

YAZILIM MÜHENDİSLİĞİNE GİRİŞ

HAFTA 4 VERİ İŞLEME

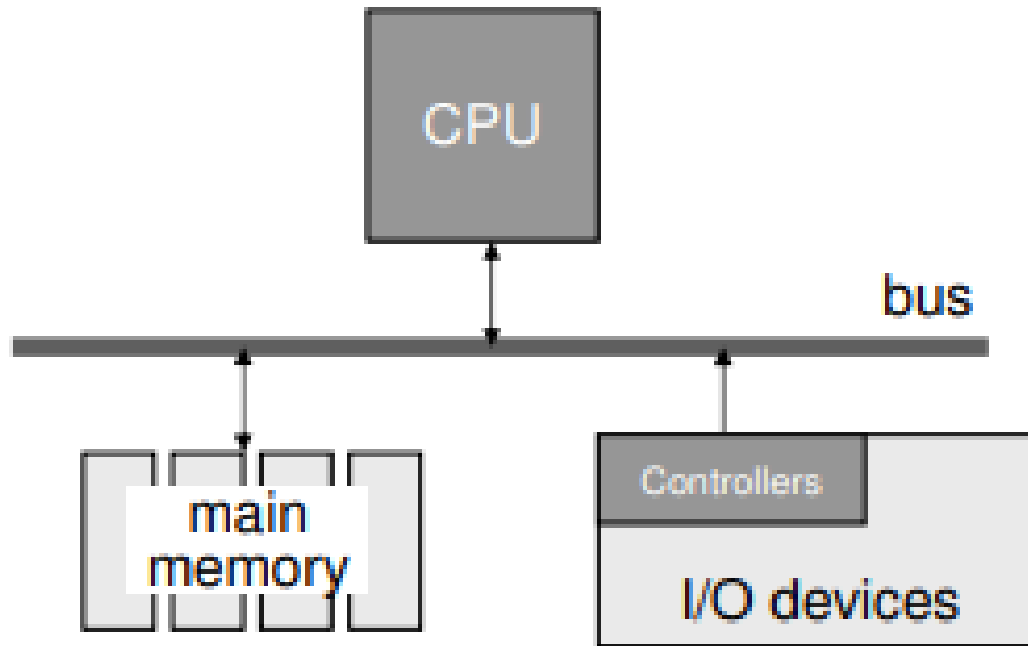
Dr. Öğr. Üyesi Ebru Aydoğan
eaydogan@mehmetakif.edu.tr

Ders İçeriği: Hafta 4

- Bilgisayar Mimarisi
- Makine Dili
- Program Yürütme
- Sanal Bir Makine Dili
- Aritmetik/Mantık İşlemleri
- Diğer Cihazlarla Haberleşme
- Diğer Mimariler

Bilgisayar Mimarisi

- Bir bilgisayarda verinin işlenmesini kontrol eden devreye Merkezi İşlem Birimi (Central Processing Unit – CPU) veya **işlemci** denir.
- Akıllı telefonlar ve diğer Mobil İnternet Cihazlarında (MID) bulunan CPU'lar ise daha küçük boyutlu olmalarından dolayı **mikroişlemci** olarak adlandırılır.



Bilgisayar Mimarisi

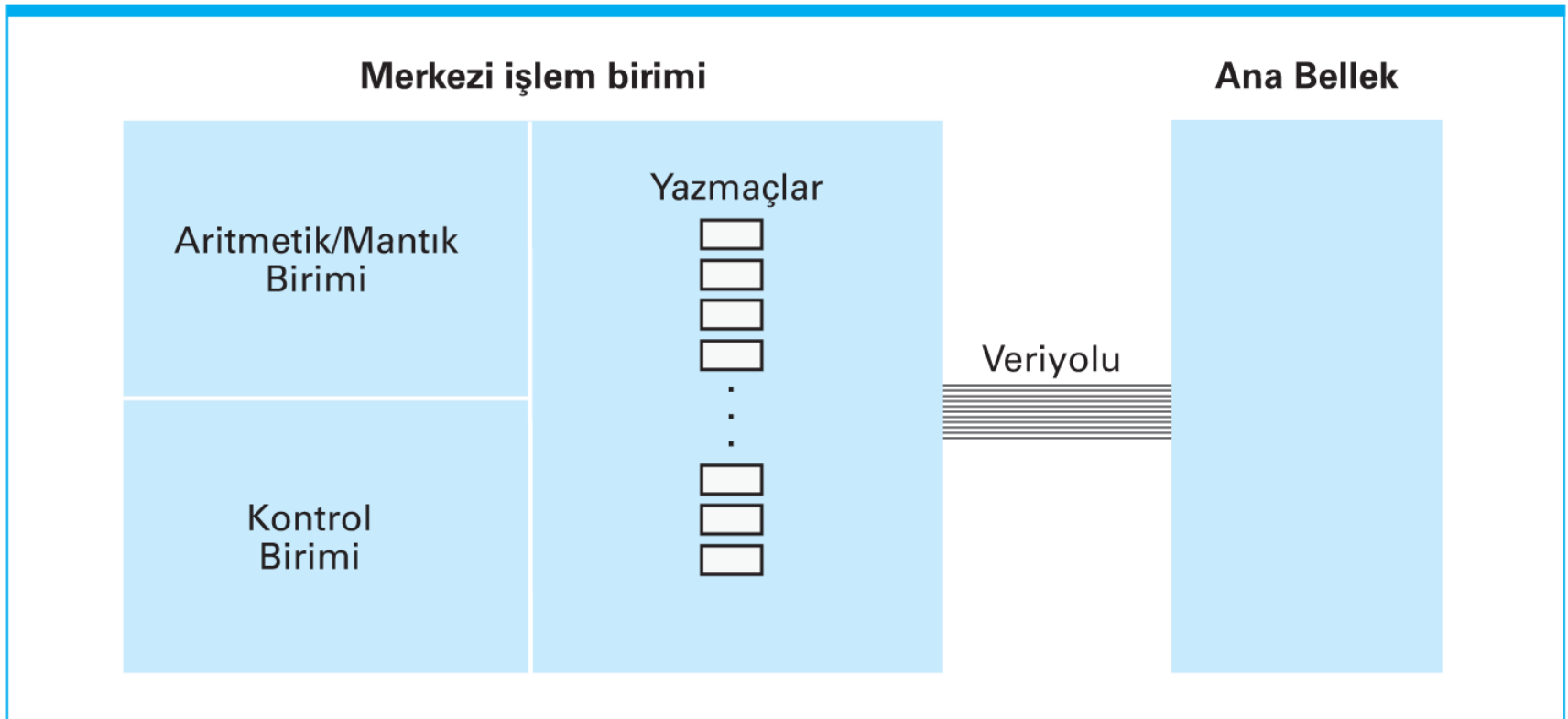
➤ CPU Mimarisi

- Bir CPU temel olarak aşağıdaki birimlerden oluşur:
 - Aritmetik / Mantık Birimi (ALU)
 - Kontrol Birimi (Control Unit)
 - Yazmaç Birimi (Registers)
- Ön Bellek (Cache Memory)
- Veri Yolu (Bus)
- Ana Bellek (Main Memory)

Bilgisayar Mimarisi

➤ CPU Mimarisi

- Bir veri yolu ile birbirine bağlanmış CPU ve Ana bellek



Bilgisayar Mimarisi

➤ CPU Mimarisi

- **Aritmetik / Mantık Birimi (ALU):** Bir mikroişlemcinin en önemli kısmını aritmetik ve lojik birimi oluşturur.
 - Aritmetik işlemler: Toplama, çıkarma, çarpma, bölme, kaydırma ve döndürme işlemlerini kapsar.
 - Mantıksal İşlemler: VE, VEYA, Özel VEYA (XOR), Değil (NOT), Karşılaştırma (=, <, >) gibi işlemleri kapsar.

