

BURSA TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
BLM-101 BİLGİSAYAR MÜHENDİSLİĞİNE GİRİŞ
PROJE KONUSU: VERİ MANİPÜLASYONLARI VE
MANTIK KAPILARI
ADI: MERVE CEMİLE
SOYADI: KAYA
OKUL NO: 25360859082

SUNUMUN İÇERİĞİ;

- Bilgisayar mimarisi temelleri
- Mantık kapıları
- Python ile mantık devresi simülasyonu



VERİ NEDİR?

- Veri, bilgisayarın işlemiği en küçük bilgi birimidir.
- Bilgisayarlar verileri **sayısal (dijital)** olarak işler.

BİNARY(İKİLİ SİSTEM):

- Bilgisayarlar yalnızca **2 durumu** anlayabilir:
 - **0 → Kapalı / Yanlış / Akım yok**
 - **1 → Açık / Doğru / Akım var**

NEDEN 0 VE 1 KULLANILIR?

- Elektronik devreler:
- Akım var → 1
- Akım yok → 0
- Daha hızlı, güvenilir ve hatasız çalışır.

•VERİ NASIL TEMSİL EDİLİR?

1. SAYILAR(BİNARY)
2. HARFLER(ASCII/UNICODE)
3. GÖRÜNTÜ&SES(BİNARY KODLAMA)



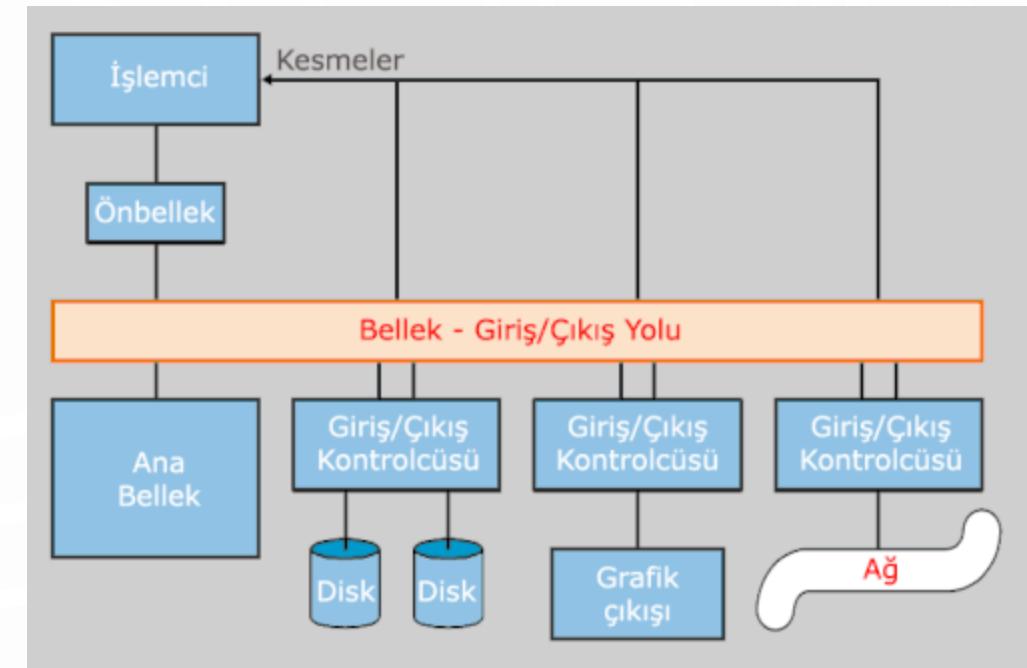
Char	ASCII	Decimal	Bits	Char	ASCII	Decimal	Bits	Char	ASCII	Decimal	Bits
0	48	0	000000	F	70	22	010110	d	100	44	101100
1	49	1	000001	G	71	23	010111	e	101	45	101101
2	50	2	000010	H	72	24	011000	f	102	46	101110
3	51	3	000011	I	73	25	011001	g	103	47	101111
4	52	4	000100	J	74	26	011010	h	104	48	110000
5	53	5	000101	K	75	27	011011	i	105	49	110001
6	54	6	000110	L	76	28	011100	j	106	50	110010
7	55	7	000111	M	77	29	011101	k	107	51	110011
8	56	8	001000	N	78	30	011110	l	108	52	110100
9	57	9	001001	O	79	31	011111	m	109	53	110101
:	58	10	001010	P	80	32	100000	n	110	54	110110
;	59	11	001011	Q	81	33	100001	o	111	55	110111
<	60	12	001100	R	82	34	100010	p	112	56	111000
=	61	13	001101	S	83	35	100011	q	113	57	111001
>	62	14	001110	T	84	36	100100	r	114	58	111010
?	63	15	001111	U	85	37	100101	s	115	59	111011
@	64	16	010000	V	86	38	100110	t	116	60	111100
A	65	17	010001	W	87	39	100111	u	117	61	111101
B	66	18	010010	'	96	40	101000	v	118	62	111110
C	67	19	010011	a	97	41	101001	w	119	63	111111
D	68	20	010100	b	98	42	101010				
E	69	21	010101	c	99	43	101011				

