William Stallings Computer Organization and Architecture 10th Edition

Orijinal slaytların çevirisidir.

Bu bölüm, bilgisayar bileşenleri ara bağlantısı için kullanılan temel yapılara odaklanmaktadır. Arka plan olarak, bu bölüm temel bileşenlerin ve bunların arayüz gereksinimlerinin kısa bir incelemesi ile başlamaktadır. Ardından işlevsel bir genel bakış sağlanır. Ardından, sistem bileşenlerini birbirine bağlamak için veri yollarının kullanımını incelemeye hazırız.

Chapter 3

A Top-Level View of Computer Function and Interconnection

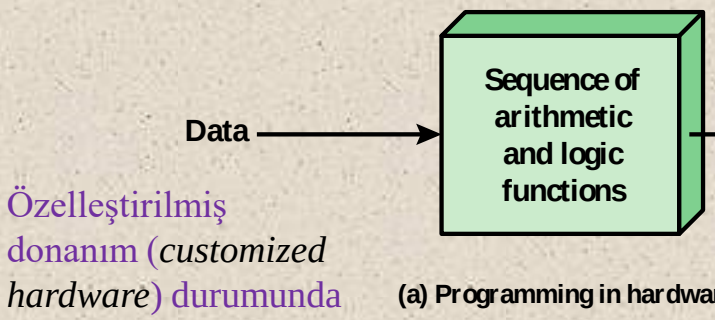
Computer Components

- Modern bilgisayar tasarımları, Princeton İleri Araştırmalar Enstitüsü'nde (IAS) John von Neumann tarafından geliştirilen kavramlara dayanmaktadır.
- *von Neumann mimarisi* olarak anılır ve üç temel kavrama dayanır:
  - Veriler ve komutlar tek bir okuma-yazma belleğinde saklanır
  - Bu belleğin içeriği, burada bulunan veri türüne bakılmaksızın konuma (*by location*) göre adreslenebilir.
  - İcra/yürütme, (açıkça değiştirilmedikçe) bir komuttan diğerine sıralı bir gerçekleşir
- *Hardwired program*
  - Çeşitli bileşenleri istenen konfigürasyonda bağlama işleminin sonucu

İkili verileri depolamak ve bu veriler üzerinde aritmetik ve mantıksal işlemler gerçekleştirmek için çeşitli şekillerde birleştirilebilen küçük bir temel mantık bileşenleri kümesi vardır. Gerçekleştirilecek belirli bir hesaplama varsa, bu hesaplama için özel olarak tasarlanmış bir mantık bileşenleri konfigürasyonu oluşturulabilir. Çeşitli bileşenlerin istenilen konfigürasyonda birleştirilmesi sürecini bir programlama biçimi olarak düşünebiliriz. Ortaya çıkan "program" donanım biçimindedir ve ***hardwired program*** olarak adlandırılır.

Hardware and Software Approaches

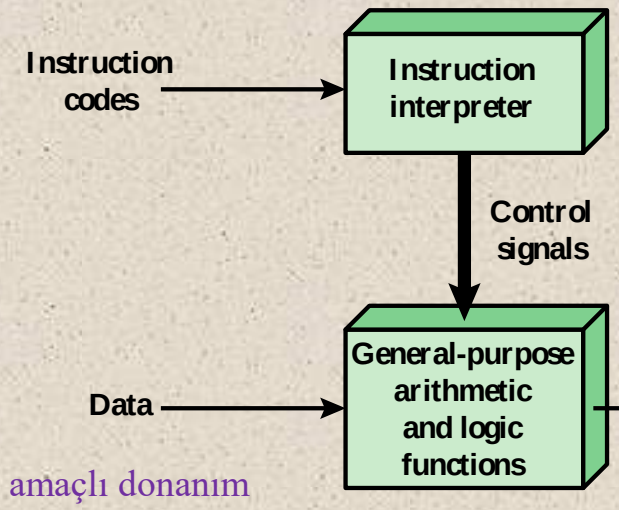
Tüm program aslında bir dizi adımdır. Her adımda, bazı veriler üzerinde bazı aritmetik veya mantıksal işlemler gerçekleştirilir. Her adım için yeni bir kontrol sinyali seti gereklidir. Olası her kontrol sinyali seti için benzersiz bir kod sağlanır ve genel amaçlı donanıma bir kodu kabul edebilen ve kontrol sinyalleri oluşturabilen bir segment eklenir (Şekil 3.1b).



Özelleştirilmiş donanım (*customized hardware*) durumunda sistem, verileri kabul eder ve sonuçlar üretir.

(a) Programming in hardware

Bu donanım seti, donanıma uygulanan kontrol sinyallerine bağlı olarak veriler üzerinde çeşitli fonksiyonları yerine getirecektir.



Genel amaçlı donanım (*general-purpose hardware*) ile sistem, verileri ve kontrol sinyallerini kabul eder ve sonuçlar üretir.

(b) Programming in software

Bu nedenle, her yeni program için donanımı yeniden kablolamak yerine, programcının yalnızca yeni bir kontrol sinyali seti sağlaması gerekir.

Figure 3.1 Hardware and Software Approaches

Software

- Bir dizi kod veya komut
- Donanımın bir kısmı her komutu yorumlar ve kontrol sinyalleri üretir
- Donanımı yeniden kablolamak/kurmak yerine her yeni program için yeni bir kod dizisi sağlanır

Major components:

- CPU
  - Komut yorumlayıcısı (*Instruction interpreter*)
  - Genel amaçlı aritmetik ve mantık fonksiyonları modülü
- I/O Components
  - Input module
    - Veri ve komutları kabul etmek ve bunları sistem tarafından kullanılabilen dahili bir sinyal biçimine dönüştürmek için temel bileşenleri içerir
  - Output module
    - Sonuçları raporlama/yansıtma araçları

Software

I/O Components

CPU, bellekle veri alışverişinde bulunur. Bu amaçla, tipik olarak iki dahili (CPU) registeri kullanır:

Memory address register (MAR)

- Bir sonraki okuma veya yazma işlemi için bellek adresini belirtir

Memory buffer register (MBR)

