(3. Grup: Veri Manipülasyonu ve Mantık Kapıları)

Bilgisayar Bilimine Giriş

Bölüm 2: Veri İşleme (Data Manipulation)

Kaynak: Computer Science: An Overview – Brookshear & Brylow

Büşra AMET
25360859417

Sunum İçeriği

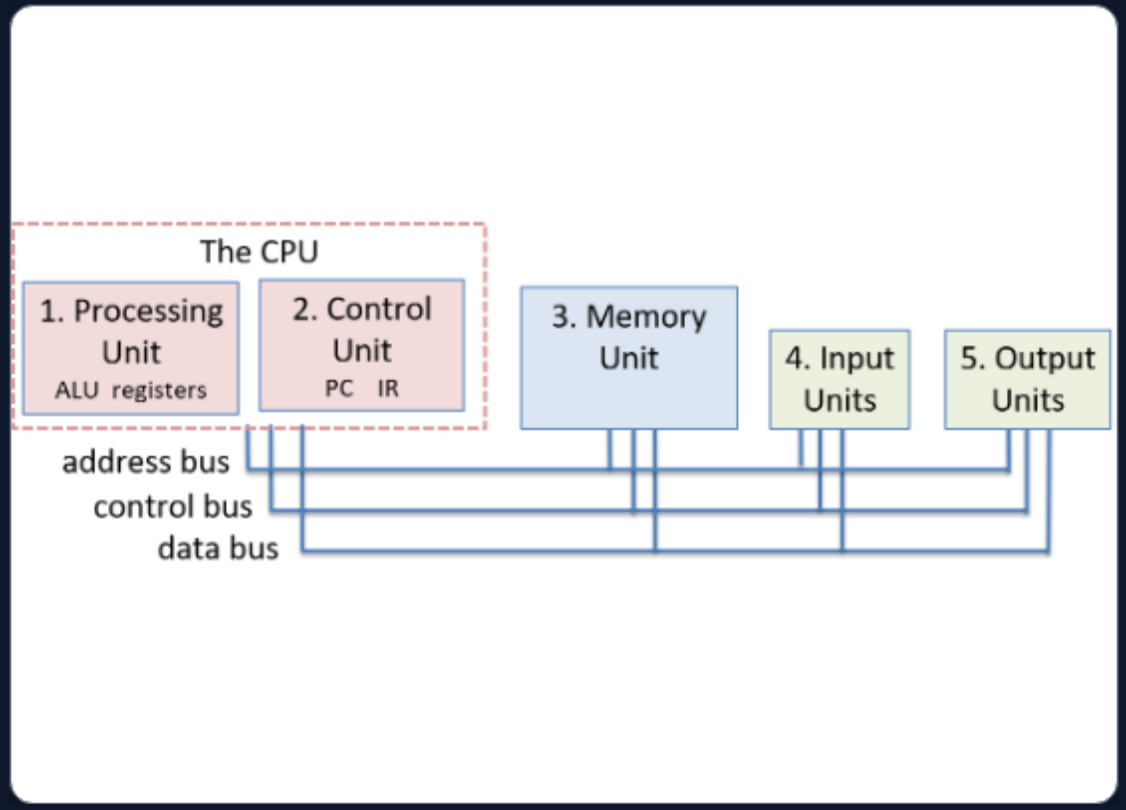
- **Bilgisayar Mimarisi** (Computer Architecture)
- **CPU Temelleri** (CPU Basics)
- **Saklı Program Kavramı** (Stored-Program Concept)
- **Makine Dili** (Machine Language)
- **Temel Mantık Kapıları**
- **Komut Listesi** (Instruction Repertoire)
- **Veri Aktarma Komutları** (Data Transfer)
- **Aritmetik & Mantık Komutları**
- **Sanal Bir Makine Dili Örneği**

1. Bilgisayar Mimarisi

Computer Architecture

Bilgisayar Mimarisi Nedir?

- Bilgisayar, programlanabilir bir veri işleme makinesidir.
- Temel olarak üç ana bölümden oluşur: **CPU**, **Bellek** ve **G/Ç (I/O)** birimleri.
- Bu birimler, verilerin işlenmesi ve saklanması için elektriksel devreler aracılığıyla birbirine bağlanır.
- Mimari, bu bileşenlerin nasıl organize edildiğini ve birbirleriyle nasıl iletişim kurduğunu tanımlar.



Temel Bileşenler

CPU

Merkezi İşlem Birimi. Veri işleme (manipulation) işlemlerinin yapıldığı beyindir.

Ana Bellek

Main Memory. Verilerin ve çalıştırılacak programların geçici olarak tutulduğu yerdir.

Veri Yolu

Veri yolu ya da bus Bileşenler arasında veri transferini sağlayan elektriksel bağlantı yoludur.

