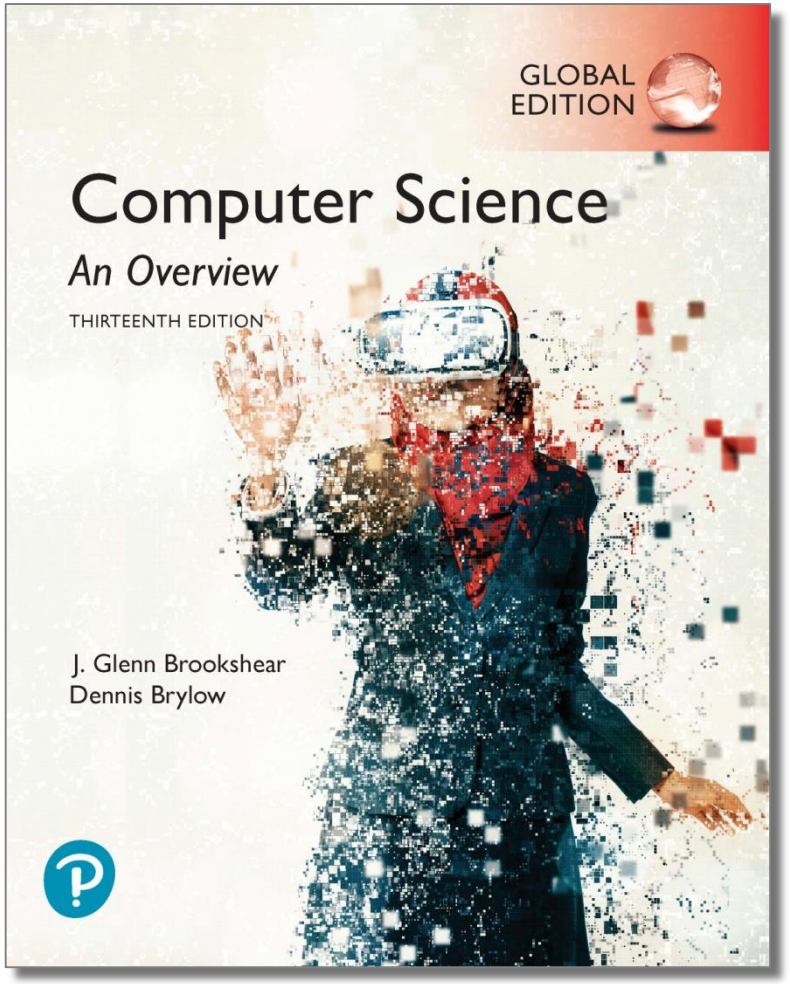
# Bilgisayar Bilimine Giriş

13. Baskı, Global Edition




## Bölüm 2

Veri İşleme

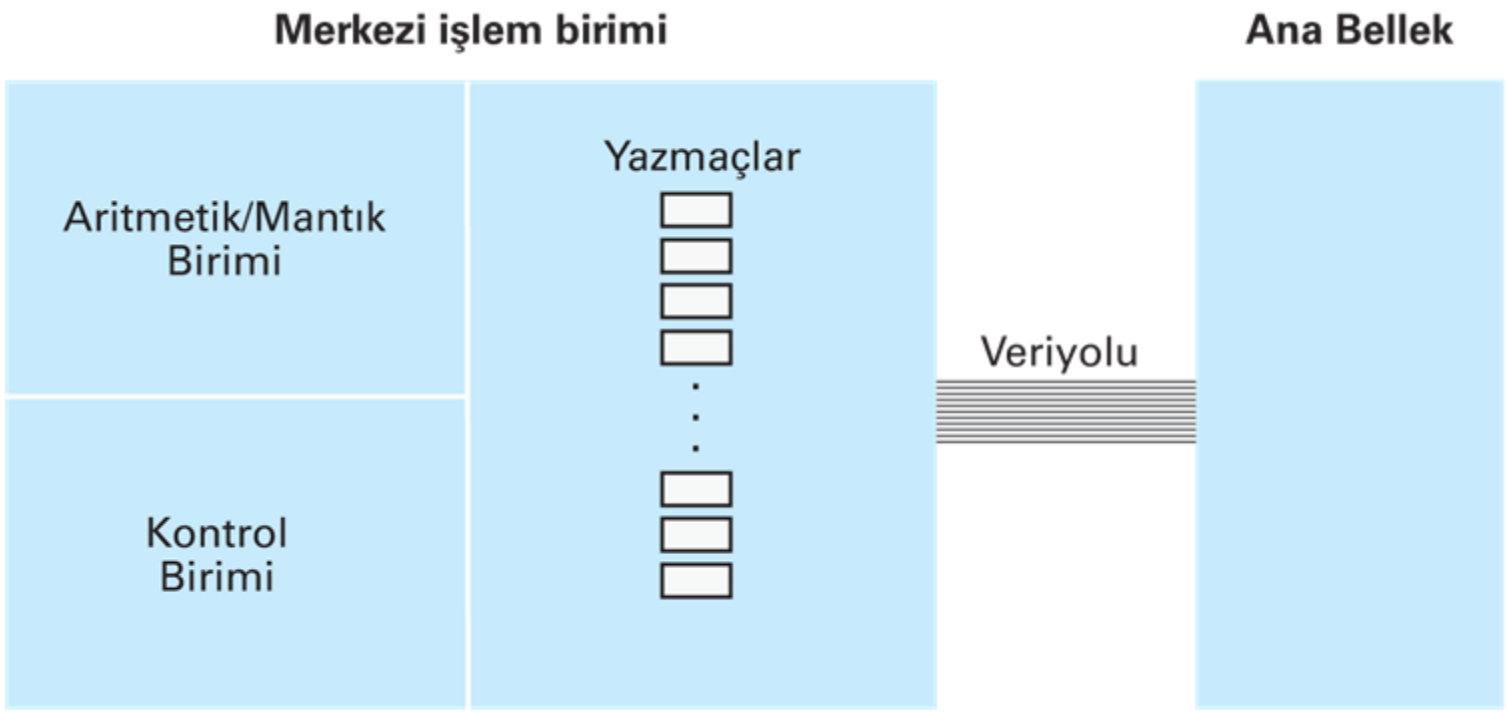
# Bölüm 2: Veri İşleme

- 2.1 Bilgisayar Mimarisi
- 2.2 Makine Dili
- 2.3 Program Yürütme
- 2.4 Aritmetik/Mantık
- 2.5 Diğer Cihazlarla İletişim
- 2.6 Veri İşlemeyi Programlamak
- 2.7 Diğer Mimariler

# 2.1 Bilgisayar Mimarisi

- Central Processing Unit (CPU) (Merkezi İşlem Birimi)
  - Aritmetik/Mantık Birimi
  - Control Birimi
  - Yazmaç Birimi (Register Unit)
    - Genel  yazmaçlar
    - Özel amaçlı yazmaçlar
- Veriyolu(Bus)
- Ana Bellek

# Şekil 2.1 Veriyolu(Bus) ile birbirine bağlanmış Ana Bellek ve Merkezi İşlem Birimi (CPU)



# Depolanmış Program Kavramı

Bir program bit desenleri şeklinde kodlanıp Ana Bellekte depolanabilir. Kontrol Birimi tarafından çıkartılabilir, çözülebilir ve çalıştırılabilir.

# 2.2 Makine Dili

- **Makine talimatı:** Bir talimat (yönerge)CPU tarafından çalıştırılabilrn bir bit deseni olarak kodlanır.
- **Makine dili:** Makinenin anladığı bütün talimatların dizisidir.

