

# YMT213 Vocational English YMH213 Mesleki İngilizce

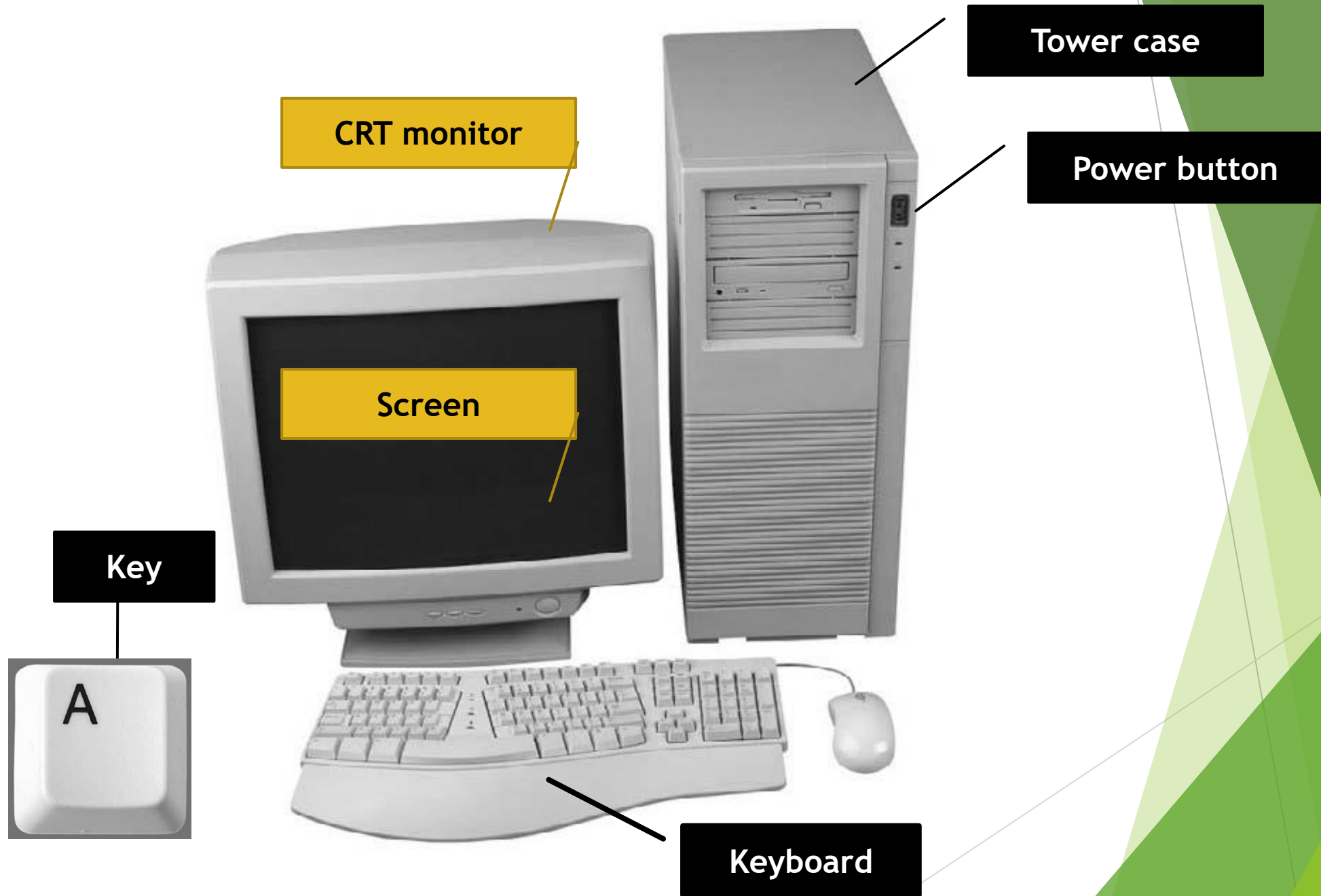
Assoc. Prof. Dr. Murat KARABATAK

# Hardware

- ▶ battery
- ▶ data cable
- ▶ Plug
- ▶ socket
- ▶ desktop computer
- ▶ digital camera
- ▶ fax machine
- ▶ Mouse
- ▶ PDA (Personal Digital Assistant)
- ▶ docking station
- ▶ printer
- ▶ projector
- ▶ Scanner
- ▶ mobile phone
- ▶ laptop computer (or notebook)

## 1.2 Some useful verbs

- ▶ 1. recharge
  - ▶ 2. click on
  - ▶ 3. dial
  - ▶ 4. give
  - ▶ 5. move
  - ▶ 6. print out
  - ▶ 7. send and receive
  - ▶ 8. take some
- ▶ a. digital photos
  - ▶ b. faxes
  - ▶ c. a number on your mobile phone
  - ▶ d. a presentation
  - ▶ e. something with the mouse
  - ▶ f. battery
  - ▶ g. mouse
  - ▶ h. twenty pages



# Computer Keyboard

Num Lock, Caps Lock, and Scroll Lock indicators

Function keys

Control keys



Keyboard

Wrist pad

Arrow keys

Keypad

# Keys

- ▶ 1. To go back one space, hit the \_\_\_\_\_.
- ▶ 2. To change to capital letters, press the \_\_\_\_\_.
- ▶ 3. To change the capital letters permanently, hit the \_\_\_\_\_.
- ▶ 4. To insert a tabulation, press the \_\_\_\_\_.
- ▶ 5. To activate the "Ctrl" functions, press the \_\_\_\_\_.
- ▶ 6. To activate the "alt" functions, hit the \_\_\_\_\_.
- ▶ 7. To stop the computer doing something, you can press the \_\_\_\_\_.
- ▶ 8. Select the text you want to remove, and hit the \_\_\_\_\_.