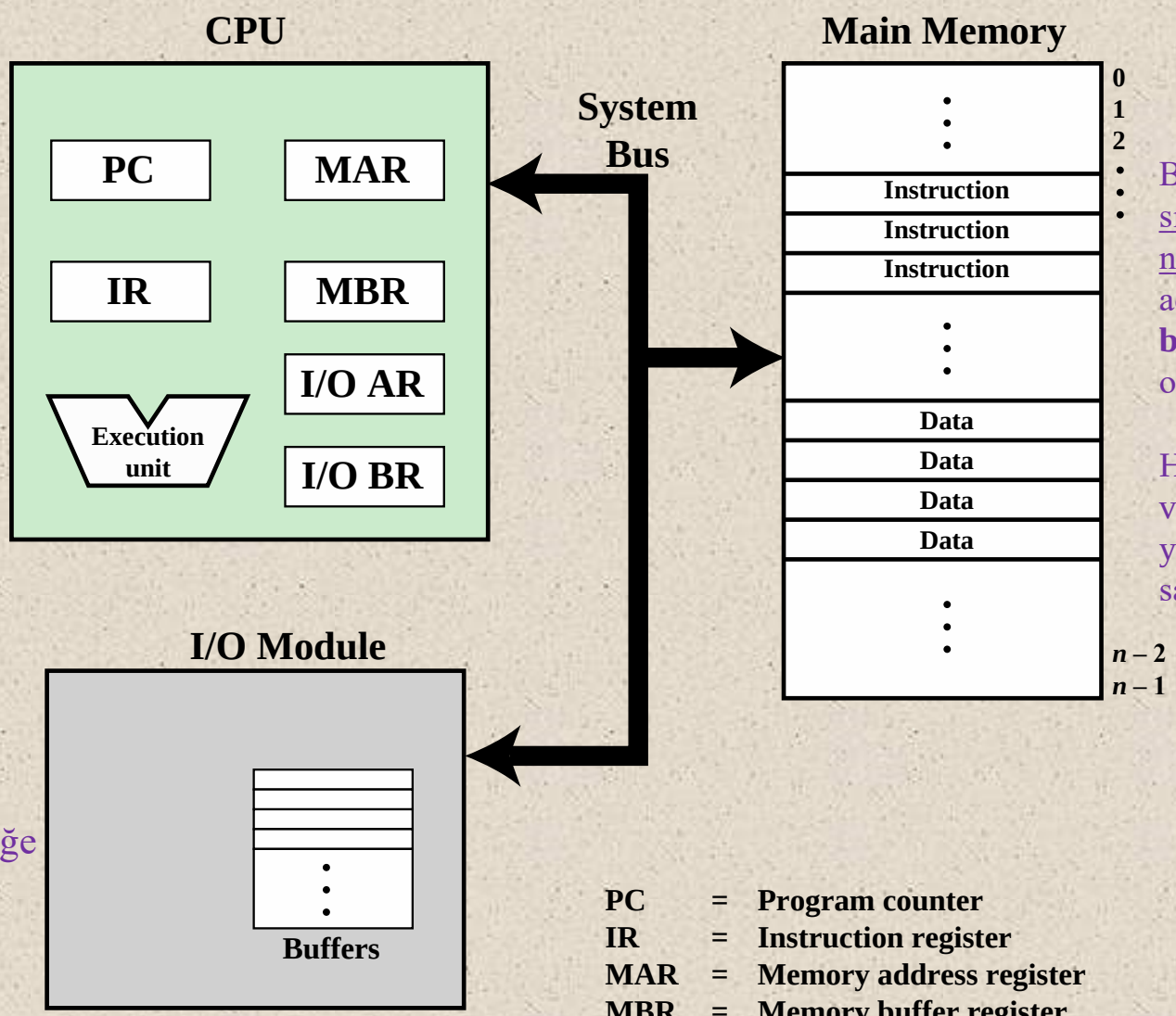
- Belleğe yazılacak veya bellekten okunan verileri tutar

I/O address register (I/OAR)

- Belirli bir I/O cihazını belirtir

I/O buffer register (I/OBR)

- Bir I/O modülü ile CPU arasında veri alışverişi için kullanılır



Bir bellek modülü, sıralı olarak numaralandırılmış adreslerle tanımlanan **bir dizi konumdan** oluşur.

Her konum, bir **komut** veya **veri** olarak yorumlanabilen bir ikili sayı içerir.

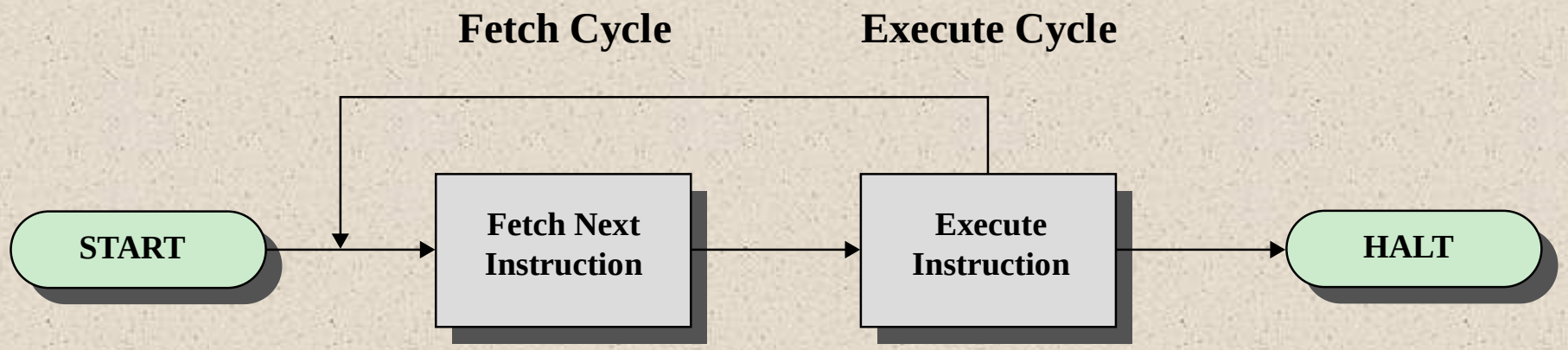
Bir I/O modülü, verileri harici cihazlardan CPU'ya ve belleğe veya tam tersi şekilde aktarır. Gönderilinceye kadar bu verileri geçici olarak tutmak için dahili **tamponlar** içerir.

PC = **Program counter**
IR = **Instruction register**
MAR = **Memory address register**
MBR = **Memory buffer register**
I/O AR = **Input/output address register**
I/O BR = **Input/output buffer register**

Figure 3.2 Computer Components: Top-Level View

program icrasının/yürütmenin temel öğeleri

.