# Makine Dili Felsefeleri

- Azaltılmış Komut Kümeli Mikroişlemciler(Reduced Instruction Set Computing)(RISC)
- Az, basit, etkili ve hızlı komutlar
  - Örnekler: Apple/IBM/Motorola'nın PowerPC'si ve ARM
- Karmaşık Komut Kümeli Mikroişlemciler(Complex Instruction Set Computing) (CISC)
  - Fazla ,kullanışlı ve güçlü komutlar
  - Örnek: Intel

# Makine Komut Tipleri

- Veri Transfer: Veriyi bir yerden başka bir yere aktarma(Örnek: LOAD, STORE)
- Aritmetik/Mantık: Bit desenlerinde işlemler (Örnek: +, -, *, /, AND, OR, SHIFT, ROTATE)
- Kontrol: Programın direkt çalıştırılması (Örnek: JUMP, BRANCH)

# 2.2 Bellekte depolanmış veride toplama

Adım 1. Eklenecek değeri bellekten al ve yazmaca yerleştir.

Adım 2. Eklenecek diğer değeri bellekten al ve diğer yazmaca yerleştir.

Adım 3. Adım 1 ve 2'de **girdi** olarak **kullanılan yazmaçlardaki bilgileri alarak toplama devresini aktifleştir ve sonucu** tutmak için tasarlanmış diğer yazmaca ver.

Adım 4. Sonucu bellekte depola.

Adım 5. Dur.

# Şekil 2.3 Bellekte depolanmış değerde bölme

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

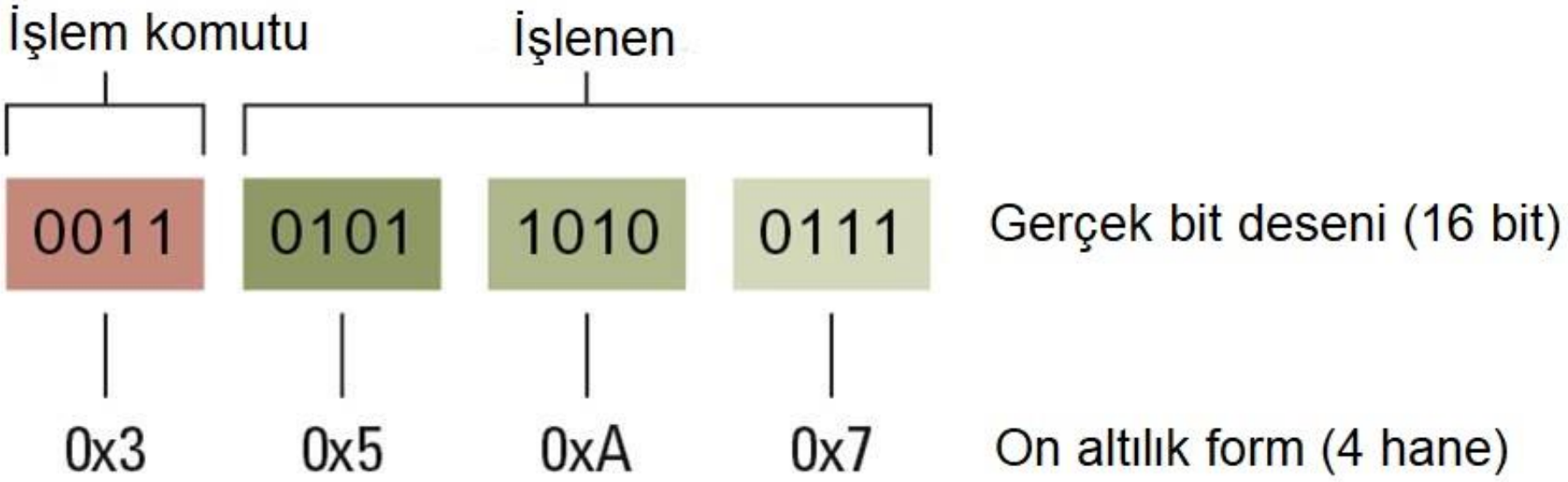
Adım 5. Üçüncü yazmacın içeriğini belleğe KAYDET