# YMT213 Vocational English

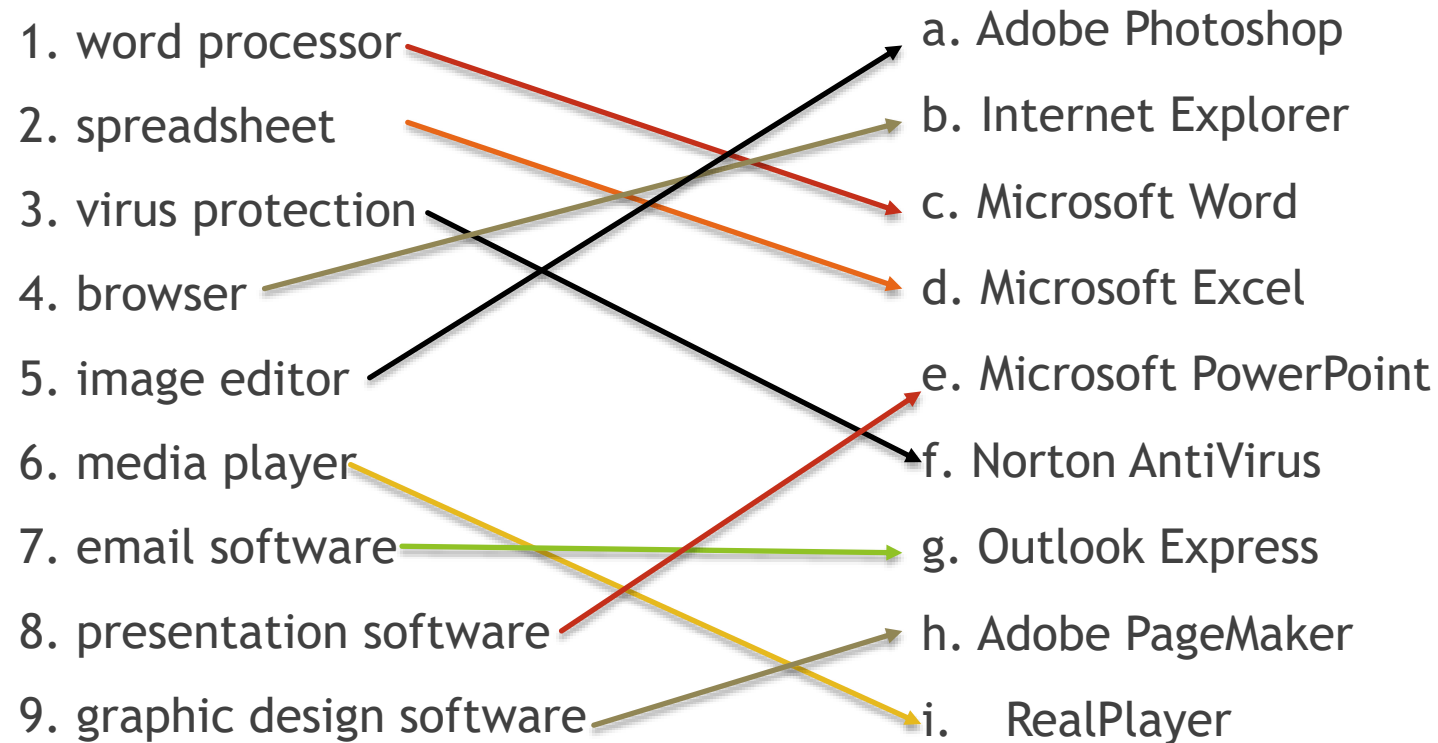
Assoc. Prof. Dr. Murat KARABATAK

# Software

1. Word Processor
2. Spreadsheet
3. Virus Protection
4. Browser
5. Image Editor
6. Media Player
7. Email Software
8. Presentation Software
9. Graphic Design Software
10. Operating Systems
11. Application
12. Folder
13. File
14. Shortcut



# Let's match the descriptions on the left with these famous applications



1 c, 2 d, 3 f, 4 b, 5 a, 6 i, 7 g, 8 e, 9 h

# Let's match the type of software with the definition

- 
- The diagram shows five software types on the left and five definitions on the right. Red arrows indicate the correct matches: 1. trial version to e, 2. shareware to d, 3. freeware to b, 4. home-use version to a, and 5. professional version to c.
- ▶ 1. trial version
  - ▶ 2. shareware
  - ▶ 3. freeware
  - ▶ 4. home-use version
  - ▶ 5. professional version
- ▶ a. A **simplified** version which is cheaper to buy.
  - ▶ b. Software which is in the **public domain**.  
Anybody can use it without paying.
  - ▶ c. The **full version** with all the features.
  - ▶ d. You can try it for a while for free. Then if you want to keep using it, you are expected to pay a small **fee** to the writer.
  - ▶ e. You can use it for free for a while (often a month). When the **trial period** finishes, you have to pay, or the program will **de-activate**.

# YMT/YMH 213

# Vocational English

Assoc. Prof. Dr. Murat KARABATAK

# Boolean Algebra

Boolean algebra, which forms the mathematical basis of digital circuit design, is a symbolic mathematical logic system that depicts the relationships between propositions or objects.