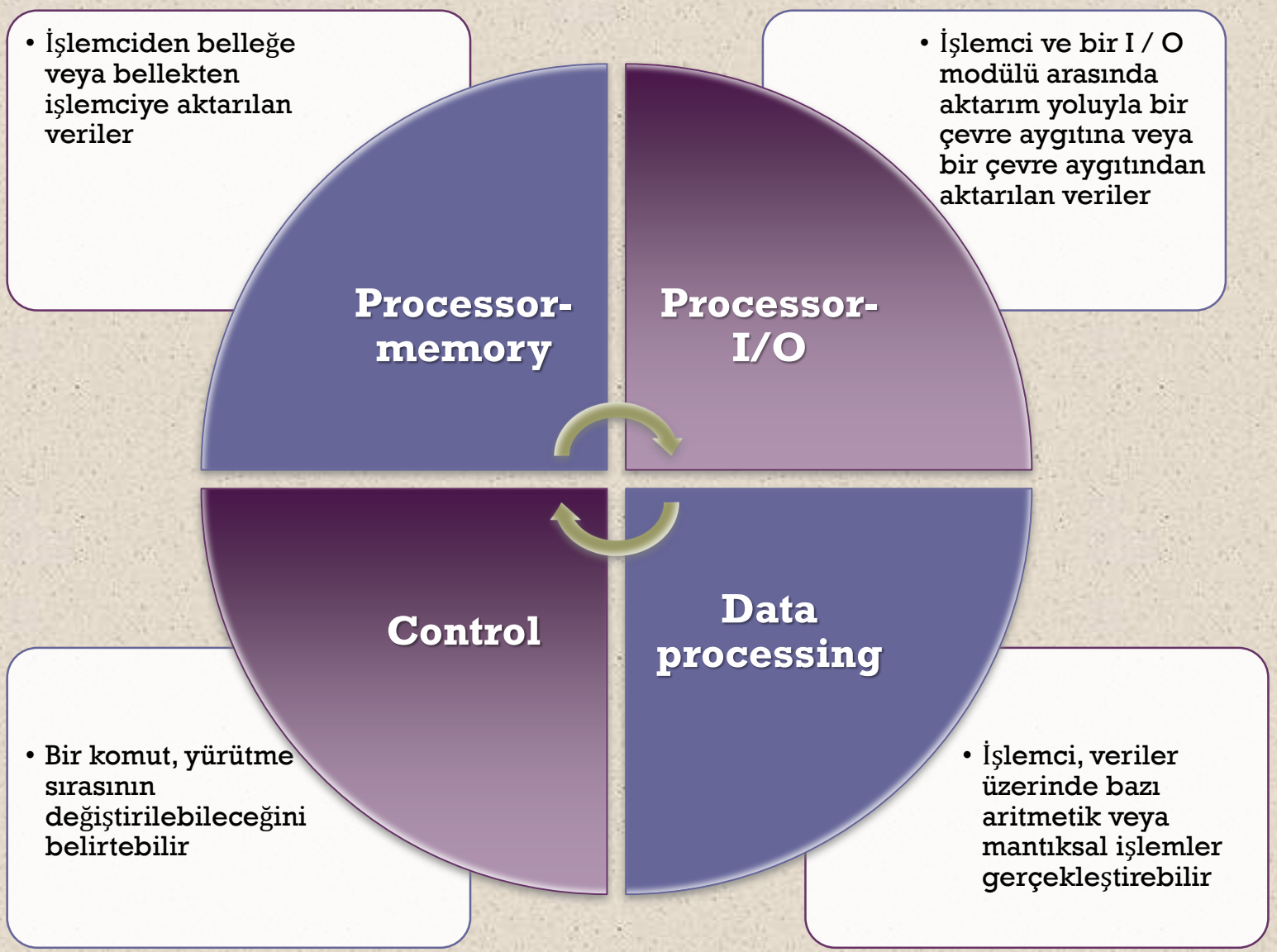
En basit haliyle, komut işleme iki adımdan oluşur: İşlemci, komutları teker teker bellekten okur (getirir) ve her komutu yürütür

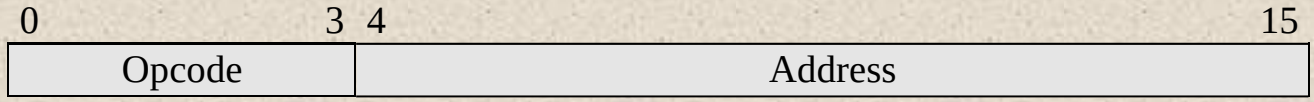
Figure 3.3 Basic Instruction Cycle

Fetch Cycle

- Her komut döngüsünün başında işlemci bellekten bir komut alır
- Program sayıcı (PC- program counter), sıradaki alınacak komutun adresini tutar.
- İşlemci, her bir komut alımından sonra PC'yi bir sonraki komutu alacak şekilde artırır.
- Alınan komut, komut yazmacına (IR) yüklenir
- İşlemci komutu yorumlar ve gerekli eylemi/aksiyonu (***action***) gerçekleştirir

Action Categories





(a) Instruction format

(b) Integer format

Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

İşlemci, akümülatör (AC) adı verilen tek bir veri registeri içerir.

Hem komutlar hem de veriler 16 bit uzunluğundadır. Bu nedenle, 16 bit kelimeler kullanarak belleği düzenlemek uygundur.