Adım 6. Dur.

# Makine komutunun bölümleri

- **İşlem komutu(Op-code):** Uygulanacak işlemi belirtir
- **İşlenen(Operand):** İşlem hakkında daha kapsamlı bir bilgi verir
  - İşlenen değişkenler işlem komutuna göre yorumlanır

# Şekil 2.5 Bir Makine dili komutunun yapısı

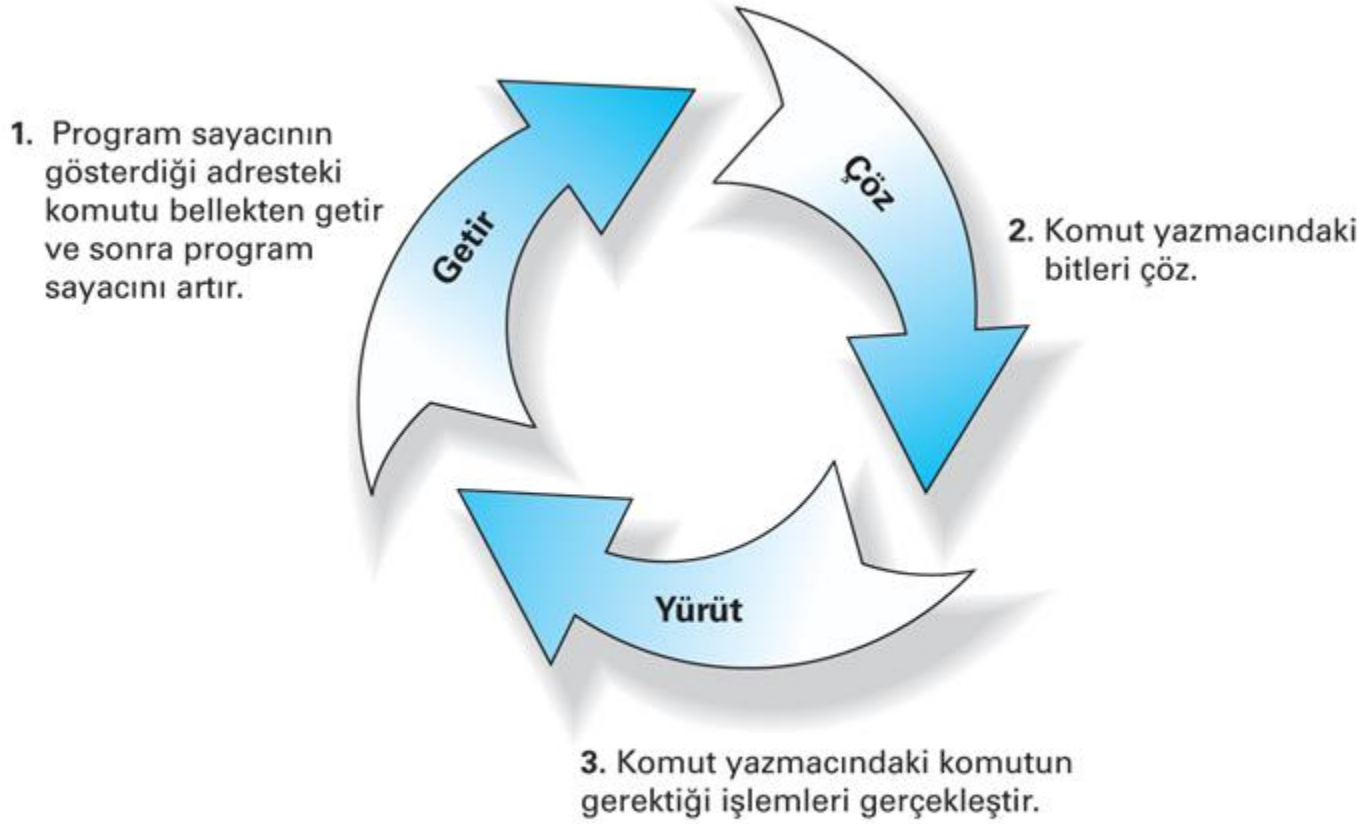