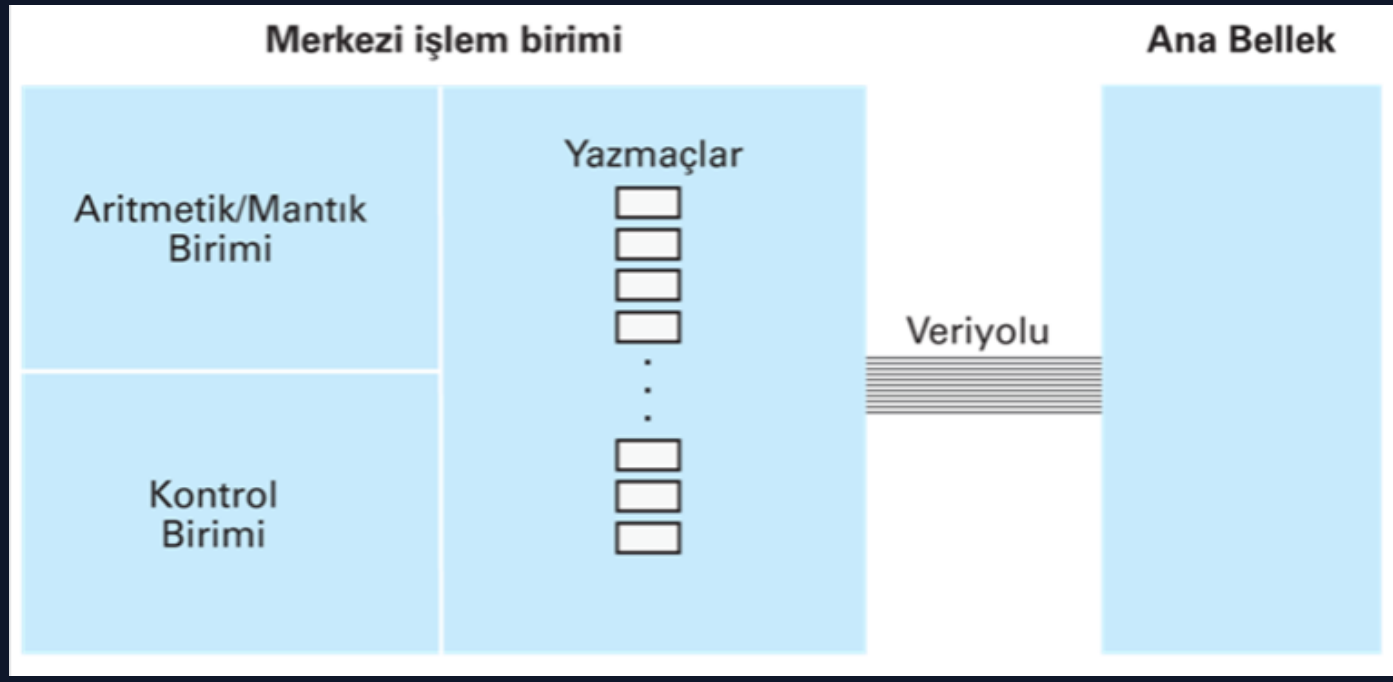
2. CPU Temelleri

Central Processing Unit Basics

Merkezi İşlem Birimi

CPU, bilgisayarın "hesaplama" kısmını oluşturur ve iki ana alt birimden meydana gelir:

- **Aritmetik/Mantık Birimi (ALU):** Matematiksel ve mantıksal işlemlerin yapıldığı devre.
- **Kontrol Birimi (Control Unit):** Bilgisayarın aktivitelerini koordine eden yönetici birim.
- Bu birimler, **Yazmaçlar (Registers)** adı verilen çok hızlı geçici hafıza hücrelerini kullanır. Kapasiteleri de çok düşüktür.



Aritmetik/Mantık Birimi (ALU)

- **Arithmetic/Logic Unit (ALU)**, veriler üzerinde asıl işlemleri gerçekleştiren kısımdır.
- Toplama, çıkarma gibi ***aritmetik işlemleri yapar.***
- AND, OR, XOR gibi **mantıksal** karşılaştırmaları gerçekleştirir.
- Kontrol biriminden gelen sinyallere göre, yazmaçlardan aldığı veriyi işler ve sonucu tekrar bir yazmaca yazar.

Kontrol Birimi (Control Unit)

- Bilgisayarın içindeki "trafik polisi" gibi çalışır.
- Bellekten hangi komutun işleneceğini ***belirler*** ve ***transferini*** yönetir.
- Komutları yorumlar ve ALU'ya ne yapması gerektiğini söyler. Bu yorumlama işlemine decode da denir
- Bileşenler arasındaki veri akışını senkronize eder.

Koordinasyon & Kontrol

Yazmaçlar (Registers)

- CPU'nun içinde bulunan, ***çok hızlı*** erişilebilen ***geçici*** veri saklama alanlarıdır. Kapasiteleri düşüktür.
- **Genel Amaçlı Yazmaçlar (General Purpose Registers):** İşlenecek verilerin veya işlem sonuçlarının geçici olarak tutulduğu yerlerdir (Örn: Toplama işlemi için iki sayının tutulması).
- Ana bellekten (Main Memory) çok daha hızlıdırlar ancak sayıları ***sınırlıdır***.

Özel Amaçlı Yazmaçlar: PC

Program Sayacı (PC)

Program Counter.

CPU'nun bir sonraki adımda çalıştıracağı komutun **bellek adresini** tutar.

Her komut işlenince değeri artırılarak ***sonraki komutu*** işaret eder.

Özel Amaçlı Yazmaçlar: IR

Komut Yazmacı (IR)

Instruction Register.

Şu anda çalıştırılmakta olan komutun **kendisini** tutar.

Kontrol birimi, bu yazmaçtaki veriyi analiz ederek (decode) ***ne yapılması gerektiğin***i anlar.