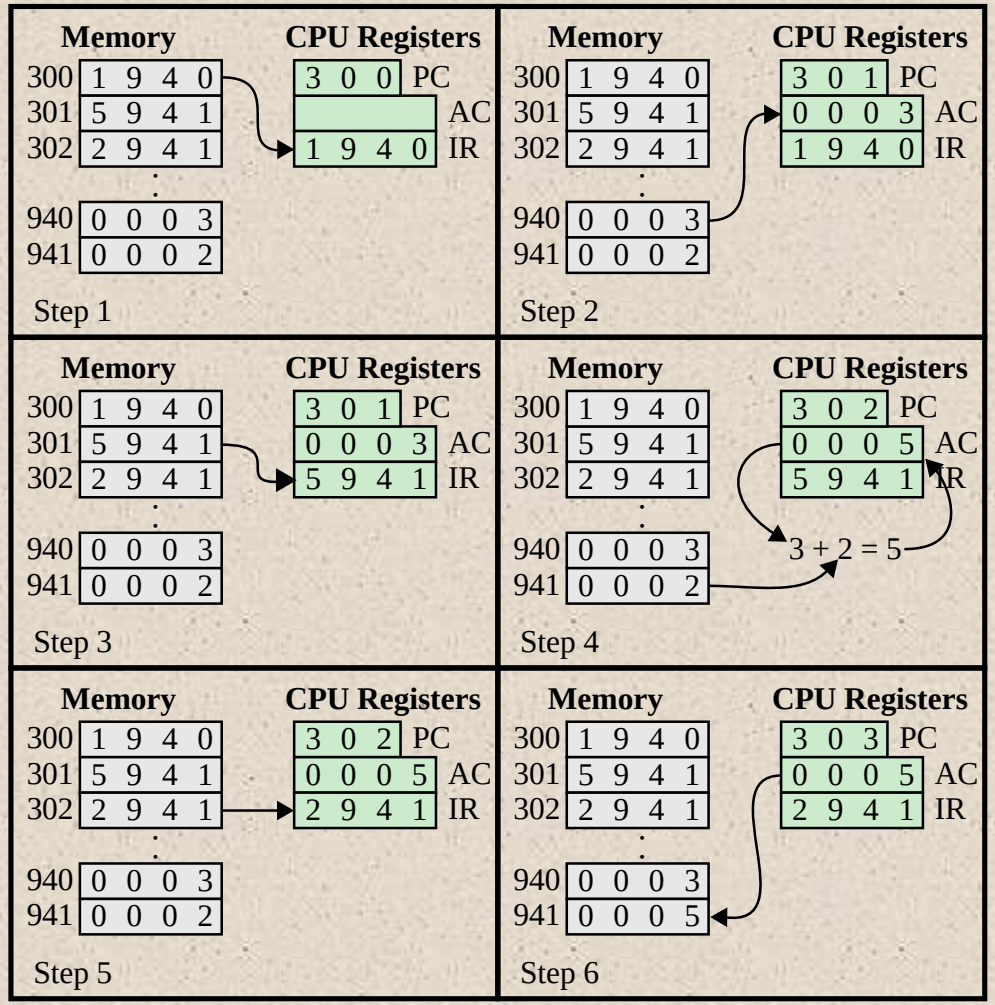
0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

Komut formatında, işlem kodu için 4 bit ayrılmıştır, böylece $2^4 = 16$ farklı işlem kodu olabilir ve $2^{12} = 4096$ (4K) kelimeye kadar bellek doğrudan adreslenebilir.

Figure 3.4 Characteristics of a Hypothetical Machine

Şekil 3.4'te listelenen özellikleri içeren varsayımsal bir makine kullanan basit bir örnek düşünün.



Gösterilen program parçası, 940 adresindeki bellek hücresinin içeriğini, 941 adresindeki bellek hücresinin içeriğine ekler ve sonucu bu hücre konumda depolar.