- **Bilgisayar Mimarisi Nedir?**
- Bilgisayar mimarisi, bilgisayarın **donanım yapısını** ve bu parçaların **birbiriyle nasıl çalıştığını** inceler.
- **Temel Donanım Bileşenleri:**
- **CPU (Merkezi İşlem Birimi)**
- **Bellek (RAM)**
- **Giriş Birimleri (Klavye, mouse vb.)**
- **Çıkış Birimleri (Ekran, yazıcı vb.)**

- **Veri Nasıl İşlenir?**

- Kullanıcı veri girer.
- Veri belleğe alınır.
- CPU veriyi işler.
- Sonuç kullanıcıya gösterilir.

- Bilgisayar mimarisi, donanım bileşenlerinin yapısını inceler.
- CPU, bellekteki verileri işler.
- Giriş birimlerinden alınan veriler işlenip çıkışa gönderilir.



CPU Nedir?

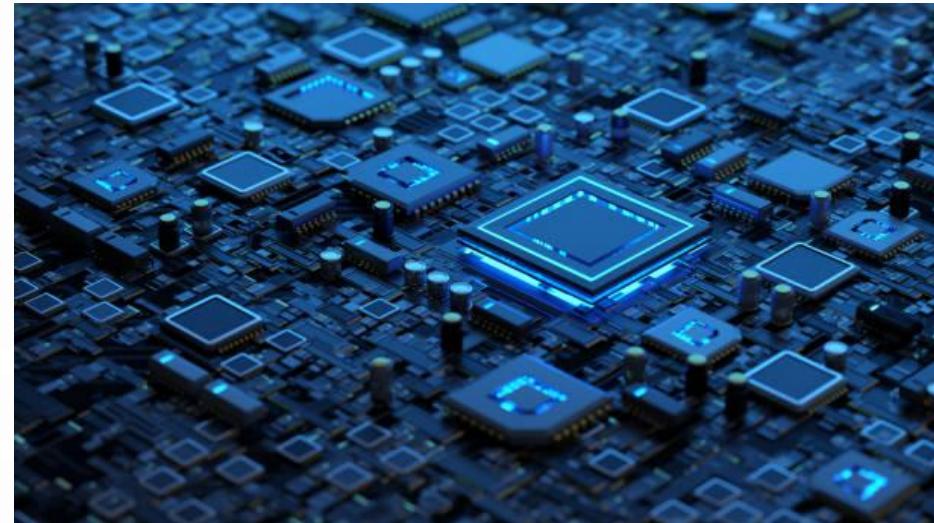
- CPU, **Merkezi İşlem Birimidir.**
- Bilgisayarın **beynidir.**
- Tüm hesaplama ve karar verme işlemleri burada yapılır.