# 2.3 Program Çalıştırma

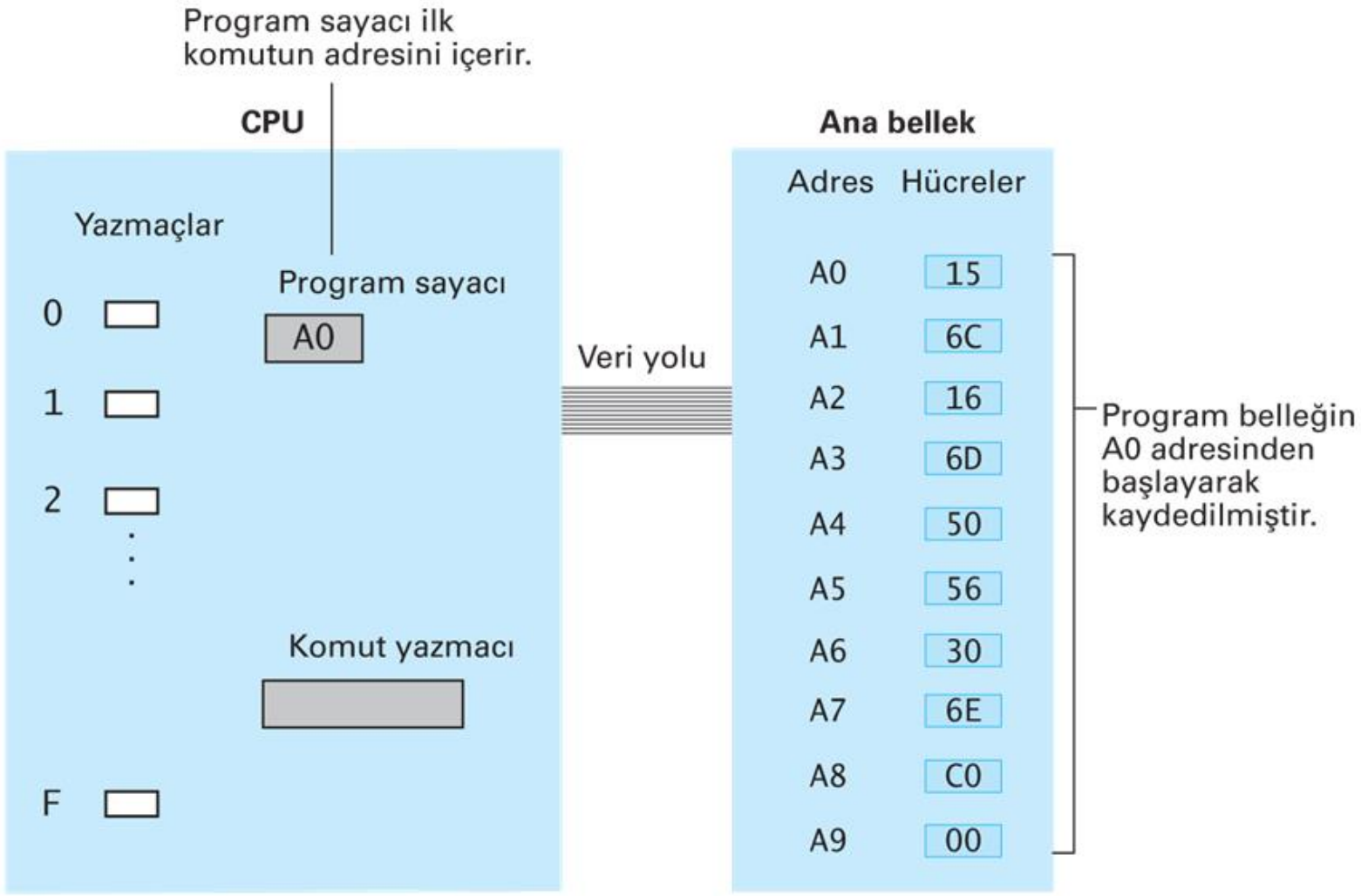
- İki özel amaçlı kayıt tarafından kontrol edilir
  - Komut yazmacı
    - Mevcut komutu tutar
  - Program sayacı
    - Sıradaki komutun adresini tutar

- Makine döngüsü: (bu üç adımı tekrarlar)
  - Bilgiyi getir,çözümle,yürüt(yap)