Örnek Program İcrası

Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

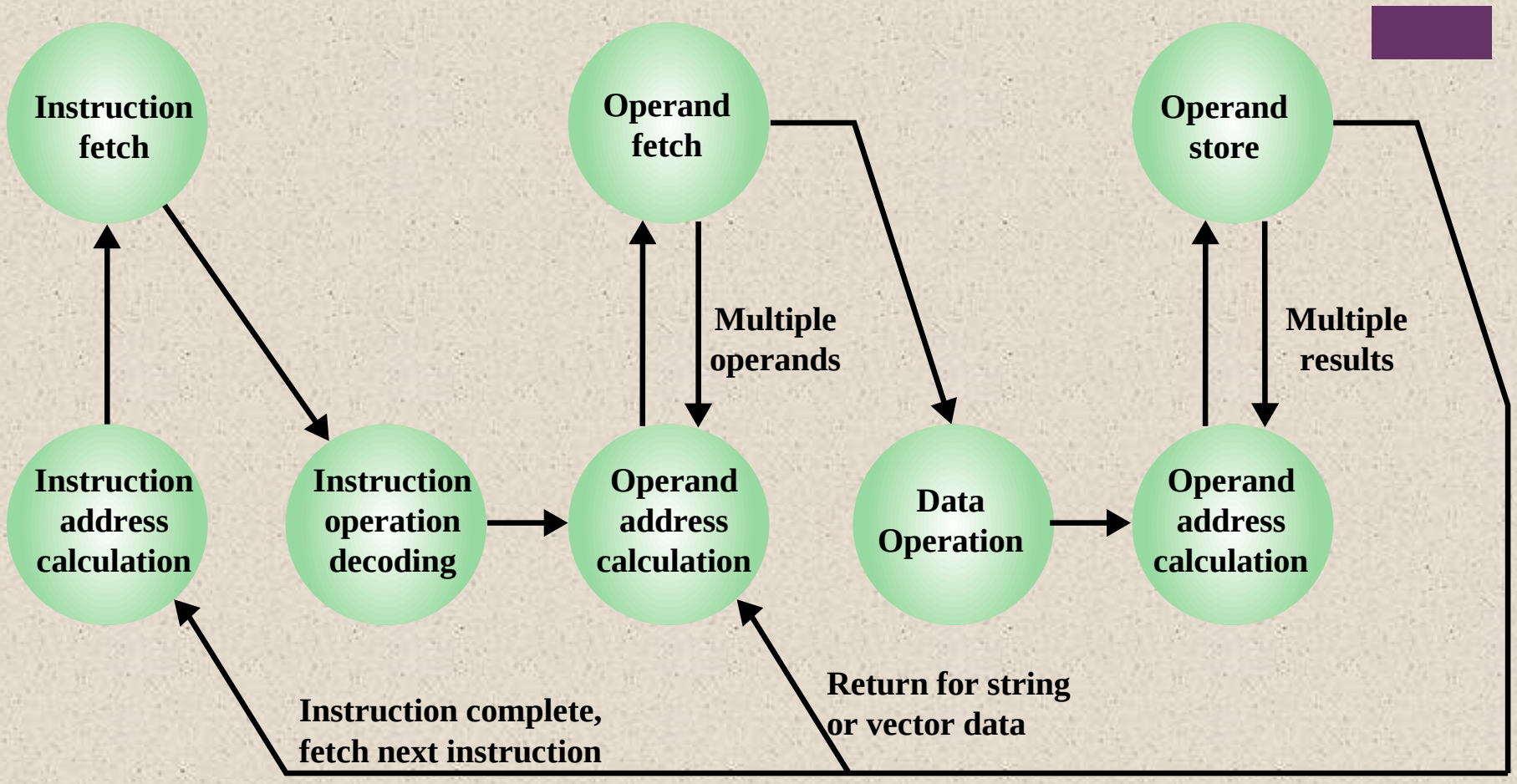


Figure 3.6 Instruction Cycle State Diagram

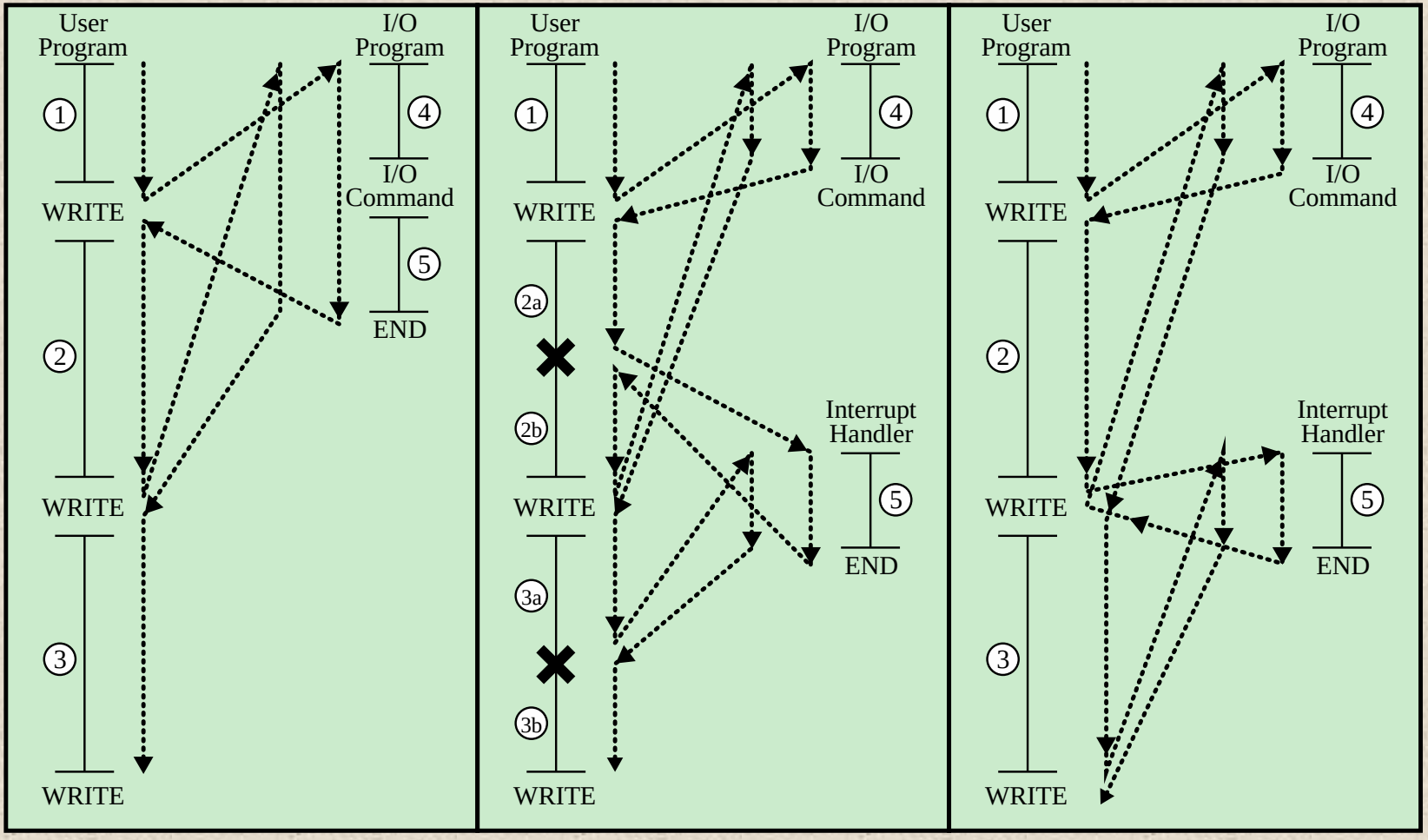
Hemen hemen tüm bilgisayarlar, diğer modüllerin (I/O, bellek) işlemcinin normal işleyişini kesintiye uğratabileceği bir mekanizma sağlar.

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation, request service from the processor, or to signal a variety of error conditions.
Hardware failure	Generated by a failure such as power failure or memory parity error.

Table 3.1

Classes of Interrupts

Kesmeler, öncelikle işleme verimliliğini artırmanın bir yolunu sağlar.