Bilgisayar Mimarisi

➤ CPU Mimarisi

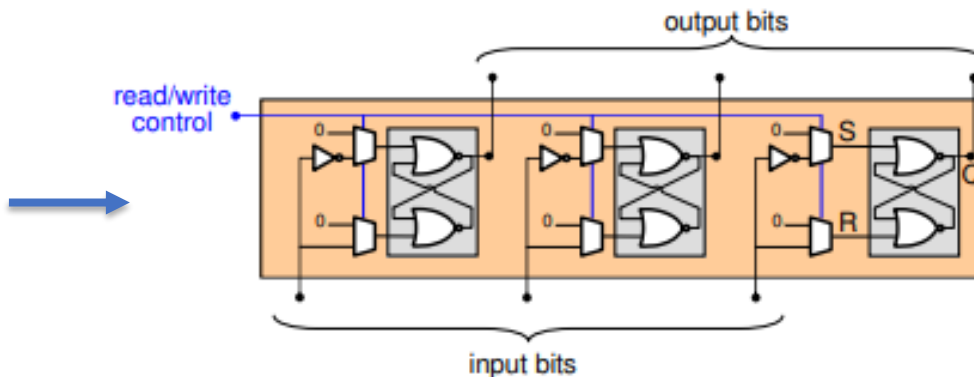
- **Kontrol Birimi:** İşlemcinin ve sistemin tüm elemanlarının senkronizasyon içinde çalışmasını sağlayan devrelerdir.
 - Zamanlama kontrolü
 - Komut kod çözücüsü
 - Kesme ve mantık birimi

Bilgisayar Mimarisi

➤ CPU Mimarisi

- **Yazmaçlar:** CPU'nun içindeki bilgilerin geçici olarak depolanması için kullanılan ve yazmaç (register) adı verilen veri depolama hücrelerinden oluşan devrelerdir.
 - Geçici verileri depolamak için genel amaçlı yazmaçlar,
 - Program işleyişini yönetmek için ise özel amaçlı yazmaçlar bulunmaktadır.
 - Modern mimarilerde CPU, her bir yazmaçta 64 bite kadar veri saklayabilmektedir.

3-bitlik bir
register
iç yapısı



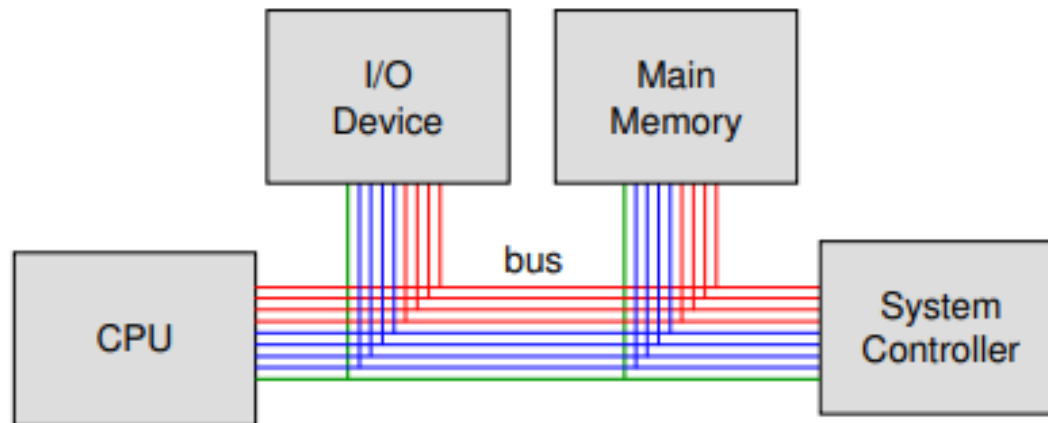
The truth table of the flip-flop is

S	R	Q
0	0	Previous value
1	0	0
0	1	1
1	1	Don't care

Bilgisayar Mimarisi

➤ CPU Mimarisi

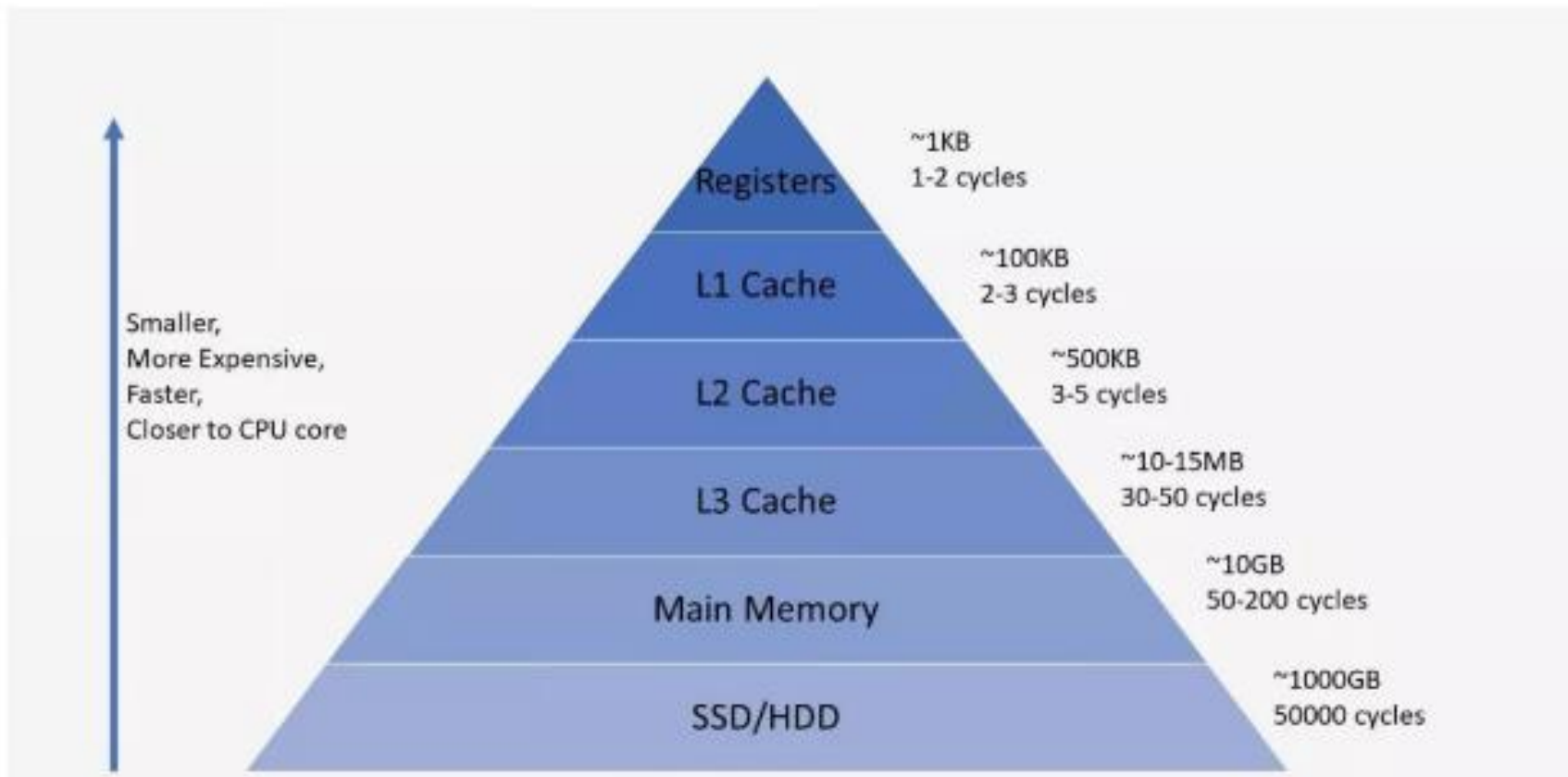
- **Veri Yolu:** Bilgisayarın CPU'sunu, belleğini ve G/Ç aygıtlarını bağlamak için kullanılan kablolar topluluğudur.
 - Bilgiler (0'lar ve 1'ler) CPU, bellek ve cihazlar arasında veri yolu aracılığıyla bazı el sıkışma kurallarına göre iletilir.
 - Adres Yolu (Address Bus)
 - Veri Yolu (Data Bus)
 - Kontrol Yolu (Control Bus)



Bilgisayar Mimarisi

➤ CPU Mimarisi

▪ Ana Bellek ve Ön Bellek



Bilgisayar Mimarisi

➤ Saklı-Program (Stored-Code) Kavramı

- Program, bir algoritmanın sıralı komutlar şeklinde uygulanmasıdır.
- Program, veri gibi bit desenleri olarak kodlanabilir ve ana bellekte saklanabilir.
- Bir bilgisayarın çalıştıracığı programın kendi ana belleğinde kaydedilmesi fikrine saklı-program kavramı adı verilir.
- CPU istediğinde depolandığı yerden programı çıkarabilir ve çalıştırabilir.
- Çalıştırılacak program kolaylıkla değiştirilebilir.

Bilgisayar Mimarisi

➤ Saklı-Program (Stored-Code) Kavramı

- Bilgisayarda bulunan komutların makine dilinin çözümlenebilmesi (decode) ve çalıştırılabilmesi (execute) için kısa ve kesin (net) olması gerekir.
- 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 kullanarak ve sonucu başka bir yazmaca yazacak şekilde toplama devresini çalıştır.

Adım 4. Sonucu belleğe kaydet.

Adım 5. Dur.

Makine Dili

- CPU'lar, saklı-program kavramını uygulayabilmek için bit desenleri şeklinde kodlanmış komutları tanımlayabilecek şekilde tasarlanırlar.
- Kodlama sistemiyle birlikte bu komutlar bütününe **makine dili** adı verilir.
- Bu dilde kullanılan herhangi bir komuta da **makine komutu** denir.
- Her CPU'nun kendine özel bir makine dili vardır ve bu CPU için Instruction Set Architecture (ISA) olarak adlandırılır.