## Basic Boolean Algebraic Identities

Additive	Multiplicative
$A + 0 = A$	$0A = 0$
$A + 1 = 1$	$1A = A$
$A + A = A$	$AA = A$
$A + \bar{A} = 1$	$A\bar{A} = 0$

Dijital devre tasarımının matematiksel temelini oluşturan Boole cebri, önermeler veya nesneler arasındaki ilişkileri gösteren sembolik bir matematiksel mantık sistemidir.

# Boolean Algebra

Boolean algebra, which provided the essential foundation for the design of circuits used in digital computers, applies in cases where the truth values - in other words truth or falseness of a logical proposition - are used as variables rather than numerical quantities as in algebra.

Dijital bilgisayarlarda kullanılan devrelerin tasarımı için gerekli temeli sağlayan Boole cebri, doğruluk değerlerinin - diğer bir deyişle mantıksal önermenin doğruluğu veya yanlışlığı - cebirde olduğu gibi sayısal büyüklükler yerine değişken olarak kullanıldığı durumlarda uygulanır.

# Boolean Algebra

This feature, which is an important advantage of Boolean algebra, allows making an operation by using propositions, the truth value of which may be 1 (one) or 0 (zero).

Boole cebirinin önemli bir avantajı olan bu özellik, doğruluk değeri 1 (bir) veya 0 (sıfır) olabilen önermeler kullanarak işlem yapılmasına izin verir.

## Properties of Boolean Algebra

PROPERTY	AND	OR
Commutative	$AB = BA$	$A + B = B + A$
Associative	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive	$A(B + C) = (AB) + (AC)$	$A + (BC) = (A + B)(A + C)$
Identity	$A1 = A$	$A + 0 = A$
Complement	$A(A') = 0$	$A + (A') = 1$
De Morgan's law	$(AB)' = A' \text{ OR } B'$	$(A + B)' = A'B'$

# Boolean Algebra

When two logical propositions are connected with AND ( $\wedge$ ) or OR ( $\vee$ ) logical conjunction, the truth value of compound logical proposition depends on the truth values of each proposition and the type of conjunction.

İki mantıksal önerme VE ( $\wedge$ ) veya VEYA ( $\vee$ ) mantıksal birleşimiyle bağlantılı olduğunda, bileşik mantıksal önermenin doğruluk değeri, her önermenin doğruluk değerine ve birleşim türüne bağlıdır.

# Fundamentals of Boolean Algebra

## (Boole Cebirinin Temelleri)

Boolean algebra is examined in terms of logic relationships. What is important here is whether the algebraic term is 0 or 1; in other words, whether the result of an expression is 'true' or 'false' in its relationship with other expressions.

Boole cebri mantık ilişkileri açısından incelenir. Burada önemli olan cebirsel terimin 0 mı yoksa 1 mi olduğu; başka bir deyişle, bir ifadenin sonucunun diğer ifadeyle olan ilişkisinde "doğru" veya "yanlış" olup olmadığıdır.



# Fundamentals of Boolean Algebra

## (Boole Cebirinin Temelleri)

Boolean algebra explains the relationship between logic changes. Input variable may be 1 or 0. AND symbol covering the input variables of a Boolean equation shows multiplications, whereas OR symbol shows additions. Complement of a variable is shown as NOT operation.

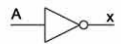



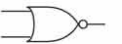


Boole cebri, mantık değişiklikleri arasındaki ilişkiyi açıklar. Giriş değişkeni 1 veya 0 olabilir. Bir Boole denkleminin giriş değişkenlerini kapsayan VE sembolü çarpmaları gösterirken, OR sembolü toplamaları gösterir. Bir değişkenin tamamlayıcısı NOT işlemi ile gösterilir.

# Logic Gates and Truth Tables

Logic gates, which are packet as the integrated circuits, contain various elements such as diodes, transistors, resistors, capacitors.

Mantık kapıları, entegre devreler olarak paketlenmiş olan, diyotlar, transistörler, dirençler, kapasitörler gibi çeşitli elemanları içerir.

## Logic Gates








Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	$\overline{A}$	$AB$	$\overline{AB}$	$A + B$	$\overline{A + B}$	$A \oplus B$	$\overline{A \oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</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></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><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>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><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></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

# Logic Gates and Truth Tables

Integrated circuits are small systems which work by low-power, but with high-speed; their external cable connection is relatively low. In the picture, logic gates and truth tables related to these gates are presented

Entegre devreler, düşük güçle, ancak yüksek hızda çalışan küçük sistemlerdir; harici kablo bağlantıları nispeten düşüktür. Resimde mantık kapıları ve bu kapılarla ilgili doğruluk tabloları sunulmuştur.