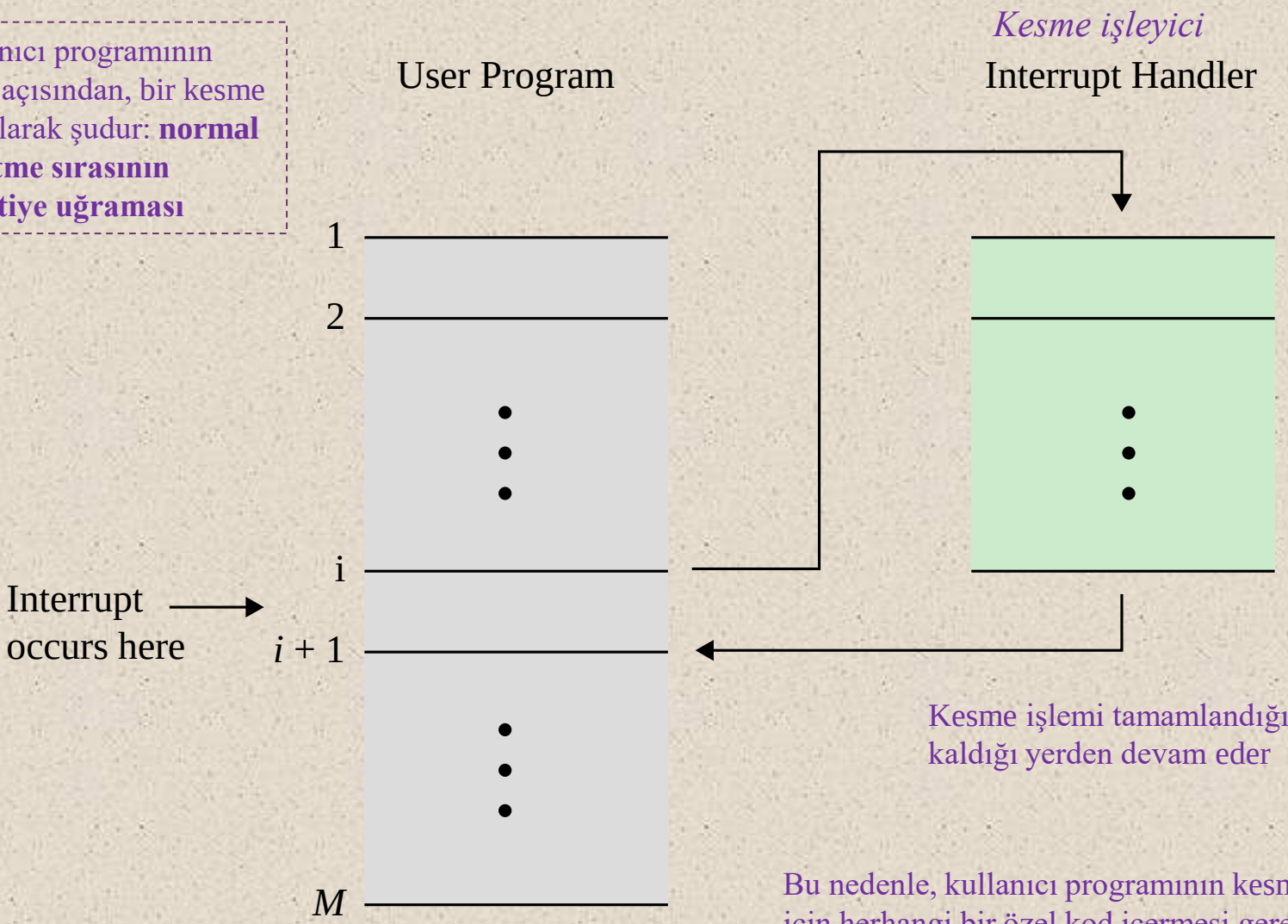


(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

✖ = interrupt occurs during course of execution of user program

Figure 3.7 Program Flow of Control Without and With Interrupts

Kullanıcı programının bakış açısından, bir kesme tam olarak şudur: **normal yürütme sırasının kesintiye uğraması**

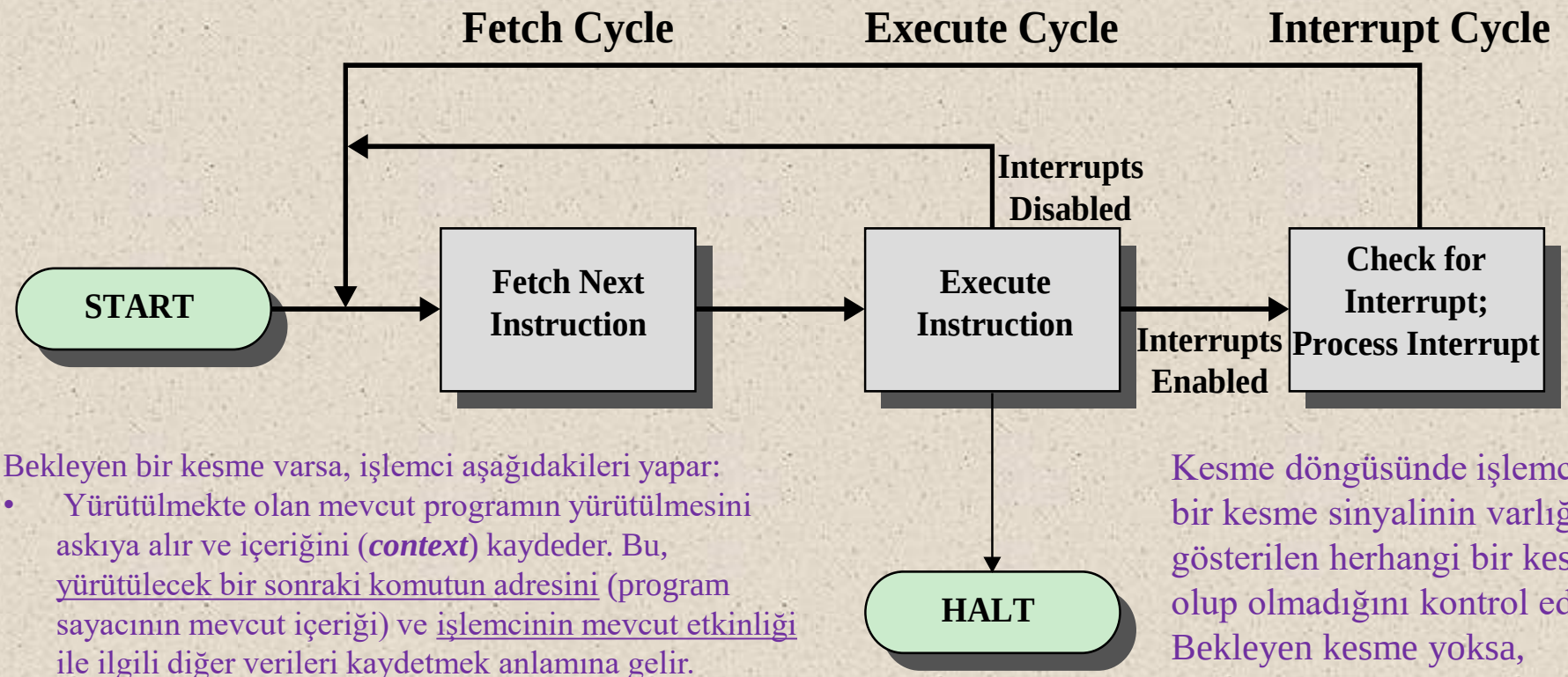


Kesme işlemi tamamlandığında, yürütme kaldığı yerden devam eder

Bu nedenle, kullanıcı programının kesmeleri barındırmak için herhangi bir özel kod içermesi gerekmez; işlemci ve işletim sistemi, kullanıcı programını askıya almaktan ve ardından aynı noktada devam ettirmekten sorumludur.

Figure 3.8 Transfer of Control via Interrupts

Kesmeleri barındırmak için, şekilde gösterildiği gibi komut döngüsüne bir kesme döngüsü eklenir.