Makine Dili

➤ RISC ve CISC Mimarileri

Reduced Instruction Set Computing / İndirgenmiş Komut Takımlı Bilgisayar

- Az sayıda, basit, verimli ve hızlı komutlar
 - Apple
 - IBM
 - Motorola'dan PowerPC
 - ARM
- Az sayıda komut içerdiği için aynı işlem için birden çok komut yürütmesi yönüyle dezavantajlı.

Complex Instruction Set Computing / Karmaşık Komut Takımlı Bilgisayar

- Çok sayıda, kullanışlı ve güçlü komutlar, yüksek enerji tüketimi
 - Intel
 - AMD
- Güçlü ve zengin komut kümesi yönüyle avantajlı
- Enerji tüketimi yönüyle dezavantajlı

➤ Komut Listesi

- Bir işlemcinin komutları, RISC veya CISC olmasına bakılmaksızın, üç ayrı gruba ayrılabilir.

1. **Veri aktarma komutları:** Bu grupta verinin bir yerden başka bir yere taşınmasını sağlayan komutlar yer alır.
2. **Aritmetik/mantık komutları:** Bu grup, işlemci içerisindeki kontrol birimine, aritmetik mantık biriminde bir işlem gerçekleştirmesini söyleyen komutlardan oluşur.
3. **Kontrol komutları:** Bu gruptaki komutlar veriyi işlemek yerine programın çalışmasını yönetirler.

➤ Komut Listesi

- Örnek: Bellekteki 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

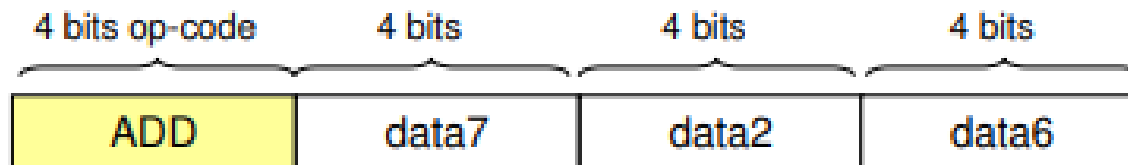
Adım 5. Üçüncü yazmacın içeriğini belleğe KAYDET

Adım 6. Dur.

Makine Dili

➤ Makine Komutunun Parçaları

- Bir makine komutunun kodlanmış hali iki bölümden oluşur:
 - Komut = İşlem Kodu (Opcode) + İşlenen (Operand)
 - **İşlem Kodu (Opcode)**: Hangi temel işlemin (STORE, SHIFT, XOR, JUMP, ADD) gerçekleştirileceğini belirler.
 - **İşlenen (Operand)**: İşlem koduyla belirlenen işlem hakkında daha detaylı bilgi içerir. İşlenenin yorumlanması, işlem koduna bağlı olarak değişir.
- Örnek: data2 ile data6'yı toplayıp sonucu data7 ile saklayan 16-bitlik bir CPU komutu gösterimi



➤ Komut Uzunluğu

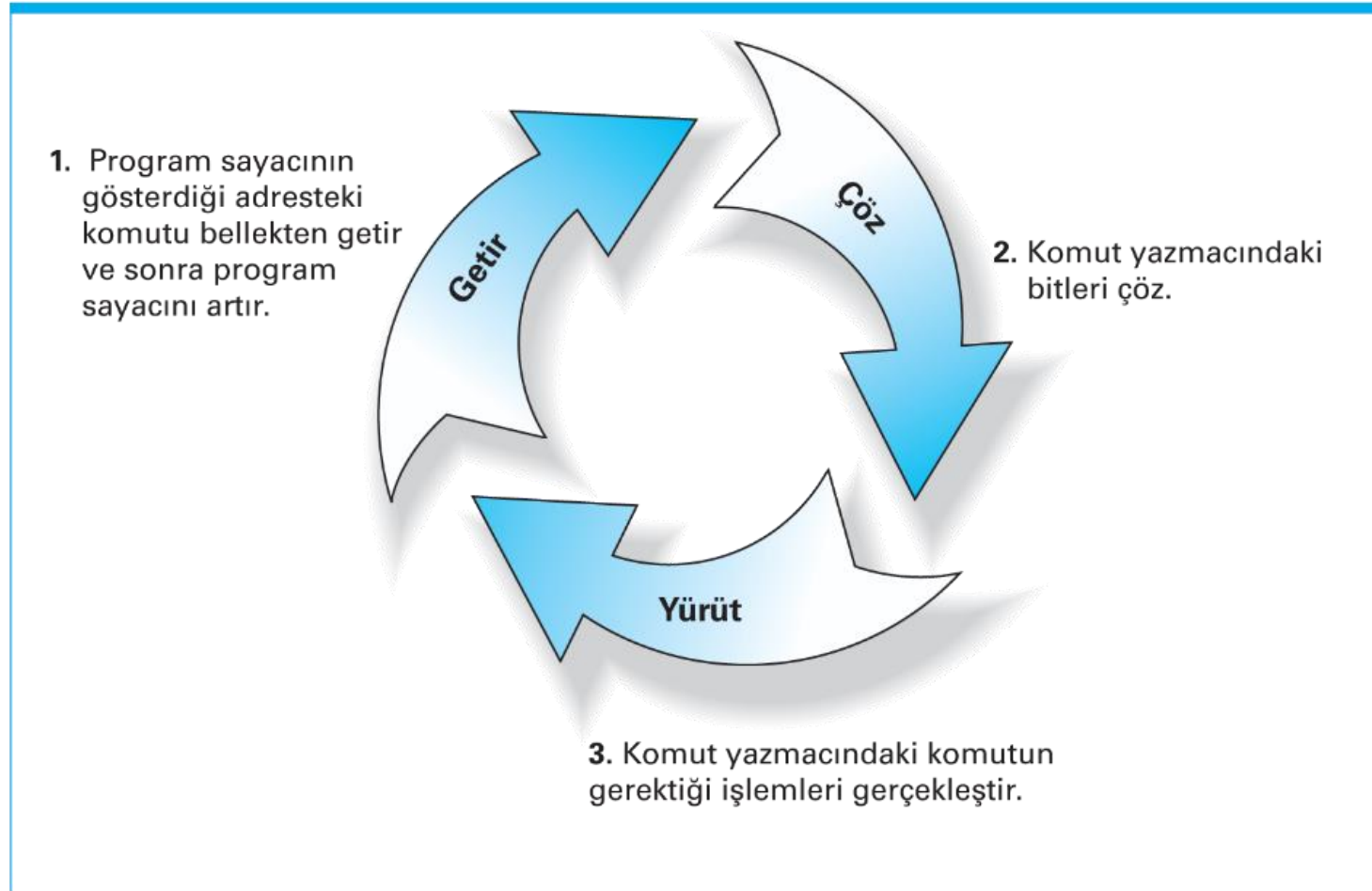
- Bir makine dili, tüm komutların sabit uzunluklu kodlamasını veya değişken uzunluklu kodlamasını kullanabilir.
 - Sabit uzunluklu kodlama: CPU'nun kodunu çözmesi daha kolaydır, genellikle dilde daha az talimata izin verir.
 - Değişken uzunluklu kodlama: CPU'nun kodunu çözmesi daha zordur, ancak daha zengin bir talimat dizisine olanak tanır.

Program Yürütme

- Bir programın bilgisayar tarafından nasıl yürütüldüğünü anlamak için öncelikle iki özel amaçlı yazmacı incelememiz gerekir.
 - **Komut yazmacı (instruction register):** işlemci tarafından işlenmekte olan komutu tutar.
 - **Program sayacı (program counter):** Bir sonra işlenecek olan komutun bellekteki adresini tutar.

