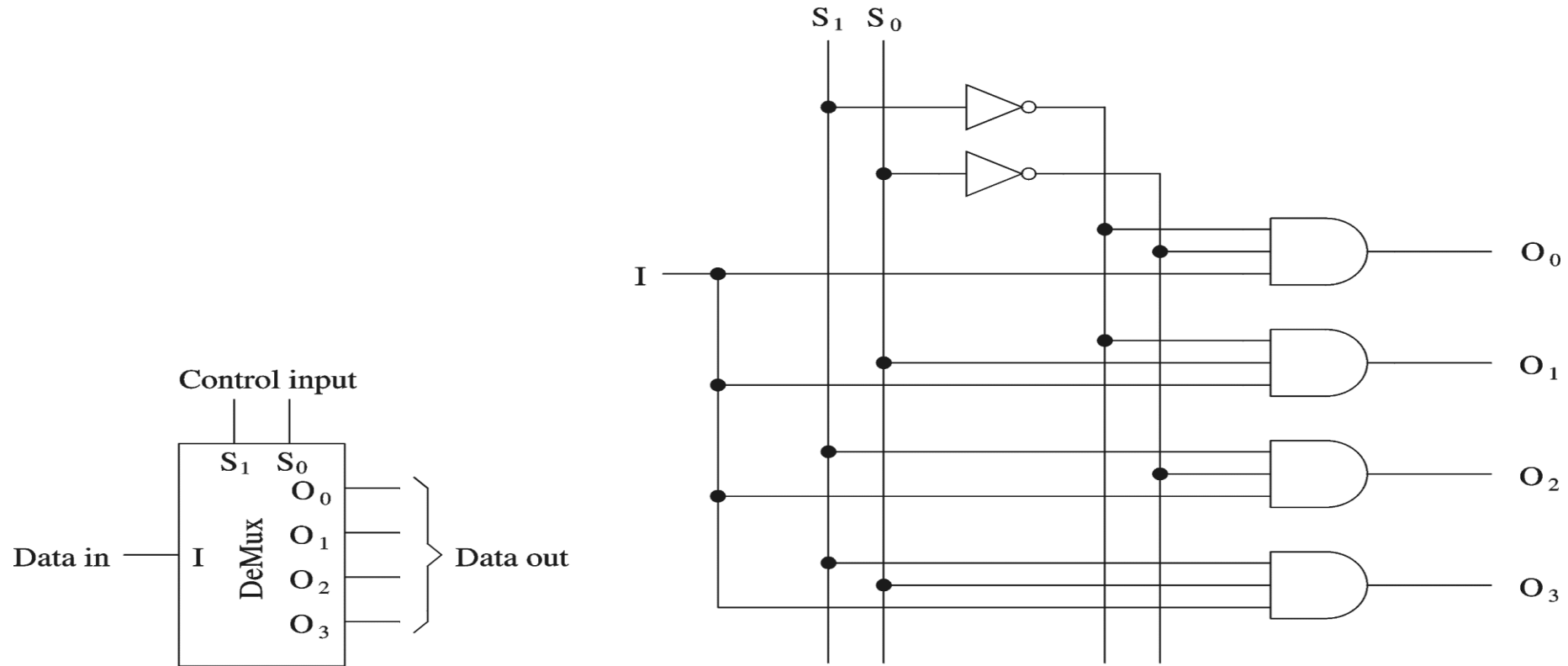
(b) Logic symbol

Multiplexers

An eight-input multiplexer circuit.



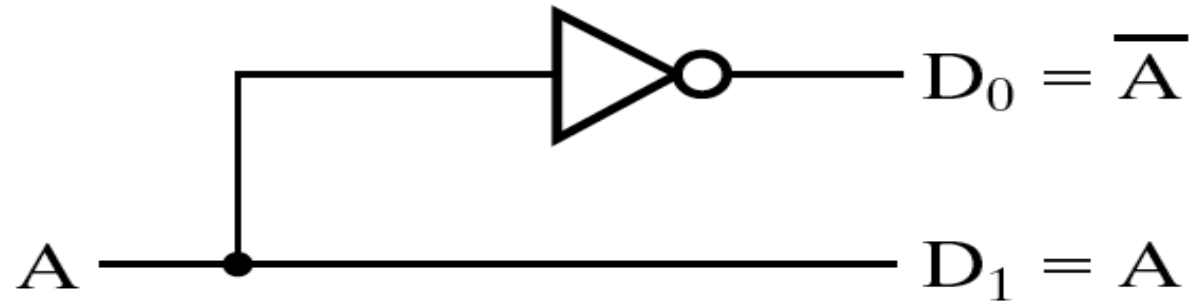
Demultiplexer (DeMUX)



1-2 Decoder

A	D₀	D₁
0	1	0
1	0	1

(a)

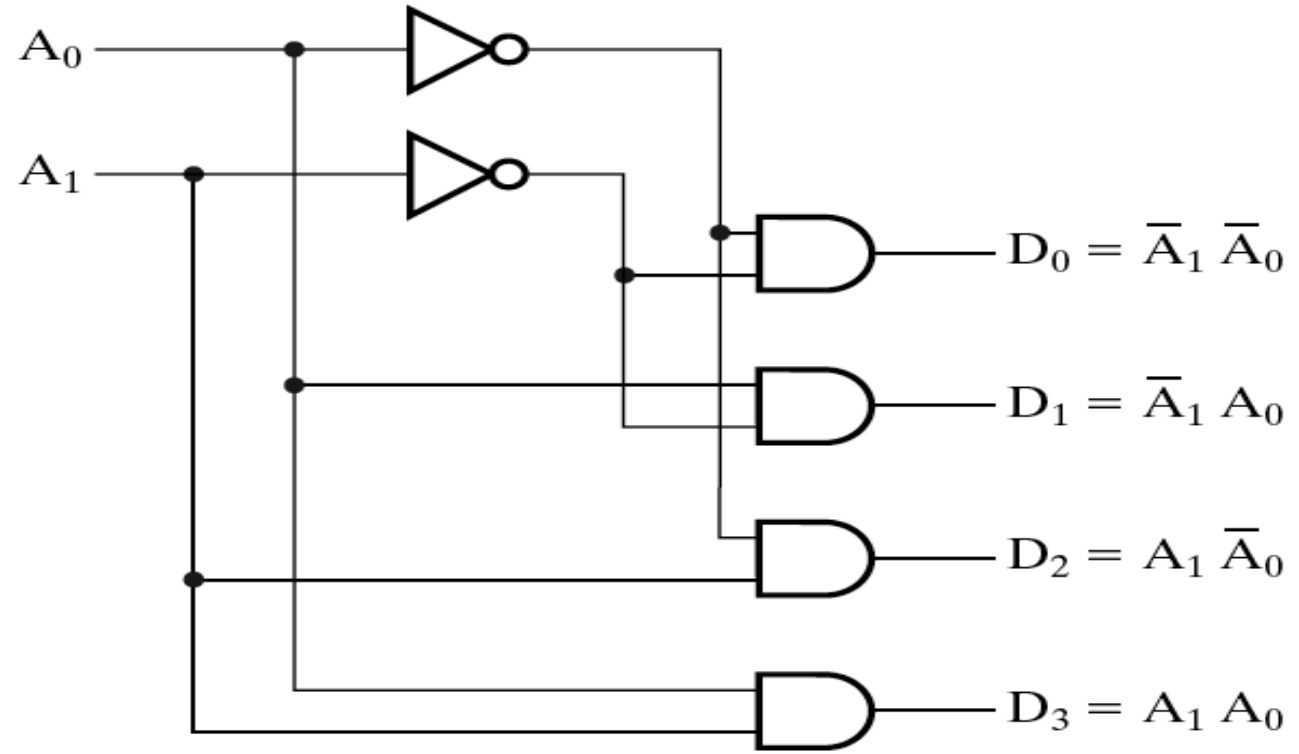


(b)

2-to-4 Decoder

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

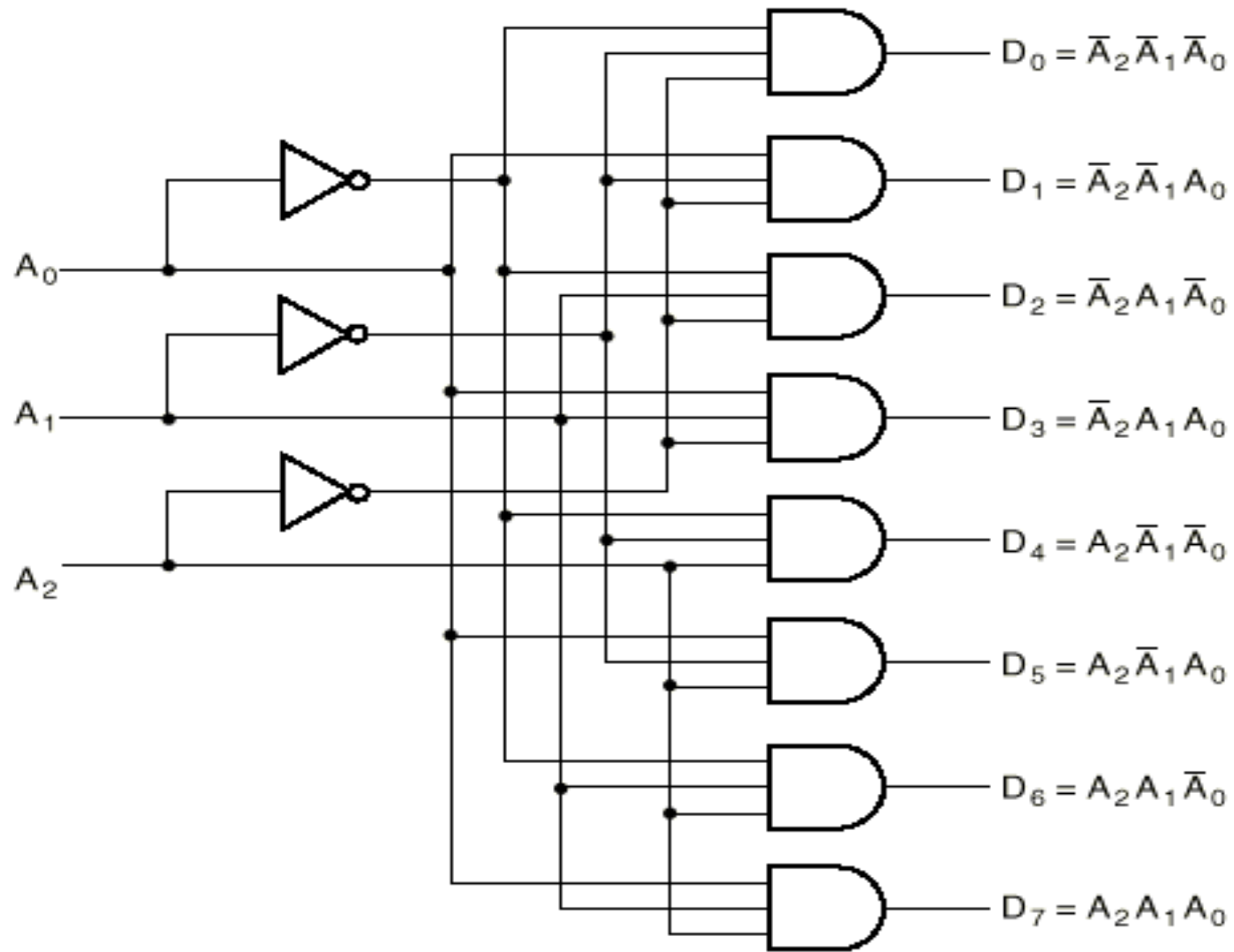
(a)



(b)

3-to-8 Decoder

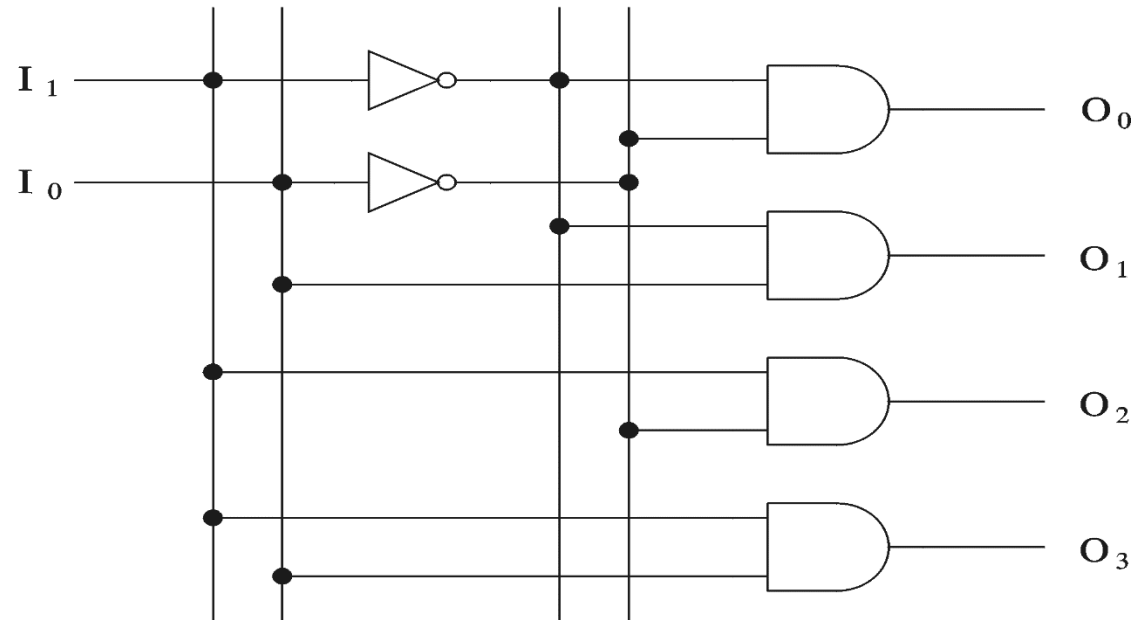
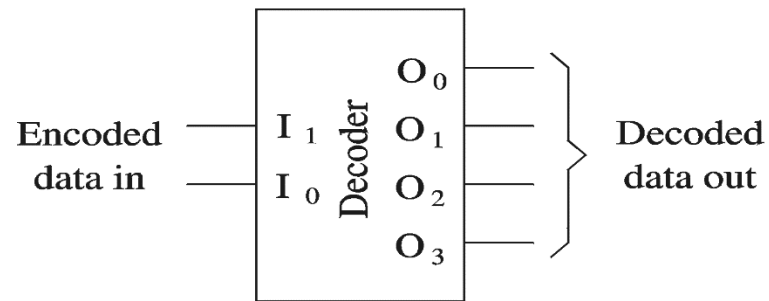
- Decoding circuit is used to select memory.



Decoders

- Decoder selects one-out-of-N inputs

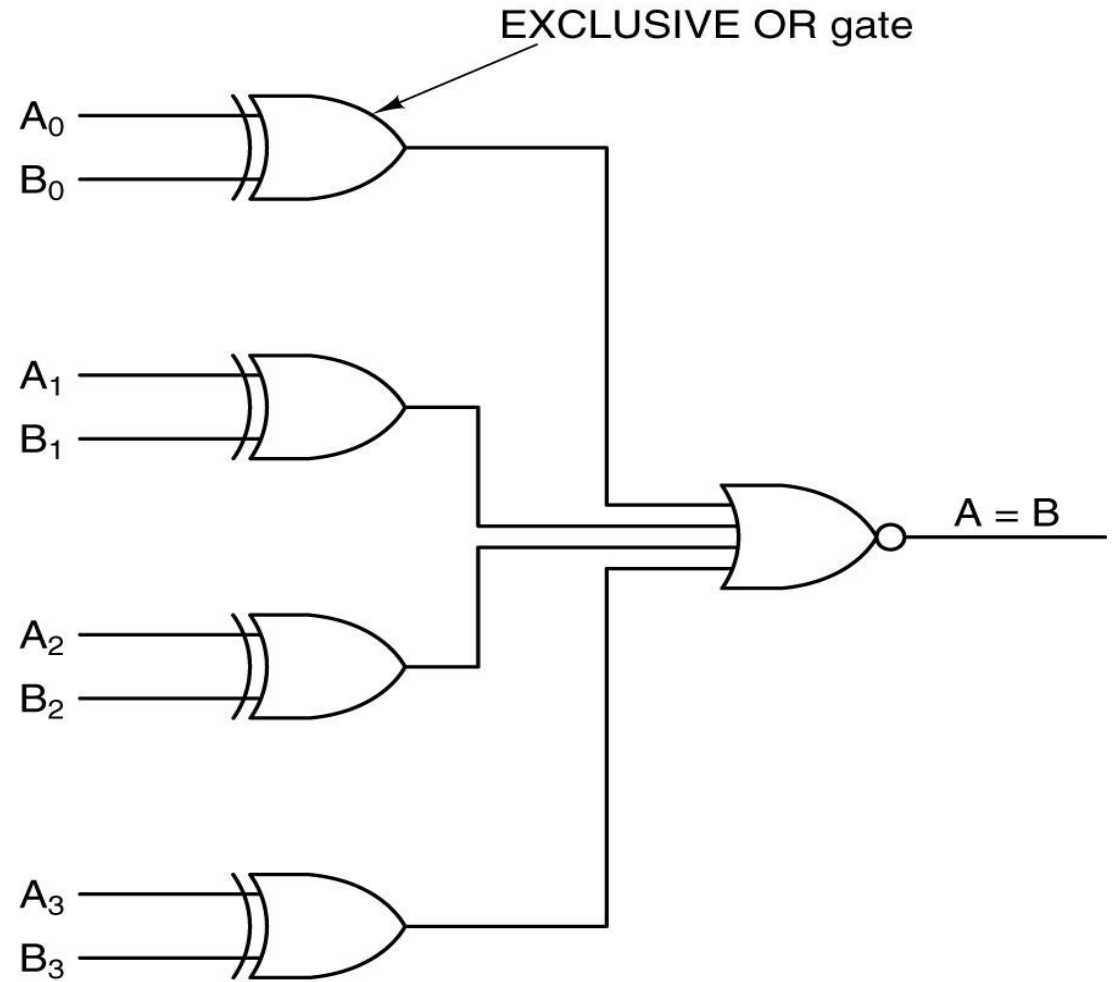
I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Comparators

A simple 4-bit comparator.

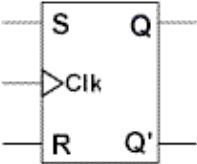
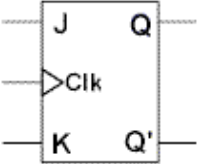

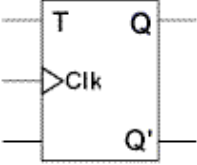
Used to implement comparison operators ($=$, $>$, $<$, \geq , \leq)



Sıralı Mantık (Sequential Logic)

- Sequential logic has memory; The circuit stores the result of the previous set of inputs. The output depends on current inputs as well as past inputs.
- The basic element in sequential logic is the two-state latch or flip-flop circuit that serves as the memory element for one bit of data.