3. Saklı Program Kavramı

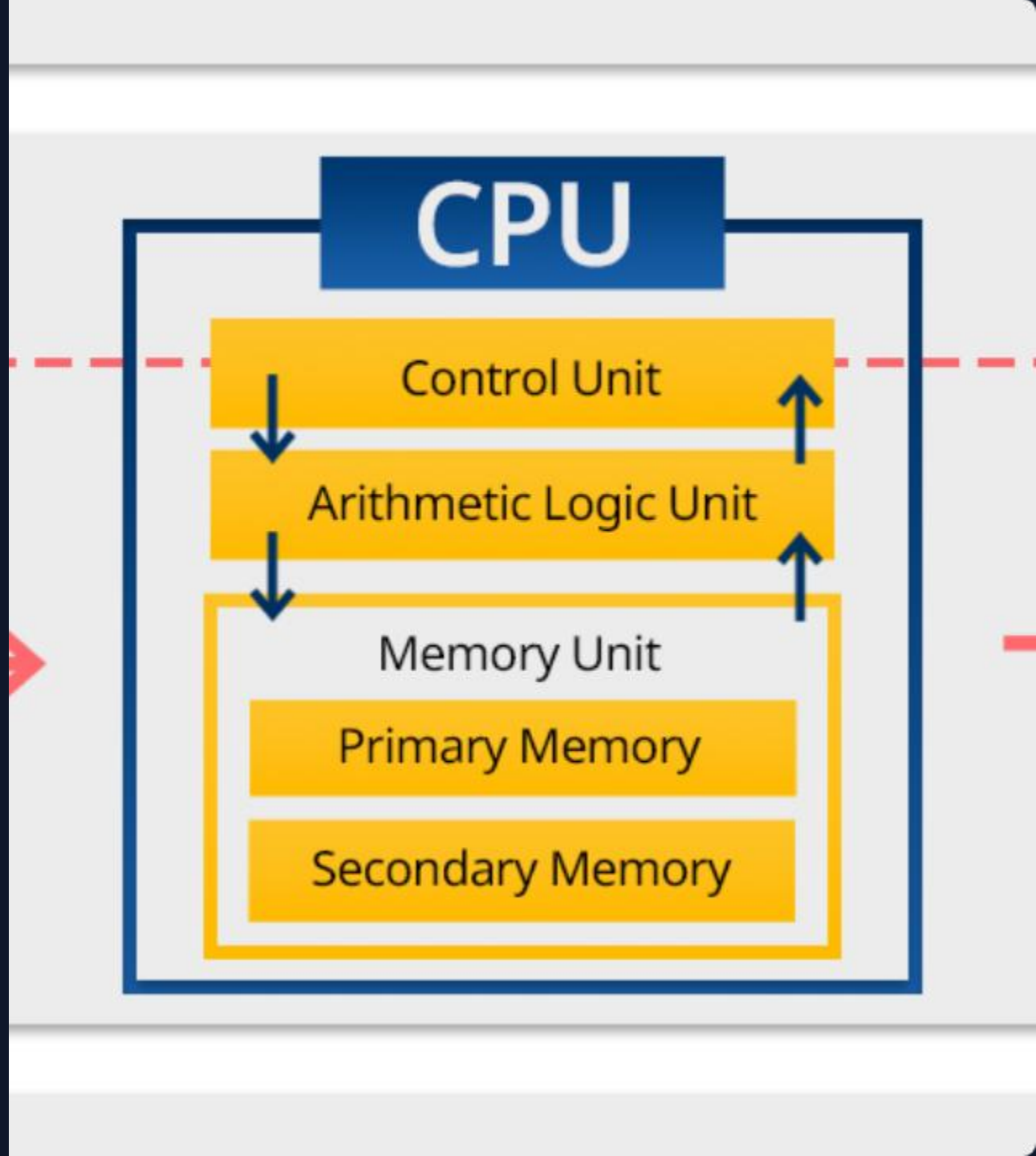
Stored-Program Concept

Saklı Program Nedir?

Erken dönem bilgisayarlarda programlar kablolarla fiziksel olarak kurulurdu.

Saklı Program Kavramı: Bir programın, tıpkı veriler gibi bilgisayarın ana belleğinde kodlanmış olarak saklanması fikridir.

Bu sayede bilgisayarın yaptığı işi değiştirmek için donanımı değiştirmeye gerek kalmaz; sadece bellekteki program değiştirmek yeterlidir.



4. Makine Dili

Machine Language

Makine Dili Mantığı

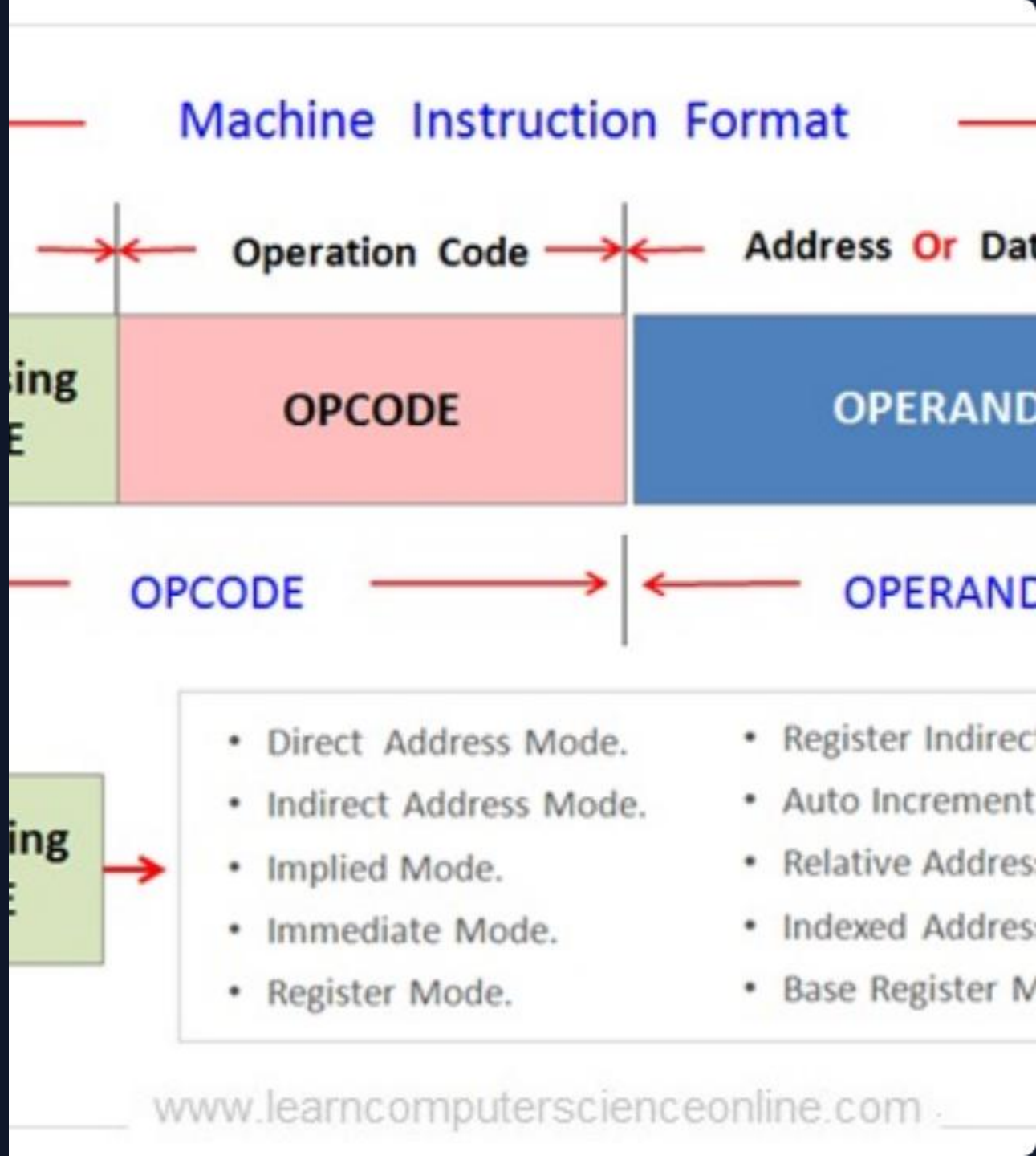
- Bilgisayarın donanımı (CPU) sadece elektriksel sinyalleri anlar.
- Bu durum 0 ve 1'lerden oluşan **İkili Sistem (Binary System)** ile temsil edilir.
- CPU'nun doğrudan anlayıp çalıştırabildiği komutlar kümesine **Makine Dili** denir.
- Her işlemci mimarisinin (x86, ARM vb.) kendine özgü bir makine dili vardır.