Program Yürütme

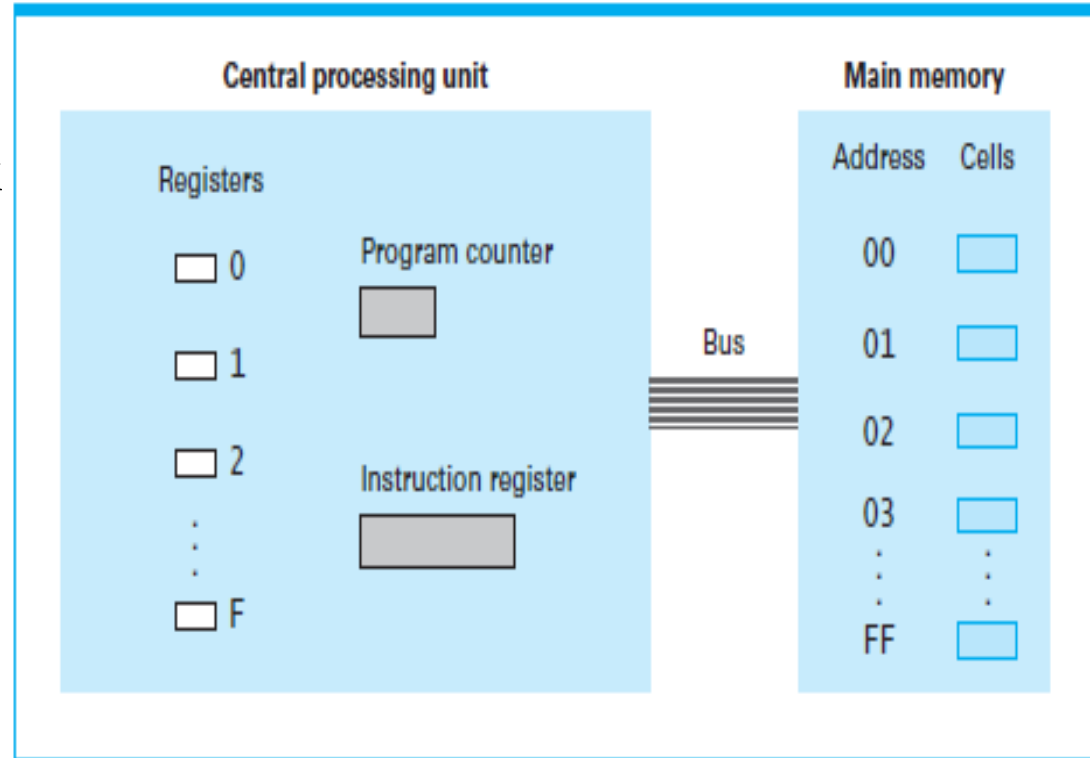
- CPU görevini, bilgisayar çevrimi adı verilen 3 adımlık bir işlemi sürekli tekrar ederek gerçekleştirir. (Fetch, decode, execute)



Sanal Bir Makine Dili

- Elimizde örnek amaçlı tanımlanmış bir CPU mimarisi olduğunu düşünelim.
- Bu CPU'nun komutları nasıl çözümlediğini analiz edelim.

- 16 genel amaçlı yazmaç
- 256 tane 8 bitlik ana bellek hücresi
- Bir PC ve bir IR
- Her bir komut 16 bit ile kodlanmış ve onaltılık sistemle temsil edilmekte



Sanal Bir Makine Dili

➤ Tanımlanan bu sanal makinenin dili Ek-C'deki gibi olsun:

Op-Code	Operand	Açıklama
1	RXY	R registerına adresi XY olan hafıza hücresinde bulunan bit desenini yaz (LOAD). örnek: 14A3, A3 adresindeki hafıza hücresinin içeriğinin 4 numaralı registera yerleştirilmesini sağlayacaktır.
2	RXY	R registerına XY bit desenini yaz (LOAD). örnek: 20A3, register 0'a A3 değerini yerleştirmeye sebep olur.
3	RXY	R registerı içinde bulunan değeri, adresi XY olan hafıza hücresine kaydet (STORE). Örnek: 35B1, 5 numaralı register içeriğinin belleğin B1 adresine yerleştirir.
4	ORS	R registerındaki bit desenini S registerına taşı (MOVE). Örnek: 40A4, A registerının içeriğinin register 4'e kopyalanmasını sağlar.
5	RST	Register S ve T'deki bit örüntülerini ikinin tümleyenini gösterimindeymiş gibi TOPLA (ADD) ve sonucu R registerına yaz. Örnek: 5726, 2 ve 6 numaralı registerlardaki ikili sayıları toplar ve sonucun 7 numaralı registera yerleştirilmesini sağlar.

Sanal Bir Makine Dili

➤ Ek-C devam:

6	RST	S ve T registerlarındaki bit örüntülerini sanki değerleri kayan-nokta gösteriminde temsil ediyorlar gibi TOPLA (ADD) ve kayan-nokta sonucu R registerına yaz. Örnek: 634E, 4 numaralı register ve E registerındaki değerlerin kayan-nokta değerleri olarak toplanmasına ve sonucun 3 numaralı registra yerleştirilmesine neden olur.
7	RST	S ve T registerlarındaki bit desenlerine OR işlemini uygula ve sonucu R registerında tut. Örnek: 7CB4, register B ve 4'teki içeriğin OR işlemine alınmasını ve sonucun C registerında tutulmasını sağlar.
8	RST	S ve T registerlarındaki bit desenlerine AND işlemini uygula ve sonucu R registerına yerleştir. Örnek: 8045, 4 ve 5 numaralı registerlardaki içeriklerin AND işlemine sokulmasıyla elde edilen sonucun 0 numaralı registra yerleştirilmesini sağlar.
9	RST	S ve T registerlarındaki bit desenlerini XOR işlemine alarak sonucu R registerına yerleştirir. Örnek: 95F3 F ve 3 numaralı registerlardaki içeriğin XOR işlemine alınarak sonuç 5 numaralı registra yerleştirilir.

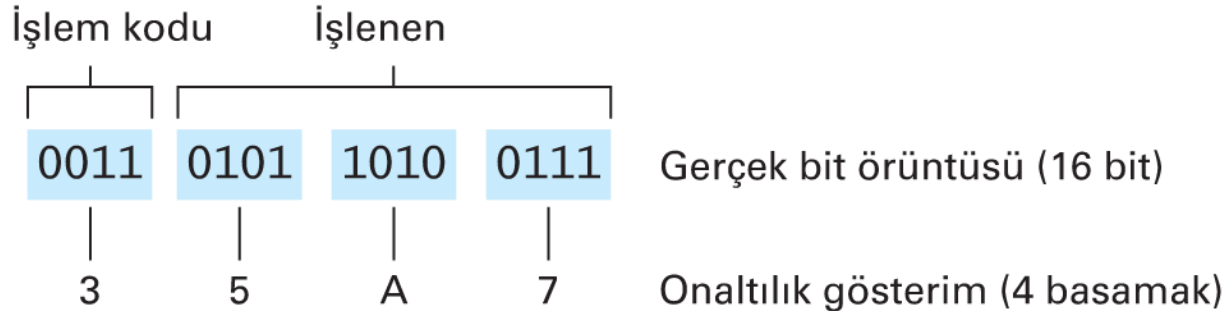
Sanal Bir Makine Dili

➤ Ek-C devam:

A	R0X	R registerındaki bit desenini X kere 1 bit sağa kaydır. Her seferinde, düşük değer ucunda başlayan biti, yüksek seviye uca yerleştir. Örnek: A403 4 numaralı registerın içeriğinin döngüsel biçimde 3 bit sağa kaydırılmasını sağlar.(ROTATE)
B	RXY	Eğer R registerındaki bit deseni 0 numaralı registerdaki bit desenine eşit ise, XY adresindeki hafıza hücresinde bulunan komuta atla (JUMP). Aksi durumda, yürütmeye normal sırada devam et. (atlama, yürütme sırasında program sayacına XY'yi kopyalayarak gerçekleşir) Örnek: B43C, önce 4 numaralı register ile 0 numaralı registerdaki içerikleri karşılaştırır. Eşit ise, 3C deseni program sayacına yüklenir ve böylece çalıştırılan sonraki komut hafıza adresinde tutulan olacaktır. Aksi durumda, hiçbir şey yapılmaz ve programın yürütülmesi normal seyrinde devam eder.
C	000	Çalışmayı durdur (HALT). Örnek: C000 programın yürütülmesinin durdurulmasına neden olur.