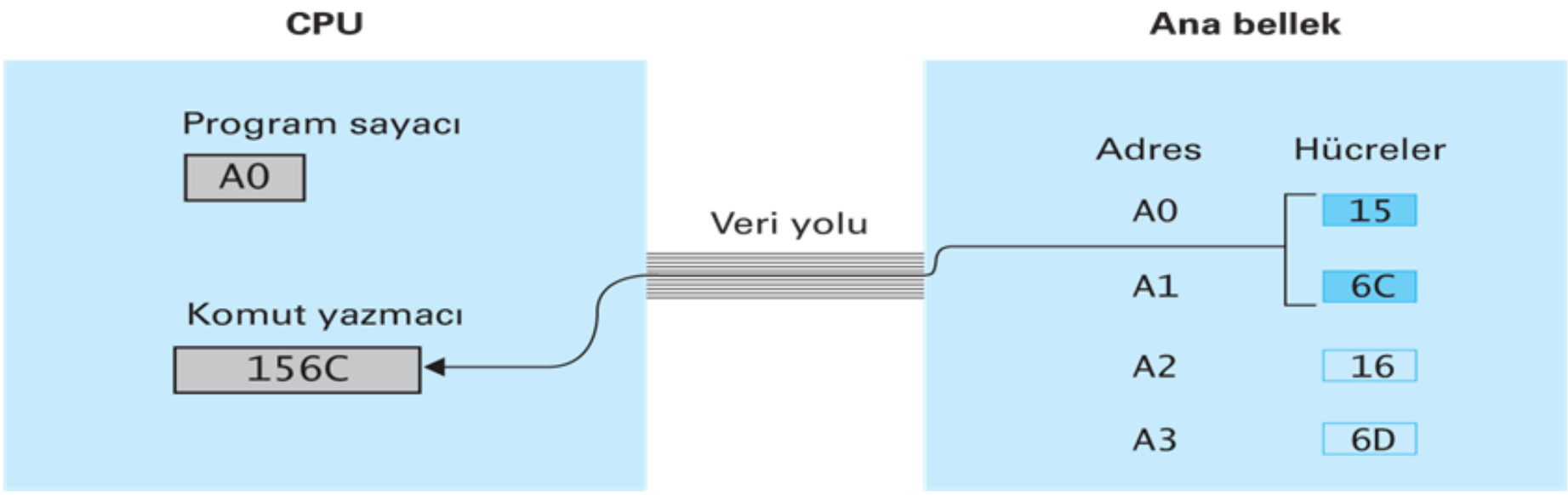
# Şekil 2.8 Makine döngüsü



# Şekil 2.10 Şekil 2.7'deki program ana bellekte depolanmış ve yürütülmeye hazır

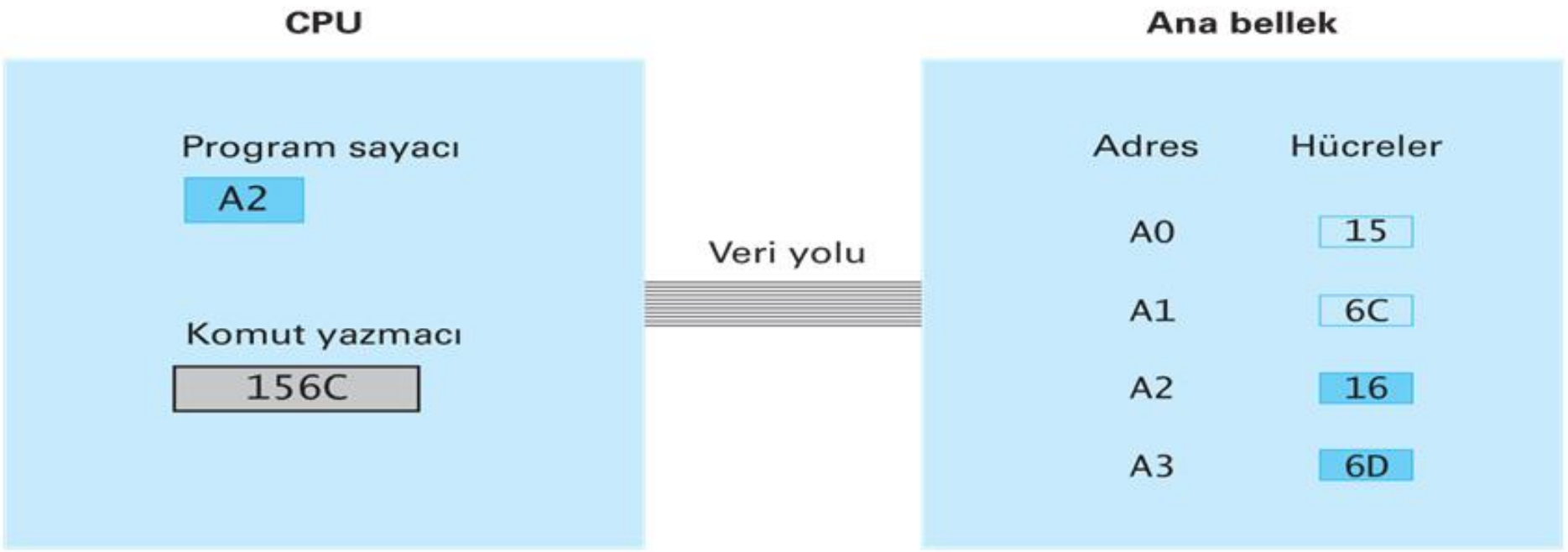


# Şekil 2.11 Makine döngüsünün bilgiyi alma adımının işleyişi




**a.** Getir adımının başında A0 adresinde başlayan komut bellekten okunur ve komut yazmacına yerleştirilir.

# Şekil 2.11 Makine döngüsünün bilgiyi alma adımının işleyişi(devamı)




**b.** Daha sonra program sayacı artırılarak bellekteki bir sonraki komutu

# 2.4 Aritmetik/Mantık Komutları

- Mantık işlemleri:
  - VE(AND), VEYA(OR), Özel OR(XOR)
- Döndürme ve kaydırma işlemleri:
  - Dairesel kaydırma, mantıksal kaydırma, aritmetik kaydırma
- Aritmetik işlemler:
  - ekle, çıkar, çarp, böl