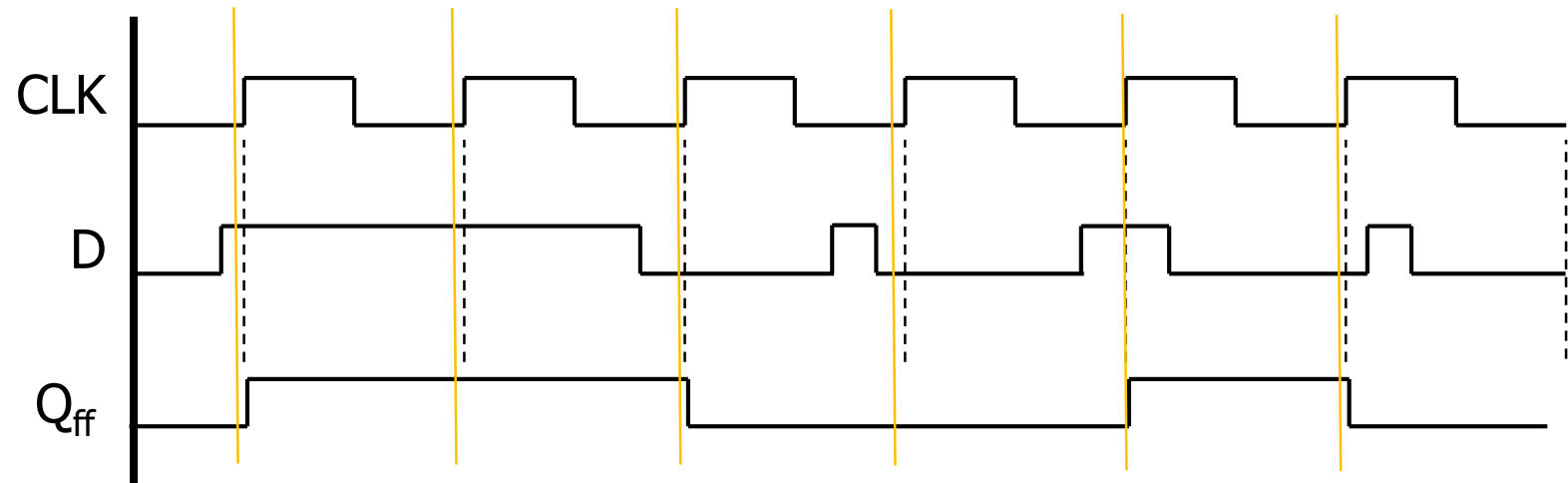
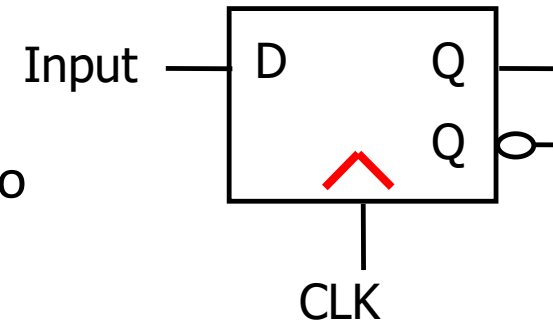
Sequential Logic (Bellek özelliğine sahiptir.)

FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC EQUATION	EXCITATION TABLE																				
SR		$Q_{(next)} = S + R'Q$ $SR = 0$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q_(next)</th> <th>S</th> <th>R</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>X</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>X</td> <td>0</td> </tr> </tbody> </table>	Q	Q _(next)	S	R	0	0	0	X	0	1	1	0	1	0	0	1	1	1	X	0
Q	Q _(next)	S	R																				
0	0	0	X																				
0	1	1	0																				
1	0	0	1																				
1	1	X	0																				
JK		$Q_{(next)} = JQ' + K'Q$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q_(next)</th> <th>J</th> <th>K</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>X</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>X</td> </tr> <tr> <td>1</td> <td>0</td> <td>X</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>X</td> <td>0</td> </tr> </tbody> </table>	Q	Q _(next)	J	K	0	0	0	X	0	1	1	X	1	0	X	1	1	1	X	0
Q	Q _(next)	J	K																				
0	0	0	X																				
0	1	1	X																				
1	0	X	1																				
1	1	X	0																				
D		$Q_{(next)} = D$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q_(next)</th> <th>D</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Q	Q _(next)	D	0	0	0	0	1	1	1	0	0	1	1	1					
Q	Q _(next)	D																					
0	0	0																					
0	1	1																					
1	0	0																					
1	1	1																					
T		$Q_{(next)} = TQ' + T'Q$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q_(next)</th> <th>T</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	Q	Q _(next)	T	0	0	0	0	1	1	1	0	1	1	1	0					
Q	Q _(next)	T																					
0	0	0																					
0	1	1																					
1	0	1																					
1	1	0																					

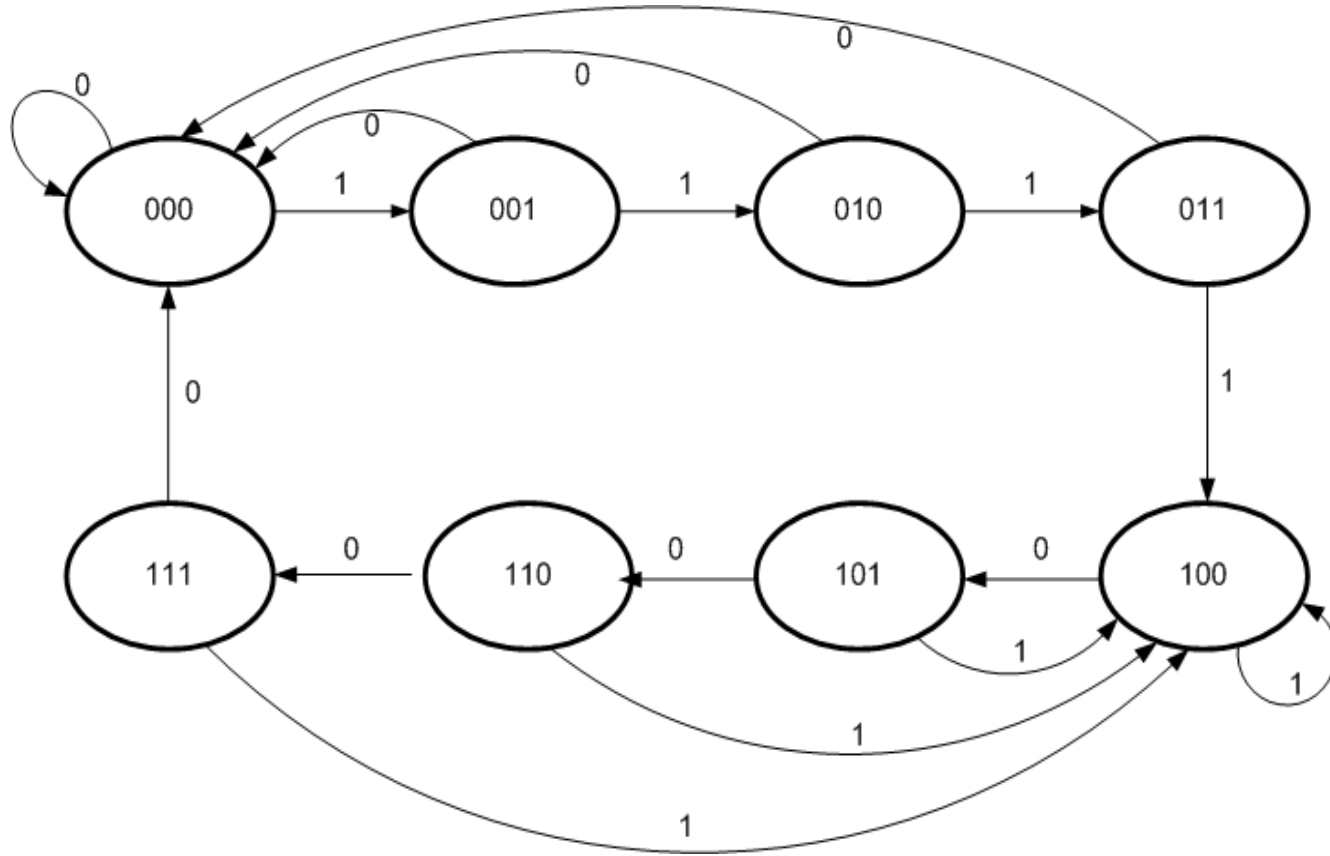
It is triggered by the rising edge of the Clk and the output takes the input value. Its state does not change until the next clock rising edge arrives. This is called memory feature.

The D flip-flop

- Input sampled at clock edge
 - Rising edge: Input passes to output
 - Otherwise: Flip-flop holds its output
- Flip-flops can be rising-edge triggered or falling-edge triggered
- On the rising edge of the clock signal, the output becomes equal to the input ($Q=D$). In all other cases of the clock signal, the output remains unchanged.
- The current state (Q) and the next state (D) are considered together.



State Diagram



- The number of binary circuits is found according to the current and next states
- Number of pairs = 3 pieces. Because in the State diagram, all states vary between 0 and 7. Total number of states = $8 = 2^3$.
- The current states are found at the Q outputs of the D-binary circuit. The next situation is at the D inputs of the D-binary circuit.
- When the D-binary circuit is triggered by the rising edge of the Clock, the Q-outputs become equal to the D-inputs.

Creating the State Table and reducing it with the help of Karnaugh diagram

Current situation

Next situation

Şu anki durum			Giriş	Bir sonraki durum		
Q2	Q1	Q0	C	D2	D1	D0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	1	0
1	0	1	1	1	0	0
1	1	0	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	0	0

$D2 =$

$Q_2Q_1 \backslash Q_0C$	00	01	11	10
00				
01			(1)	
11	(1)	(1)	(1)	
10	(1)	(1)	(1)	(1)

$D1 =$

$Q_2Q_1 \backslash Q_0C$	00	01	11	10
00			(1)	
01		(1)		
11	(1)			
10				(1)

$D0 =$

$Q_2Q_1 \backslash Q_0C$	00	01	11	10
00		(1)		
01		(1)		
11	(1)			
10	(1)			

- $D2 = Q_2Q_1' + Q_2Q_0' + Q_1Q_0C$
- $D1 = Q_2Q_1Q_0'C' + Q_2'Q_1Q_0'C + Q_2'Q_1'Q_0C + Q_2Q_1'Q_0C'$
- $D0 = Q_2'Q_0'C' + Q_2Q_0'C$

SUMMARY

- A binary number is a weighted number in which the weight of each whole number digit is a positive power of 2 and the weight of each fractional digit is a negative power of 2.
- The 1's complement of a binary number is derived by changing 1s to 0s and 0s to 1s
- The 2's complement of a binary number can be derived by adding 1 to the 1's complement.
- The octal number system consists of eight digits, 0 through 7.
- The hexadecimal number system consists of 16 digits and characters, 0 through 9 followed by A through F.
- The ASCII is a 7-bit alphanumeric code that is widely used in computer systems for input/output of information.
- The output of an inverter is the complement of its input
- The output of an AND gate is high only if all the inputs are high
- The output of an OR gate is high if any of the inputs is high
- The output of an NOR gate is low if any of the inputs is high
- The output of an NAND gate is low only if all the inputs are high
- The output of an exclusive-OR gate is high when the inputs are not the same

Binary Numbering System

Binary System

The binary system uses the number 2 as the base. The only allowable digits are 0 and 1. With digital circuits it is easy to distinguish between two voltage levels (i.e., +5 V and 0 V), which can be related to the binary digits 1 and 0 (Figure 6.2). Therefore, the binary system can be applied quite easily to computer systems. Since the binary system uses only two digits, each position of a binary number can go through only two changes, and then a 1 is carried to the immediate left position

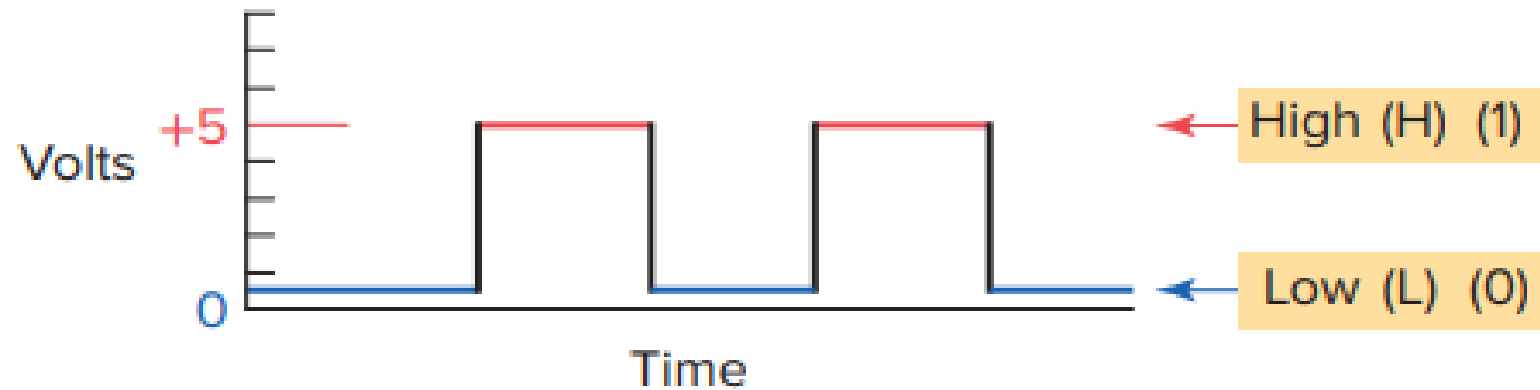


Figure 6.2 Digital signal waveform.

Binary Numbering System

- Binary is base 2
- Symbols: 0, 1
- Convention: $(2)_{10} = (10)_2 = (10)_b$
- What is $(110)_b$ in base 10?
- $(110)_b = (110)_2 = (1 \times 2^2) + (1 \times 2^1) + (0 \times 1^0) = (6)_{10}$

Base 10	Base 2
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

Binary number system

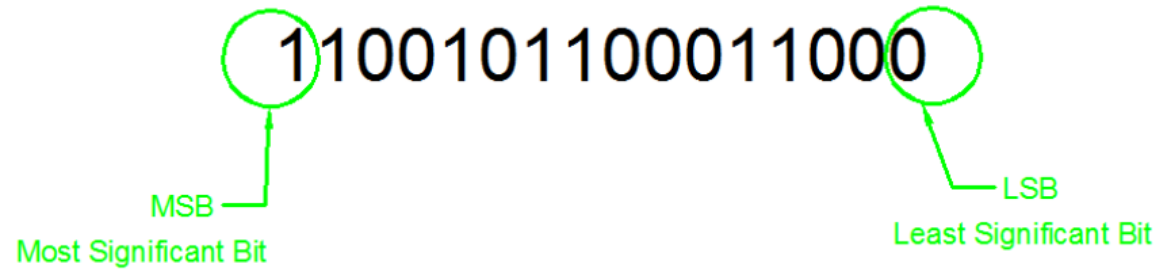
- All computers use the binary system :
 - Binary number system: Base = 2. Thus there are 2 numbers: 0 and 1.
 - A single binary number is called a Binary digit, or bit.
- Computers perform operations on binary number groups called bytes.
- Today, most computers use 32- or 64- bit :
- Bytes are subdivided into 8-bit groups called bytes.
- One-half a byte is sometimes referred to as a nibble (a term not often used anymore).

16 bitten yani 2 bayttan oluşan sayılara 1 word adı verilir.

1100101100011000

8 bitten oluşan sayılara 1 byte adı verilir.

Bir binary sayının en solundaki bit MSB, en sağındaki bit de LSB olarak isimlendirilir.



The Binary Number System

- The binary number system is also known as **base 2**. The values of the positions are calculated by taking 2 to some power.
- Why is the base 2 for binary numbers?
 - Because we use 2 digits, the digits 0 and 1.
- The binary number system is also a positional numbering system.
- Instead of using ten digits, 0 - 9, the binary system uses only two digits, 0 and 1.
- Example of a binary number and the values of the positions:

$$\begin{array}{ccccccc} \underline{1} & \underline{0} & \underline{0} & \underline{1} & \underline{1} & \underline{0} & \underline{1} \\ 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$



Binary Conversion

Figure 3 illustrates **how the binary number** 10101101 is **converted to its decimal equivalent**: 173.

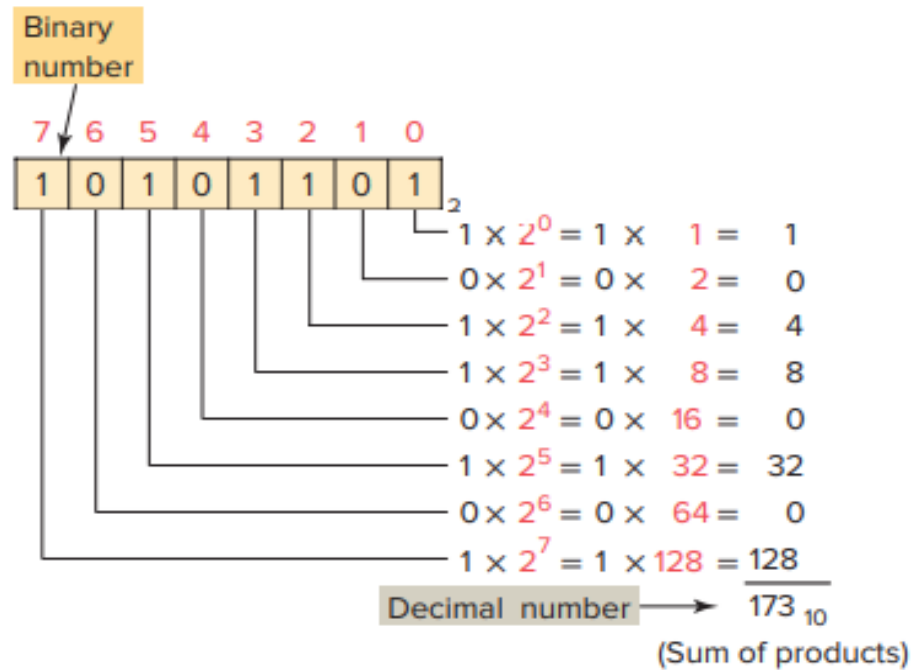


Figure 6.3 Converting a binary number to a decimal number.

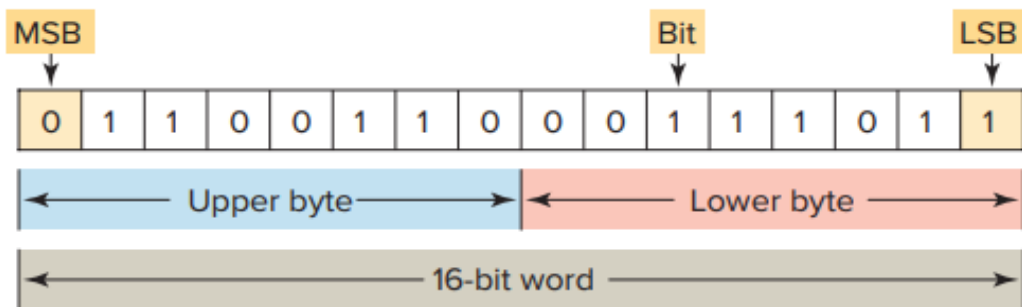


Figure 6.4 A 16-bit word.

Each digit of a binary number is known as a bit. In a processor-memory element consists of hundreds or thousands of locations. These locations, or registers, are referred to as **words**. Each word is capable of storing data in the form of binary digits, or bits. Bits can also be grouped within a word into **bytes**. **A group of 8 bits is a byte**, and a group of 2 or more bytes is a **word**. Figure 4 illustrates a 16-bit word made up of 2 bytes. The least significant bit (LSB) is the digit that represents the smallest value, and the most significant bit (MSB) is the digit that represents the largest value. A bit within the word can exist only in two states: a logical 1 (or ON) condition, or a logical 0 (or OFF) condition.

To convert a decimal number to its binary equivalent, we must perform a series of divisions by 2. Figure 5 illustrates the conversion of the decimal number 47 to binary. We start by dividing the decimal number by 2.

If there is a remainder, it is placed in the LSB of the binary number. If there is no remainder, a 0 is placed in the LSB. The result of the division is brought down and the process is repeated until the result of successive divisions has been reduced to 0.