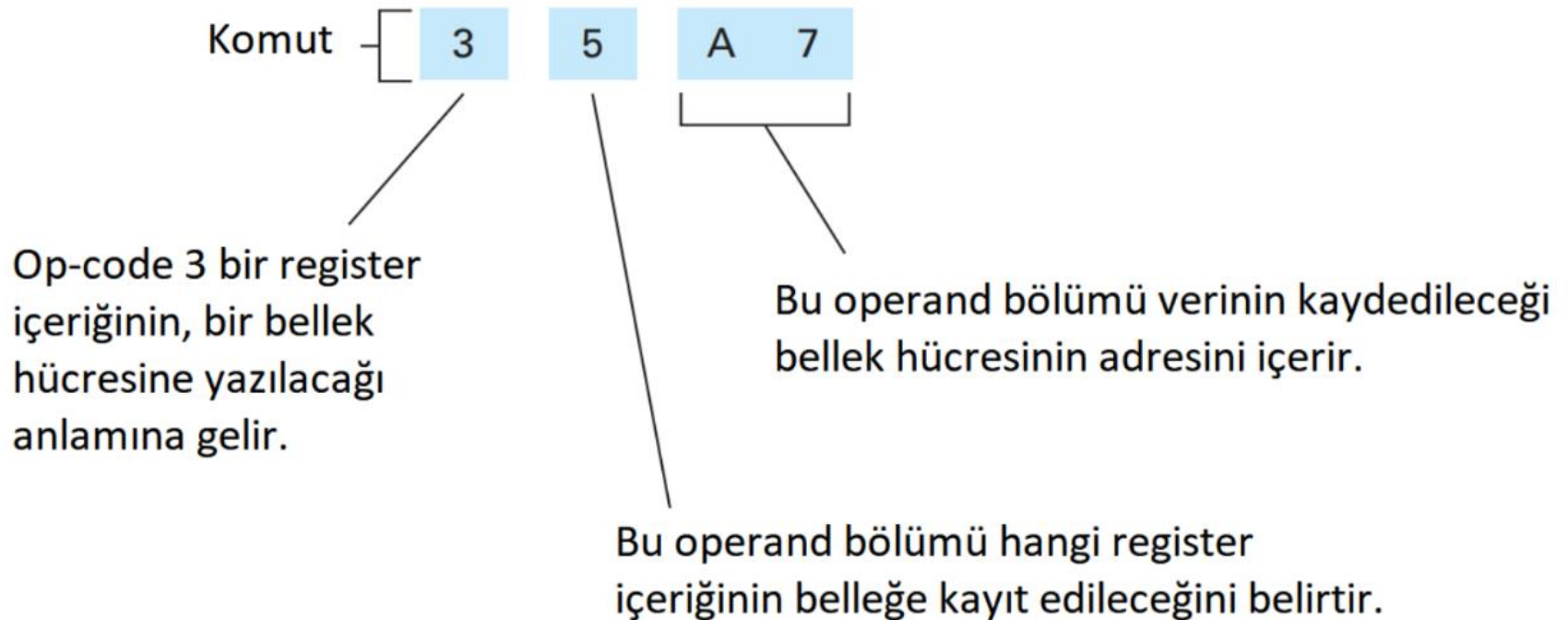
Sanal Bir Makine Dili

- CPU'ya bir komut geldiğinde öncelikle onun çözümülenmesi (decode) gerekli.
- Sanal makine diline göre komutların bölümleri bulunur.
- Örnek bir komutu inceleyelim.



Sanal Bir Makine Dili

➤ «35A7» komutunun çözülmesi



Sanal Bir Makine Dili

➤ Sanal Makine Dili ile Program Yürütme

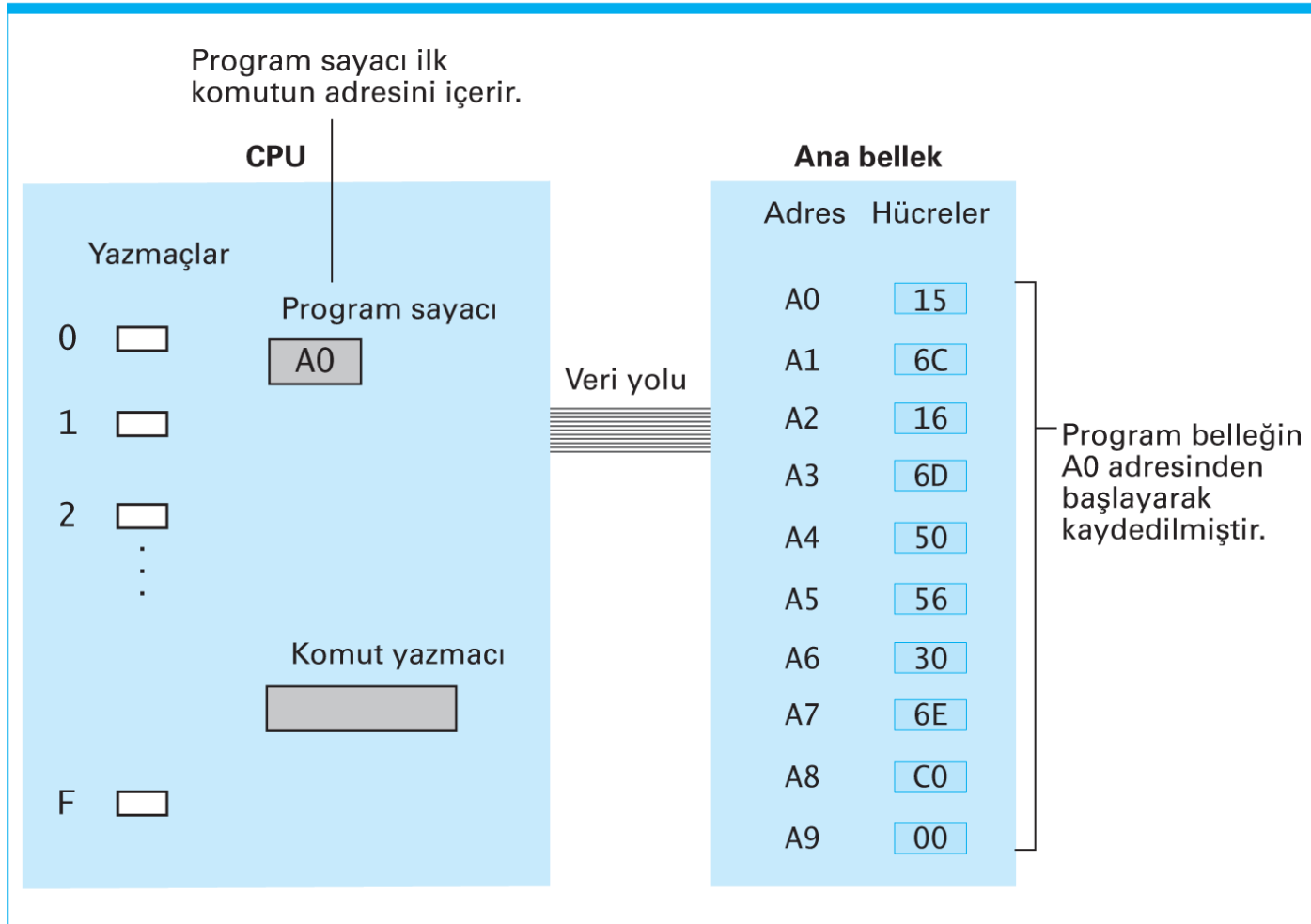
- **Örnek:** Bellekte yer alan iki sayının toplanması için komutların kodlanması

Kodlanmış Komut	Anlamı (Translation)
156C	6C adresli bellek hücresinin içeriğini register 5'e yükle
166D	6D adresli bellek hücresinin içeriğini register 6'ya yükle
5056	Register 5 ve 6'nın içeriklerini (2'ye tümleyen gösterimine göre) topla ve sonucu register 0'da tut
306E	Register 0'ın içeriğini 6E adresli bellek hücreğine kaydet
C000	Dur. (HALT)

Sanal Bir Makine Dili

➤ Sanal Makine Dili ile Program Yürütme

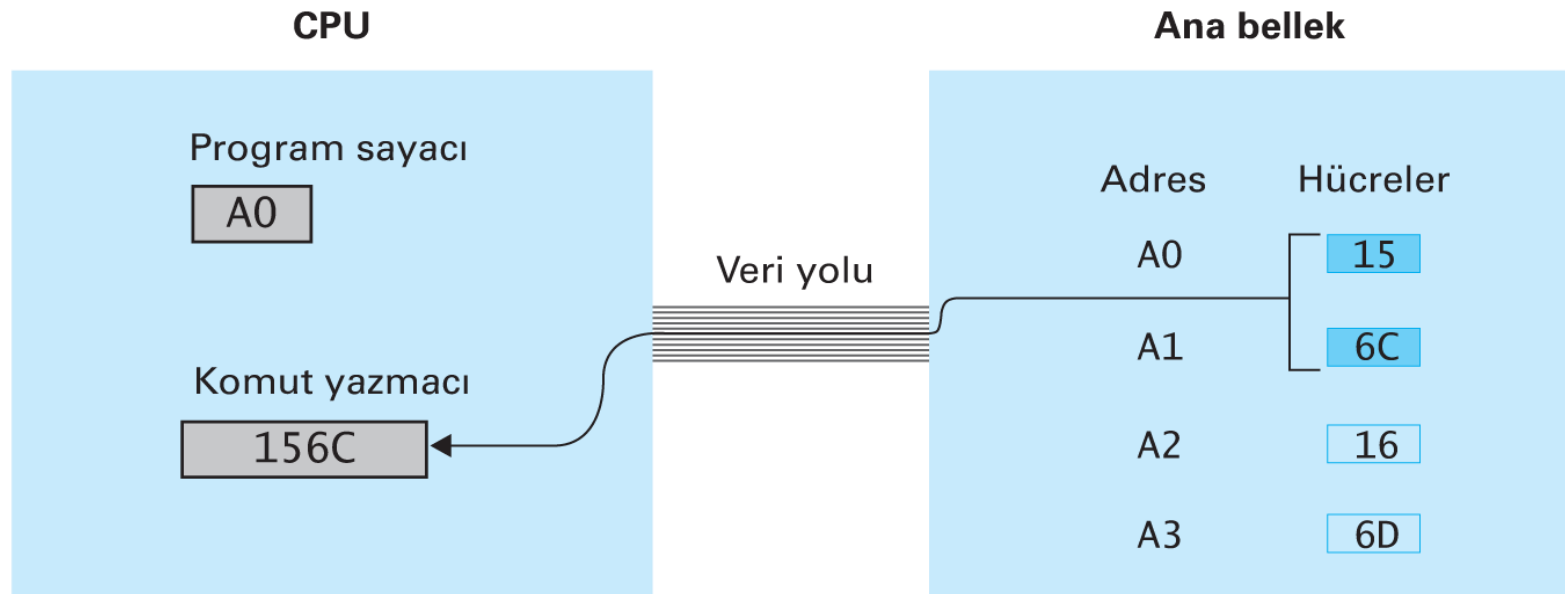
- İlk aşamada, program ana bellekte ve başlangıç adresi PC'de:



Sanal Bir Makine Dili

➤ Sanal Makine Dili ile Program Yürütme – Getir (Fetch)

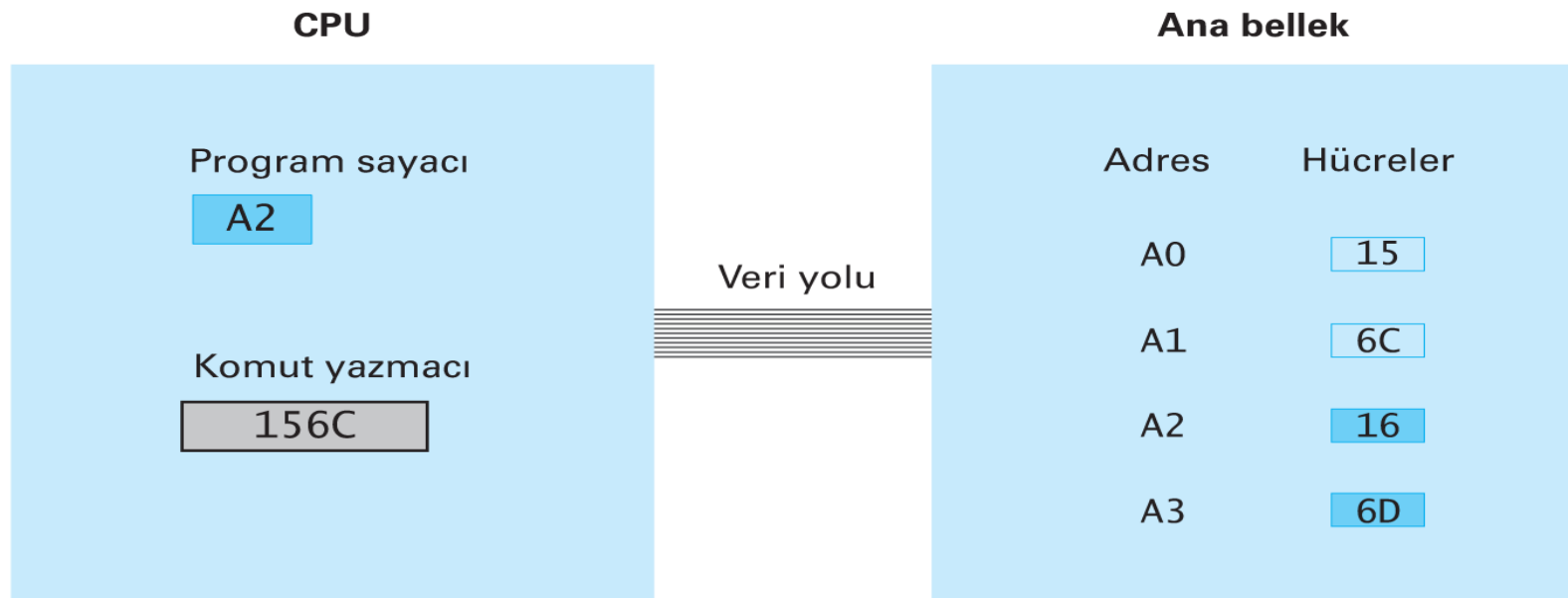
- Getir adımının başında A0 adresinden başlayan komut bellekten okunur ve komut yazmacına yerleştirilir.



Sanal Bir Makine Dili

➤ Sanal Makine Dili ile Program Yürütme – Getir (Fetch)

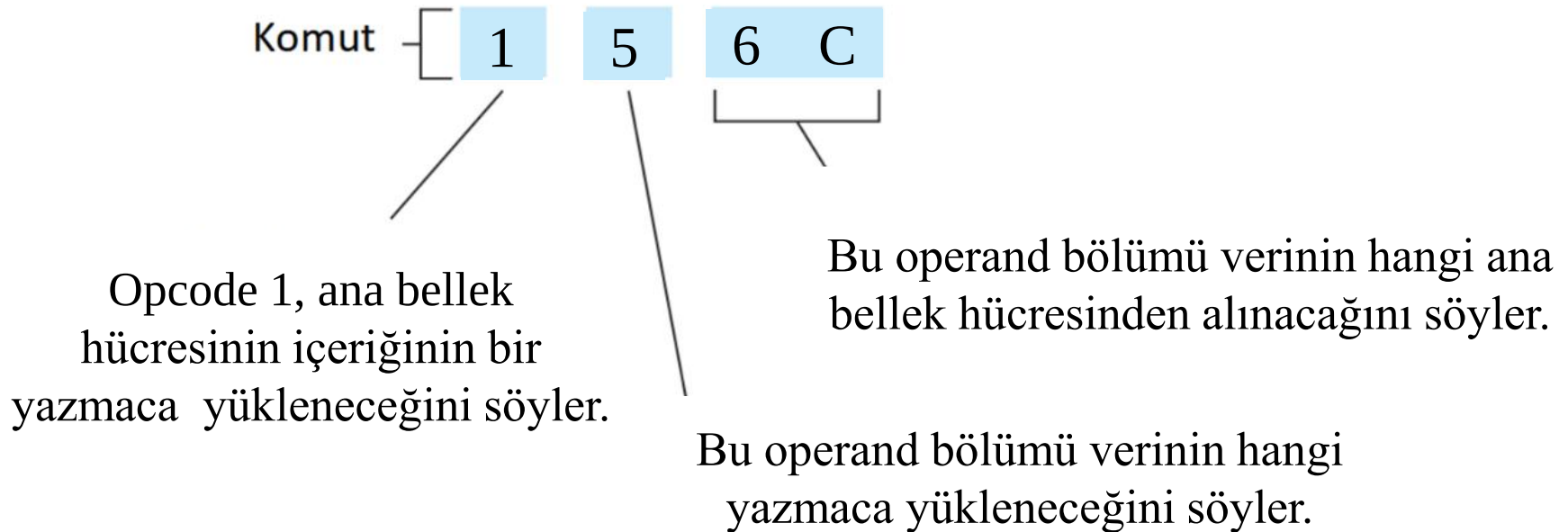
- Daha sonra ise program sayacı arttırılır ve bir sonraki komutun adresini gösterir.