CPU'nun Görevleri:

- Program komutlarını çalışırmak.
- Verileri işlemek.
- Diğer donanım birimlerini kontrol etmek.

CPU'nun Temel Bileşenleri

- ALU (Arithmetic Logic Unit)
- Kontrol Birimi (Control Unit)
- Register'lar (Yazmaçlar)



CPU Neden Önemlidir?

- Programların çalışma hızını belirler.
- Mantık kapıları CPU içinde kullanılır.
- Donanım–yazılım arasındaki köprüdür.



shutterstock.com • 2601794947

ALU Nedir?

- ALU, CPU'nun **hesaplama ve mantık işlemlerini yapan** birimidir.
- Aritmetik ve mantıksal işlemler burada gerçekleştirilir.

ALU'nun Yaptığı İşlemler

- Toplama, çıkarma
- Karşılaştırma işlemleri
- Mantıksal işlemler (AND, OR, NOT, XOR)

ALU ve Mantık Kapıları

- ALU, mantık kapılarını kullanarak çalışır.
- Tüm karşılaştırmalar 0 ve 1 üzerinden yapılır.
- Mantık kapıları donanım seviyesinde yer alır.

ÖRNEK:

Bellekte yer alan iki sayının toplanması

- **Adım 1:** Toplanacak sayılardan birini bellekten al ve bir yazmaca yerleştir.
- **Adım 2:** Bellekten diğer sayıyı al ve başka bir yazmaca yerleştir.
- **Adım 3:** Adım 1 ve 2'deki yazmaçları giriş olarak kullanacak ve sonucu başka bir yazmaca yazacak şekilde toplama devresini çalıştır.
- **Adım 4:** Sonucu belleğe kaydet.
- **Adım 5:** Dur.

Kontrol Birimi Nedir?

- Kontrol birimi, CPU'nun **yönetici kismıdır**.
- Program komutlarının **hangi sırayla ve nasıl çalışacağını** belirler.

Kontrol Biriminin Görevleri:

- Komutları bellekten almak
- Komutları çözümlemek (decode)
- ALU'ya hangi işlemin yapılacağını söylemek
- Veri akışını kontrol etmek

Yazmaç (Register) Nedir?

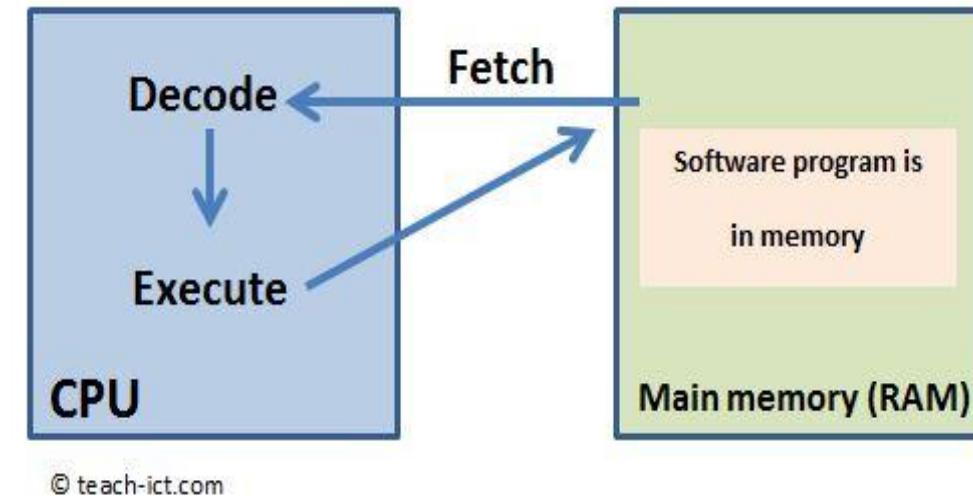
- CPU içinde bulunan **en hızlı küçük belleklerdir**.
- İşlem yapılacak veriler geçici olarak burada tutulur ,işlemler ALU tarafından bu verilerle yapılır.