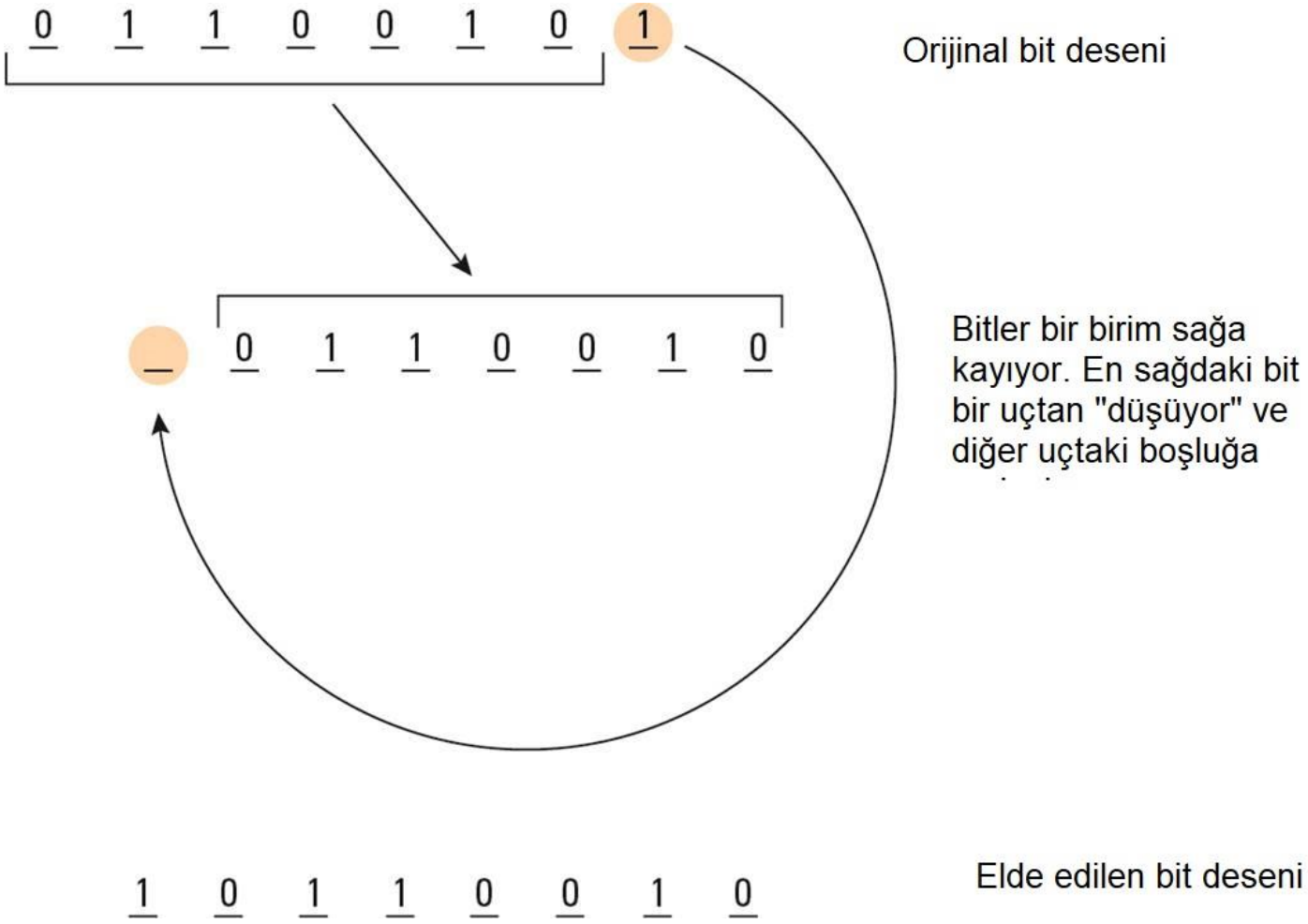
# ARİTMETİK MANTIK Operasyonları

## Mantık İşlemleri

•Tek bir çıkış dizgesi oluşturmak için iki dizgenin bitlerini birleştiren **bitsel işlemler**e genişletilebilirler. İki dizgeyi bir mantık işlemine sokmak için dizgelerin aynı sıralarında bulunan bitler basit mantık işlemine sokulurlar.

```
  10011010        10011010        10011010
AND 11001001    OR  11001001    XOR 11001001
  10001000        11011011        01010011
```