Sanal Bir Makine Dili

➤ Sanal Makine Dili ile Program Yürütme – Çöz (Decode) ve Yürüt (Execute)

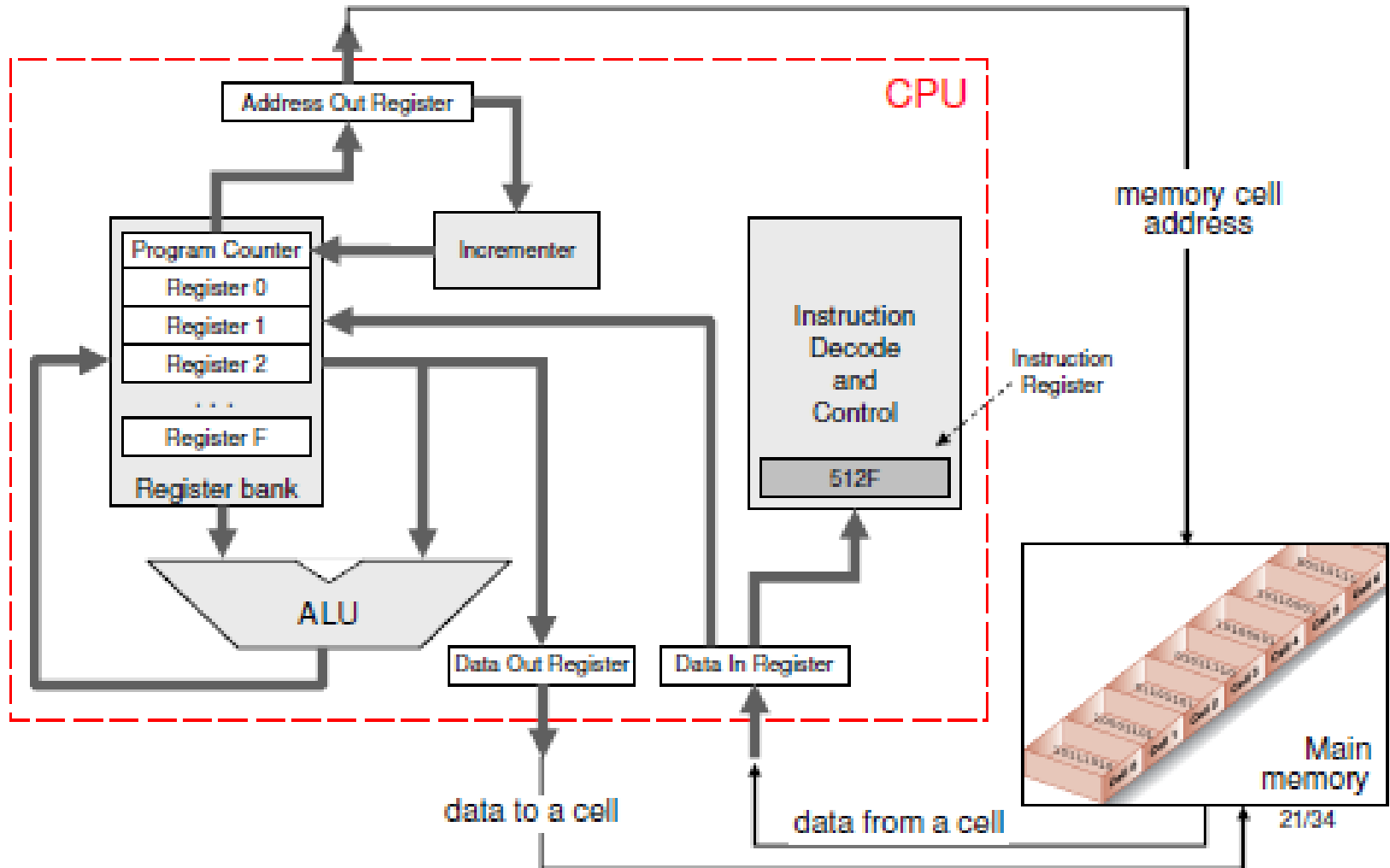
- Çöz aşamasında CPU komut yazmacındaki komutu inceler.



- Yürüt aşamasında ise ana belleğin 6C adresli hücresinin içeriği yazmaç 5'e yüklenir.

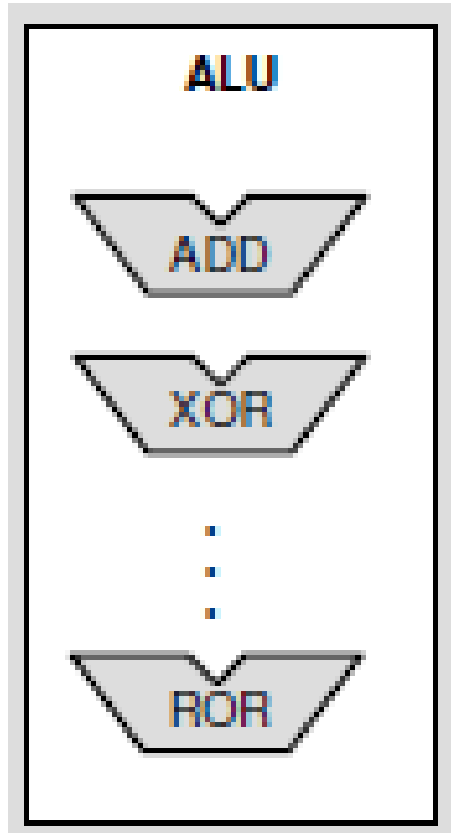
Aritmetik/Mantık İşlemleri

➤ Gerçekçi Bir CPU'nun İç Yapısı



Aritmetik/Mantık İşlemleri

➤ Aritmetik/Mantık Birimi



- ALU, veri işlemlerini gerçekleştiren kastır.
- ALU tarafından üç tür işlem desteklenir:
 - **Mantık:** VE, VEYA, XOR, DEĞİL
 - **Döndürme ve Kaydırma:** döndürme (dairesel kaydırma), mantıksal kaydırma, aritmetik kaydırma
 - **Aritmetik:** toplama, çıkarma, çarpma, bölme

Aritmetik/Mantık İşlemleri

➤ Mantık (Logic) İşlemleri

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

$$\begin{array}{r} 10011010 \\ \text{AND } 11001001 \\ \hline 10001000 \end{array}$$

$$\begin{array}{r} 10011010 \\ \text{OR } 11001001 \\ \hline 11011011 \end{array}$$

$$\begin{array}{r} 10011010 \\ \text{XOR } 11001001 \\ \hline 01010011 \end{array}$$

Aritmetik/Mantık İşlemleri

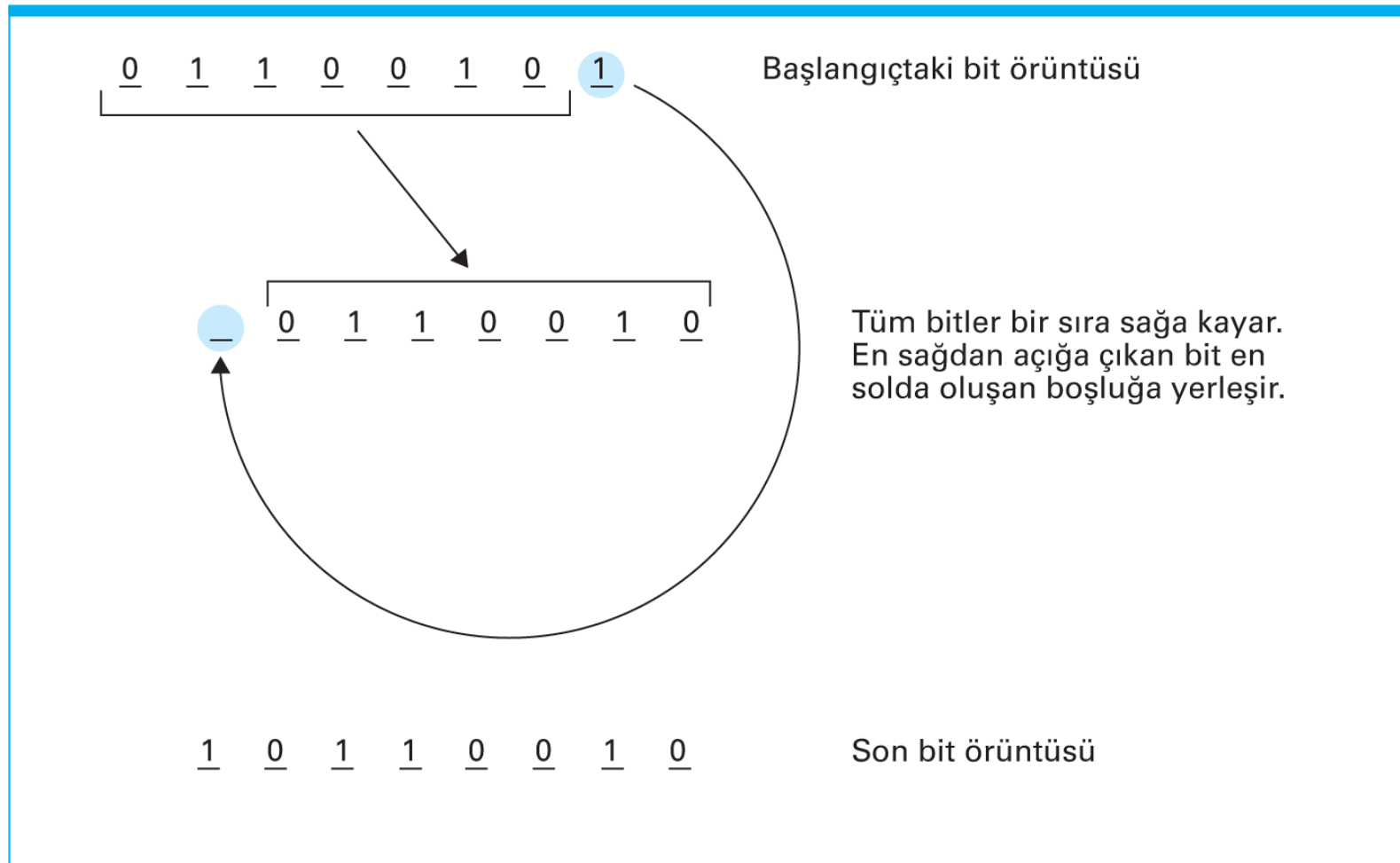
➤ Döndürme ve Kaydırma İşlemleri

- Bu gruptaki komutlar bir yazmaç içerisindeki bitlerin hareket etmesini sağlar.
- Bu komutlar hareketin yönüne (sağa veya sola) ve işlemin dairesel olup olmadığına göre sınıflandırılır.