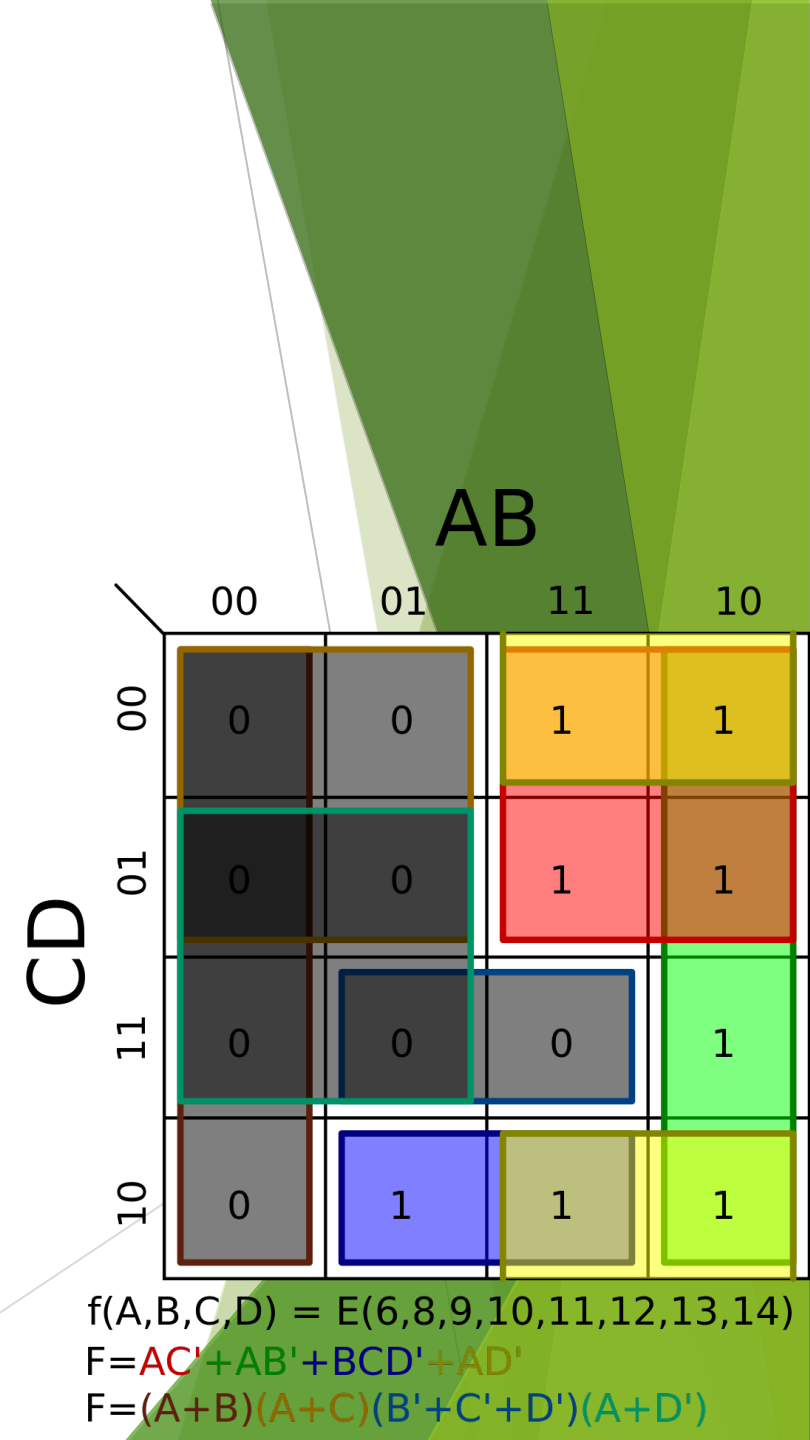
**Logic Gates**

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	$\overline{A}$	$AB$	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</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></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><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>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><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></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

# Karnaugh Maps

It is possible to simplify Boolean functions with the rules of Boolean algebra, but it is not useful because of the fact that there is no systematic rule of this minimization method. Thus, Karnaugh map method has emerged as a more useful method in reduction of Boolean functions.

Boolean fonksiyonları, Boole cebri kurallarıyla basitleştirmek mümkündür, ancak bu minimizasyon yönteminin sistematik bir kuralı olmadığı için kullanışlı değildir. Bu nedenle, Boole fonksiyonlarının azaltılmasında daha kullanışlı bir yöntem olarak Karnaugh harita yöntemi ortaya çıkmıştır.



# SUMMARY

Value of a binary variable can be 0 or 1. A Boolean function consists of binary variable, parentheses, equality sign and binary operations such as AND, OR and NOT. For specific variable values, function's value is either 0 or 1. In order to express a function in truth table, it is necessary to write  $2^n$  combination with 1 and 0 corresponding to  $n$  binary variable and to present the combinations, the function value of which is 1 or 0, in a column.

İkili bir değişkenin değeri 0 veya 1 olabilir. Bir Boole fonksiyonu, ikili değişken, parantezler, eşitlik işareti ve AND, OR ve NOT gibi ikili işlemlerden oluşur. Belirli değişken değerleri için, fonksiyonun değeri 0 veya 1'dir. Doğruluk tablosunda bir fonksiyon ifade etmek için,  $n$  ikili değişkene karşılık gelen 1 ve 0 ile  $2^n$  kombinasyonunu yazmak ve fonksiyon değeri 1 veya 0 olan kombinasyonları bir sütunda sunmak gerekir.

# SUMMARY

Each Boolean function can be represented by a truth table. For each row of the table function takes the value 0 or 1. Boolean algebra operations are usually used to find more simple expressions that has the same function.

Her Boole işlevi bir doğruluk tablosu ile temsil edilebilir. Tablonun her satırı, 0 veya 1 değerini alır. Boole cebri işlemleri genellikle aynı işleve sahip daha basit ifadeleri bulmak için kullanılır.