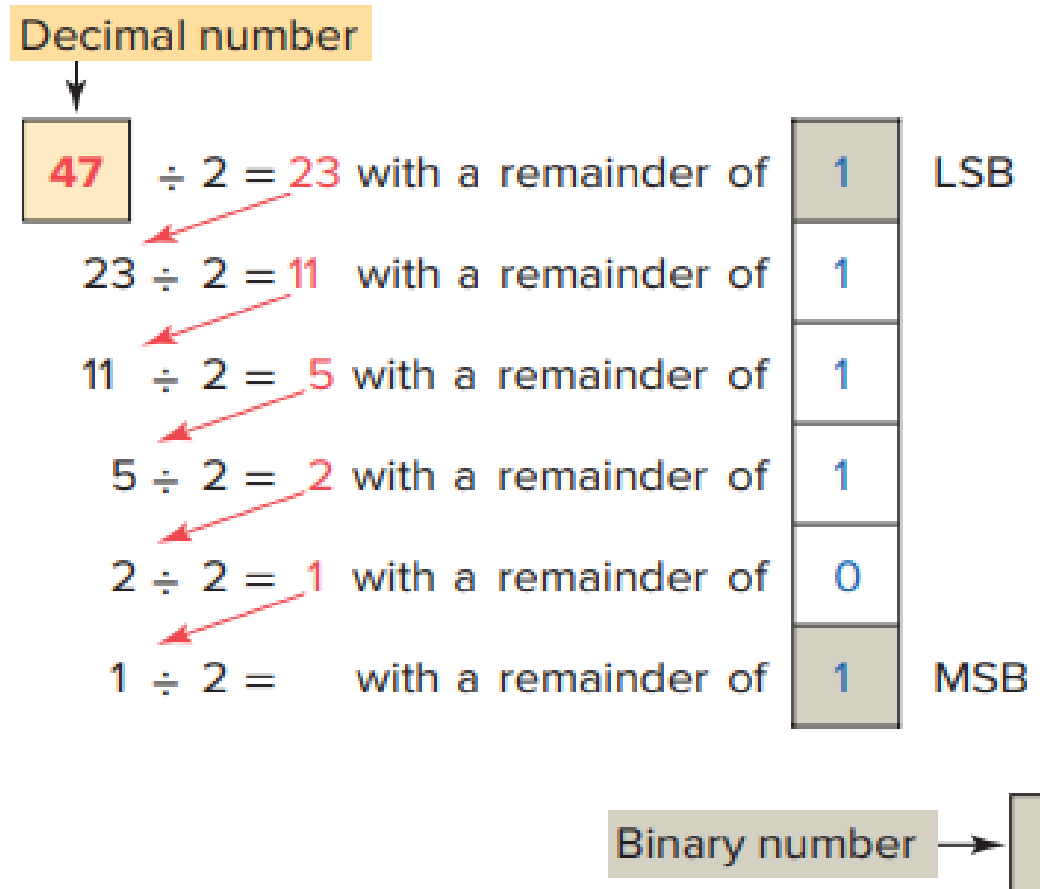


Figure 5.5 Converting a decimal number to a binary number.

Converting from Binary to Decimal

<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>		$1 \times 2^0 = 1$
2^6	2^5	2^4	2^3	2^2	2^1	2^0		$0 \times 2^1 = 0$
								$1 \times 2^2 = 4$
$2^0 = 1$	$2^4 = 16$							$1 \times 2^3 = 8$
$2^1 = 2$	$2^5 = 32$							$0 \times 2^4 = 0$
$2^2 = 4$	$2^6 = 64$							$0 \times 2^5 = 0$
$2^3 = 8$								$1 \times 2^6 = \underline{64}$
77_{10}								

Converting From Decimal to Binary

- Make a list of the binary place values up to the number being converted.
- Perform successive divisions by 2, placing the remainder of 0 or 1 in each of the positions from right to left.
- Continue until the quotient is zero.
- Example: 42_{10}

	2^5	2^4	2^3	2^2	2^1	2^0
	32	16	8	4	2	1
	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>

Working with Large Numbers

$$(0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1)_b = ?$$

Index: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

$$=2^{14}+2^{12}+2^7+2^5+2^2+2^1+2^0$$

- Humans can't work well with binary numbers; there are too many digits to deal with.
- Memory addresses and other data can be quite large. Therefore, we sometimes use the **hexadecimal number system**.

Binary Number Examples

- $11 = 1 \times 2^0 + 1 \times 2^1 = 3_{10}$
- $101 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4 + 1 = 5_{10}$.
- $1001 = 1 \times 2^3 + 1 \times 2^0 = 8 + 1 = 9_{10}$.
- $1100 = 1 \times 2^3 + 1 \times 2^2 = 8 + 4 = 12_{10}$.
- $11101 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 = 16 + 8 + 4 + 1 = 29_{10}$.
- $0.1 = 1 \times 2^{-1} = \frac{1}{2} = 0.5_{10}$
- $0.111 = 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 0.5 + 0.25 + 0.125 = 0.875_{10}$
- $0.10001 = 1 \times 2^{-1} + 1 \times 2^{-5} = 0.5 + 0.03125 = 0.53125_{10}$
- $1101.01 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-2} = 8 + 4 + 1 + 0.25 = 13.25_{10}$
- $11.001 = 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-3} = 2 + 1 + 0.125 = 3.125_{10}$
- $10.0011 = 1 \times 2^1 + 1 \times 2^{-3} + 1 \times 2^{-4} = 2 + 0.125 + 0.0625 = 2.1875_{10}$

Converting from Decimal to Binary

Given a decimal number N:

- List increasing powers of 2 from right to left until $\geq N$
- From left to right, ask is that (power of 2) $\leq N$?
 - If YES, put a 1 below and subtract that power from N
 - If NO, put a 0 below and keep going

Example for ~~13~~:

~~5~~
~~1~~
0

$2^4=16$	$2^3=8$	$2^2=4$	$2^1=2$	$2^0=1$
0	1	1	0	1

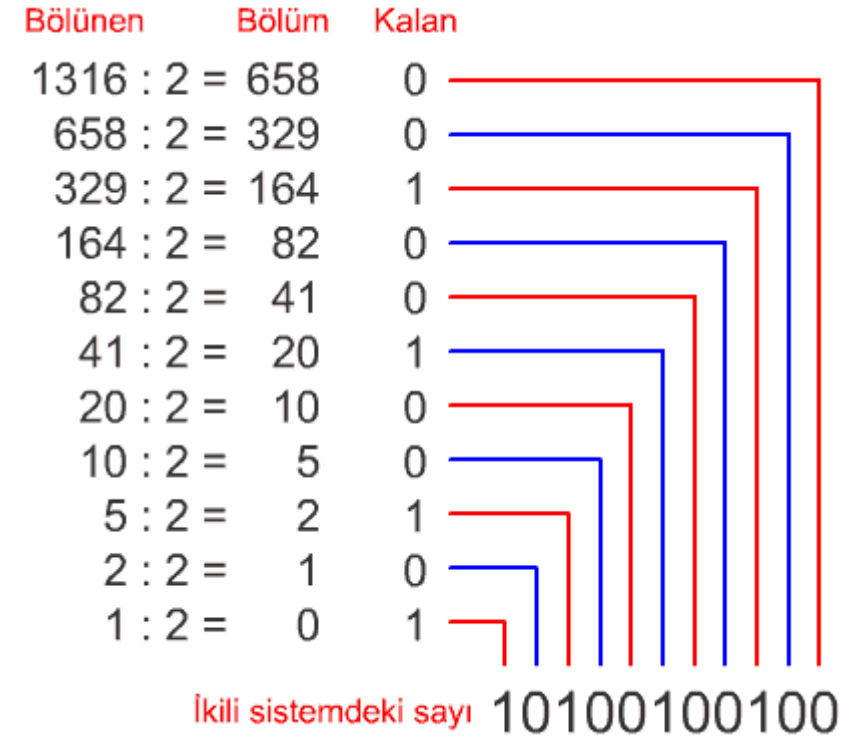
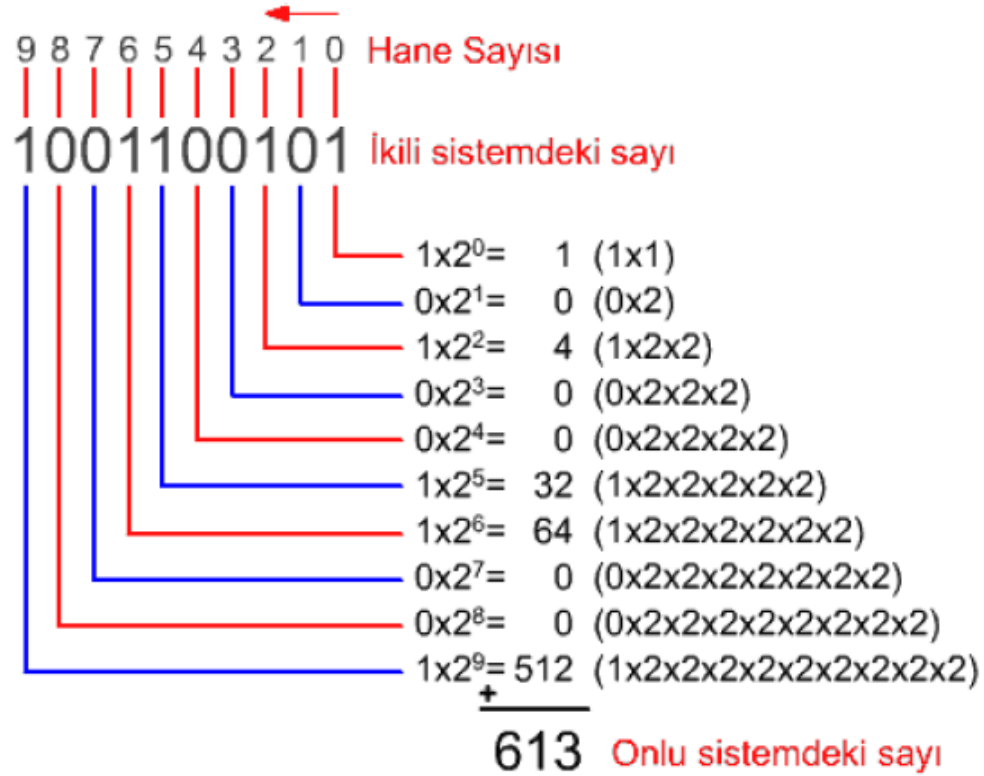
Binary Sayıların Decimal Sayılara Dönüştürülmesi:

- $(100011)_2 = (?)_{10}$
indis: 5,4,3,2,1,0
1 olanların indis karşılığı = 2^{indis}
 $2^5 + 2^1 + 2^0 = 32 + 2 + 1 = (35)_{10} = (23)_h$
- Ondalıklı Binary Sayıların Decimal Sayılara Dönüştürülmesi: $(111,101)_2 = 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-3} = 4 + 2 + 1 + 1/2 + 1/8 = 7,625$
- Decimal Sayıların Binary Sayılara Çevrilmesi: $(172)_{10} = (128 + 32 + 8 + 4)_{10} = (2^7 + 2^5 + 2^3 + 2^2)_{10} = (1010\ 1100)_2 = (AC)_{16}$
- Ondalıklı Decimal Sayıların Binary Sayılara Dönüştürülmesi
- $(10,75)_{10} = ?$ $(10)_{10} = (2^3 + 2^1)_{10} = (1010)_2$, $2^{-1} = 1/2 = 0,5$ $2^{-2} = 1/4 = 0,25$, $(10,75)_{10} = (1010,11)_2$

Numbers

- A 16-bit binary number
- 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
- 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1
- $= 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
- $= 256 + 128 + 16 + 8 + 4 + 2 + 1$
- $= 415$

Binary sayı sisteminden decimal'a çevirme



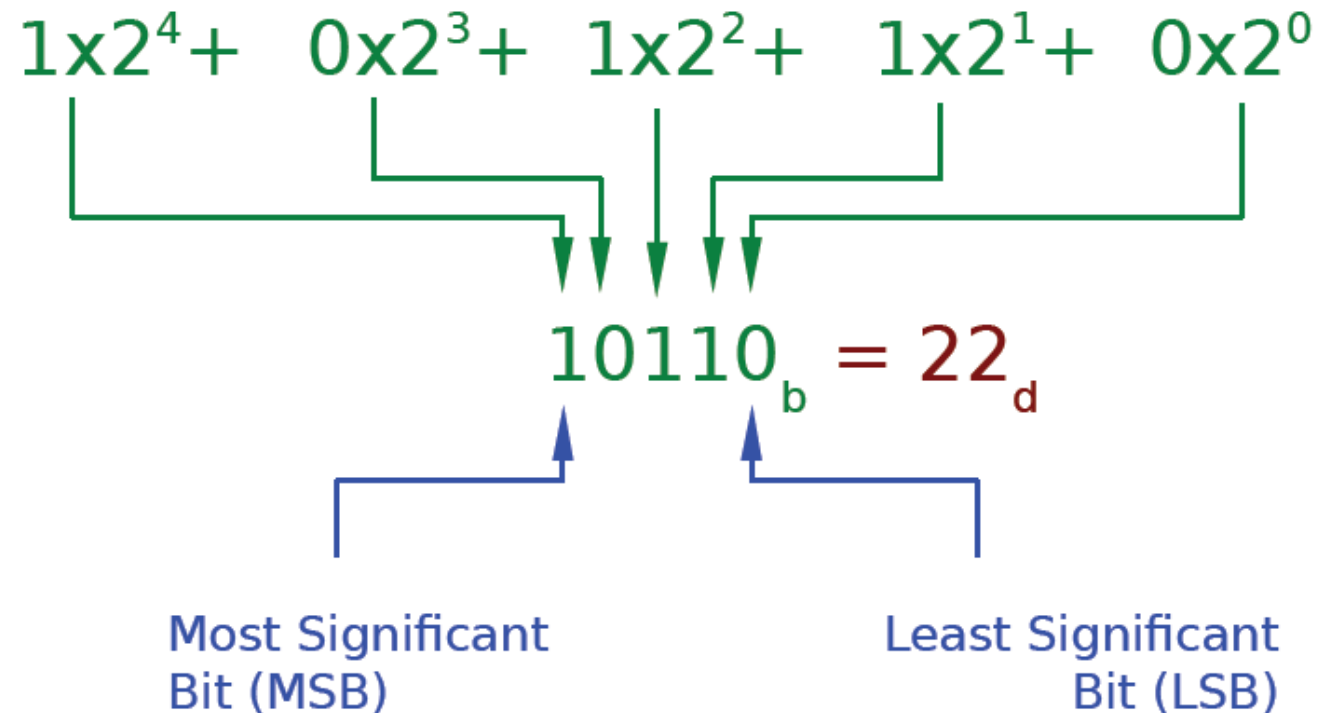
Binary to Decimal

Decimal	→ → conversion → →	Binary
0 =	0 × 2 ⁰	= 0
1 =	1 × 2 ⁰	= 1
2 =	1 × 2 ¹ + 0 × 2 ⁰	= 10
3 =	1 × 2 ¹ + 1 × 2 ⁰	= 11
4 =	1 × 2 ² + 0 × 2 ¹ + 0 × 2 ⁰	= 100
5 =	1 × 2 ² + 0 × 2 ¹ + 1 × 2 ⁰	= 101
6 =	1 × 2 ² + 1 × 2 ¹ + 0 × 2 ⁰	= 110
7 =	1 × 2 ² + 1 × 2 ¹ + 1 × 2 ⁰	= 111
8 =	1 × 2 ³ + 0 × 2 ² + 0 × 2 ¹ + 0 × 2 ⁰	= 1000

Binary

- Binary is exactly the same, only instead of ten digits/states (0 to 9) we have just two, so the base becomes 2:

Bilgisayar sistemlerinde 2 üzeri indeksleme 0,1,2,3,.. Biçimindedir. İndeksleme sağdan başlar.



Binary'den decimal'a çevirme örnekleri

- Örnek:

- $(1010)_2 = (?)_{10}$

- $(1010)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

- $(1010)_2 = 8 + 0 + 2 + 0$

- $(1010)_2 = 10$

- Örnek:

- $(11001)_2 = (?)_{10}$

- $(11001)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

- $(11001)_2 = 16 + 8 + 0 + 0 + 1$

- $(11001)_2 = 25$

Soru

- $(1011\ 0010)_b = (?)_d$
- Bir bit sağdan sola öteleyin
- Bir bit soldan sağa öteleyin
- Yorumlayın

Numbers

- If we let the most significant bit (MSB) signify positive or negative numbers (1 for negative; 0 for positive) Word olara tanımlayalım (16 bit)
- Then
- $+9 = 0\ 000\ 0000\ 0000\ 1001$
- $-9 = 1\ 000\ 0000\ 0000\ 1001$
- $+9 + (-9) = 1\ 000000000000010010$
- The result is NOT zero. This is a problem!

Numbers

- So we need a different representation for negative numbers
- Called 2s-complement
- Take the 1s-complement of the positive number:
- (+9)₁₀: 0 0 0 0 0 0 0 0 1 0 0 1
- 1'ler 0; 0'lar 1 yapılır:
- 1 1 1 1 1 1 1 1 0 1 1 0
- Note that the addition of the 1s-complement and the original number is not zero. +1 eklenir.
- To get the 2s-complement, add 1
- 1 1 1 1 1 1 1 1 1 1 0 1 1 0 + 1
- ---
- 1 1 1 1 1 1 1 1 1 1 0 1 1 1
- Now do the addition:
- 0 0 0 0 0 0 0 0 0 0 1 0 0 1 +
- 1 1 1 1 1 1 1 1 1 1 0 1 1 1
- ---
- 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Numbers

- Another example:
- $+1 + (-1)$
- 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 +
- 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
- ---
- 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Örnek: (16bit)

- $(-27)_d$
- $(+27)_d = 16 + 8 + 2 + 1 = 2^4 + 2^3 + 2^1 + 2^0$
- İndis: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
- 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
- $(+27)_d = 0000\ 0000\ 0001\ 1011 = (001B)_h$
- 1111 1111 1110 0100 (1'ler 0; 0'lar 1 yapıldı)
- 1111 1111 1110 0100
- 1
- $1111\ 1111\ 1110\ 0101 = (-27)_d$

Binary Sayıların Decimal Sayılara Dönüştürülmesi:

- $(100011)_2 = (?)_{10}$
indis: 5,4,3,2,1,0
1 olanların indis karşılığı = 2^{indis}
 $2^5 + 2^1 + 2^0 = 32 + 2 + 1 = (35)_{10} = (23)_h$
- Ondalıklı Binary Sayıların Decimal Sayılara Dönüştürülmesi: $(111,101)_2 = 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-3} = 4 + 2 + 1 + 1/2 + 1/8 = 7,625$
- Decimal Sayıların Binary Sayılara Çevrilmesi: $(172)_{10} = (128 + 32 + 8 + 4)_{10} = (2^7 + 2^5 + 2^3 + 2^2)_{10} = (1010\ 1100)_2 = (AC)_{16}$
- Ondalıklı Decimal Sayıların Binary Sayılara Dönüştürülmesi
- $(10,75)_{10} = ?$ $(10)_{10} = (2^3 + 2^1)_{10} = (1010)_2$, $2^{-1} = 1/2 = 0,5$ $2^{-2} = 1/4 = 0,25$, $(10,75)_{10} = (1010,11)_2$

Örnek

$(3)_d = 2^1 + 2^0 = 2 + 1$
 $(1 \ 1)_b$

$(215)_d = (b)_2$ maksimum 2
 $= 2^7 + 2^6 + 2^4 + 2^2 + 2^1 + 2^0$

128
 $\underline{87}$
 64
 $\underline{23}$
 16
 $\underline{7}$
 4
 $\underline{3}$
 2
 $\underline{1}$

$7 \ 6 \ 5 \ 4 \ 3 \ 2 \ 1 \ 0$
 $1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1$
 $(11010111)_b$
 $= 2^7 + 2^6 + 2^4 + 2^3 + 2^1 + 2^0$

$10 \ 10$
 $\underline{+ 1} \quad \underline{+ 1}$
 $11 \ 11$

$(104)_b = 2^2 + 2^0 = 5$

$(22)_d = 2^4 + 2^2 + 2^1 = 20$ 0000010110
 $\underline{+ 16}$
 06

$4 \ 3 \ 2 \ 1 \ 0$
 $(1 \ 0 \ 1 \ 1 \ 0)_b$

$(10 \ 110)_b \rightarrow$ Ötdem oklama
 $(10110)_b$
 $5 \ 4 \ 3 \ 2 \ 1 \ 0$
 $5 \ 3 \ 2 = 32 + 8 + 4 = 44 = (22 \times 2)$