Komut Formatı

Tipik bir makine dili komutu iki ana alandan oluşur:

- **İşlem Kodu (Op-code):** Hangi işlemin yapılacağını belirtir (Örn: Topla, Yükle, Kaydet).
- **İşlenen (Operand):** İşlemin hangi veri veya bellek adresi üzerinde yapılacağını belirtir.

Örneğin: 16-bitlik bir komutta ilk 4 bit işlem türünü, kalan 12 bit adresi gösterebilir.



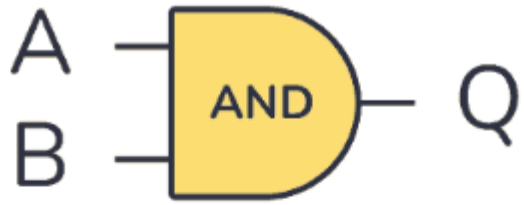
5. Temel Mantık Kapıları

AND, OR, NOT, XOR

Bilgisayar donanımının en temel yapı taşlarıdır. 0 (Yanlış) ve 1 (Doğru) sinyalleri üzerinde işlem yaparlar.

AND (VE) Kapısı

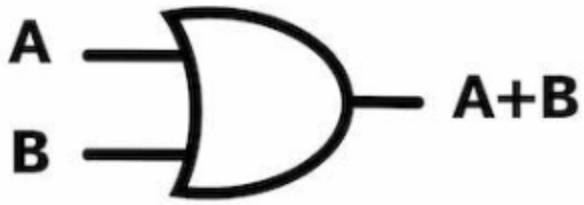
- **Tanım:** İki girişi vardır. Sadece ve sadece her iki giriş de **1** ise çıktı **1** olur.
- Diğer tüm durumlarda çıktı 0'dır.
- **ALU İlişkisi:** Maskeleme (Masking) işlemlerinde kullanılır. Belirli bitleri 0'a çevirmek için idealdir.



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

OR (VEYA) Kapısı

- **Tanım:** İki girişi vardır. Girişlerden **en az biri 1** ise çıktı **1** olur.
- Sadece her iki giriş de 0 ise çıktı 0 olur.
- **ALU İlişkisi:** Belirli bitleri 1'e ayarlamak) için de kullanılır.



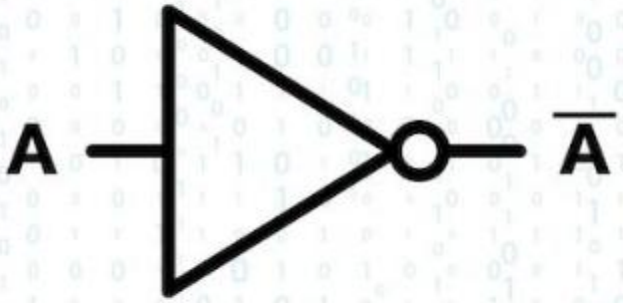
2 input OR gate

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT (DEĞİL) Kapısı

- **Tanım:** Tek bir girişi vardır. Girişi tersine çevirir.
- Giriş 1 ise çıktı 0; Giriş 0 ise çıktı 1 olur.
- Genellikle “Ters Çevirici” olarak da adlandırılır.

- **ALU İlişkisi:** Sayıların tümleyenini almak için kullanılır (çıkarma işleminde).