# YMT/YMH 213

# Vocational English

Assoc. Prof. Dr. Murat KARABATAK

# Algorithms and Flowcharts (Algoritmalar ve Akış Şemaları)

Algorithm means «putting forward a specific task step by step by using pre-defined processing steps, and to encode these steps in computer environment by using any programming language». In other aspect, algorithm is «logical and symbolic description of processes required for the solution of a problem».

Algoritma, «önceden tanımlanmış işlem adımlarını kullanarak belirli bir görevi adım adım ileriye doğru yürütmek ve bu adımları herhangi bir programlama dili kullanarak bilgisayar ortamında kodlamak» demektir. Diğer bir açıdan, algoritma, «bir problemin çözümü için gerekli süreçlerin mantıksal ve sembolik açıklamasıdır».



# Algorithms and Flowcharts (Algoritmalar ve Akış Şemaları)

An Algorithm puts forward the steps and conditions to be followed in order to get the results of a specific job or problem. These steps can reach a conclusion if the conditions are followed step by step. In computer application, many algorithms are required while developing software applications.

Bir Algoritma, belirli bir işin veya problemin sonuçlarını almak için takip edilen adımları ve koşulları ortaya koyar. Bu adımlar, koşullar adım adım takip edildiğinde bir sonuca varabilir. Bilgisayar uygulamasında, yazılım uygulamaları geliştirilirken birçok algoritmaya ihtiyaç duyulmaktadır.

## Necessary Features for an Algorithm (Bir Algoritma İçin Gerekli Özellikler)

***To be effective and general:*** Algorithm should be efficient and should not be repetitious; it needs to be usable in other algorithms. In addition, it should be designed with a general purpose; in other words, it should get the correct conclusion in each case and in each input value.

***Etkili ve genel olma:*** Algoritma verimli olmalı ve tekrarlı olmamalıdır; diğer algoritmalarda kullanılabilir olması gerekir. Ayrıca genel bir amaç için tasarlanmalıdır; diğer bir deyişle, her durumda ve her girdi değerinde doğru sonuca varmalıdır.

## Necessary Features for an Algorithm (Bir Algoritma İçin Gerekli Özellikler)

***To be finite:*** An algorithm should include strictly finite number of operations and also time for these operations should be finite. Algorithm consists of a certain number of steps; it starts from a start point and has necessarily an end point.

***Sonlu olma:*** Bir algoritma kesin olarak sonlu sayıda işlem içermeli ve bu işlemler için zaman da sonlu olmalıdır. Algoritma belirli sayıda adımdan oluşur; bir başlangıç noktasından başlar ve bir bitiş noktasına sahip olması gerekir.

## Necessary Features for an Algorithm (Bir Algoritma İçin Gerekli Özellikler)

***Infallibility:*** For algorithm, infallibility is the most important criterion. When algorithm is re-executed, same results are obtained for same input values.

***Hatasızlık:*** Algoritma için hatasızlık en önemli kriterdir. Algoritma yeniden yürütüldüğünde, aynı girdi değerleri için aynı sonuçlar elde edilir.

## Necessary Features for an Algorithm (Bir Algoritma İçin Gerekli Özellikler)

***Valid Input/Output:*** An algorithm should have input and output values. Input value is data that algorithm processes in order to produce a result; output value is the result obtained by algorithm.

***Geçerli Giriş / Çıkış :*** Bir algoritma giriş ve çıkış değerlerine sahip olmalıdır. Girdi değeri, algoritmanın bir sonuç üretmek için işlediği verilerdir; çıktı değeri, algoritma ile elde edilen sonuçtur.

## Necessary Features for an Algorithm (Bir Algoritma İçin Gerekli Özellikler)

***Performance*** : An algorithm should be designed to offer good performance; repetitions should be avoided. Memory requirement and working time should be as balanced as possible.

***Performans***: Bir Algoritma, iyi performans sunmak için bir tasarlanmalıdır; tekrarlardan kaçınılmalıdır. Bellek gereksinimi ve çalışma süresi olabildiğince dengeli olmalıdır.

# Pseudo Code and Real Code (Sözde Kod ve Gerçek Kod)

Pseudo code means putting forward or defining algorithm by using both programming language and natural language. On the other hand, real code means implementing algorithm by using a programming language, such as C, Java or C++.

Sözde kod, hem programlama dilini hem de doğal dili kullanarak algoritmayı sunmak veya tanımlamak anlamına gelir. Öte yandan, gerçek kod, C, Java veya C ++ gibi bir programlama dili kullanarak algoritma uygulamak anlamına gelir.

# Flowcharts

## (Akış çizelgeleri)

Flowcharts are used to put forward the algorithm by explaining the steps that need to be done in a visual way; it shows what should be done to fix the problem from beginning to end by using the symbols that consists of geometric shapes. Each symbol in general indicates a command or task to do. Flowcharts begins with Start icon and ends with Stop icon.

Akış şemaları, yapılması gereken adımları görsel bir şekilde açıklayarak algoritmayı ortaya koymak için kullanılır; Geometrik şekillerden oluşan sembolleri kullanarak sorunun baştan sona giderilmesi için yapılması gerekenleri gösterir. Genel olarak her sembol, yapılacak bir komut veya görevi belirtir. Akış çizelgeleri Başlat simgesiyle başlar ve Durdur simgesiyle biter.



## Loops and Iterative Operations (Döngüler ve Yinelemeli İşlemler)

For example, when reading value to  $n$ -element array, it is necessary to write line that consist of  $n$  reading commands. To make such repetitive operations, it is more reasonable to use loops in algorithms.