$2^3 + 2^1 + 2^0 = 8 + 2 + 1 = 11 = (22/2)$

$(101)_b = (5)$
 $\rightarrow (10)_b = (2)$
 $10.111 \cdot \frac{5}{2}$
 $\times \frac{1}{2}$
 $\frac{1}{2} + \frac{1}{4} + \frac{1}{8}$

Binary (ikili): 0/1 → Bit

Bilgisayar: Veri, işleyen, I/O

Transistor → Yarıiletken teknolojisi,
elektron akışı kontrol ediliyor

→ Anahtarlama

→ Kuvvetlendirme

→ Şu anki durum ve bir sonraki durumu

Sembol (mesaj)

Veri (16)

Karakter (klavye tuşları) → ASCII → her karakter → 8 bit

Renk tonları → 8 bit / 16 bit / 32 bit / ...

$$(278)_d = (b)_2 = 2^8 + 2^4 + 2^2 + 2^1$$

En yüksek 2^n

maksimum indis

olarlar → 1, olmayalar → 0

$$\begin{array}{cccccccc} 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ (1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0)_2 \end{array}$$

$$\begin{array}{r} 100010110 \\ 8 \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \quad 0 \\ = 2^8 + 2^4 + 2^2 + 2^1 \end{array}$$

Sağdan solara indisle
olarlara 2^n

Topla:

$$\begin{array}{r} 256 \\ 16 \\ 4 \\ 2 \end{array}$$

$$\hline (278)_d$$

$$\begin{array}{r} 278 \\ 256 \\ \hline 22 \\ 16 \\ \hline 6 \\ 4 \\ \hline 2 \end{array}$$

OR - Veya (birtanesi 1 ise 1)
OR NOR

0	OR	0	=	0	1
0	OR	1	=	1	0
1	OR	0	=	1	0
1	OR	1	=	1	0

AND (ve) (hepsi 1 ise 1)
AND NAND

0	AND	0	=	0	1
0	AND	1	=	0	1
1	AND	0	=	0	1
1	AND	1	=	1	0

Not Teri

1 → 0
0 → 1

XOR (farklıdır)

0	XOR	0	→	0
0	XOR	1	→	1
1	XOR	0	→	1
1	XOR	1	→	0

$0 \text{ OR } 0 = 0$
 $0 \text{ OR } 1 = 1$
 $1 \text{ OR } 0 = 1$
 $1 \text{ OR } 1 = 1$

$$(10111011)_2 \text{ OR } (01110001)_2 = ?$$

$$(11111011)_2 = (FB)_{16}$$

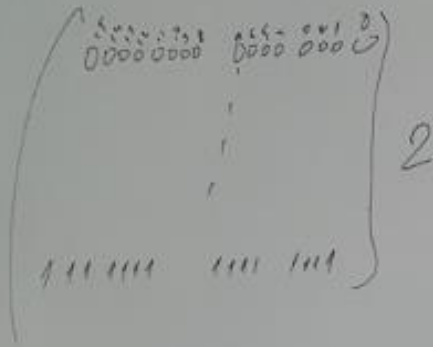
0 and 0 = 0
 0 and 1 = 0
 1 and 0 = 0
 1 and 1 = 1

$$\begin{array}{r}
 (FB)_{16} \text{ OR } (09)_{16} \\
 1100 \ 1010 \\
 0000 \ 1100 \\
 \hline
 1100 \ 1010 \\
 (0B)_{16}
 \end{array}$$

278 kbyte kaç byte?

i° $278 \leq 2^m$ ile ifade edilmeli.

$$2^9 \text{ kbyte} = 2^9 * 2^{10} = 2^{19} \text{ byte.}$$



2^9 - byte / 8 bitlik belleğin Adres hatları

10^{10} - bit/sec, bit Adres hatları sayısının

Veri transferinde Veri işlemde.

$$(781)_d = (?)_b = (?)_h$$
$$512 = 2^9 + 2^8 + 2^7 + 2^2 + 2^0$$

9 8 7 6 5 4 3 2 1 0
(1 1 0 0 0 0 1 1 0 1)_b

$$= (30D)_h$$

From Base 10 to Base 2: using table

- Input : a decimal number
- Output: the equivalent number in base 2
- Procedure:
 - Write a table as follows
 1. Find the largest two's power that is smaller than the number
 1. Decimal number 234 => largest two's power is 128
 2. Fill in 1 in corresponding digit, subtract 128 from the number => 106
 3. Repeat 1-2, until the number is 0
 4. Fill in empty digits with 0

...	512	256	128	64	32	16	8	4	2	1
			1	1	1	0	1	0	1	0

- Result is 11101010

Converting Binary to Decimal

Binary number system is base 2

- **0, 1**
- **Uses 2 numbers**

$$10010001 = 145$$

Base 2 representation	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal representation	128	64	32	16	8	4	2	1
Base 2 representation	1	0	0	1	0	0	0	1

Binary to Decimal

- Technique
 - Multiply each bit by 2^n , where n is the “weight” of the bit
 - The weight is the position of the bit, starting from 0 on the right
 - Add the results

$$101011_2 \Rightarrow \begin{array}{r} 1 \times 2^0 = 1 \\ 1 \times 2^1 = 2 \\ 0 \times 2^2 = 0 \\ 1 \times 2^3 = 8 \\ 0 \times 2^4 = 0 \\ 1 \times 2^5 = 32 \\ \hline 43_{10} \end{array}$$

$$\begin{array}{ccc} 1 & 1 & 1 \\ \uparrow & \uparrow & \uparrow \\ \text{position: } 2 & 1 & 0 \end{array}$$

$$\begin{aligned} 111 &= 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 1 \times 4 + 1 \times 2 + 1 \times 1 \\ &= 4 + 2 + 1 = 7 \end{aligned}$$

Decimal Sayıların Binary Sayılara Çevrilmesi:

$$(172)_{10} = (128 + 32 + 8 + 4)_{10} = (2^7 + 2^5 + 2^3 + 2^2)_{10} = (1010 \ 1100)_2 = (AC)_{16}$$

Binary'den decimal'a çevirme örnekleri

- **Örnek:**

- $(1010)_2 = (?)_{10}$

- $(1010)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

- $(1010)_2 = 8 + 0 + 2 + 0$

- $(1010)_2 = 10$

- **Örnek:**

- $(11001)_2 = (?)_{10}$

- $(11001)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

- $(11001)_2 = 16 + 8 + 0 + 0 + 1$

- $(11001)_2 = 25$

Decimal sayı sisteminden binary sayı sistemine çevirme örnekleri

- $(217)_d = (?)_b$
- $(217)_d = (11011001)_b$
- Sağlaması:
- Maksimum ne var, $2^7 = 128$
- $128 + 64 + 16 + 8 + 1 = 217$
- İndislenir: 7 6 5 4 3 2 1 0 ise 2 üzeri olanlara 1 olmayanlar 0 yazılır.
- $(11011001)_b$

- $(54)_d = (?)_2$
- $32 + 16 + 4 + 2$
- $(54)_d = (110110)_b$



Hex Numbering System

6.5 Hexadecimal System

The hexadecimal (hex) numbering system is used in programmable controllers because a word of **data consists of 16 data bits**, or two 8-bit bytes. **The hexadecimal system is a base 16 system, with A to F used to represent decimal numbers 10 to 15** (Table 6.5). The hexadecimal numbering system allows the status of a large number of binary bits to be represented in a small space, such as on a computer screen.

The techniques used when converting hexadecimal to decimal and decimal to hexadecimal are the same as those used for binary and octal. To convert a hexadecimal number to its decimal equivalent, the hexadecimal digits in the columns are multiplied by the base 16 weight, depending on digit significance.

Table 6.5 Hexadecimal Numbering System

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

Figure 6.6 illustrates how the conversion would be done for the hex number 1B7

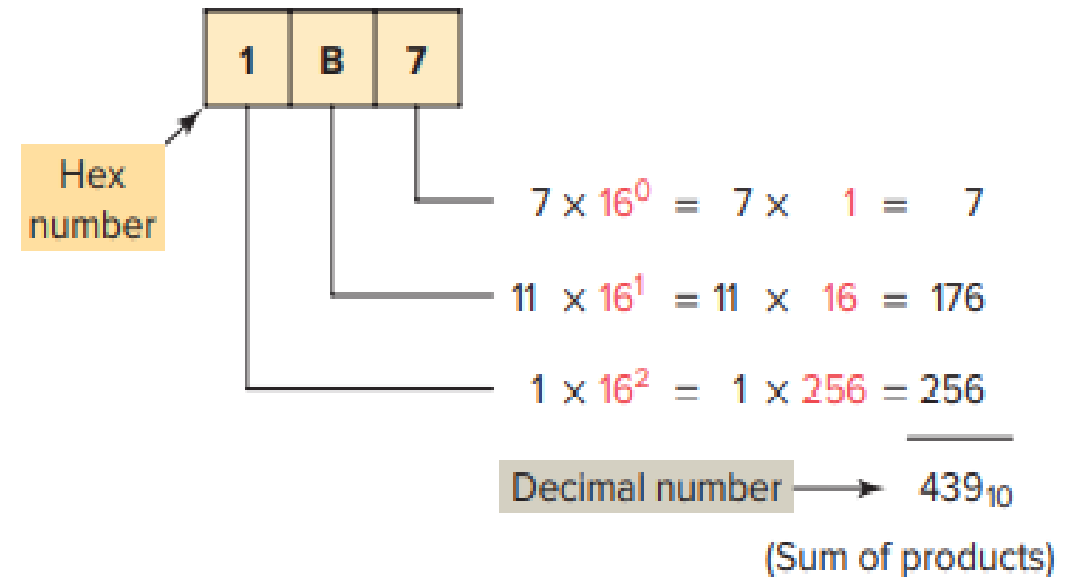


Figure 6.6 Converting a hexadecimal number to a decimal number.

Hexadecimal numbers can easily be converted to binary numbers. Conversion is accomplished by writing the 4-bit binary equivalent of the hex digit for each position, as illustrated in Figure 6.7

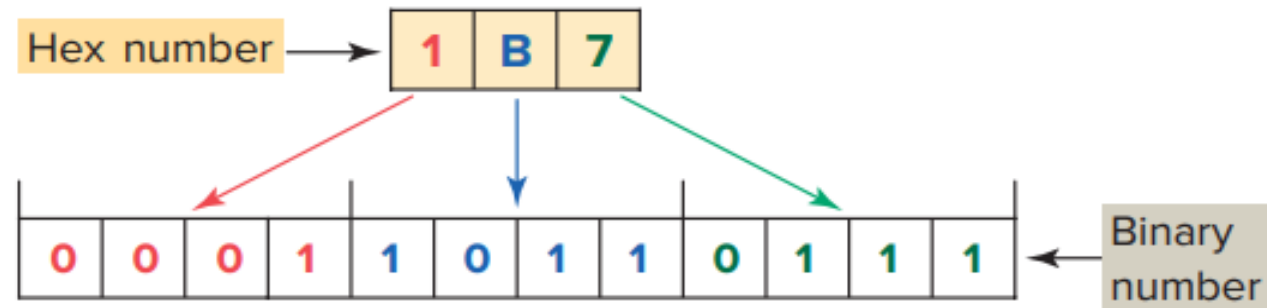


Figure 6.7 Converting a hexadecimal number to a binary number.

The Hexadecimal Number System

- The hexadecimal number system is also known as **base 16**. The values of the positions are calculated by taking 16 to some power.
- Why is the base 16 for hexadecimal numbers ?
 - Because we use 16 symbols, the digits 0 and 1 and the letters A through F.

Hexadecimal Number System

- Hexadecimal is base 16 (>10)
- Symbols? 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Convention: $(16)_{10} = (10)_{16} = 0x10h$
- Example: What is $0xA5$ in base 10?
- • $0xA5 = (A5)_{16} = (10 \times 16^1) + (5 \times 16^0) = (165)_{10}$

Converting Binary Hexadecimal

Hex Binary

- Substitute hex digits, then drop leading zeros
- Example: 0x2D in binary
 - 0x2 is (0010)b, 0xD is (1101)b
 - Drop two leading zeros, answer is (0010 1101)b

Binary Hex

- Pad with leading zeros until multiple of 4, then substitute groups of 4
- Example: (101101)b
- Pad to (0010 1101)
- Substitute to get 0x2D

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

Binary Hex Practice

- Convert (100110110101101)b
- How many digits?
- Pad: (0100 1101 1010 1101)b
- Substitute: 0x4DAD

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

The Hexadecimal Number System

- Example of a hexadecimal number and the values of the positions:

$$\begin{array}{ccccccc} \underline{3} & \underline{C} & \underline{8} & \underline{B} & \underline{0} & \underline{5} & \underline{1} \\ 16^6 & 16^5 & 16^4 & 16^3 & 16^2 & 16^1 & 16^0 \end{array}$$

Example of Equivalent Numbers

Binary: 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1₂

Decimal: 20647₁₀

Hexadecimal: 50A7₁₆

Notice how the number of digits gets smaller as the base increases.

Binary-Hexadecimal

- Since $2^4 = 16$, each hex digit effectively represents the same numeric count as four binary digits.
- Another way to say this is that one column in a hex number is the same as four columns of a binary number.

16^2 16^1 16^0 16^{-1} 16^{-2}

100101011011.01111010 = **0x95B.7A***

2^{11} 2^{10} 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6} 2^{-7} 2^{-8}

*Note: The “0x” prefix before a number signifies “hexadecimal.”

Binary to Hexadecimal number system

The binary number: 0001 0010 0100 1000 1001 1010 1101 1111
 └──┘ └──┘ └──┘ └──┘ └──┘ └──┘ └──┘ └──┘
The hexadecimal number: 1 2 5 8 9 A D F

Ex :- Convert the binary number $(1010101010)_2$ in to hexadecimal number.

$$(1010101010)_2 = 0010 \ 1010 \ 1010$$

$$= (2AA)_{16}$$

Radix Number System

- Base – 2 (binary numbers), bit:0/1
– 0 1
- Base – 8 (octal numbers)
– 0 1 2 3 4 5 6 7
- Base – 16 (hexadecimal numbers), 0 ve 1 lerin dizisinden oluşan ikil sayı sisteminde sağdan başlayarak 4'lü gruplar ayrılır.
– 0 1 2 3 4 5 6 7 8 9 A B C D E F

Binary/Hexidecimal

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

Conversions:

1000 1110 (binary)

8 E (hex)

Notations for hex:

- 0x8E
- 8Eh
- 8E₁₆

Bilgisayarda, tüm işlevler 1/0 ikili sayı sisteminde. 1 ve 0 lardan oluşan dizi var.

- Giriş ve Çıkış
- Veri Transferi
- Saklama
- Manipülasyon (Aritmetik, Mantıksal; kontrol işlemleri, kesme, ...)

Veri: Sinyal (Ses, EM, Elektriksel, ..),
Görüntü, Resim, Metin; portlar,
klavye, mouse

Hex Sayı Sistemi Nedir ?

- Binary ve desimal sistemden sonra mantık olarak aynı ancak fark olarak 16 lıkta bana sahip sayılar hexa desimal sayılardır.
- Heks sisteminde kullanabileceğimiz rakamlar benzer mantıkla yine 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 şeklindedir.
- Ancak hexa desimal sisteme göre yazılan sayılar 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F, şeklindedir.
- Dijital Elektronikte binary sayıların heksa desimala çevrilip bellekte kodlanması sağlanır.
- Sayılar binary'den heksadesimale çevrilirken sağdan sola doğru dörder basamak olmak üzere gruplandırılır.
- Çünkü heksa decimal sayı sisteminin tabanı 16 dir ve binary sayı sisteminde 0-15 sayıları, 4 bit ile ifade edilebilmektedir.

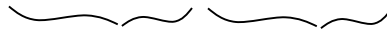
Binary/Hexidecimal

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

Conversions:

1000 1110 (binary)

8 E (hex)



Notations for hex:

•0x8E

•8Eh

•8E₁₆

Binary – Hexa Numbering System

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

- Short-hand for all these 1s and 0s
- HEX notation
- Each group of 4 bits represents a number in the range 0 – 15
- Hex is used as a notation for any sequence of bits (e.g. ASCII characters require just two hex digits)

Converting Binary to Hexadecimal

- Mark groups of *four* (from right)
- Convert each group

10101011

1010 1011
A B

10101011 is AB in base 16

And this?

Numbers

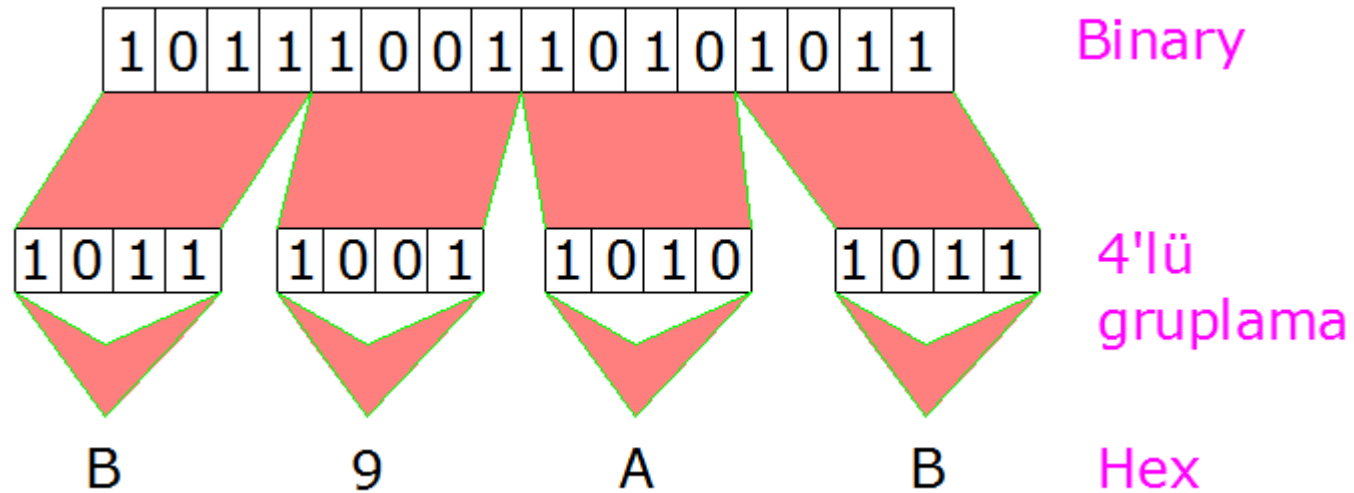
- Short-hand for all these 1s and 0s
- HEX notation
- Hex is used as a notation for any sequence of bits (e.g. ASCII characters require just two hex digits)
- Each group of 4 bits represents a number in the range 0 – 15

• Thus:

- 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0009
- 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 FFF7
- 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0000

0 0 0 0 =	0	1 0 0 0 =	8
0 0 0 1 =	1	1 0 0 1 =	9
0 0 1 0 =	2	1 0 1 0 =	A
0 0 1 1 =	3	1 0 1 1 =	B
0 1 0 0 =	4	1 1 0 0 =	C
0 1 0 1 =	5	1 0 0 1 =	D
0 1 1 0 =	6	1 1 1 0 =	E
0 1 1 1 =	7	1 1 1 1 =	F

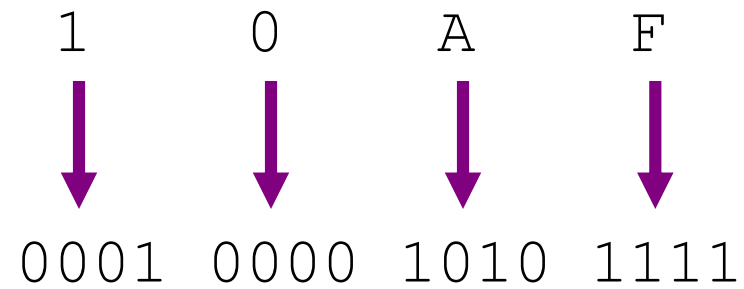
Binary Sayının Hexa decimal'e Çevrilmesi



Hexadecimal to Binary

- Convert each hexadecimal digit to a 4-bit equivalent binary representation

$$10AF_{16} = ?_2$$



$$10AF_{16} = 0001000010101111_2$$

Octal System

Octal System

To express the number in the binary system requires many more digits than in the decimal system. Too many binary digits can become cumbersome to read or write. To solve this problem, other related numbering systems are used. **The octal numbering system**, a **base 8 system**, is used because 8 data bits make up a byte of information that can be addressed. Octal is a convenient means of handling large binary numbers

As shown in Table 6-4, one octal digit can be used to express three binary digits. As in all other numbering systems, each digit in an octal number has a weighted decimal value according to its position. Figure 6.6 illustrates how the octal number 462 is converted to its decimal equivalent: 306.