2 input NOT gate

A	$\overline{A}$
0	1
1	0

XOR (ÖZEL VEYA) Kapısı

- **Tanım:** Exclusive OR olarakta adlandırılır girişler **farklı** ise çıktı **1** olur.
- Girişlerin ikisi de aynıysa (0-0 veya 1-1) çıktı 0 olur.
- **ALU İlişkisi:** İki verinin eşit olup olmadığını kontrol etmek için kullanılır. Sonuç 0 ise veriler eşittir.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

6. Komut Listesi

Instruction Repertoire & RISC vs CISC

Komut Kümeleri

- Bir CPU'nun tanıyıp çalıştırabildiği tüm komutların listesine "Instruction Set" denir.
- Farklı işlemci tasarımları, verimlilik ve karmaşıklık açısından farklı yaklaşımlar benimser.
- İki temel tasarım felsefesi vardır: **RISC** ve **CISC**.

RISC Yaklaşımı

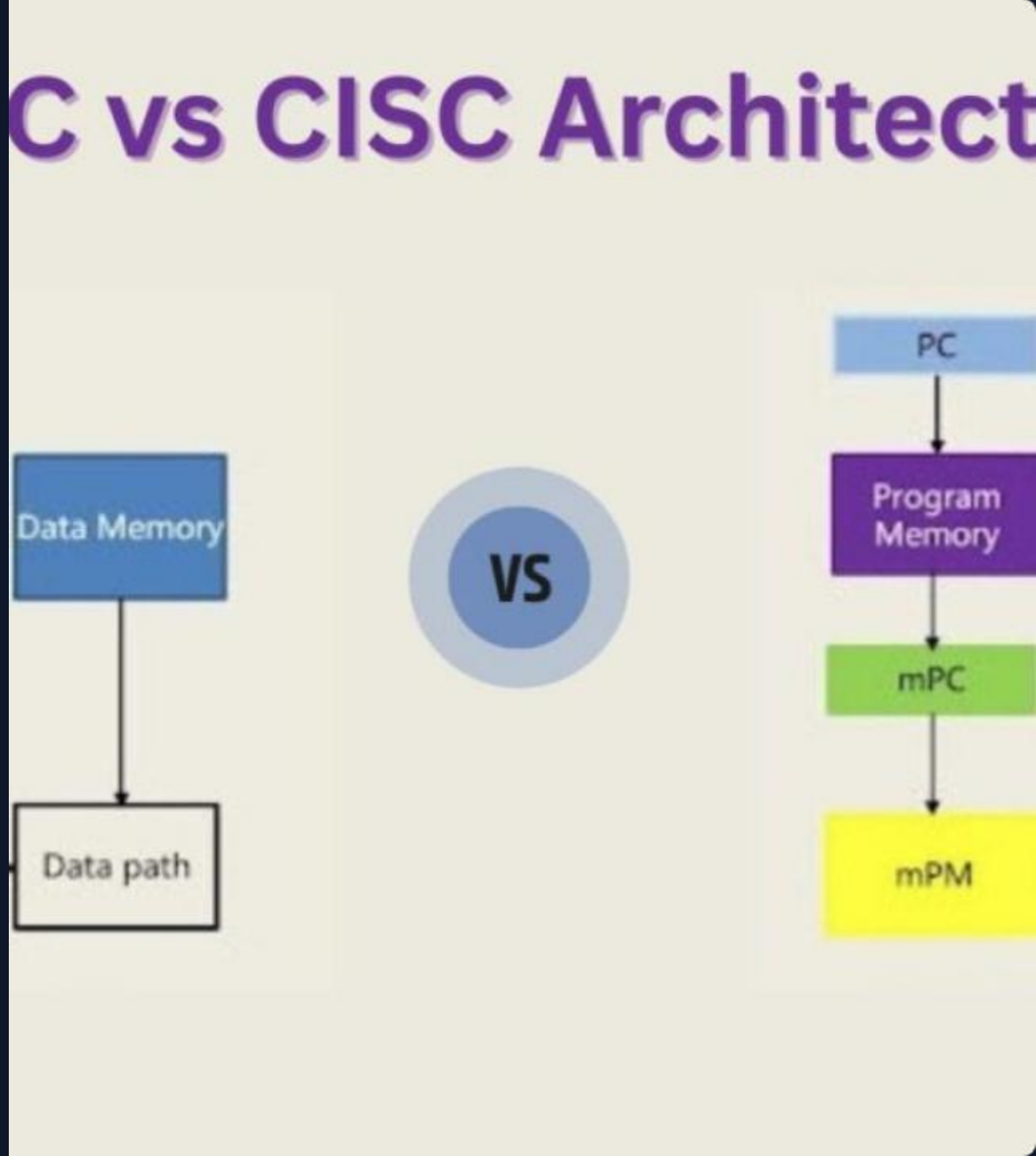
- **Reduced Instruction Set Computer** (İndirgenmiş Komut Takımlı Bilgisayar).
- Felsefe: Az sayıda, basit ve hızlı çalışan komutlar kullanmak.
- Her komut genellikle sabit uzunluktadır ve tek bir saat döngüsünde çalışır.
- Karmaşık işlemler, basit komutların birleştirilmesiyle yapılır (Örn: PowerPC, ARM).
- Donanım daha basittir, ancak programlar daha fazla satırdan oluşabilir.

CISC Yaklaşımı

- **Complex Instruction Set Computer** (Karmaşık Komut Kümeli Bilgisayar).
- Felsefe: Tek bir komutla karmaşık ve çok adımlı işlemleri yapabilmek.
- Komutlar değişken uzunlukta olabilir (Örn: x86 mimarisi).
- Programlar daha az bellek kaplar ancak CPU tasarımı daha karmaşıktır.
- Tek bir komut hem veri yükleyip, hem toplayıp, hem de kaydedebilir.