Makine Dili Nedir?

- Bilgisayarın doğrudan anlayabildiği tek dildir.
- Sadece 0 ve 1'lerden oluşur.
- Donanıma en yakın programlama seviyesidir.

Makine Dili Nasıl Çalışır?

- Her komut binary olarak ifade edilir.
- **CPU bu komutları:**
 - Alır (Fetch), Çözümler (Decode), Çalıştırır(Execute).



KOMUT LİSTESİ

RISC (Reduced Instruction Set Computer)

- Az sayıda, **basit komutlar**
- Komutlar genellikle **tek clock cycle**
- Hız ve verimlilik ön plandadır.

CISC (Complex Instruction Set Computer)

- Çok sayıda, **karmaşık komutlar**
- Bir komut birden fazla işlem yapabilir
- Daha az kod satırı ile işlem yapılır

RISC ve CISC Karşılaştırması

Özellik	RISC	CISC
Komut sayısı	Az	Çok
Komut yapısı	Basit	Karmaşık
Çalışma süresi	Genelde 1 cycle	Birden fazla cycle
Donanım	Daha sade	Daha karmaşık
Örnek	ARM	x86

- Makine Dili Komut Türleri

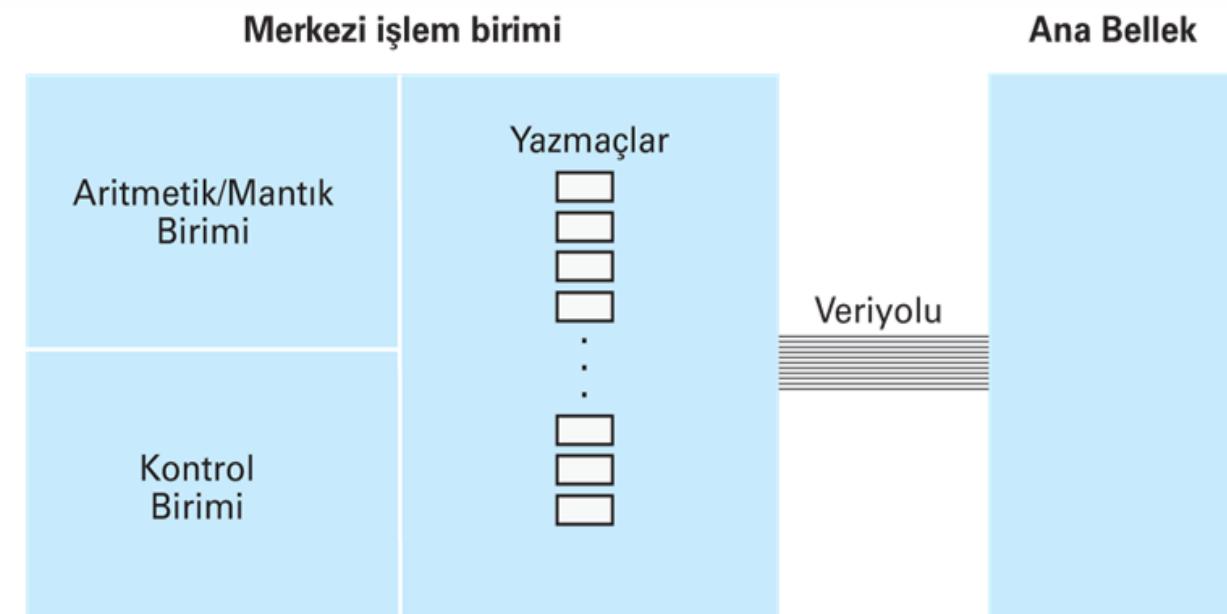
Makine Dili Komutları Nedir?