Table 6.4 Binary and Related Octal Code	
Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

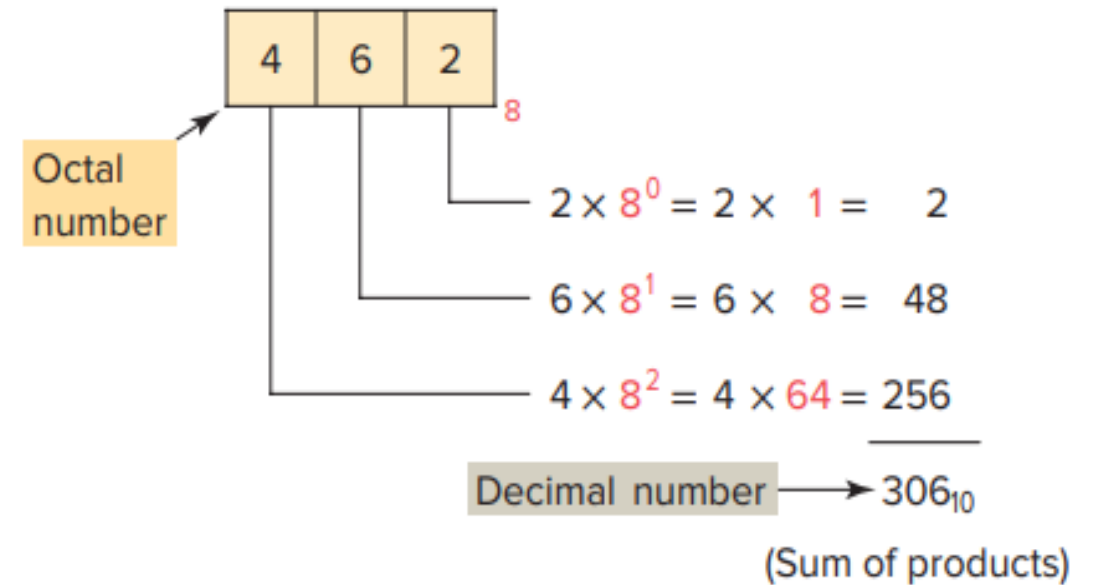


Figure 6.6 Converting an octal number to a decimal number.

Octal converts easily to binary equivalents. For example, the octal number 462 is converted to its binary equivalent by assembling the 3-bit groups, as illustrated in Figure 6.7.

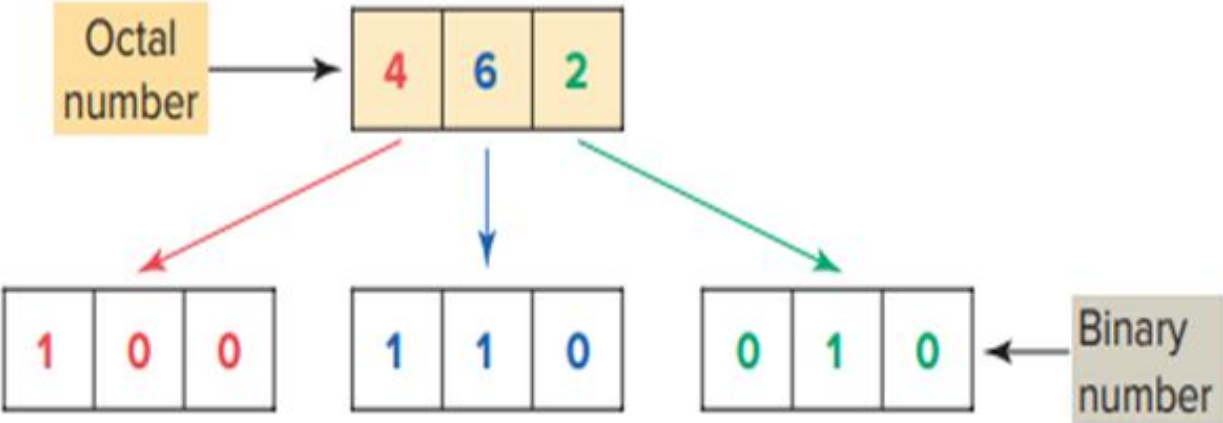
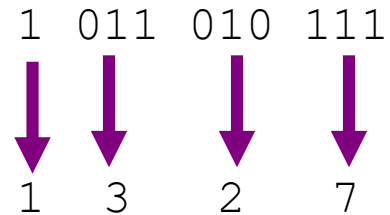


Figure 6.7 Converting an octal number to a binary number.

Binary to Octal

- Technique
 - Group bits in threes, starting on right
 - Convert to octal digits

$$1011010111_2 = ?_8$$

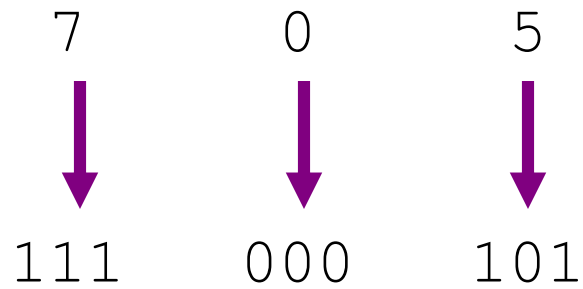


$$1011010111_2 = 1327_8$$

Octal to Binary

- Technique
 - Convert each octal digit to a 3-bit equivalent binary representation

$$705_8 = ?_2$$



$$705_8 = 111000101_2$$

Fractions

Fractional Binary number to Decimal number

Ex :- Convert the binary number $(0.1101)_2$ into decimal number.

$$\begin{aligned}(0.1101)_2 &= (1 \times \frac{1}{2}) + (1 \times \frac{1}{4}) + (0 \times \frac{1}{8}) + (1 \times \frac{1}{16}) \\ &= 1 \times 0.5 + 1 \times 0.25 + 0 \times 0.125 + 1 \times 0.0625 \\ &= 0.5 + 0.25 + 0 + 0.0625 \\ &= (0.8125)_{10}\end{aligned}$$

Ex :- Convert the binary number $(1010.0101)_2$ into decimal number.

$$\begin{aligned}\text{Sol :- } \underline{\text{Integer part}} (1010)_2 &= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) \\ &= (1 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1) \\ &= (10)_{10}\end{aligned}$$

$$\begin{aligned}\underline{\text{Fractional part}} (0.0101)_2 &= (0 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4}) \\ &= (0 \times \frac{1}{2}) + (1 \times \frac{1}{4}) + (0 \times \frac{1}{8}) + (1 \times \frac{1}{16}) \\ &= 0 \times 0.5 + 1 \times 0.25 + 0 \times 0.125 + 1 \times 0.0625 \\ &= 0 + 0.25 + 0 + 0.0625 \\ &= (0.3125)_{10}\end{aligned}$$

Fractional Decimal number to binary

The fractional decimal number is converted into binary number by using successive fraction multiplications method.

1. The fractional decimal number is multiplied with 2 by successive fraction multiplications method.
2. If '1' or '0' occurs in units place in the product, transfer that '1' or '0' to the binary record.
3. The multiplication is continued with the remaining fraction.
4. The same procedure is followed in each multiplication.
5. The first transferred number (1 or 0) to binary record is taken as most significant bit (MSB).
6. The last transferred number (1 or 0) to binary record is taken as least significant bit (LSB).
7. If the multiplication does not end, it can be stopped at any of our desired level.

Ex : - Convert the fractional decimal number $(0.638)_{10}$ in to fractional binary number.

Sol :-

Successive multiplications		Binary	
$0.638 \times 2 = 1.276$	→	1	MSB
$0.276 \times 2 = 0.552$	→	0	
$0.552 \times 2 = 1.104$	→	1	
$0.104 \times 2 = 0.208$	→	0	
$0.208 \times 2 = 0.416$	→	0	
$0.416 \times 2 = 0.832$	→	0	
$0.832 \times 2 = 1.664$	→	1	
$0.664 \times 2 = 1.328$	→	1	
$0.328 \times 2 = 0.656$	→	0	
$0.656 \times 2 = 1.312$	→	1	LSB

So, $(0.638)_{10} = (0.1010001101)_2$

Virgüllü ondalık sayıyı binarysayıya çevirme

- 10 tabanında verilen sayının tamsayı kısmı sürekli olarak istenen tabana bölünerek kalanlar sağdan sola doğru yazılır.
- 10 tabanında verilen sayının kesir kısmı için ise, bu kısım sürekli olarak istenen taban ile çarpılır ve çarpımın sonucu istenen basamağa erişene kadar (ya da 0 çıkana kadar) işlemin tamsayı kısmı alınır.
- **Örnek: $(121.125)_{10}$ sayısını 2 tabanına dönüştürünüz.**

- $121 / 2 = 60$
- $60 / 2 = 30$
- $30 / 2 = 15$
- $15 / 2 = 7$
- $7 / 2 = 3$
- $3 / 2 = 1$
- $1 / 2 = 0$

Kalan

1
0
0
1
1
1
1



- Kesirli kısmı dönüştürelim.

- $0.125 * 2 = 0.250$
- $0.250 * 2 = 0.500$
- $0.500 * 2 = 1.000$

0
0
1



- Sonuç = 1111001.001

Ondalık sayı sistemini ikil sayı sistemine dönüştürülürken 2'ye bölünerek en son böüm geröeklene devam edilir. Kalan var ise 1 yok ise 0 yapılır.

Here are some handy facts:

Power of 2	Base 2	Base 10
2^{-4}	.0001	.0625
2^{-3}	.001	.125
2^{-2}	.01	.25
2^{-1}	.1	.5
2^0	1	1
2^1	10	2
2^2	100	4
2^3	1000	8
2^4	10000	16
2^5	100000	32
2^6	1000000	64
2^7	10000000	128
2^8	100000000	256

N	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Fractions

- We can extend our unsigned representational system of binary to include a decimal point
 - After the decimal point, the i exponent, in 2^i , becomes negative
 - So, we now have the $\frac{1}{2}$ column, the $\frac{1}{4}$ column, etc
 - $(1011.1001)_b =$
 - Indisleme: iki grupta yapılır. Noktadan sola ve sağa doğru... 3,2,1,0.,-1,-2,-3,-4
 - $1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 0*2^{-3} + 1*2^{-4} =$
 - $8 + 2 + 1 + \frac{1}{2} + \frac{1}{16} =$
 - $11 \frac{9}{16} = 11.5625$
 - What is .4304? Use 8-bits with 4 fraction bits
 - .4304 has a .25, .125, .03125, .015625, and more fractions, but this exceeds the number of fraction bits so the number is 0000.0110
 - But $0000.0110 = .125 + 0.3125 = .375$, we have a loss in precision!
 - In the fraction representation, our decimal point is typically fixed, so this is often known as *fixed point representation*
 - We will cover a floating point representation later

Fractions

- Decimal to decimal (just for fun)

$$\begin{aligned} 3.14 \Rightarrow & & 4 \times 10^{-2} & = & 0.04 \\ & & 1 \times 10^{-1} & = & 0.1 \\ & & 3 \times 10^0 & = & 3 \\ & & & & 3.14 \end{aligned}$$

- Binary to decimal

$$\begin{aligned} 10.1011 \Rightarrow & & 1 \times 2^{-4} & = & 0.0625 \\ & & 1 \times 2^{-3} & = & 0.125 \\ & & 0 \times 2^{-2} & = & 0.0 \\ & & 1 \times 2^{-1} & = & 0.5 \\ & & 0 \times 2^0 & = & 0.0 \\ & & 1 \times 2^1 & = & 2.0 \\ & & & & 2.6875 \end{aligned}$$

Works for Fractional Numbers too...

$$3 \times 10^1 + 5 \times 10^0 + 4 \times 10^{-1} + 6 \times 10^{-2} + 2 \times 10^{-3}$$

35.462

$$1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3}$$

$10.110_b = 2.75_d$

$$1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$$

$11.011_b = 3.375_d$

Binary'den decimal'a çevirme örnekleri

- Örnek:

- $(111,101)_2 = (?)_{10}$

- $(111,101)_2 =$

- $1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

- $(111,101)_2 =$

- $1 \times 4 + 1 \times 2 + 1 \times 1 + 1 \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{8}$

- $(111,101)_2 =$

- $4 + 2 + 1 + 0,5 + 0 + 0,125$

- $(111,101)_2 = (7,625)_{10}$

- Örnek:

- $(100111,01)_2 = (?)_{10}$

- $(100111,01)_2 = (39,25)_{10}$



Binary Manuplation

Why Binary Arithmetic?

- All data is recorded as 1 or 0 in the computer's memory.
- All operations in the processor are done with 1s and 0s.
- The hardware is made with electrical signals only in the binary number system of 1 and 0.
- Must represent all numbers, integers or floating point, positive or negative, by binary digits, called *bits*.
- Can devise electronic circuits to perform arithmetic operations: add, subtract, multiply and divide, and logical operations, comparisons on binary numbers.

Why Binary Arithmetic?

$$3 + 5$$



$$= 8$$

$$0011 + 0101$$



$$= 1000$$

Binary Arithmetic

Mathematical operations include **addition, subtraction, multiplication, and division**. Binary addition follows rules similar to decimal addition. When adding with binary numbers, there are only four conditions that can occur:

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ +0 & +0 & +1 & +1 \\ \hline 0 & 1 & 1 & 0 \text{ carry } 1 \end{array}$$

The first three conditions are easy because they are like adding decimals, but the last condition is slightly different. In decimal, $1 + 1 = 2$. In binary, a 2 is written 10. Therefore, in binary, $1 + 1 = 0$, with a carry of 1 to the next most significant place value. When adding larger binary numbers, the resulting 1s are carried into higher-order columns, as shown in the following examples.

Decimal

$$\begin{array}{r} 5 \\ +2 \\ \hline 7 \end{array}$$

Equivalent binary

$$\begin{array}{r} 101 \\ + 10 \\ \hline 111 \end{array}$$

$$\begin{array}{r} 10 \\ + 3 \\ \hline 13 \end{array}$$

$$\begin{array}{r} \text{carry} \\ 1 \\ 10 \quad 10 \\ + \quad 11 \\ \hline 11 \quad | \quad 01 \end{array}$$

$$\begin{array}{r} 26 \\ +12 \\ \hline 38 \end{array}$$

$$\begin{array}{r} \text{carry} \quad \text{carry} \\ 1 \quad 1 \\ 1 \quad 1010 \\ + \quad 1100 \\ \hline 1 \quad | \quad 0 \quad | \quad 0110 \end{array}$$

In arithmetic functions, the initial numeric quantities that are to be combined by subtraction are the minuend and subtrahend. The result of the subtraction process is called the difference, represented as:

$$\begin{array}{r} A \text{ (minuend)} \\ - B \text{ (subtrahend)} \\ \hline C \text{ (difference)} \end{array}$$

To subtract from larger binary numbers, subtract column by column, borrowing from the adjacent column when necessary. Remember that when borrowing from the adjacent column, there are now two digits, i.e., 0 borrow 1 gives 10.

EXAMPLE

Subtract 1001 from 1101.

$$\begin{array}{r} 1101 \\ -1001 \\ \hline 0100 \end{array}$$

Subtract 0111 from 1011.

$$\begin{array}{r} 1011 \\ -0111 \\ \hline 0100 \end{array}$$

EXAMPLE

Subtract 111 from 100.

$$\begin{array}{r} 111 \\ -100 \\ \hline -011 \end{array}$$

Subtract 11011 from 10111.

$$\begin{array}{r} 11011 \\ -10111 \\ \hline -00100 \end{array}$$

Binary numbers can also be negative. The procedure for this calculation is identical to that of decimal numbers because the smaller value is subtracted from the larger value and a negative sign is placed in front of the result.

There are other methods available for doing subtraction:

1's complement

2's complement

The procedure for subtracting numbers using the 1's complement is as follows:

Step 1 Change the subtrahend to 1's complement.

Step 2 Add the two numbers.

Step 3 Remove the last carry and add it to the number (end-around carry)

When there is a **carry** at the end of the result, **the result is positive**.

When there is **no carry**, we take the 1's complement of the **result is** and a minus sign has to be placed in front of it.

Decimal	Binary
$\begin{array}{r} 10 \\ - 6 \\ \hline 4 \end{array}$	$\begin{array}{r} 1010 \\ - 0110 \\ \hline 100 \end{array}$
	$\begin{array}{r} 1010 \\ + 1001 \\ \hline 10011 \\ \hline \end{array}$
	$\begin{array}{r} 10011 \\ + 1 \\ \hline 100 \end{array}$

EXAMPLE

Subtract 11011 from 01101.

$$\begin{array}{r}
 01101 \\
 + \cancel{0}00100 \\
 \hline
 10001
 \end{array}$$

The 1's complement

There is no carry, so we take the 1's complement and add the minus sign:

$$\begin{array}{r}
 -01110 \\
 \hline
 \end{array}$$

For subtraction using the 2's complement, the 2's complement is added instead of subtracting the numbers. In the result, **if the carry is a 1, then the result is positive**; **if the carry is a 0, then the result is negative and requires a minus sign.**

EXAMPLE

Subtract 101 from 111.

$$\begin{array}{r}
 111 \\
 + \cancel{0}011 \\
 \hline
 1010
 \end{array}$$

The 2's complement

The first 1 indicates that the result is positive, so it is disregarded:

$$\begin{array}{r}
 010 \\
 \hline
 \end{array}$$

EXAMPLE

Subtract 11011 from 01101.

$$\begin{array}{r}
 01101 \\
 + \cancel{0}00101 \\
 \hline
 10010
 \end{array}$$

The 2's complement

There is no carry, so the result is negative; therefore a 1 has to be subtracted and the 1's complement taken to give the result:

$$\begin{array}{r}
 \text{subtract 1} \quad 10010 - 1 = 10001 \\
 \text{1's complement} \quad \underline{-01110}
 \end{array}$$

REVIEW QUESTIONS

1. Convert each of the following binary numbers to decimal numbers:
 - a. 10
 - b. 100
 - c. 111
 - d. 1011
 - e. 1100
 - f. 10010
 - g. 10101
 - h. 11111
 - i. 11001101
 - j. 1110001
2. Convert each of the following decimal numbers to binary numbers:
 - a. 7
 - b. 19
 - c. 28
 - d. 46
 - e. 57
 - f. 86
 - g. 94
 - h. 112
 - i. 148
 - j. 230

3. Convert each of the following **octal numbers** to **decimal numbers**:

- a. 36
- b. 104
- c. 120
- d. 216
- e. 360
- f. 1516

4. Convert each of the following **octal numbers** to **binary numbers**:

- a. 74
- b. 130
- c. 250
- d. 1510
- e. 2551
- f. 2634

5. Convert each of the following **hexadecimal numbers** to **decimal numbers**:

- a. 5A
- b. C7
- c. 9B5
- d. 1A6

6. Convert each of the following **hexadecimal numbers** to **binary numbers**:

- a. 4C
- b. E8
- c. 6D2
- d. 31B

7. Add the following binary numbers:

a. $110 + 111$

b. $101 + 011$

c. $1100 + 1011$

8. Subtract the following binary numbers:

a. $1101 - 101$

b. $1001 - 110$

c. $10111 - 10010$

9. Convert each piece of binary information to the appropriate hexadecimal code

a. $0001\ 1111$

b. $0010\ 0101$

c. $0100\ 1110$

d. $0011\ 1001$

10. Express the decimal number 18 in each of the following number codes:

a. Binary

b. Octal

c. Hexadecimal

Bilgisayarda İşlemler

- Aritmetik: +, (-, *, /)
- Mantıksal: VE, VEYA, NOT, ...
- Karşılaştırma: <, >, ≥, ≤, ==, ≠, ...
- Mikroişlemci ALU (Aritmetik Logic Unit) sadece toplama işlemei vardır ve çarpma ve bölme işlemlerini ise öteleme ile yapar.
- ROL: Rotate operand1 left (Çarpma)

Binary Arithmetic

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} {}^1 1 \\ + 1 \\ \hline 10 \end{array}$$

$$\begin{array}{r} 1 \\ 1 \\ 1 \\ + \\ \hline 11 \end{array}$$

İkili Sayılarda Toplama

Binary(İkilik) sayı sistemindeki temel toplama kuralları;

0+0	= 0	→ Elde 0	Toplam 0
0+1	= 1	→ Elde 0	Toplam 1
1+0	= 1	→ Elde 0	Toplam 1
1+1	= 10	→ Elde 1	Toplam 0
1+1+1	= 11	→ Elde 1	Toplam 1

Numerical examples for addition follow:

+ 6	00000110	- 6	11111010
+13	<u>00001101</u>	+13	<u>00001101</u>
+19	00010011	+ 7	00000111
+ 6	00000110	- 6	11111010
-13	<u>11110011</u>	-13	<u>11110011</u>
- 7	11111001	-19	11101101

Ondalık toplama işlemini düşünün: $3+1=4$

İkili sayı sistemine dönüştürülür.