Aritmetik/Mantık İşlemleri

➤ Döndürme ve Kaydırma İşlemleri

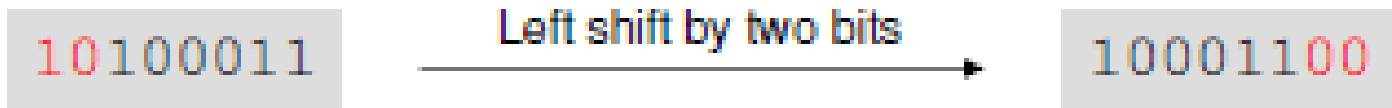
- 65 (onaltılık) sayısının bir bit sağa döndürülmesi (rotation)



Aritmetik/Mantık İşlemleri

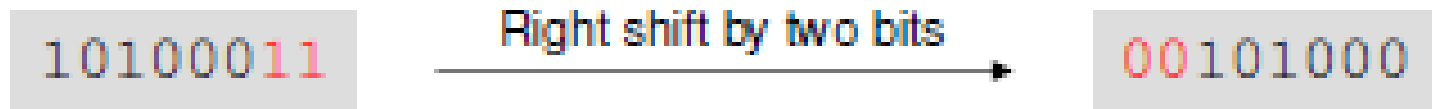
➤ Döndürme ve Kaydırma İşlemleri

- n-bit sola kaydırma: Tek bir yolla yapılır

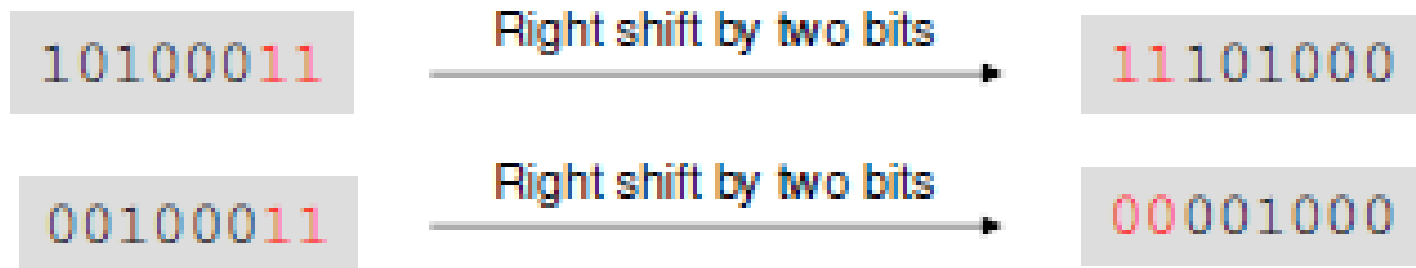


- N-bit sağa kaydırma: İki yolla yapılabilir

- Mantıksal sağa kaydırma



- Aritmetik sağa kaydırma



Diğer Cihazlarla Haberleşme

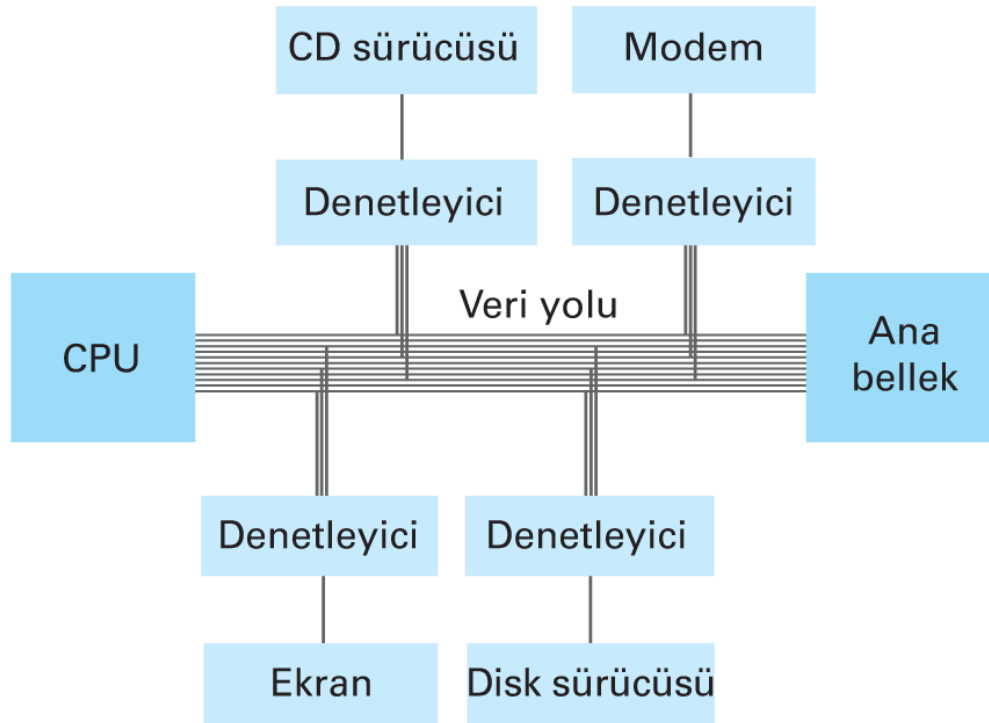
➤ Denetleyicilerin Rolü

- Bir bilgisayar ve diğer cihaz arasındaki iletişim genellikle **denetleyici** adı verilen ara birimler tarafından yönetilir.
- İşlemci ve ana belleğin de bağlı olduğu veri yoluna bağlanırlar.
- Böylece hem işlemci ve bellek arasındaki sinyalleri takip eder hem de kendi sinyallerini aktarabilir.

Diğer Cihazlarla Haberleşme

➤ Denetleyicilerin Rolü

■ Veri Yoluna Bağlanmış Denetleyiciler



Diğer Cihazlarla Haberleşme

➤ Doğrudan Bellek Erişimi

- Bir denetleyici bilgisayarın veri yoluna bağlı olduğu için CPU'nun kullanmadığı zamanlarda veri yolu üzerinden ana bellekle direkt olarak iletişim kurabilir.
- Denetleyicilerin ana belleğe bu şekilde erişmesine **Doğrudan Bellek Erişimi (Direct Memory Access – DMA)** adı verilir.

Diğer Cihazlarla Haberleşme

➤ El Sıkışma

- Bilgisayar ve çevresel aygıt arasındaki veri aktarımını koordine etme sürecine **el sıkışma (handshaking)** adı verilir.
- Çift veya tek yönlü bir haberleşme yöntemidir.
- Örneğin yazıcıya bir doküman gönderme işlemi çift yönlü el sıkışmadır.

Diğer Cihazlarla Haberleşme

➤ Popüler Haberleşme Ortamları

- İki cihaz arasındaki haberleşme iki farklı türde olabilir;
 - **Paralel İletişim:** Her biri ayrı bir "hatta" (bilgisayarın dahili veri yolu) aynı anda aktarılan birkaç sinyal
 - **Seri İletişim:** Sinyaller tek bir hat üzerinden (USB, FireWire) arka arkaya aktarılır.

Diğer Mimariler

➤ Küme Komut İşleme

- Yürütme hızını artırmadan bilgisayarın üretilen işini artırmanın bir yolu küme komut işleme (iş hattı veya boru hattı olarak da kullanılmaktadır, pipelining) kavramıdır.
- Bu teknikte bilgisayar çevrimindeki adımların örtüşmesine izin verilir.
- Bir komut yürütülürken, bir sonraki komutun GETİR adımı başlatılabilir.

Diğer Mimariler

➤ Çok İşlemcili Makineler

- Gerçek paralel işleme için birden fazla işlemci birimi gereklidir.
- Bu tür sistemlere çok işlemcili veya çok çekirdekli makine adı verilir.
 - SISD: Single Instruction, Single Data
 - Paralel işlem yoktur
 - MIMD: Multiple Instruction, Multiple Data
 - Farklı programlar, farklı veriler
 - SIMD: Single Instruction, Multiple Data
 - Aynı program, farklı veriler