- CPU'nun çalıştırıldığı en temel talimatlardır.
- Her komut belirli bir görevi yerine getirir.

1	AND	AND operand1, operand2
2	OR	OR operand1, operand2
3	XOR	XOR operand1, operand2
4	TEST	TEST operand1, operand2
5	NOT	NOT operand1

Temel Komut Grupları

- Veri Aktarma Komutları
- Aritmetik / Mantık Komutları
- Kontrol Komutları



Veri Aktarma Komutları

- Veriyi bir yerden başka bir yere taşır.
- İşlem yapmaz, sadece **aktarır**.

Örnek İşlemler:

- Bellekten register'a veri alma
- Register'dan belleğe veri yazma
- Register-register veri aktarımı
- CPU ve ana bellek arasındaki aktarımında da bazı terimler kullanılır:
 - ✓ Yükleme(LOAD)
 - ✓ Kayıt(STORE)
- CPU ve ana bellek dışındaki cihazlarla ise haberleşmeyi sağlayan Giriş/Çıkış(G/Ç) komutları kullanılır.

ÖRNEK: Bellekte iki sayıyı bölme algoritması

Adım 1:Bellekten bir sayıyı bir yazmaca YÜKLE.

Adım 2:Bellekten başka bir sayıyı başka bir yazmaca YÜKLE.

Adım 3:Eğer ikinci sayı 0 ise Adım 6'ya ATLA.

Adım 4:İlk yazmaçtaki sayıyı ikinci yazmaçtaki sayıya böl ve sonucu bir üçüncü yazmaçta tut.

Adım 5:Üçüncü yazmacın içeriğini belleğe KAYDET.

Adım 6:Dur.

Kontrol Komutları

- Programın akışını değiştirir.
- **Koşullu** veya **koşulsuz** dallanma sağlar.

Örnek Kontrol Komutları

- JUMP(Atlama)
- BRANCH (Koşullu dallanma)
- LOOP (Döngü)

Kontrol Birimi ile İlişki

- Bu komutlar **Kontrol Birimi** tarafından yönetilir.
- Karar yapıları mantık kapılarına dayanır.

Aritmetik Komutlar

- Toplama, çıkarma, çarpma, sayısal işlemler

Mantık Komutları

- AND, OR, NOT, XOR, Karşılaştırma işlemleri
- Kaydırma SHIFT, Döndürme ROTATE komutları

ALU ile İlişki

- Bu komutlar doğrudan **ALU'da çalıştırılır.**
- Mantık kapıları burada devreye girer.

NOT

x	F
0	1
1	0



AND

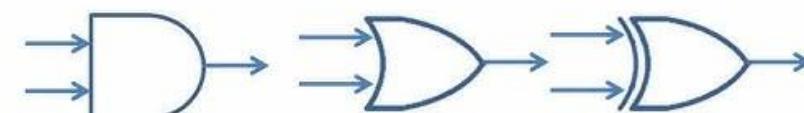
x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0



AND KAPISI

- AND kapısı, **iki veya daha fazla giriş** alır.
- **Tüm girişler 1 ise çıkış 1 olur.** Aksi durumda çıkış 0'dır.

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

OR Kapısı

- OR kapısı, **iki veya daha fazla giriş** alır.
- **En az bir giriş 1 ise çıkış 1 olur.** Tüm girişler 0 ise çıkış 0'dır.

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

XOR Kapısı (Exclusive OR)

- XOR kapısı iki giriş alır.
- Girişler farklıysa çıkış 1 olur. Girişler aynıysa çıkış 0 olur.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

NOT Kapısı

- NOT kapısı tek girişlidir.
- Giriş tersine çevirir. $0 \rightarrow 1 \quad 1 \rightarrow 0$

A	NOT A
0	1
1	0

Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	\bar{A}	AB	$\bar{A}\bar{B}$	$A+B$	$\bar{A}+\bar{B}$	$A \oplus B$	$\bar{A} \oplus \bar{B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table border="1"><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 border="1"><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 border="1"><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 border="1"><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 border="1"><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 border="1"><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 border="1"><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	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

Kombinasyonel Devre Nedir?