2 üssü toplamlardan ondalık sayı sistemi eldeilir: $4=2^2$

İndisleme: 2 1 0

Dönüştürme: (100)

(3)d=(011)b

(1)d=(001)b

(011)b+(001)b=(100)b

Binary addition

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

- Represent sum of binary numbers as a binary number

decimal addition

$$1+1 = 2$$

$$1+1+1 = 3$$

binary addition

$$1+1 = 10$$

$$1+1+1 = 10+1 = 11$$

Adding binary numbers

Binary addition

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

- Represent sum of binary numbers as a binary number

decimal addition

$$1+1 = 2$$

$$1+1+1 = 3$$

binary addition

$$1+1 = 10$$

$$1+1+1 = 10+1 = 11$$

Adding binary numbers

$$\begin{array}{r} 101 \\ + 10 \\ \hline 111 \end{array}$$

$$\begin{array}{r} 11 \leftarrow \text{carry} \\ 101 \\ + 11 \\ \hline 1000 \end{array}$$

$$\begin{array}{r} 11 \leftarrow \text{carry} \\ 111 \\ + 110 \\ \hline 1101 \end{array}$$

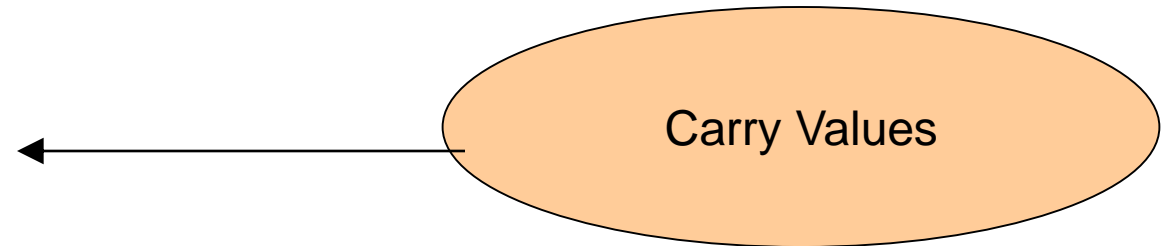
$$\begin{array}{r} 1 \quad 11 \leftarrow \text{carry} \\ 10101010111 \\ + 110000110 \\ \hline 11011011101 \end{array}$$

Arithmetic in Binary

Remember that there are only 2 digit symbols in binary, 0 and 1

1 + 1 is 0 with a carry

$$\begin{array}{r} 1011111 \\ 1010111 \\ +1001011 \\ \hline 10100010 \end{array}$$



Binary sayılarda toplama-çıkarma işlemleri

Aşağıda verilen toplama işlemlerini gerçekleştirin.

$$\begin{array}{r} \text{a- } (11)_2 \\ + (11)_2 \\ \hline (110)_2 \end{array}$$

$$\begin{array}{r} 3 \\ + 3 \\ \hline 6 \end{array}$$

$$\begin{array}{r} \text{b- } (100)_2 \\ + (11)_2 \\ \hline (111)_2 \end{array}$$

$$\begin{array}{r} \text{c- } (111)_2 \\ + (11)_2 \\ \hline (1010)_2 \end{array}$$

$$\begin{array}{r} \text{d- } (0110)_2 \\ + (1111)_2 \\ \hline (10101)_2 \end{array}$$

$$\begin{array}{r} \text{c- } (11101)_2 \\ (1001)_2 \\ + (111)_2 \\ \hline (101101)_2 \end{array}$$

Aşağıda verilen çıkarma işlemlerini gerçekleştirin.

$$\begin{array}{r} \text{a- } (11)_2 \\ - (10)_2 \\ \hline (01)_2 \end{array}$$

$$\begin{array}{r} \text{b- } (100)_2 \\ - (011)_2 \\ \hline (001)_2 \end{array}$$

$$\begin{array}{r} \text{c- } (101)_2 \\ - (011)_2 \\ \hline (010)_2 \end{array}$$

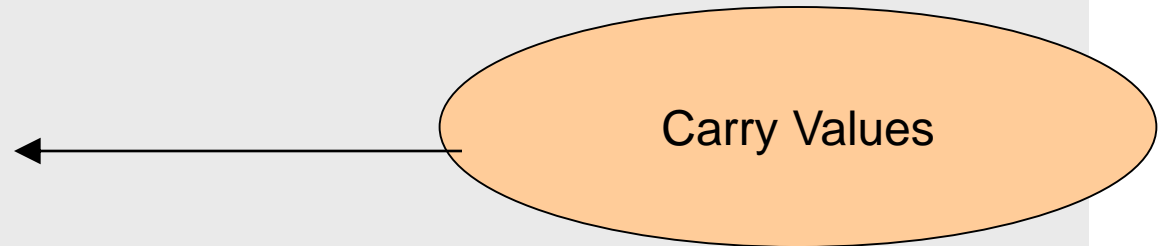
$$\begin{array}{r} \text{d- } (1010)_2 \\ - (0011)_2 \\ \hline (0111)_2 \end{array}$$

Arithmetic in Binary

Remember that there are only 2 digits in binary, 0 and 1

1 + 1 is 0 with a carry

$$\begin{array}{r} 111111 \\ 1010111 \\ +1001011 \\ \hline 10100010 \end{array}$$



Doing Binary Math.

$\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array}$	$\begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array}$
--	--	--	---

Adding With A Carry

c_{i-1}	0	0	0	0	1	1	1	1
$+a_i$	0	0	1	1	0	0	1	1
$+b_i$	<u>+0</u>	<u>+1</u>	<u>+0</u>	<u>+1</u>	<u>+0</u>	<u>+1</u>	<u>+0</u>	<u>+1</u>
F	0	1	1	10	1	10	10	11

Adding Answers

$\begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array}$	$\begin{array}{r} 1 \\ +11 \\ \hline 100 \end{array}$	$\begin{array}{r} 100 \\ +11 \\ \hline 111 \end{array}$	$\begin{array}{r} 101 \\ +110 \\ \hline 1011 \end{array}$
---	---	---	---

$\begin{array}{r} 110 \\ +111 \\ \hline 1101 \end{array}$	$\begin{array}{r} 1101 \\ +1100 \\ \hline 11001 \end{array}$	$\begin{array}{r} 101110 \\ +110001 \\ \hline 1011111 \end{array}$	$\begin{array}{r} 1100101 \\ +1110001 \\ \hline 11010110 \end{array}$
---	--	--	---

Negative Numbers

If a decimal number is positive, it has a plus sign; if a number is negative, it has a minus sign. In binary number systems, such as computers, it is not possible to use positive and negative symbols to represent the polarity of a number. **One method of representing a binary number as either a positive or negative value is to use an extra digit, or sign bit, at the MSB side of the number. In the sign bit position, a 0 indicates that the number is positive, and a 1 indicates a negative number** (Table 2)

Table 2 Signed Binary Numbers

Magnitude Sign		Decimal Value
Same as binary numbers	0111	+7
	0110	+6
	0101	+5
	0100	+4
	0011	+3
	0010	+2
	0001	+1
	0000	0
	1001	-1
	1010	-2
	1011	-3
	1100	-4
	1101	-5
	1110	-6
	1111	-7

Another method of expressing a negative number in a digital system is by using the **complement of a binary number**. To complement a binary number, **change all the 1s to 0s and all the 0s to 1s**. This is known as the **1's complement** form of a binary number. For example, the 1's complement of 1001 is 0110. The most common way to express a negative binary number is to show it as a **2's complement number**. The **2's complement is the binary number that results when 1 is added to the 1's complement**. This system is shown in Table 6-3. A zero sign bit means a positive number, whereas a 1 sign bit means a negative number. **Using the 2's complement makes it easier for the computer to perform mathematical operations**. The correct sign bit is generated by forming the 2's complement.

Table 6.3 1's and 2's Complement Representation of Positive and Negative Numbers

Signed Decimal	1's Complement	2's Complement	2's Complement-16 bit
+7	0111	0111	0000 0000 0000 0111
+6	0110	0110	0000 0000 0000 0110
+5	0101	0101	0000 0000 0000 0101
+4	0100	0100	0000 0000 0000 0100
+3	0011	0011	0000 0000 0000 0011
+2	0010	0010	0000 0000 0000 0010
+1	0001	0001	0000 0000 0000 0001
0	0000	0000	0000 0000 0000 0000
-1	1110	1111	1111 1111 1111 1111
-2	1101	1110	1111 1111 1111 1110
-3	1100	1101	1111 1111 1111 1101
-4	1011	1100	1111 1111 1111 1100
-5	1010	1011	1111 1111 1111 1011
-6	1001	1010	1111 1111 1111 1010
-7	1000	1001	1111 1111 1111 1001

Same as binary numbers

The computer knows that a number retrieved from memory is a negative number if the MSB is 1. Whenever a negative number is entered from a keyboard, the computer stores it as a 2's complement. What follows is the original number in true binary followed by its 1's complement, its 2's complement, and finally, its decimal equivalent

ROL: Rotate operand1 left (Çarpma)

- **ROL Operand1, Operand2**
- Algorithm: Tüm bitleri sola kaydırılır, giden bit CF'ye ayarlanır ve aynı bit en sağdaki konuma eklenir.
- Not: Operand1'i sola döndürülür. Döndürme sayısı Operand2 tarafından belirlenir.

Operand:

- memory, immediate
- REG, immediate
- memory, CL
- REG, CL

- **Example:**

```
MOV AL, 1Ch ; AL = 00011100b
```

```
ROL AL, 1 ; AL = 00111000b, CF=0.
```

```
RET
```

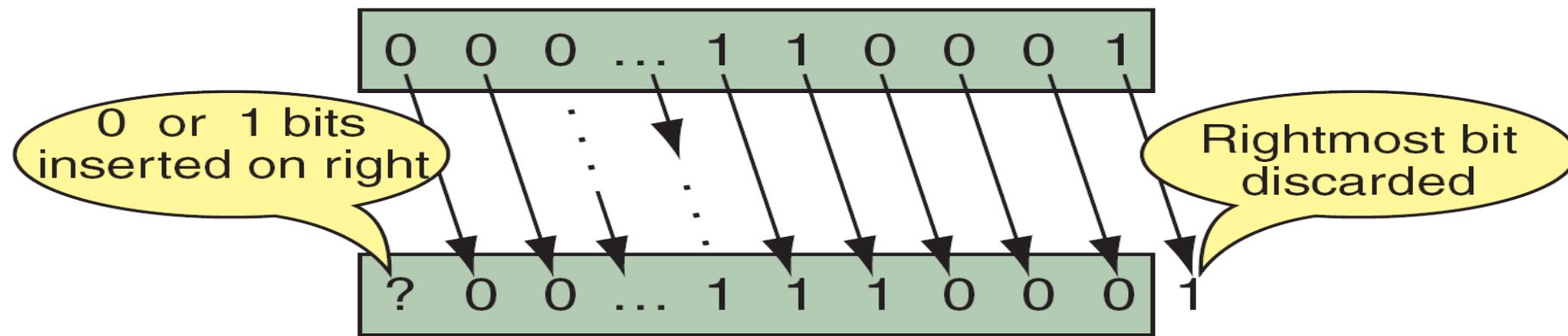


FIGURE Shift-right Operation

$2^{\text{shift value}}$	Divides by	Shift Operator
1	2	$\gg 1$
2	4	$\gg 2$
3	8	$\gg 3$
4	16	$\gg 4$
...
n	2^n	$\gg n$

Table

Divide by Shift

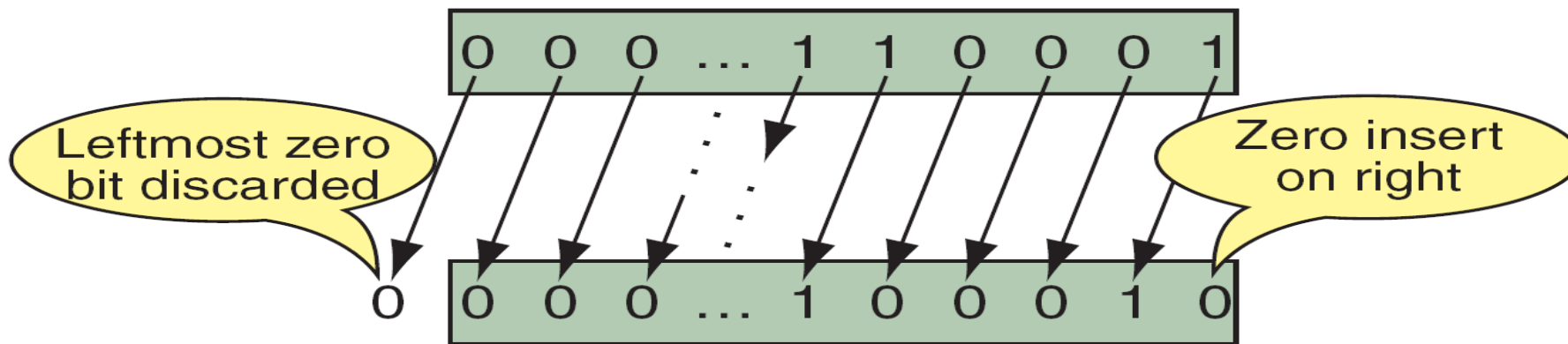


FIGURE Shift-left Operation

$2^{\text{shift value}}$	Multiplies by	Shift Operator
1	2	$\ll 1$
2	4	$\ll 2$
3	8	$\ll 3$
4	16	$\ll 4$
...
n	2^n	$\ll n$

Table

Multiply by Shift

Twos Complement Representation

Unsigned Binary Integers

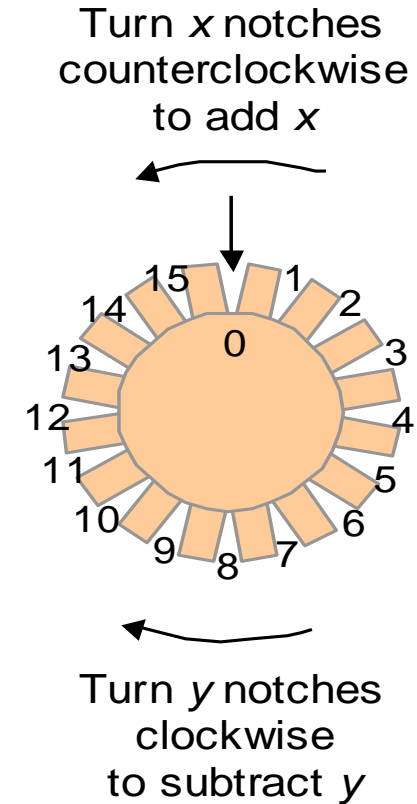
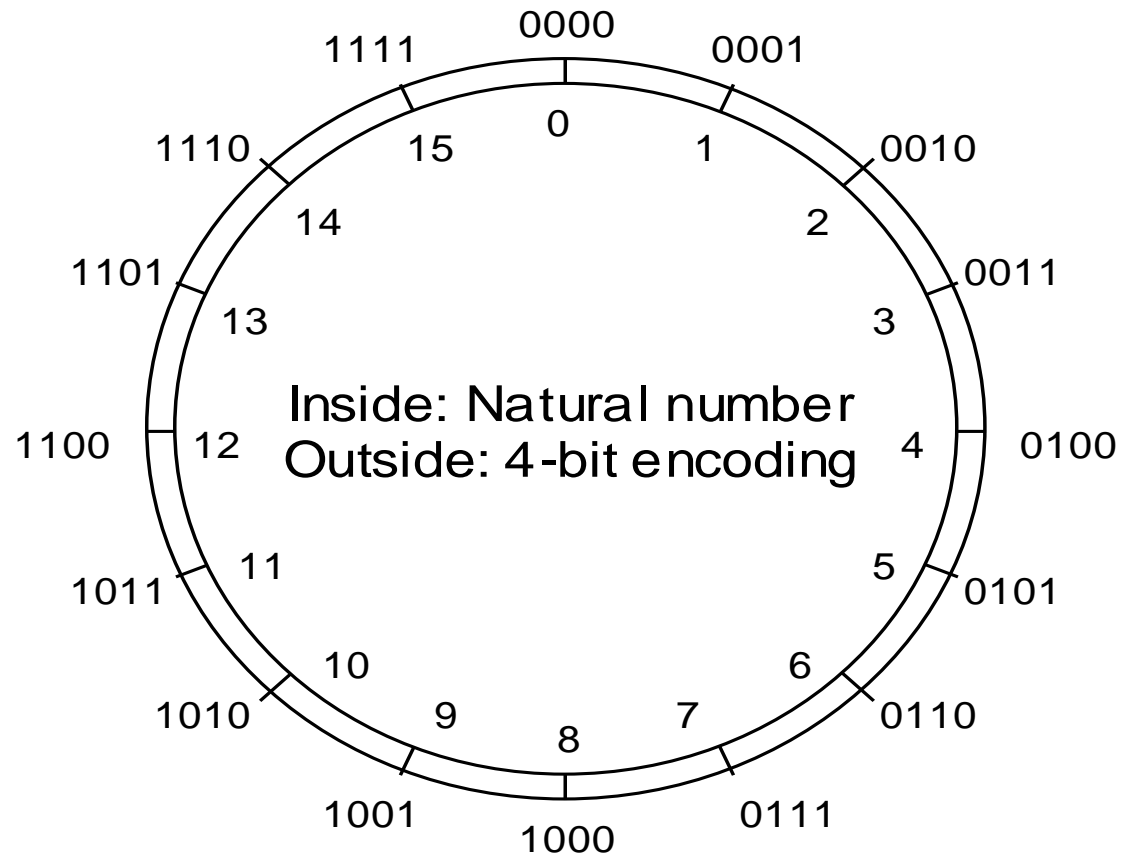


Figure Schematic representation of 4-bit code for integers in [0, 15].

Two's-Complement Representation

With k bits, numbers in the range $[-2^{k-1}, 2^{k-1} - 1]$ represented.
Negation is performed by inverting all bits and adding 1.