RISC vs CISC

Özellik	RISC	CISC
Komut Sayısı	Az ve Basit	Çok ve Karmaşık
Komut Boyutu	Sabit	Değişken
Donanım	Daha Basit	Daha Karmaşık
Enerji Verimi	Genelde Yüksek	Genelde Düşük



7. Veri Aktarma Komutları

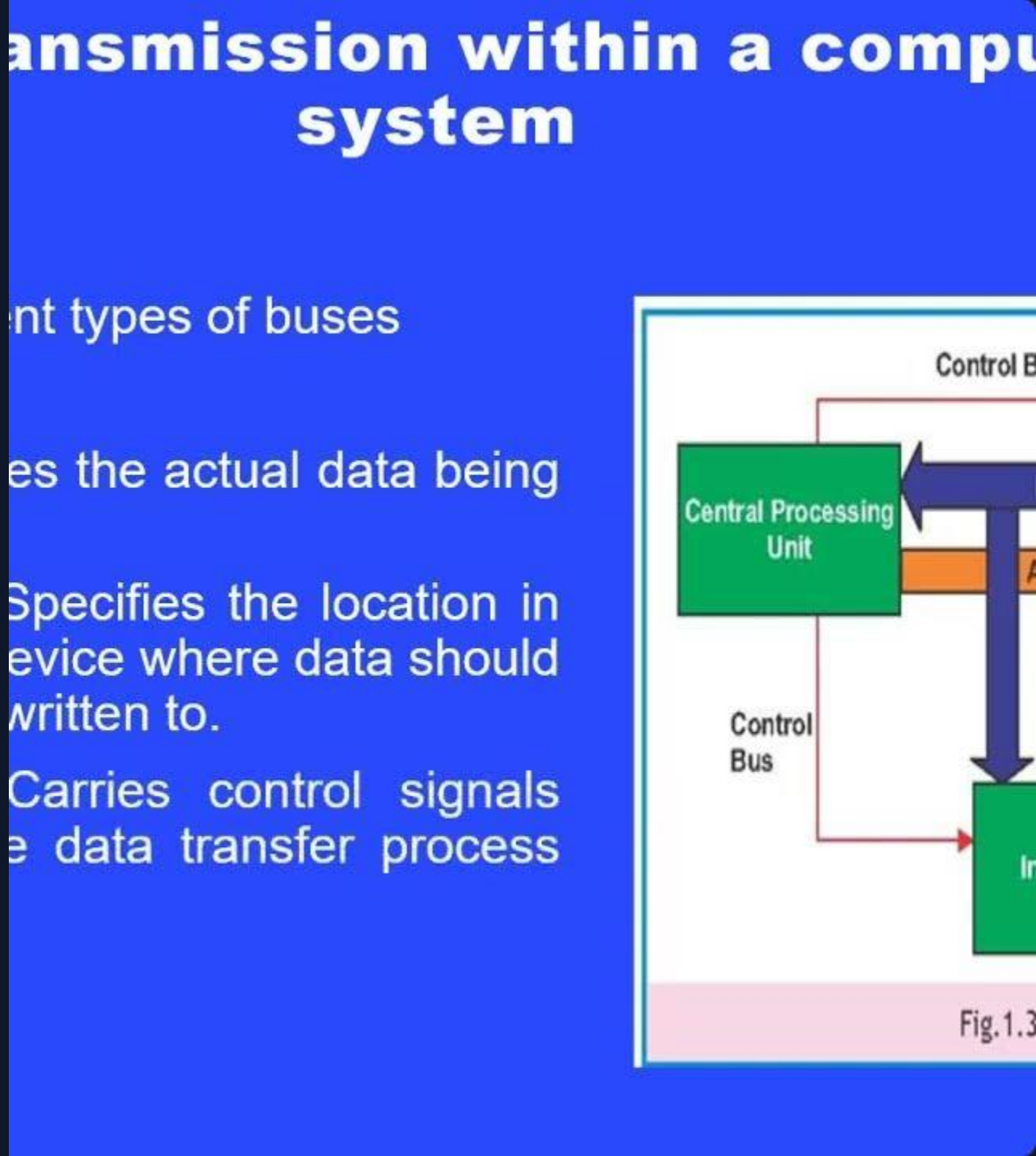
Data Transfer Instructions

Kopyalama İşlemi

Veri aktarma komutları aslında veriyi "taşımaz", **kopyalar**.

Kaynak (Source) konumundaki veri değişmeden kalır, hedef (Destination) konumundaki verinin üzerine yazılır.

Bu komutlar temel olarak CPU ile Ana Bellek arasındaki veri trafiğini yönetir.



LOAD (Yükle) Komutu

- **Amaç:** Ana bellekten CPU içindeki bir yazmaca (Register) veri getirmek.
- Örnek İşlem: "Bellek adresi 1A'daki veriyi al, Register 3'e koy."
- Bu işlem sırasında bellek hücresindeki orijinal veri silinmez, sadece bir kopyası alınır.
- CPU'nun veriyi işlemesi için önce onu LOAD etmesi gerekir.

STORE (Kaydet) Komutu

- **Amaç:** CPU içindeki bir yazmaçta bulunan işlem sonucunu ana belleğe göndermek.
- Örnek İşlem: "Register 5'teki sonucu al, Bellek adresi B2'ye yaz."
- Bu işlem, bellekteki o adreste daha önce ne varsa üzerine yazar (overwrite) ve eski veriyi siler.
- Programın çıktısını saklamak için kullanılır.