Örneğin,  $n$  elemanlı diziye değer okurken  $n$  okuma komutundan oluşan bir satır yazmak gerekir. Bu tekrarlayan işlemleri yapmak için, algoritmalarda döngü kullanmak daha mantıklıdır.

# Algorithm Design with an Object-Oriented Approach (Nesne Tabanlı Bir Yaklaşımla Algoritma Tasarımı)

Programming languages such as C++ and Java are object-oriented languages; if an object-oriented language is to be used to design a program, flowcharts can be used to express an algorithm. However, it is also possible to use ORM(Object Rule Modeling) or UML(Unified Modeling Language) method which has been developed for object-oriented language.

C ++ ve Java gibi programlama dilleri nesne yönelimli dillerdir; Bir programı tasarlamak için nesne yönelimli bir dil kullanılacaksa, bir algoritmayı ifade etmek için akış şemaları kullanılabilir. Ancak nesne yönelimli dil için geliştirilmiş olan ORM (Object Rule Modeling) veya UML (Unified Modeling Language) yöntemini kullanmak da mümkündür.

## Summary (Özet)

Algorithm design could be called as "the initial step of programming". A task should be put forth with an approach close to the spoken language, regardless of programming language. Thus, it should be easily implemented by using a programming language.

Algoritma tasarımı, "programlamanın ilk adımı" olarak adlandırılabilir. Programlama dilinden bağımsız olarak, konuşulan dile yakın bir yaklaşımla görev ortaya konulmalıdır. Bu nedenle, bir programlama dili kullanılarak kolayca uygulanmalıdır.

# YMT/YMH 213

# Vocational English

Assoc. Prof. Dr. Murat KARABATAK

# Programming Languages (Programlama Dilleri)

The theory of programming languages is a branch of computer engineering and include design, application, analysis and programming language classification as well as general characteristics of programming language.

Programlama dilleri teorisi, bilgisayar mühendisliğinin bir dalıdır ve tasarım, uygulama, analiz ve programlama dili sınıflandırmasının yanı sıra programlama dilinin genel özelliklerini içerir.

# Programming Languages (Programlama Dilleri)

The theory of programming languages, which is interdisciplinary subject including mathematics, software engineering, linguistics and even educational sciences, is one of the most fundamental branches that has been the subject hundreds of research publications.

Matematik, yazılım mühendisliği, dilbilim ve hatta eğitim bilimlerini içeren disiplinler arası bir konu olan programlama dilleri teorisi, yüzlerce araştırma yayınına konu olan en temel dallardan biridir.

# Software Development Process (Yazılım Geliştirme Süreci)

Software development process can be examined in five stages. These stages generally take place as a consecutive five steps in software development. These steps can be listed as follows:

Yazılım geliştirme süreci beş aşamada incelenebilir. Bu aşamalar genellikle yazılım geliştirmede birbirini izleyen beş adım olarak gerçekleşir. Bu adımlar şu şekilde sıralanabilir:

## Software Development Process (Yazılım Geliştirme Süreci)

**Requirement Analysis:** Software is developed to meet the requirements of a certain group of users. Software system that is to be designed and developed should meet the user requirements completely. These requirement should be created by a group of users clearly and in a suitable form. At this stage, joint work between people who will develop and use the software will be appropriate.

**İhtiyaç analizi:** Yazılım, belirli bir kullanıcı grubunun gereksinimlerini karşılamak için geliştirilmiştir. Tasarlanacak ve geliştirilecek yazılım sistemi, kullanıcı gereksinimlerini tam olarak karşılamalıdır. Bu gereksinimler, bir grup kullanıcı tarafından açık ve uygun bir biçimde oluşturulmalıdır. Bu aşamada yazılımı geliştirecek ve kullanacak kişilerin ortak çalışması uygun olacaktır.



**Software Design:** Software designers start the design of the system by the above-mentioned requirement documents. As a result of this stage, documentation is set out, which provides the definitions about system design. In this documentation, all modules of the system and their interfaces should be defined.

**Yazılım Tasarımı:** Yazılım tasarımcıları, yukarıda belirtilen gereksinim belgeleri ile sistemin tasarımına başlarlar. Bu aşamanın sonucunda sistem tasarımı ile ilgili tanımları sağlayan dokümantasyon oluşturulur. Bu dokümantasyonda, sistemin tüm modülleri ve ara yüzleri tanımlanmalıdır.

## Software Development Process (Yazılım Geliştirme Süreci)

**Coding:** It is the stage where definitions made in the second stage are encoded. This stage is the sole step where programming language is applied directly. It is a software system fully implemented and reported.

**Kodlama:** İkinci aşamada yapılan tanımların kodlandığı aşamadır. Bu aşama, programlama dilinin doğrudan uygulandığı tek adımdır. Tamamen uygulanan ve raporlanan bir yazılım sistemidir.

## Software Development Process (Yazılım Geliştirme Süreci)

**Certification:** Software, that is put forward here, is deployed to end users after quality control. Certification can be made after all software is put forward, it is also possible to check each modules and interfaces between modules incrementally. In all of these transactions, it is considered whether software fully meet the expectations and have seamless interfaces.

**Sertifikasyon :** Burada ortaya konulan yazılım, kalite kontrolünden sonra son kullanıcılara dağıtılır. Sertifikasyon, tüm yazılımlar ortaya konulduktan sonra yapılabilir, ayrıca her bir modülü ve modüller arasındaki ara yüzleri aşamalı olarak kontrol etmek de mümkündür. Tüm bu işlemlerde yazılımın beklentileri tam olarak karşılayıp karşılamadığı ve sorunsuz ara yüzlere sahip olup olmadığı dikkate alınır.