- Birden fazla mantık kapısının birlikte kullanıldığı devrelerdir.
- Çıkış, **sadece o anki girişlere** bağlıdır.
- Hafıza içermez.

ÖRNEK:

A AND (B OR C)

- Önce **B OR C** hesaplanır.
- Sonra sonuç **A ile AND** yapılır.

3 Değişkenli Doğruluk Tablosu

➤ A AND (B OR C) Kombinasyonel devresinin **tüm olası girişler için çıktısını** gösteririz.

A	B	C	B OR C	A AND (B OR C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Python Projesi – Sanal Mantık Devresi Simülatörü

Projenin Amacı

- Mantık kapılarını yazılım ortamında simüle etmek
- Kullanıcıdan giriş alarak sonuç üretmek

Programın Özellikleri

- AND, OR, NOT, XOR işlemleri
- Kullanıcıdan 0 ve 1 değerleri alma
- 3 değişkenli ifadeler için doğruluk tablosu üretme

Proje Açıklaması

- Bu projede **Sanal Mantık Devresi Simülatörü** geliştirilmiştir. Amaç, temel mantık kapılarının (AND, OR, NOT, XOR) çalışma mantığını Python dili kullanarak simüle etmek ve kullanıcıdan alınan girişlere göre sonuç üretmektir. Ayrıca 3 değişkenli mantıksal ifadeler için doğruluk tablosu oluşturulmaktadır.

Kullanılan Teknolojiler ve Kütüphaneler

- **Python 3**
- Ekstra bir kütüphane kullanılmamıştır.
- Program yalnızca Python'un temel yapıları (if-else, fonksiyonlar, döngüler) kullanılarak geliştirilmiştir.

Programın Çalışma Mantığı

- Program iki ana bölümden oluşmaktadır:

1 Temel Mantık Kapıları Simülasyonu

- Bu bölümde kullanıcıdan:
 - Birinci giriş (0 veya 1)
 - İkinci giriş (0 veya 1)
 - Mantık kapısı türü (AND, OR, XOR, NOT)
 - bilgileri alınır.
 - Seçilen kapı türüne göre ilgili mantıksal işlem yapılır ve sonuç ekrana yazdırılır.

2 3 Değişkenli Doğruluk Tablosu Oluşturma

- Bu bölümde program, aşağıdaki mantıksal ifade için doğruluk tablosu üretir.
 - A AND (B OR C)
- A, B ve C değişkenleri için tüm olası kombinasyonlar (0 ve 1) oluşturulur.
- Önce **B OR C** işlemi yapılır.
- Elde edilen sonuç **A AND (B OR C)** işleminde kullanılır.
- Tüm sonuçlar tablo halinde ekranaya yazdırılır.

Algoritma Mantığı (Adım Adım)

- Kullanıcıdan giriş değerleri alınır.
- Seçilen mantık kapısına göre işlem yapılır.
- Sonuç ekrana yazdırılır.

3 değişkenli ifade için:

- A, B ve C için tüm kombinasyonlar üretilir.
- Mantıksal işlemler sırayla uygulanır.
- Doğruluk tablosu oluşturulur.

Programın Çalıştırılması

- Python yüklü olduğundan emin olunuz.
- Terminal veya komut satırında aşağıdaki komutu çalıştırınız:

Bu proje sayesinde:

- Mantık kapılarının çalışma mantığı öğrenilmiştir.
- Donanım seviyesindeki mantıksal işlemler yazılım ortamında simüle edilmiştir.
- Python dili kullanılarak teorik bilgiler uygulamaya dökülmüştür.
- Bilgisayar mimarisi ve mantık kapıları öğrenildi.
- Python ile donanım mantığı simüle edildi.
- Teorik bilgi uygulamaya döküldü.