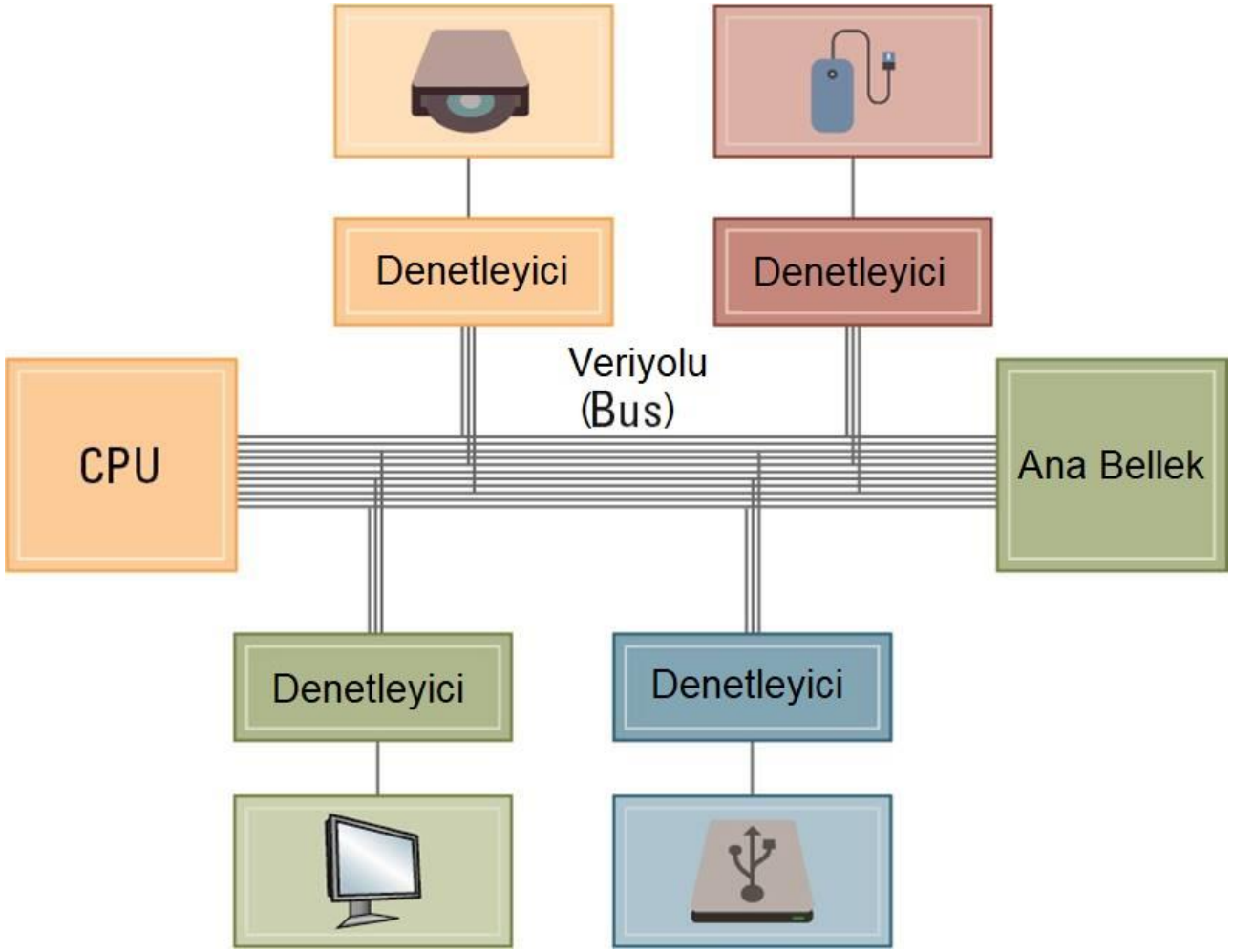
# Şekil 2.12 0x65 bit desenini sağa doğru bir bit kaydırma işlemi



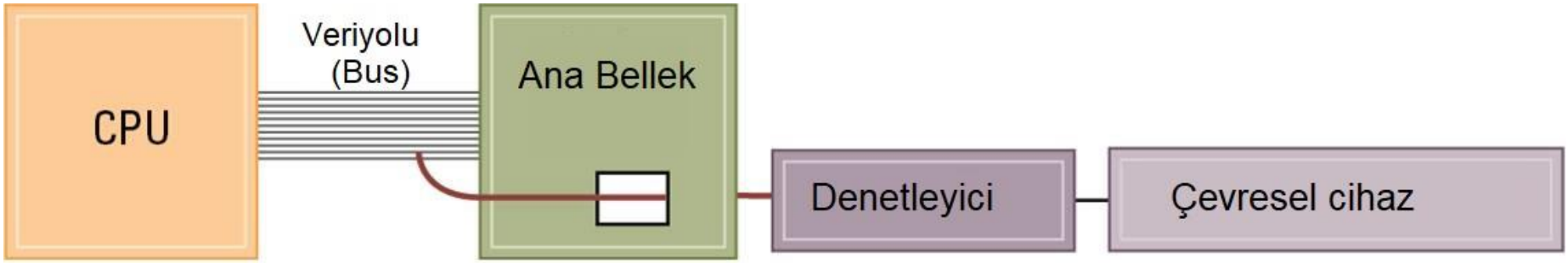
# 2.5 Diğer aygıtlarla iletişim

- **Denetleyici:** Bilgisayarın diğer aygıtlarla iletişimini yönetir
  - Özelleştirilmiş (aygıtın cinsine göre)
  - Genel amaçlı (USB, HDMI)
- **Kapı (Port):** Aygıtın bilgisayara bağlandığı yer
- **Bellek-haritalı giriş/çıkış:** CPU'ya bellek konumuymuş gibi gözüken aygıtlardır

# Şekil 2.13 Makinenin veriyoluna bağlı Denetleyiciler



# Şekil 2.14 Bellek-haritalı giriş/çıkış'ın görsel bir gösterimi



# Diğer aygıtlarla iletişim (devamı)

- **Direkt bellek erişimi (DMA):** Ana belleğin veriyolu üstündeki bir denetleyici tarafından erişimi
- **El sıkışma:** Bilgisayar ve çevresel cihaz arasındaki veri transferi süreci