## Software Development Process (Yazılım Geliştirme Süreci)

***Maintenance:*** This step includes the addition of new components and the elimination of errors. The importance of this stage can be understood better when the costs are considered. Experience has shown that a system maintenance costs may be close to or even higher than the total cost of other stages of the system.

***Bakım:*** Bu adım, yeni bileşenlerin eklenmesini ve hataların ortadan kaldırılmasını içerir. Maliyetler düşünüldüğünde bu aşamanın önemi daha iyi anlaşılabilir. Deneyimler, bir sistem bakım maliyetlerinin, sistemin diğer aşamalarının toplam maliyetine yakın veya hatta daha yüksek olabileceğini göstermiştir.

# Object Oriented Programming (Nesne Yönelimli Programlama)

Object-oriented programming is based on the concept of object. Here, "object" is a logical entity that exists in the real world or created by the programmer. Object is considered together with data that identifies it and all operations to be performed. A generic name is given to the same group of data objects and thus a new data type is created.

Nesneye yönelik programlama, nesne kavramına dayanır. Burada "nesne", gerçek dünyada var olan veya programcı tarafından yaratılan mantıksal bir varlıktır. Nesne, onu tanımlayan veriler ve gerçekleştirilecek tüm işlemler ile birlikte değerlendirilir. Aynı veri nesnesi grubuna genel bir ad verilir ve böylece yeni bir veri türü oluşturulur.

## Elements of Programming Languages (Programlama Dillerinin Unsurları)

**Syntax:** As with conventional languages, programming languages have also syntax. Syntax of a program is a set of rules that determines the order according to which symbols should be written in order to be accepted as valid. The best known formal notation for syntax is "Extended Backus Naur Form"(EBNF) is known as. Code, which is written in correct syntax, may not be correct semantically.

**Sözdizimi** : Geleneksel dillerinde olduğu gibi, programlama dillerinin de sözdizimi vardır. Bir programın sözdizimi, geçerli olarak kabul edilebilmesi için hangi sembollerin yazılması gerektiğini belirleyen bir kurallar dizisidir. Sözdizimi için en iyi bilinen biçimsel gösterim "Genişletilmiş Backus Naur Formu" (EBNF) olarak bilinir. Doğru sözdiziminde yazılan kod, anlamsal olarak doğru olmayabilir.

## Elements of Programming Languages (Programlama Dillerinin Unsurları)

***Semantics:*** Semantics is the meaning of a statements in a programming language. Even the syntax of a language is very well understood, it is difficult to grasp the meaning.

***Semantik(Anlambilim)*** : Anlambilim, bir programlama dilinde bir ifadenin anlamıdır. Bir dilin sözdizimi çok iyi anlaşılmış olsa bile anlamını kavramak zordur.

# YMT/YMH 213

# Vocational English

Assoc. Prof. Dr. Murat KARABATAK



# Operating Systems (İşletim Sistemleri)

An operating system, in short, is a collection of programs that serves as an interface between computer resources and users. Its aim is to offer an environment in which users run programs and to provide the efficient use of computer resources including both hardware and software.

Kısacası bir işletim sistemi, bilgisayar kaynakları ve kullanıcılar arasında bir ara yüz görevi gören bir program koleksiyonudur. Amacı, kullanıcıların programları çalıştırdıkları bir ortam sunmak ve hem donanım hem de yazılım dahil olmak üzere bilgisayar kaynaklarının verimli kullanımını sağlamaktır.

# Well-Known Operating Systems (Bilinen İşletim Sistemleri)

Many general-purpose and special-purpose operating systems are available; for example, Windows, Linux, Unix and Macintosh operating systems come to mind immediately. If you go a little further back, you can see operating systems that are not mentioned very much nowadays; for example VM operating system.

Birçok genel amaçlı ve özel amaçlı işletim sistemi mevcuttur; örneğin Windows, Linux, Unix ve Macintosh işletim sistemleri akla hemen geliyor. Biraz daha geriye giderseniz, günümüzde pek bahsedilmeyen işletim sistemlerini görebilirsiniz; örneğin VM işletim sistemi.

# VM Operating Systems (VM İşletim Sistemleri)

VM is abbreviation for Virtual Machine; so VM may be termed a kind of virtual computer. A virtual computer is a misleading image of a real computer; a virtual computer shows the real system as if there is more than one real computer. Users of virtual computers, which are generated by the operating system on a real system, think that they are working in a real system.

VM, Sanal Makinenin kısaltmasıdır; bu nedenle VM bir tür sanal bilgisayar olarak adlandırılabilir. Sanal bilgisayar, gerçek bir bilgisayarın yanıltıcı bir görüntüsüdür; sanal bir bilgisayar gerçek sistemi sanki birden fazla gerçek bilgisayar varmış gibi gösterir. Gerçek bir sistem üzerinde işletim sistemi tarafından üretilen sanal bilgisayar kullanıcıları, gerçek bir sistemde çalıştıklarını düşünürler.

# Tasks of Operating Systems (İşletim Sistemlerinin Görevleri)

Tasks of operating systems, in general, is to ensure the efficient use of hardware resources of the computer system and pre-installed software part by distributing to users and to the system itself if it is necessary.