8. Aritmetik & Mantık Komutları

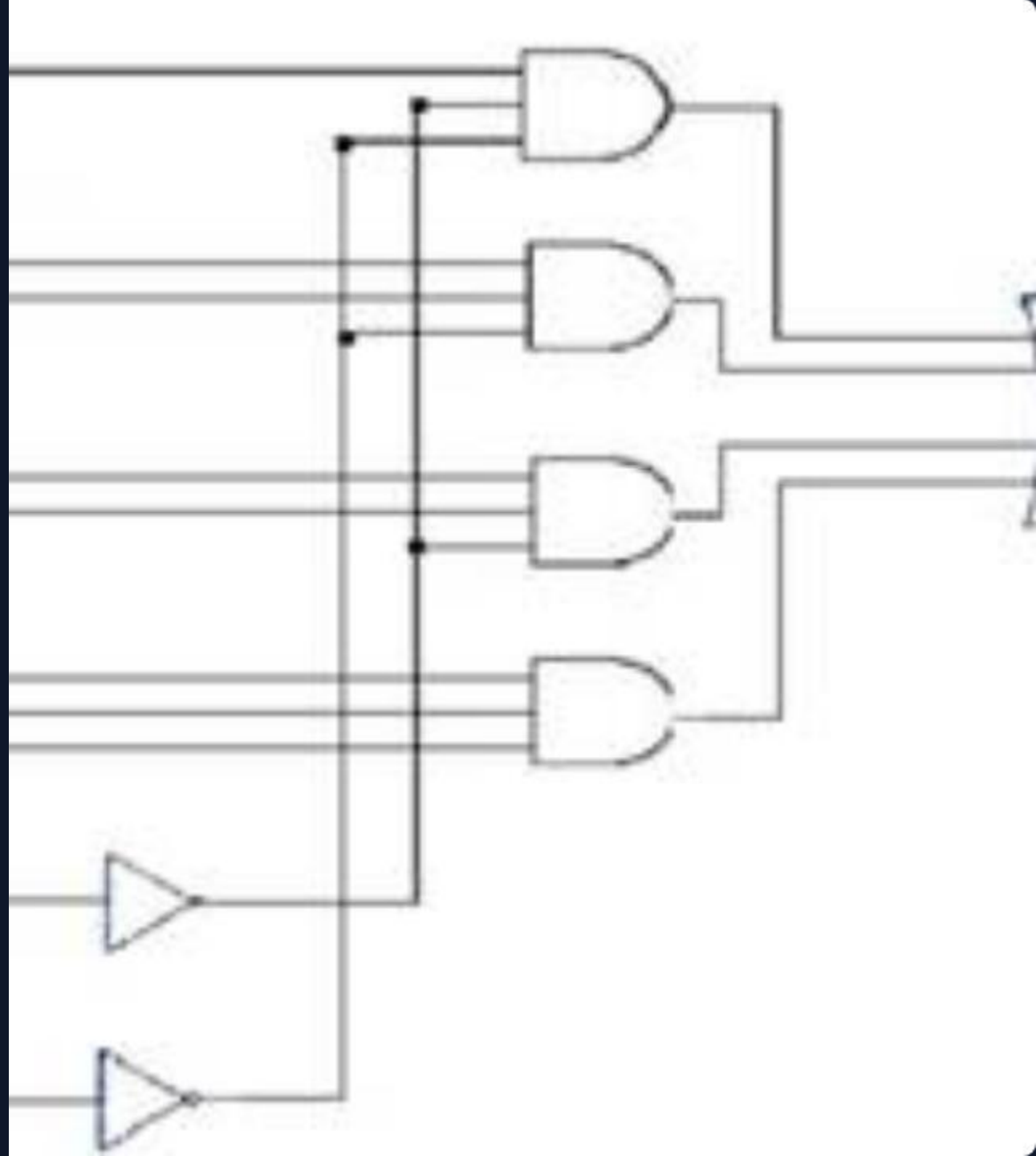
Arithmetic & Logic Instructions

Aritmetik İşlemler

Bu komutlar ALU'ya (Arithmetic/Logic Unit) hesaplama yapma emri verir.

Temel işlemler:

- **Toplama (ADD):** İki yazmaçtaki değeri toplar.
- **Çıkarma (SUBTRACT):** Değerleri birbirinden çıkarır.
- **Bölme/Çarpma:** Mimarisine göre doğrudan veya tekrarlı toplamalarla yapılır.



Kaydırma (SHIFT) ve Döndürme (ROTATE)

- Verinin bitlerini sağa veya sola hareket ettirme işlemleridir.
- **SHIFT:** Bitleri kaydırır, boşalan yere 0 gelir. (Matematiksel olarak 2 ile çarpma veya bölmeye denk gelir).
- **ROTATE:** Bir uçtan çıkan bit, diğer uçtan tekrar girer (Veri kaybı olmaz).
- Kitapta verilen örnek makinede "Sağa Döndürme” komutu bulunmaktadır.