# Diğer aygıtlarla iletişim (devamı)

- **Popüler iletişim medyası**
  - **Paralel iletişim:** Çoklu sinyalin aynı anda, her biri ayrı “hatta”gitmesi (bilgisayarın iç veriyolu)
  - **Seri iletişim:** Sinyallarin teker teker aynı ve tek “hat” üzerinden gitmesi (USB, FireWire)

# Veri İletişim Oranları

- Ölçü birimleri
  - bps:  saniye başına gönderilen bit(bits per second)
  - Kbps:  Kilo-bps (1,000 bps)
  - Mbps:  Mega-bps (1,000,000 bps)
  - Gbps:  Giga-bps (1,000,000,000 bps)
- Bant-genişliği: Mümkün olan maksimum hız

# 2.6 Veri programlama işlemi

- Programlama dilleri kullanıcıyı ana makine dilinin karmaşıklığından kurtarır:
  - Bir tek Python ifadesi onlarca belki de yüzlerce makine komut haritası kaplayabilir
  - Programcının, işlemcinin hangi mimariyi kullandığını bilmesine gerek yoktur(RISC yada CISC)

# Python Kodlama dilinde bit düzeni problemleri

```
print(bin(0b10011010 & 0b11001001))
#'0b10001000‘ Çıktısı verir


print(bin(0b10011010 | 0b11001001))
# '0b11011011'Çıktısı verir


print(bin(0b10011010 ^ 0b11001001))
# '0b1010011'Çıktısı verir
```

# Kontrol yapıları

- If ifadesi:

```
if (su_sicakligi > 140):
    print(‘Su cok sicak!')
```

- While ifadesi:

```
while (n < 10):
    print(n)
    n = n + 1
```

# Fonksiyonlar

- **Fonksiyon**: Verilen parametre(ler) üstünde gösterilmesi gereken işlem serisine verilen addır
- **Fonksiyon çıktısı**: İfadedeki fonksiyonun görünümü

```
x = 1034
y = 1056
z = 2078
biggest = max(x, y, z)
print(biggest)  # '2078‘ yazdırır
```

# Fonksiyonlar (devamı)

- **Argüman değeri**: Parametreye girilmiş değer
- Fonksiyonlar bir değere **geri döner**
- **void fonksiyonları**, veya **prosedürler,** bir değere geri dönmezler

```
sideA = 3.0
sideB = 4.0
# Pisago teoremini kullanarak üçüncü kenarın
uzunluğunu hesaplar
hypotenuse = math.sqrt(sideA**2 + sideB**2)
print(hypotenuse)
```

# Giriş / Çıkış

```
# Nizami bir üçgenin hipotenüsünü hesaplar

import math

# Kenar uzunluklarının girişi, ilk deneme
sideA = int(input(‘A kenarının uzunluğu
nedir?'))
sideB = int(input('B kenarının uzunluğu
nedir?'))

# Üçün kenarı Pisagor Teoremi ile hesaplar
hypotenuse = math.sqrt(sideA**2 + sideB**2)

print(hypotenuse)
```

# Maraton İdman Programı

```
# Maraton idman yardımcısı.

import math

# Bu fonksiyon dakikaları ve saniyeleri
# sadece saniye şekline dönüştürür.
def total_seconds(min, sec):
    return min * 60 + sec

# bu fonksiyon saatte alınan mil bilgisini kullanıp bir
milin kaç saniyede alınacağını hesaplar
def speed(time):
    return 3600 / time
```

# Maraton İdman Programı(devamı)

```
# Kullanıcıdan bir mil için gerekli süreyi ve mesafeyi
al.
pace_minutes = int(input('Minutes per mile? '))
pace_seconds = int(input('Seconds per mile? '))
miles = int(input('Total miles? '))

# Hızı hesapla ve yazdır.
mph = speed(total_seconds(pace_minutes, pace_seconds))
print('Hızınız ' + str(mph) + ' mph'dir')

# Çalışma için ayrılmış zamanı hesapla.
total = miles * total_seconds(pace_minutes, pace_seconds)
elapsed_minutes = total // 60
elapsed_seconds = total % 60

print('Çalışma süreniz ' + str(elapsed_minutes) +
                ' dakikadır ' + str(elapsed_seconds) + '
secs')
```

# Şekil 2.15 Örnek maraton idman verileri

| Bir mil için geçen süre | | | | Toplam geçen süre | |
|---|---|---|---|---|---|
| Dakika | Saniye | Mil | Hız (mil/saat) | Dakika | Saniye |
| 9 | 14 | 5 | 6.49819494584 | 46 | 10 |
| 8 | 0 | 3 | 7.5 | 24 | 0 |
| 7 | 45 | 6 | 7.74193548387 | 46 | 30 |
| 7 | 25 | 1 | 8.08988764044 | 7 | 25 |

# 2.7 Diğer Mimariler

- Üretilen işi artıran teknolojiler:
  - Küme komut işleme (Pipelining): Makine döngülerinin birden fazlasının aynı anda çalıştırılması
  - Paralel işleme: Birbirine benzeyen birçok işlemci kullanır
    - SISD: Tek komut, Tek veri
      - Paralel işleme olmaz
    - MIMD: Çoklu komut, Çoklu veri
      - Farklı programlar, farklı veri
    - SIMD: Tek komut, Çoğul veri
      - Aynı program, farklı veri

BURAK EMRE TERZİ