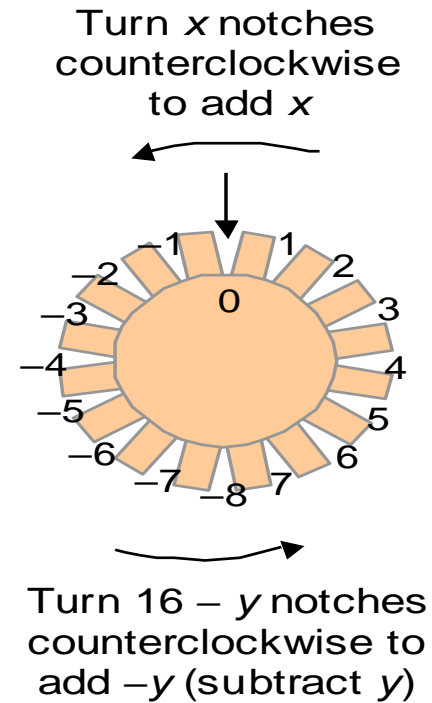
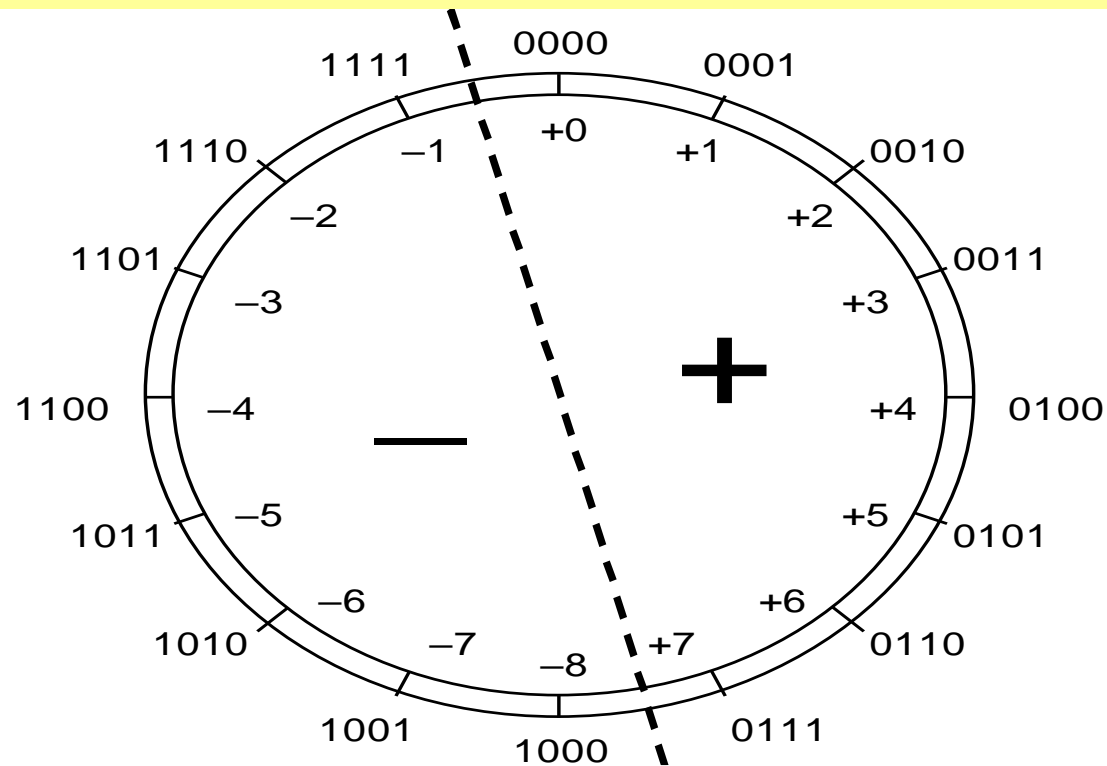


Figure Schematic representation of 4-bit 2's-complement code for integers in $[-8, +7]$.

Fixed-Point 2's-Complement Numbers

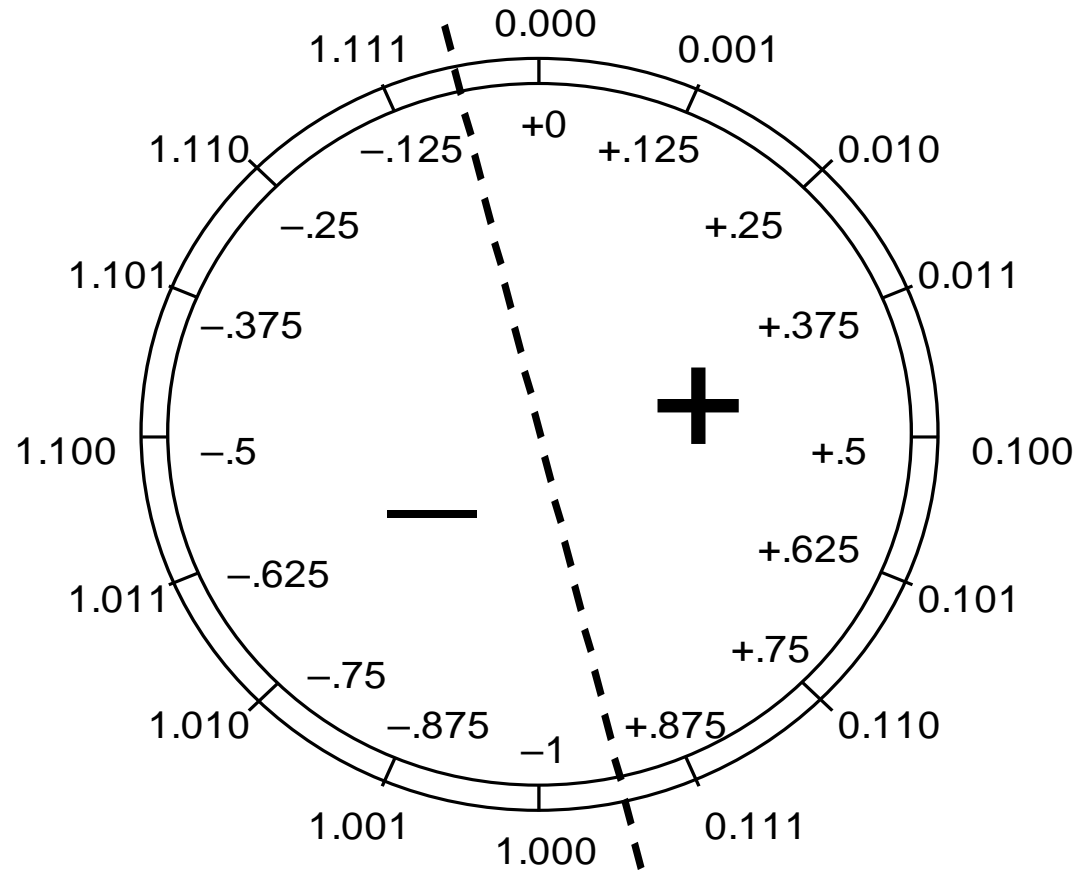
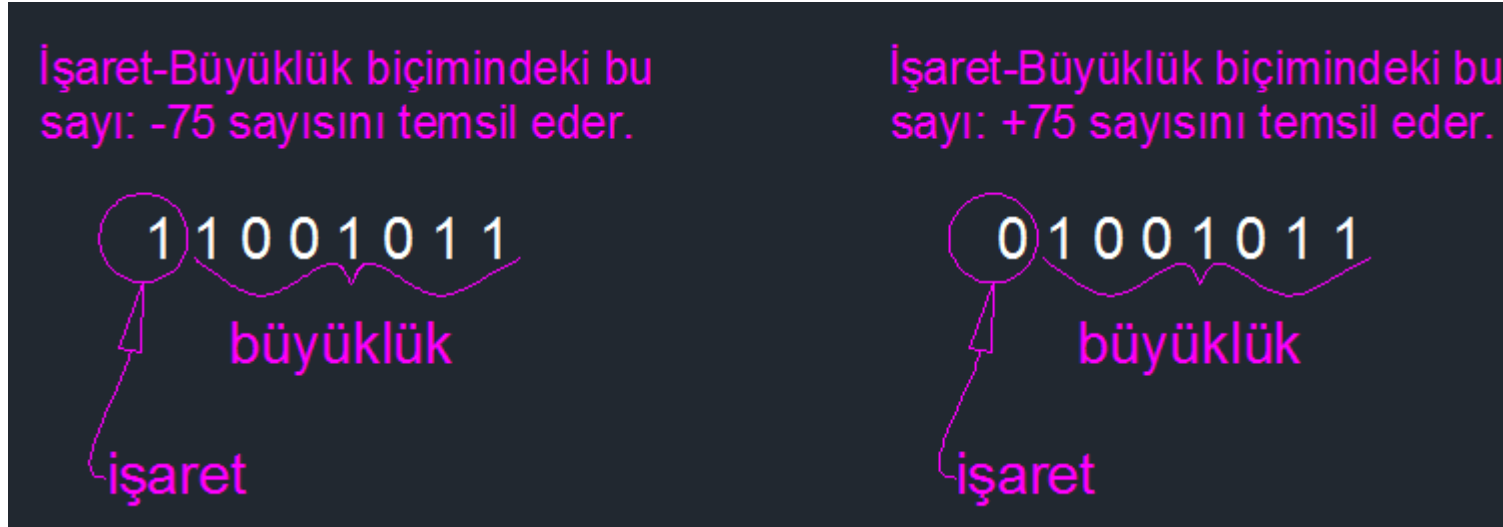
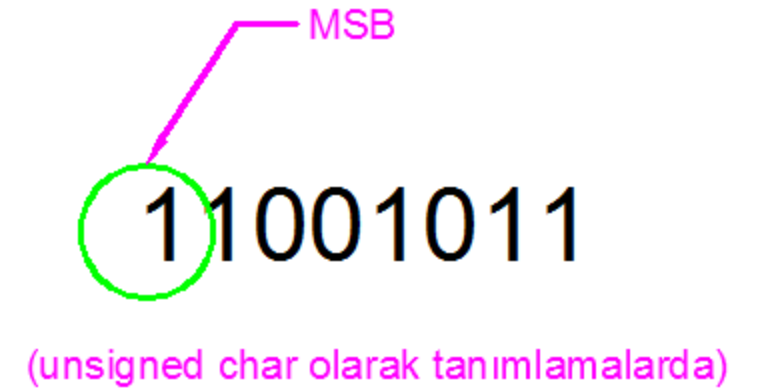
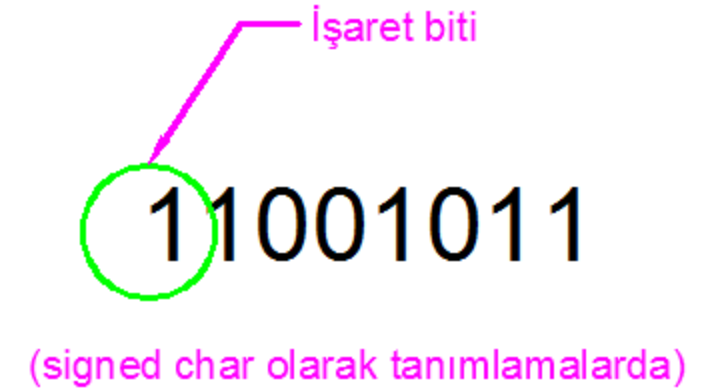


Figure Schematic representation of 4-bit 2's-complement encoding for (1 + 3)-bit fixed-point numbers in the range $[-1, +7/8]$.

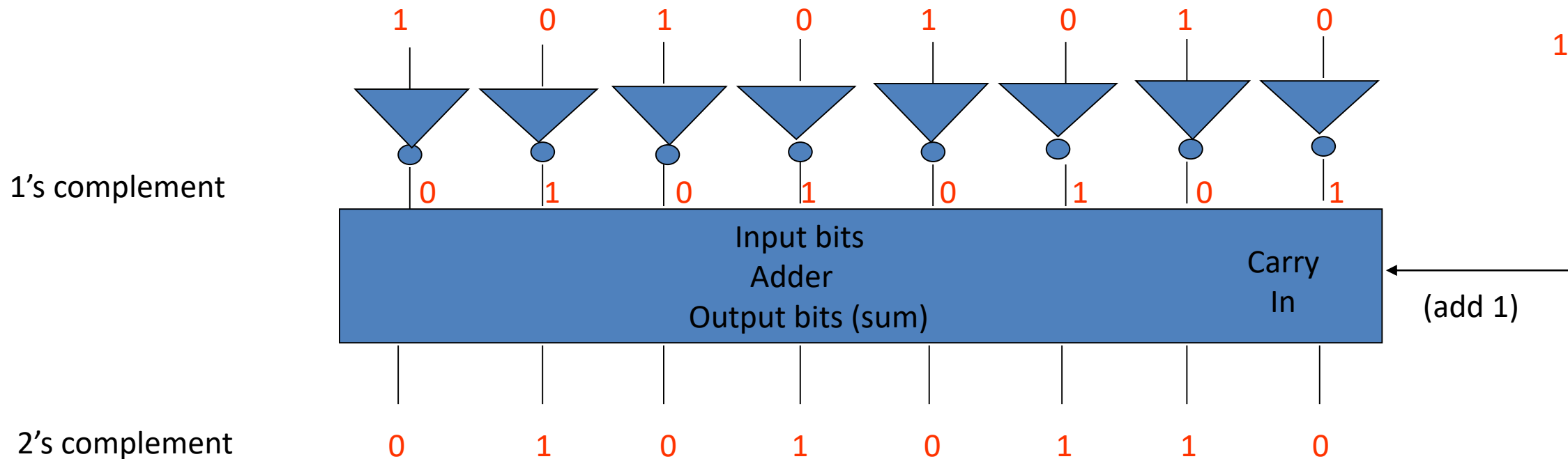
İşaretili binary sayılar

- İkili sayı sistemlerinde 1 byte eile (8 bit) 0-255 arası pozitif sayıları ifade edebiliriz, fakat
- •negatif sayıları ifade etmek için kullanıldığında en soldaki bit işaret bitidir.
- •Eğer işaret biti 1 ise sayı negatif, 0 ise sayı pozitiftir.



Complements of Binary Numbers

- 2's complement
- Find 1's complement and then add 1



İşaretli (signed) binary sayıların bir gösterim şekli de işaret-büyüklik biçiminde gösterimdir.

ÖRNEK: +19 ve -19 desimal sayılarını işaret-büyüklik formunda 1 byte binary sayı ile ifade ediniz.

- Çözüm:
- $(19)_{10} = 16 + 3 = 16 + 2 + 1 = 2^4 + 2^1 + 2^0$;
- İndisleme: 4, 3, 2, 1, 0
- $(10011)_2$
- $(+19)_d = (00010011)_b$
- $(-19)_d = (?)_b$ Not: 8, 16, 32, 64 bit düzeninde işlem yapılır.
- $(+19)_d = (00010011)_b \rightarrow$ ikil sayı sisteminde tersi alınır; 1 \rightarrow 0, 0 \rightarrow 1 konur. Sonra +1 eklenir.
- $(-19)_d = (11101100)_b + 1 = (1110\ 1101)_b$

Negatif Sayıların ikil sayı sistemine çevrilmesi

- Önce negatif işareti olmadan ondalık sayıyı ikili sayı sistemine çevrilir.
- En sağdan başlayarak dörterli gruplara ayrılır.
- En soldaki bit 1 ise 0'lar ilave ederek dörtlü gruba tamamlanır ya da sıfırlardan oluşan yeni bir dörtlü grup ilave edilir. (8, 16, 32, 64, ...)
- Elde ettiğiniz ikili sayı sisteminde 1 yerine 0; 0 yerine 1 yazılır.
- Son olarak yeni ikili sayı sistemi +1 ile toplanır.

$$\begin{aligned} +18 &= 0001\ 0010 \text{ (twos complement)} \\ \text{bitwise complement} &= 1110\ 1101 \\ &+ \quad \quad \quad \underline{1} \\ &1110\ 1110 = -18 \end{aligned}$$

$$\begin{aligned} -18 &= 1110\ 1110 \text{ (twos complement)} \\ \text{bitwise complement} &= 0001\ 0001 \\ &+ \quad \quad \quad \underline{1} \\ &0001\ 0010 = +18 \end{aligned}$$

Negatif sayılar

- Byte: 8 bit data tanımlar; 8 bitlik bellek gözünü işaret eder.

- Örnek db -4

$$(4)d = (0000\ 0100)b$$

$$(-4)d = \text{Binary } (4) \text{ Tersisi } +1 :$$

$$(-4)d = 1111\ 1011 +1 = 1111\ 1100 = (\text{FC})h$$

- Örnek dW -4

$$(4)d = (0000\ 0000\ 0000\ 0100)b$$

$$(-4)d = \text{Word}(4) \text{ Tersisi } +1$$

$$(-4)d = (1111\ 1111\ 1111\ 1011)b +1 = (1111\ 1111\ 1111\ 1100)b = (\text{FFFC})h$$

Overflow

- Taşma, toplamdaki bit sayısı, toplanan ve artırılan bitlerin sayısını aştığında meydana gelir.
- Taşma, yanlış işaret ile gösterilir. Yalnızca her iki sayı da pozitif olduğunda veya her iki sayı da negatif olduğunda oluşur



Coding

Kodlama

- 0 ile 9 arasındaki ikil sayı sistemi kullanılır.
- Genel olarak kodlama, görülebilen, okunabilen, yazı, sayı ve işaretlerin değiştirilmesi olarak tanımlanır.
- Binary Coded Decimal Kodlaması(BCD) sisteminde her desimalsayı karakteri için, dört bit kullanılır.

<u>Decimal</u>	BCD
25	0010 0101
32	0011 0010
679	0110 0111 1001
2571	0010 0101 0111 0001

<u>Decimal Sayıları</u>	BCD Kodu
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Characters and Other Codes

- To represent information as strings of alpha-numeric characters.
- **Binary Coded Decimal (BCD)**
 - **Used to represent the decimal digits 0 - 9.**
 - 4 bits are used.
 - Each bit position has a weight associated with it (***weighted code***).
 - Weights are: 8, 4, 2, and 1 from MSB to LSB (called 8-4-2-1 code).
 - BCD Codes:
0: 0000 1: 0001 2: 0010 3: 0011 4: 0100
5: 0101 6: 0110 7: 0111 8: 1000 9: 1001
 - Used to encode numbers for output to numerical displays
 - Used in processors that perform decimal arithmetic.
 - **Example:** $(9750)_{10} = (1001011101010000)_{BCD}$

BCD kodlama örnekleri

Örnek: $(10100110)_2$ sayısını BCD koduna çeviriniz.

- Binary sayıyı ilk olarak desimale çevirelim;
- $2+4+32+128=166$
- $(10100110)_2 = (166)_{10}$
- BCD koduna çevirirsek
- $(000101100110)_{BCD}$

Örnek: $(1010110101)_2$ sayısını BCD koduna çeviriniz.

- Binary sayıyı ilk olarak desimale çevirelim;
- $1+4+16+32+128+512=693$
- $(1010110101)_2 = (693)_{10}$
- BCD koduna çevirirsek
- $(011010010011)_{BCD}$ bulunur.

0 -12 Decimal sayıları için Gray kodlaması

- Birinci bit aynen aşağıya indirilir,
- ikinci bit aşağıya indirilen bit ile toplanır ve sonuç hemen aşağıya indirilen birinci bitin sağına yazılır.
- Üçüncü bit, aşağıya yazılan ikinci bit ile toplanır ve ikinci bit 'in yanına yazılır.
- son bite kadar işlem böyle devam ettirilir.

Örnek:(011010111101)₂sayısını graykoduna çeviriniz.

- **İlk rakam aynen alınır,**
- **Diğer basamaklarda sayı değişiyorsa 1, sayı değişmiyorsa 0 alınarak sonuca ulaşılır.**
- (011010111101)₂
- (010111100011)_{gray}

Örnek:(1110001101101)₂sayısını graykoduna çeviriniz.

- İlk rakam aynen alınır,
- Diğer basamaklarda sayı değişiyorsa 1, sayı değişmiyorsa 0 alınarak sonuca ulaşılır.
- (1110001101101)₂
- (1001001011011)_{gray}

Örnek: 297 decimalsayısını graykodu ile kodlayınız?

(297)₁₀= (100101001)₂

(100101001)₂=(110111101)_{gray}

<u>Decimal</u>	<u>Binary</u>	<u>Gray</u>
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010

Gray Kodu

- Gray kodunun, sütun taraması esasına göre çalışan cihazlarda oldukça geniş bir kullanım alanı vardır.
- Geçişler sırasında hatayı minimuma indirmek için geliştirilmiş bir koddur. İsmi mucidi Frank GRAY'den alır.
- Özelliği, Ardarda iki sayı arasında sadece tek bit değişikliğidir.
- Normal Binary kodunda ardışık sayılarda çoğu kez birden fazla bit değişikliği söz konusudur. Devrelerde 0-1 durumları arasındaki değişim sırasında okuyucu farklı değerler okuyabilir.
- Gray kodunda sadece tek bit değiştiğinden, ve binary sayı olduğundan, gray kodunda bu hata ortadan kaldırılmıştır.