Bekleyen bir kesme varsa, işlemci aşağıdakileri yapar:

- Yürütülmekte olan mevcut programın yürütülmesini askıya alır ve içeriğini (***context***) kaydeder. Bu, yürütülecek bir sonraki komutun adresini (program sayacının mevcut içeriği) ve işlemcinin mevcut etkinliği ile ilgili diğer verileri kaydetmek anlamına gelir.
- Program sayacını bir kesme işleyici (*interrupt handler*) rutininin başlangıç adresine ayarlar.

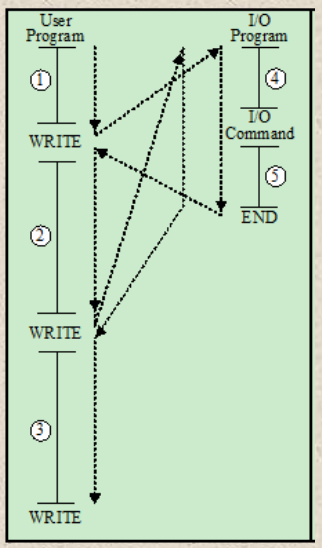
Kesme döngüsünde işlemci, bir kesme sinyalinin varlığı ile gösterilen herhangi bir kesme olup olmadığını kontrol eder. Bekleyen kesme yoksa, işlemci «fetch» döngüsüne geçer ve mevcut programın bir sonraki komutunu getirir.

Figure 3.9 Instruction Cycle with Interrupts

user program
code segments ->yeşil

I/O program code
segments -> gri

Şekil 3.10a, kesmelerin kullanılmadığı durumu göstermektedir. İşlemci, bir I / O işlemi gerçekleştirilirken beklemelidir.



(a) No interrupts

Şekil 3.7

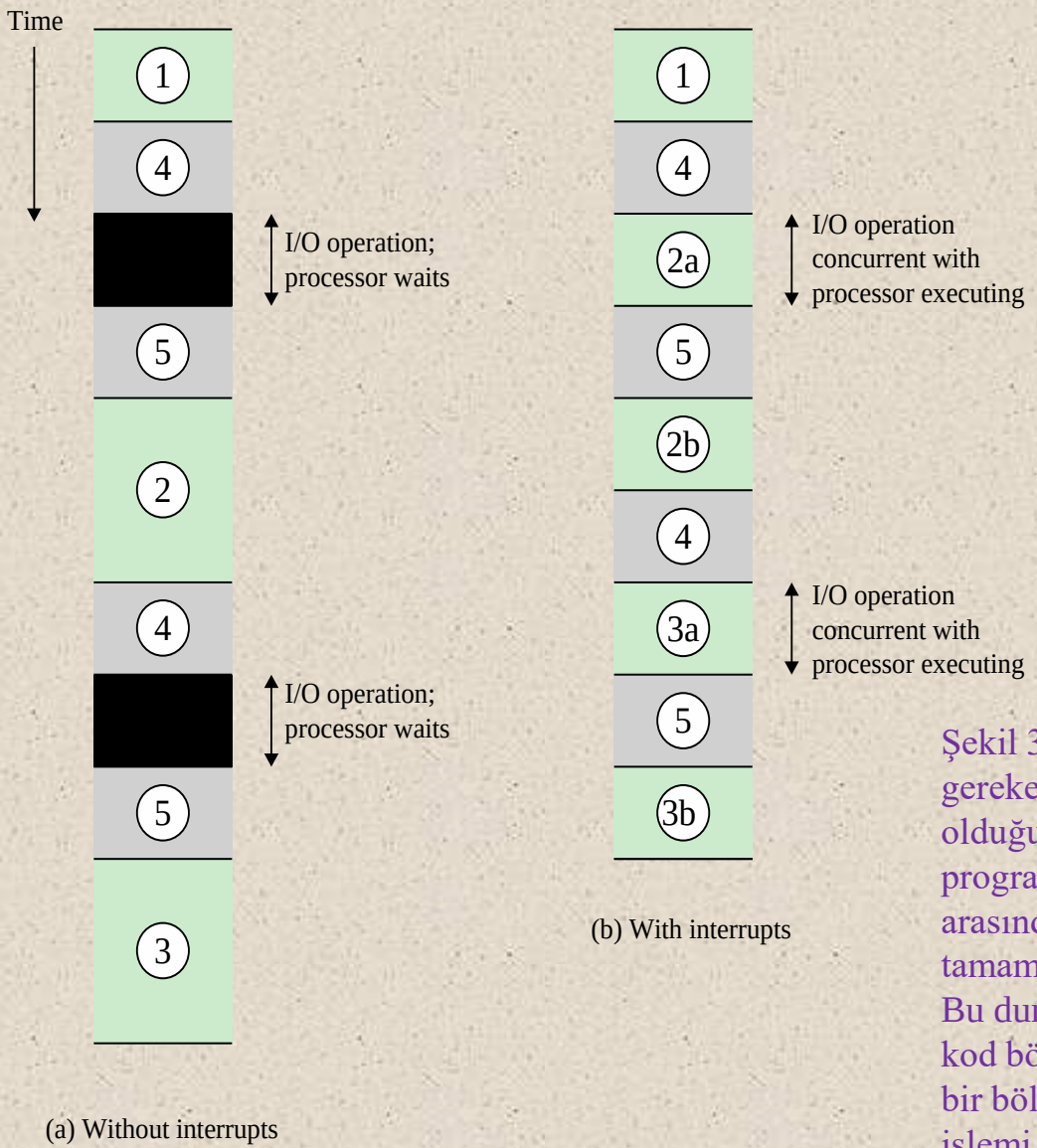
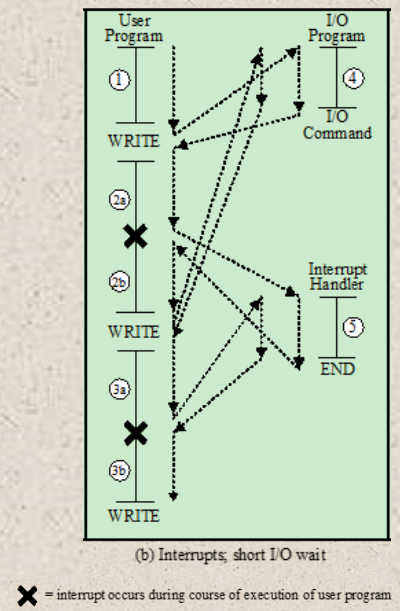


Figure 3.10 Program Timing: Short I/O Wait



Şekil 3.7

Şekil 3.7b ve 3.10b, I/O işlemi için gereken sürenin nispeten kısa olduğunu varsaymaktadır: kullanıcı programındaki yazma işlemleri arasında komutların yürütülmesini tamamlama süresinden daha azdır. Bu durumda, 2 olarak etiketlenmiş kod bölümü kesintiye uğrar. Kodun bir bölümü (2a) yürütülür (I/O işlemi gerçekleştirilirken) ve ardından kesme gerçekleşir (I/O işlemi tamamlandıktan sonra). Kesme servis rutini icra edildikten sonra, yürütme, kod segmenti 2 (2b) 'nin geri kalanıyla devam eder.