9. Kontrol Komutları

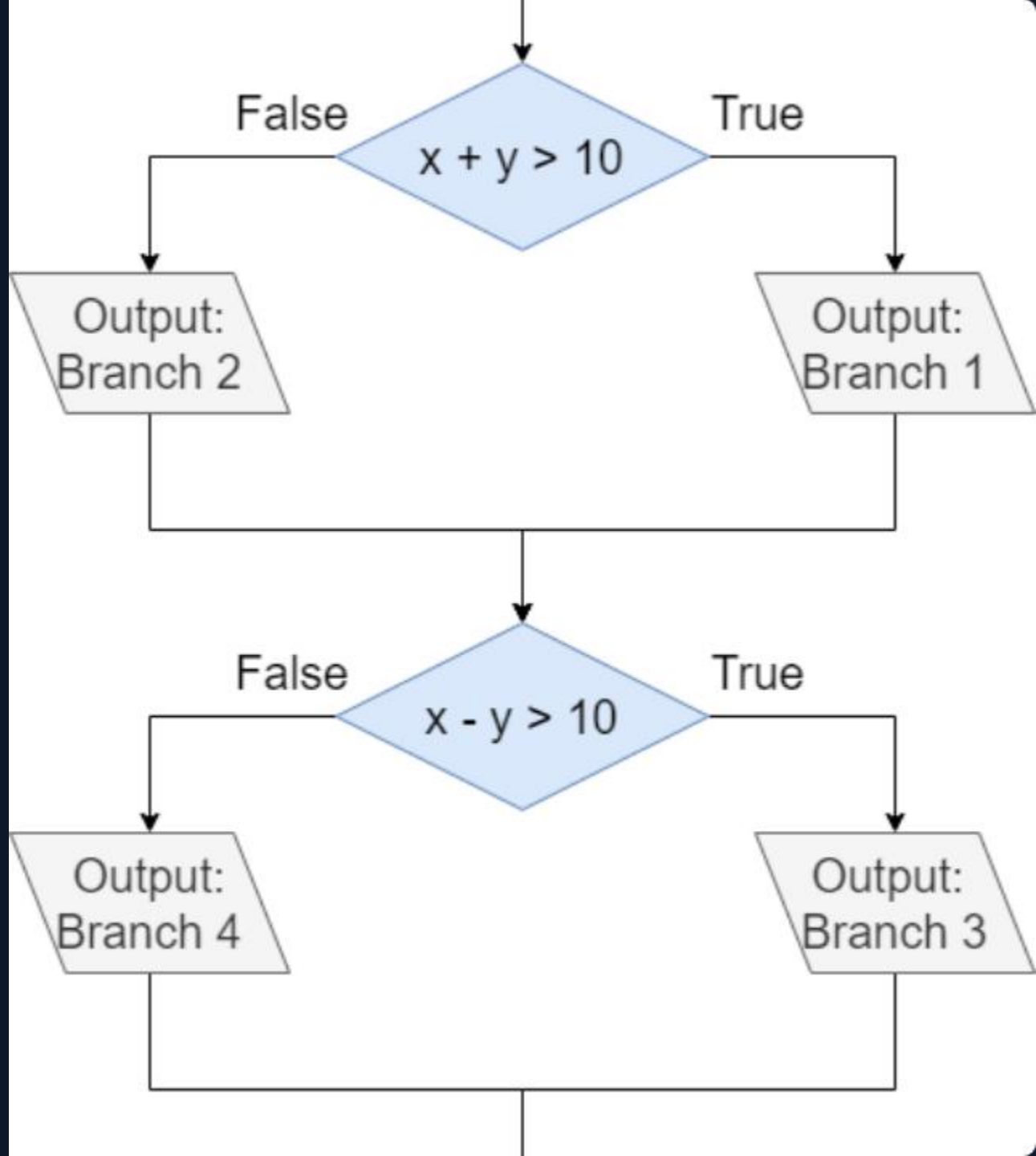
Control Instructions

Program Akışını Yönetmek

Normalde CPU komutları sırayla işler (PC otomatik artar).

Kontrol komutları, bu sırayı değiştirerek programın başka bir satıra atlamasını sağlar (Döngüler ve IF yapıları).

İki türü vardır: Koşulsuz ve Koşullu Dallanma.



JUMP (Dallanma) Komutları

Koşulsuz JUMP

Hiçbir şart aramadan doğrudan belirtilen bellek adresine gitmeyi sağlar (GOTO mantığı). Döngülerin başa dönmesi için kullanılır.

Koşullu JUMP

Sadece belirli bir şart sağlanıyorsa (Örn: "0 ise") atlama yapar. Aksi halde sıradaki komuttan devam eder (IF-ELSE mantığı).

HALT Komutu

- Programın çalışmasının bittiğini CPU'ya bildirir.
- Makine döngüsünü (Fetch-Decode-Execute) durdurur.
- Bu komut verilmezse CPU bellekteki rastgele verileri komut sanıp işlemeye devam edebilir, bu da hatalara yol açar.

10. Sanal Bir Makine Dili

Example Machine Architecture (Vole)

Örnek Makine Özellikleri

Kitapta anlatılan basitleştirilmiş makine şu özelliklere sahiptir:

16 Adet Yazmaç

Adresleri: 0'dan F'ye kadar (Hexadecimal).

256 Bellek Hücresi

Adresleri: 00'dan FF'ye kadar. Her hücre 8 bit (1 byte).

16-Bit Komut

Her komut 2 bellek hücresi kaplar (4 Hex rakamı).

Komut Formatı: 156C

Örnek Komut Kodu: 1 5 6 C

Op-code (1)	Operand 1 (5)	Operand 2 (6)	Operand 3 (C)
Ne yapılacak?	Hangi Yazmaç?	Hangi Bellek Adresi?	
1 = LOAD	Register 5	Adres 6C	

Anlamı: "Bellekteki 6C adresindeki veriyi al, 5 numaralı Yazmaca yükle."

Temel İşlem Kodları (Op-codes)

Kod	İşlem	Açıklama
1 RXY	LOAD	XY adresindeki veriyi R yazmacına yükle.
2 RXY	LOAD (Immediate)	XY sayısını (veri olarak) R yazmacına yükle.
3 RXY	STORE	R yazmacındaki veriyi XY adresine kaydet.
5 RST	ADD (Integer)	S ve T yazmaçlarını topla, sonucu R'ye yaz.
B RXY	JUMP	R yazmacı, 0 yazmacına eşitse XY adresine atla.
C 000	HALT	Programı durdur.

Basit Bir Toplama Programı

Kodlar

- 156C: 6C adresinden R5'e sayı al.
- 166D: 6D adresinden R6'ya sayı al.
- 5056: R5 ve R6'yı topla, sonucu R0'a yaz.
- 306E: R0'daki sonucu 6E adresine kaydet.
- C000: Dur.

Açıklama

Bu program bellekteki iki sayıyı alır, CPU içinde toplar ve sonucu belleğe geri yazar.

Makine dili seviyesinde işlem bu kadar küçük adımlarla gerçekleşir.