Graykodu ile kodlanmış sayıyı tekrar elde etme örnekleri

Örnek: $(01011101)_{gray}$ graykodlu sayıyı binary'e çevirelim.

- $(01011101)_{gray}$
- $(01101001)_2$
- İlk sayı aynen aşağı alınır,
- Sağındaki diğer sayılar
- yukarıdaki sayı 1 olduğunda rakamın değiştiğine,
- 0 olduğunda değişmediğine alamet olarak değerlendirilir.

Örnek: $(10101100111)_{gray}$

- graykodlu sayıyı binary'e çevirelim.
- $(10101100111)_{gray}$
- $(11001000101)_2$

Characters and Other Codes

- ***Gray Code***
 - ***Cyclic code***: A circular shifting of a code word produces another code word.
 - ***Gray code***: A cyclic code with the property that two consecutive code words differ in only 1 bit (the *distance* between the two code words is 1).
 - Gray code for decimal numbers 0 - 15: See Table 1.12

ASCII Character Code

ASCII Character Code

- Digital computers handle not only the numbers, but also other characters or symbols, such as the letters of the alphabet.
- An alphanumeric character set is a set of elements that includes the 10 decimal digits, the 26 letters of the alphabet, and a number of special characters. Such a set contains between 36 and 64 elements if only capital letters are included, or between 64 and 128 elements if both uppercase and lowercase letters are included.
- The American Standard Code for Information Interchange (ASCII) uses seven bits to code 128 characters, as shown in Table 1.
- The 7 bits of the code are designated by b1 through b7, with b7 the most significant bit.
- The letter A, for example, is represented in ASCII as (1000001)_b (column 100, row 0001).
- The ASCII code also contains 94 graphic characters that can be printed and 34 nonprinting characters used for various control functions.
- The graphic characters consist of the 26 uppercase letters (A through Z), the 26 lowercase letters (a through z), the 10 numerals (0 through 9), and 32 special printable characters, such as %, *, and \$.

There needs a standard way

- ASCII code: **American Standard Code for Information Interchange**
 - ASCII codes represent [text](#) in [computers](#), [communications](#) equipment, and other devices that use text.
 - ASCII kodları, bilgisayarlardaki, klavyedeki tüm tuşların 8 bitlik değerini temsil eder. Böylece klavyedeki tuşlar ile bellek arasındaki iletişim sayısal olarak yapılmış olur.
 - 128 characters:
 - 33 are non-printing [control characters](#) (now mostly obsolete)^[7] that affect how text and space is processed
 - 94 are printable characters
 - [space](#) is considered an invisible graphic

ASCII Coding

- Bilgisayarda 0 ve 1'lerle karakterleri ifade etmek için ikili kodlama sistemleri kullanılır.
- ASCII, karakterleri göstermek için sekiz bit (bir bayt) kullanır. Yeni geliştirilen Unicode ise karakterleri göstermek için onaltı bit kullanır:
- ASCII (American Standard Code for Information Interchange - bilgi deęiřimi için Amerikan standart kodlaması)
- EBCDIC (Extended Binary Coded Decimal Interchange Code - genişletilmiş ikili kodlamalı onluk sistem deęiřtirme kodlaması): IBM řirketi tarafından ana bilgisayarlar da kullanılmak için geliştirilmiştir.
- Unicode: Çince ve Japonca gibi dilleri desteklemek için tasarlanmış onaltı bit kullanan kodlamadır. Bu diller sekiz bit kullanan ASCII ve EBCDIC kodlamaları ile gösterilemeyecek kadar çok sayıda karakter kullanılır. Unicode kodlaması, IBM, Apple ve Microsoft řirketlerinin destekledięi Unicode řirketi tarafından geliştirilmiştir.

■ Kodlama Kullanımı

ASCII Kiřisel bilgisayarlar

EBCDIC Anabilgisayarlar

Unicode Uluslararası diller

- Klavyede bir tuřa bastığınız zaman, tuřa karřılık gelen karakter, bilgisayarın anlayabileceęi bir dizi bite çevirilir. Örneęin, klavyede A harfine basmak bilgisayar bunu $(41)_h = (01000001)_b$ ASCII koduna çevirir.
- Dökümanlar deęiřik bilgisayarlar veya uygulama programları tarafından paylařıldıęı zaman, aynı kodlama sistemi kullanılmalıdır.

ASCII Character Set

American Standard Code for Information Interchange (ASCII)

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII Character Codes (2)

Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char
20	(Space)	30	0	40	@	50	P	60	'	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	\$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	'	37	7	47	G	57	W	67	g	77	w
28	(38	8	48	H	58	X	68	h	78	x
29)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	
2D	-	3D	=	4D	M	5D]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

The ASCII Character set: characters 32 – 127.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Bits as Codes

- Ascii kodu bizim bilgisayarda görsel olarak girdiğimiz karakter, harf ve rakamların bilgisayar dilindeki temsil edilme şeklidir diyebiliriz.
- Yani bilgisayarımızın o karakteri, harfi veya rakamı belleğinde saklama biçimidir,
- Açılımı ASCII (American Standard Code for Information Interchange) olan
- Bu kodlama sistemi ilk olarak telgraf kodlarında kullanılmıştır
- Klavyedeki her tuşun 8 bitlik bir karşılığı vardır.
- İkili Kodlar her harf, rakam ve özel karakteri temsil eder.
- ASCII: Her karakter benzersiz bir 8 bitlik koddur
- ASCII: 26 harf, 10 hane, özel karakterler için 256 benzersiz kod
- Unicode: 100.000'den fazla benzersiz karakteri destekler.
- Klavyeden girilen her karakter bilgisayara ASCII kodu ile giriş yapar.

ASCII	SYMBOL	ASCII	SYMBOL
00110000	0	01001110	N
00110001	1	01001111	O
00110010	2	01010000	P
00110011	3	01010001	Q
00110100	4	01010010	R
00110101	5	01010011	S
00110110	6	01010100	T
00110111	7	01010101	U
00111000	8	01010110	V
00111001	9	01010111	W
01000001	A	01011000	X
01000010	B	01011001	Y
01000011	C	01011010	Z
01000100	D	00100001	!
01000101	E	00100010	"
01000110	F	00100011	#
01000111	G	00100100	\$
01001000	H	00100101	%
01001001	I	00100110	&
01001010	J	00101000	(
01001011	K	00101001)
01001100	L	00101010	*
01001101	M	00101011	+



Hamming Codes

Hamming Codes (1)

- Multiple check bits are employed.
- Each check bit is defined over (or covers) a subset of the information bits.
- Subsets overlap so that each information bit is in at least two subsets.
- d_{min} is equal to the weight of the minimum-weight nonzero code word.
- **Hamming Code 1**
 - $d_{min} = 3$, single error correction code.
 - Let C be the set of all code words:
 - an error word with single error: c_e
 - the correct code word for the error word: cthen, $d(c_e, c) = 1$ and $d(c_e, w) > 1$ for all other $w \in C$
 - So, a single error can be detected and corrected by finding out the code word which differs in 1 bit position from the error word.

Hamming Codes (2)

- A code word consists of 4 information bits and 3 check bits:

$$c = (i_3 i_2 i_1 i_0 c_2 c_1 c_0)$$

- Each check bit covers:

$$c_2: i_3, i_2, i_1 \quad c_1: i_3, i_2, i_0 \quad c_0: i_3, i_1, i_0$$

- This relationship is specified by the *generating matrix*, G :

$$G = \begin{bmatrix} 1000111 \\ 0100110 \\ 0010101 \\ 0001011 \end{bmatrix} = \begin{bmatrix} 1000 p_{11} p_{12} p_{13} \\ 0100 p_{21} p_{22} p_{23} \\ 0010 p_{31} p_{32} p_{33} \\ 0001 p_{41} p_{42} p_{43} \end{bmatrix}$$

- Encoding of an information word i to produce a code word, c :

$$c = iG$$

Hamming Codes (3)

- Decoding can be done using the *parity-check matrix*, H :

$$H = \begin{bmatrix} p_{11} & p_{21} & p_{31} & p_{41} & 1 & 0 & 0 \\ p_{12} & p_{22} & p_{32} & p_{42} & 0 & 1 & 0 \\ p_{13} & p_{23} & p_{33} & p_{43} & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

- H matrix is can be derived from G matrix.
- An n -tuple c is a code word generated by G if and only if
$$Hc^T = 0$$
- Let d be a data word corresponding to a code word c , which has been corrupted by an error pattern e . Then

$$d = c + e$$

- Decoding:
 - Compute the syndrome, s , of d using H matrix.
 - s tells the position of the erroneous bit.

Hamming Codes (4)

– Computation of the syndrome:

$$s = Hd^T$$

$$= H(c + e)^T$$

$$= Hc^T + He^T$$

$$= 0 + He^T$$

$$= He^T$$

- Note: All computations are performed using *modulo-2 arithmetic*.

Hamming Codes (5)

- **Hamming Code 2)**

- $d_{min} = 4$, single error correction and double-error detection.
- The generator and parity-check matrices are:

$$G = \begin{bmatrix} 10000111 \\ 01001110 \\ 00101101 \\ 00011011 \end{bmatrix}$$

$$H = \begin{bmatrix} 01111000 \\ 11100100 \\ 11010010 \\ 10110001 \end{bmatrix}$$

Odd-weight-column code:

- H matrix has an odd number of ones in each column.
- Example: Hamming Code 2.
- Has many properties; single-error correction, double-error detection, multiple-error detection, low cost encoding and decoding, etc.

Hamming Codes (6)

- Hamming codes are most easily designed by specifying the H matrix.
- For any positive integer $m \geq 3$, there exists an (n, k) SEC Hamming code with the following properties:
 - Code length: $n = 2^m - 1$
 - Number of information bits: $k = 2^m - m - 1$
 - Number of check bits: $n - k = m$
 - Minimum distance: $d_{min} = 3$
- The H matrix is an $n \times m$ matrix with all nonzero m -tuples as its column.
- A possible H matrix for a (15, 11) Hamming code, when $m = 4$:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Hamming Codes (7)

- **Example:** A Hamming code for encoding five ($k = 5$) information bits.
 - Four check bits are required ($m = 4$). So, $n = 9$.
 - A (9, 5) code can be obtained by deleting six columns from the (15,11) code shown above.
 - The H and G matrices are:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

ROR: Rotate operand1 right (Bölme)

- **ROR Operand1, Operand2**
- Algorithm: Tüm bitleri sağa kaydırılır, giden bit CF'ye ayarlanır ve aynı bit en soldaki konuma eklenir.
- Not: Operand1'i sağa döndürülür. Döndürme sayısı Operand2 tarafından belirlenir.

Operand1, Operand2:

- memory, immediate
- REG, immediate
- memory, CL
- REG, CL

• **Example:**

```
MOV AL, 1Ch ; AL = 00011100b
ROR AL, 1 ; AL = 00001110b, CF=0.
RET
```

Decimal Number System

İndis sağdan başlar; 0,1,2,3,4, ...

Neden indis 0'dan başlar? Birler basamağı $10^0=1$

Herbir katsayı ile 10'indis .arpılıp toplandığında ondalık sayının kendisi elde edilir.

$$3 \times 10^4 + 5 \times 10^3 + 4 \times 10^2 + 6 \times 10^1 + 2 \times 10^0$$

35462



Karekod(barkod)

Karekod(barkod)

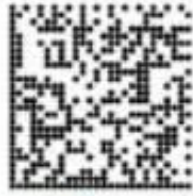
- Karekod, kare veya dikdörtgen biçimlerde basılabilen 2 boyutlu barkodun ismidir. Kare veya dikdörtgen şeklinde olan bu yapının genel adı ise Data Matrixtir.
- Karekod kelimesi ilk olarak, Beşeri Tıbbi Ürünler Barkod Uygulama Kılavuzunda kullanılmıştır.
- Ülkemizdeki ilk uygulama alanı, ilaç sektörüdür.
- Karekodbarkodların başlıcaları; QR Kod, Data MatrixKod, AztekKod



QR-Code



DataMatrix



Cool-Data-Matrix



Aztec



000133



Trillcode



Quickmark



Shotcode



mCode



Beetagg





Error Detection Codes and Correction Codes

Error Detection Codes and Correction Codes

- **An error:** An incorrect value in one or more bits.
- **Single error:** An incorrect value in only one bit.
- **Multiple error:** One or more bits are incorrect.
- Errors are introduced by hardware failures, external interference (noise), or other unwanted events.

- **Error detection/correction code:** Information is encoded in such a way that a particular class of errors can be detected and/or corrected.

- Let I and J be n -bit binary information words
 - $w(I)$: the number of 1's in I (*weight*)
 - $d(I, J)$: the number of bit positions in which I and J differ (*distance*)
- **Example:** $I = (01101100)$ and $J = (11000100)$
 - $w(I) = 4$ and $w(J) = 3$
 - $d(I, J) = 3$.

Error Detection Codes and Correction Codes

- General Properties

- *Minimum distance*, d_{min} , of a code C: for any two code words I and J in C,

$$d(I, J) \geq d_{min}$$

- A code provides t *error correction* plus *detection of s additional errors* if and only if the following inequality is satisfied.

$$2t + s + 1 \leq d_{min} \quad (1.25)$$

- **Example:**

- Single-error detection (SED): $s = 1, t = 0, d_{min} = 2$.

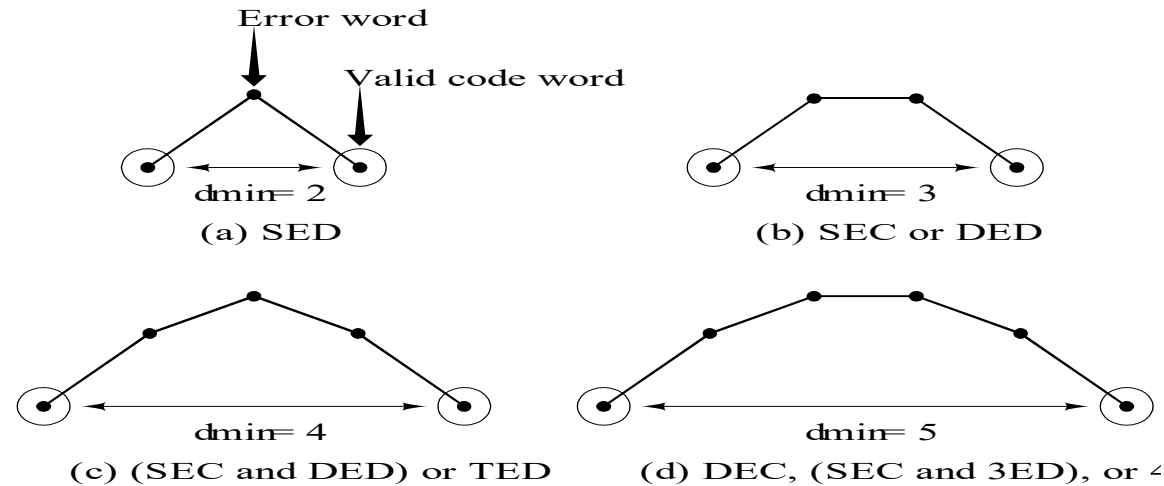
- Single-error correction (SEC): $s = 0, t = 1, d_{min} = 3$.

- Single-error correction and double-error detection (SEC and DED):

$$s = t = 1, d_{min} = 4.$$

Error Detection Codes and Correction Codes

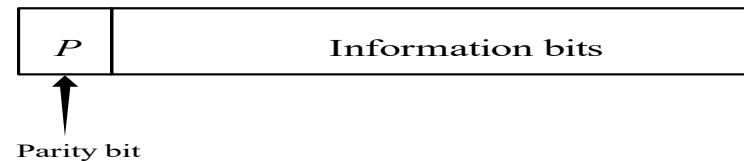
- Relationship between the minimum distance between code words and the ability to detect and correct errors:



Error Detection Codes and Correction Codes

- **Simple Parity Code**

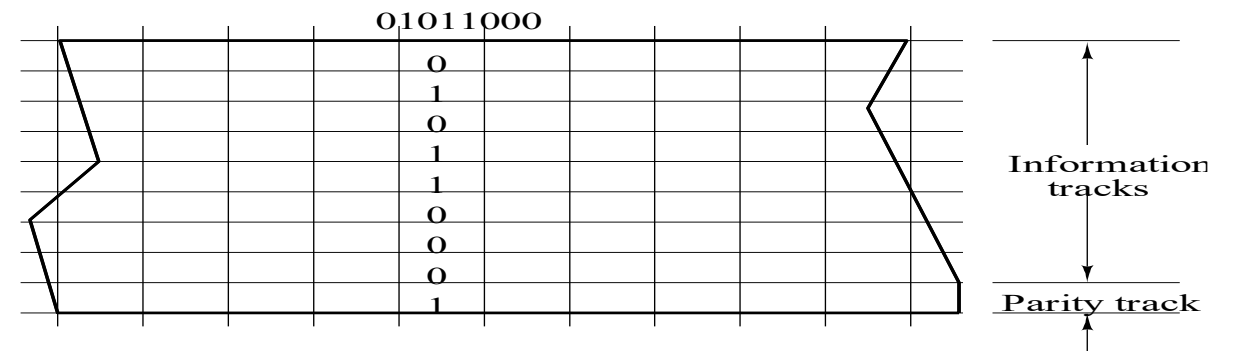
- Concatenate (|) a *parity bit*, P , to each code word of C .



- *Odd-parity code*: $w(P|C)$ is odd.

- *Even-parity code*: $w(P|C)$ is even.

- Parity coding on magnetic tape:



Error Detection Codes and Correction Codes

- **Example:** Odd-parity code for ASCII code characters:

Character	ASCII Code	Odd-parity Code
0	0110000	10110000
X	1011000	01011000
=	0111100	1111100
BEL	0000111	00000111

- Error detection: Check whether a code word has the correct parity.
- Single-error detection code ($d_{\min} = 2$).
- **Two-out-of-Five Code**
 - Each code word has exactly two 1's and three 0's.
 - Detects single errors and multiple errors in adjacent bits.

Check Your Progress

- $2 \times 101 + 8 \times 100$ is equal to (a) 10 (b) 280 (c) 2.8 (d) 28
- The binary number 1101 is equal to the decimal number (a) 13 (b) 49 (c) 11 (d) 3
- The decimal 17 is equal to the binary number (a) 10010 (b) 11000 (c) 10001 (d) 01001
- The sum of 11010 + 01111 equals (a) 101001 (b) 101010 (c) 110101 (d) 101000
- The difference of 110 – 010 equals (a) 001 (b) 010 (c) 101 (d) 100
- The 1's complement of 10111001 is (a) 01000111 (b) 01000110 (c) 11000110 (d) 10101010
- The 2's complement of 11001000 is (a) 00110111 (b) 00110001 (c) 01001000 (d) 00111000
- The binary number 101100111001010100001 can be written in octal as
(a) 54712308 (b) 54712418 (c) 26345218 (d) 231625018
- The binary number 10001101010001101111 can be written in hexadecimal as
(a) AD46716 (b) 8C46F16 (c) 8D46F16 (d) AE46F16
- The BCD number for decimal 473 is
(a) 111011010 (b) 1110111110101001 (c) 010001110011 (d) 010011110011

Usage Notes

- A lot of slides are adopted from the presentations and documents published on internet by experts who know the subject very well.
- I would like to thank who prepared slides and documents.
- Also, these slides are made publicly available on the web for anyone to use
- If you choose to use them, I ask that you alert me of any mistakes which were made and allow me the option of incorporating such changes (with an acknowledgment) in my set of slides.

Sincerely,

Dr. Cahit Karakuş

cahitkarakus@gmail.com