İşletim sistemlerinin görevleri genel olarak bilgisayar sisteminin donanım kaynaklarının ve önceden kurulmuş yazılım bölümünün kullanıcılara ve gerekirse sistemin kendisine dağıtarak verimli kullanılmasını sağlamaktır.

## Types of Operating Systems (İşletim Sistemleri Türleri)

***Monoprogramming:*** In a system based on mono-programming, only one virtual environment can be activated and a user can use all the resources of the system. Errors that can occur in runtime do not reflect to another user or system; so, protection measures take place only between the operating system and a user. In a mono-programming order, re-source assignment, integrity problems etc. can be easily solved.

***Tekli programlama:*** Mono programlamaya dayalı bir sistemde, yalnızca bir sanal ortam etkinleştirilebilir ve bir kullanıcı sistemin tüm kaynaklarını kullanabilir. Çalışma zamanında meydana gelebilecek hatalar başka bir kullanıcıya veya sisteme yansımaz; bu nedenle koruma önlemleri yalnızca işletim sistemi ve kullanıcı arasında gerçekleşir. Tekli programlama düzeninde, yeniden kaynak atama, bütünlük sorunları vb. kolayca çözülebilir.

## Types of Operating Systems (İşletim Sistemleri Türleri)

***Multiprogramming:*** Systems based on multiprogramming-based systems were designed to evaluate the waiting or suspended period of resources such as processor. If any job program running in the system waiting for input or output, synchronization etc. in this standby state processor can start another job and thus processor is used efficiently.

***Çoklu Programlama:*** Çoklu programlama tabanlı sistemlere dayalı sistemler, işlemci gibi kaynakların bekleme veya askıya alma sürelerini değerlendirmek için tasarlanmıştır. Eğer sistemde çalışan herhangi bir program giriş yada çıkış senkronizasyon vb. gibi bekliyorsa, bu bekleme durumunda işlemci başka bir işi başlatabilir ve böylece işlemci verimli bir şekilde kullanılır.

## Types of Operating Systems (İşletim Sistemleri Türleri)

**Multitasking:** A task can be defined as the smallest operating unit that can operate independently in a system task. It is the main tool for performing a job on computer. The operating system performs a job by assigning at least one task and executing it. A job can be performed by a single task; this task can run the programs that meet the steps required by the job; respectively compilation, binding and application steps.

**Çoklu Görev:** Bir görev, bir sistem görevinde bağımsız olarak çalışabilen en küçük işletim birimi olarak tanımlanabilir. Bilgisayarda bir iş yapmak için ana araçtır. İşletim sistemi, en az bir görev atayarak ve onu çalıştırarak bir işi gerçekleştirir. Bir iş, tek bir görevle gerçekleştirilebilir; bu görev işin gerektirdiği adımları karşılayan programları çalıştırabilir; sırasıyla derleme, bağlama ve uygulama adımlarıdır.

## Operating Systems Structure and Architecture (İşletim Sistemleri Yapısı ve Mimarisi)

**Resource Sharing:** Resources of the computer system are often not completely consumed by an individual user. Such resources are idle, and reduce the average efficiency of the system. Therefore, it is necessary to allow other user to use these resources simultaneously. This has been one of the major problems to be solved by system manufacturers. Processing unit and memory, from 1960s to the mid-1970s, are the first resources that was aimed to be shared in a system because of their rates in total.

**Kaynak Paylaşımı:** Bilgisayar sisteminin kaynakları genellikle tek bir kullanıcı tarafından tamamen tüketilmez. Bu tür kaynaklar boşta ve sistemin ortalama verimliliğini düşürür. Bu nedenle, diğer kullanıcıların bu kaynakları aynı anda kullanmasına izin vermek gerekir. Bu, sistem üreticileri tarafından çözülmesi gereken en büyük sorunlardan biri olmuştur. 1960'lardan 1970'lerin ortalarına kadar olan işlem birimi ve bellek, toplam oranlarından dolayı bir sistemde paylaşılması amaçlanan ilk kaynaklardır.



## Operating Systems Structure and Architecture (İşletim Sistemleri Yapısı ve Mimarisi)

**Job:** A job is a service set which is desired to be performed as an independent whole and which will be processed by the operating system without addressing any other job. Jobs may be composed of steps in which one or more programs will be run.

**İş:** İş, bağımsız bir bütün olarak gerçekleştirilmek istenen ve işletim sistemi tarafından başka herhangi bir iş adresleme yapılmadan işlenecek bir hizmet setidir. İşler, bir veya daha fazla programın çalıştırılacağı adımlardan oluşabilir.

## Processes and Process Management (Süreçler ve Süreç Yönetimi)

**Deadlock:** Deadlock occurs when processes are blocked because of the unit or resource that they can never reach. In addition, if a process waits another process to complete its work and also that processor waits it at the same time, deadlock occur again. For example, deadlock occur when Job A and Job B have request from each other while running in parallel (A & B are resources such as processor, memory block, peripherals, etc.).

**Kilitlenme:** Kilitlenme, hiçbir zaman ulaşamayacakları birim veya kaynak nedeniyle süreçler engellendiğinde oluşur. Ayrıca, bir süreç başka bir sürecin işini tamamlamasını beklerse ve aynı zamanda o işlemci onu beklerse, tekrar kilitlenme oluşur. Örneğin, İş A ve İş B paralel çalışırken birbirlerinden istek aldığında kilitlenme meydana gelir (A ve B, işlemci, bellek bloğu, çevre birimleri vb. Gibi kaynaklardır).