Özellikle yazıcı gibi yavaş bir cihaz için daha tipik durum, **I / O işleminin** bir dizi kullanıcı komutunu yürütmekten **çok daha fazla zaman almasıdır**.

Şekil 3.7c bu durumu göstermektedir. Bu durumda, kullanıcı programı, ilk çağrı ile ortaya çıkan I / O işlemi tamamlanmadan önce ikinci WRITE çağrısına ulaşır.

Sonuç, kullanıcı programının bu noktada beklemeye alınmasıdır. Önceki I/O işlemi tamamlandığında, bu yeni WRITE çağrısı işlenebilir ve yeni bir I/O işlemi başlatılabilir.

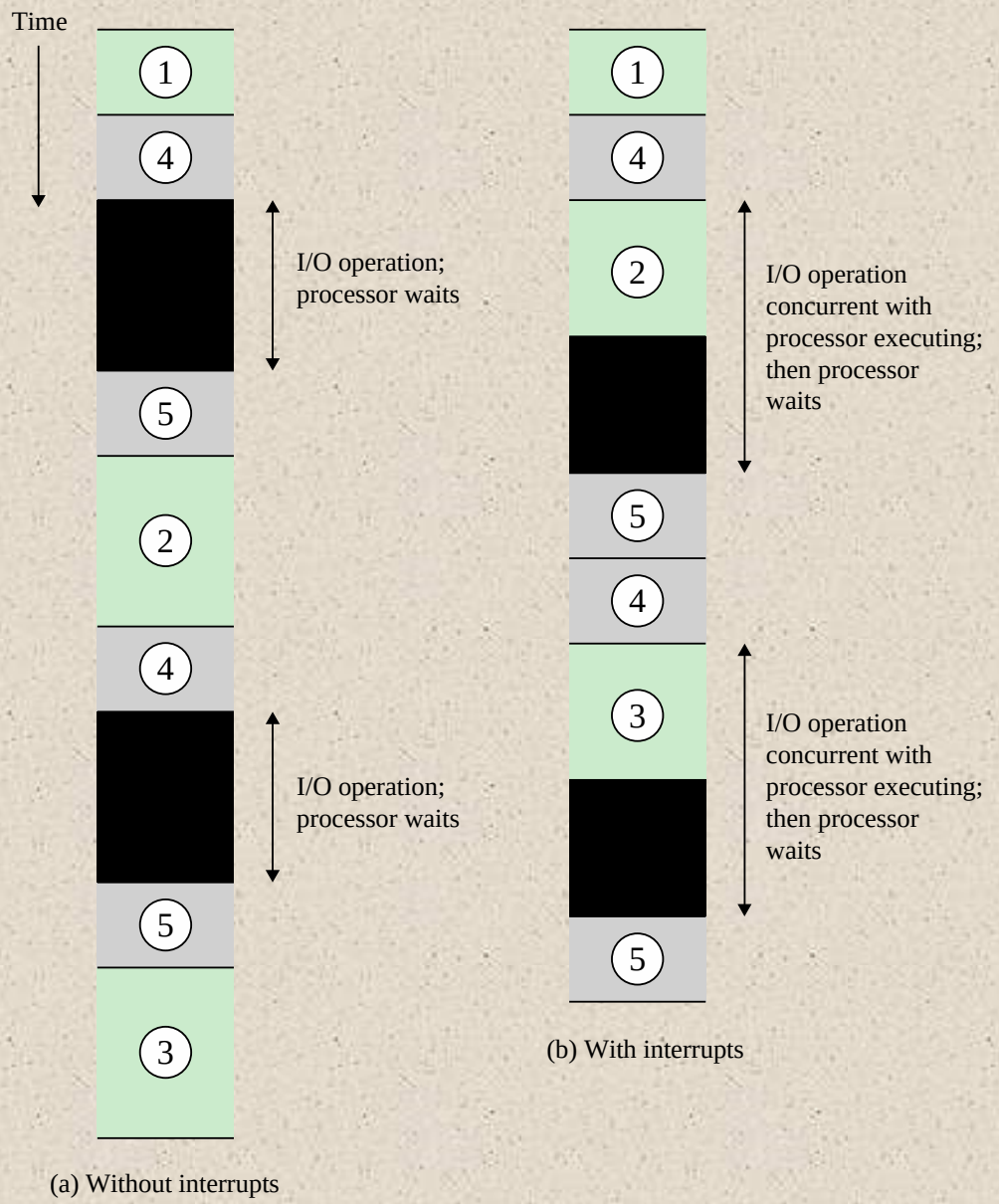
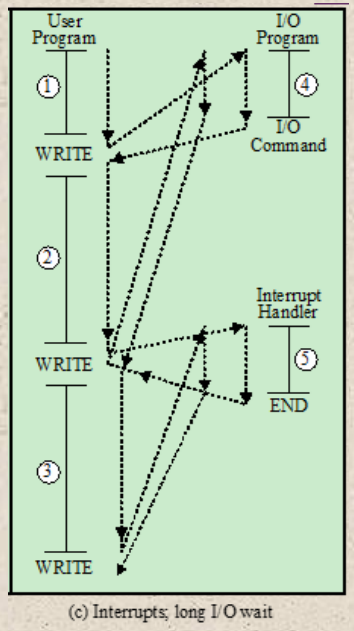


Figure 3.11 Program Timing: Long I/O Wait



Şekil 3.7

Şekil 3.11, kesme kullanımıyla ve kullanılmadan bu durum için zamanlamayı göstermektedir. Verimlilikte hala bir kazanç olduğunu görebiliriz çünkü I/O işleminin devam ettiği sürenin bir kısmı kullanıcı komutlarının yürütülmesiyle çakışır/üst üste gelir.

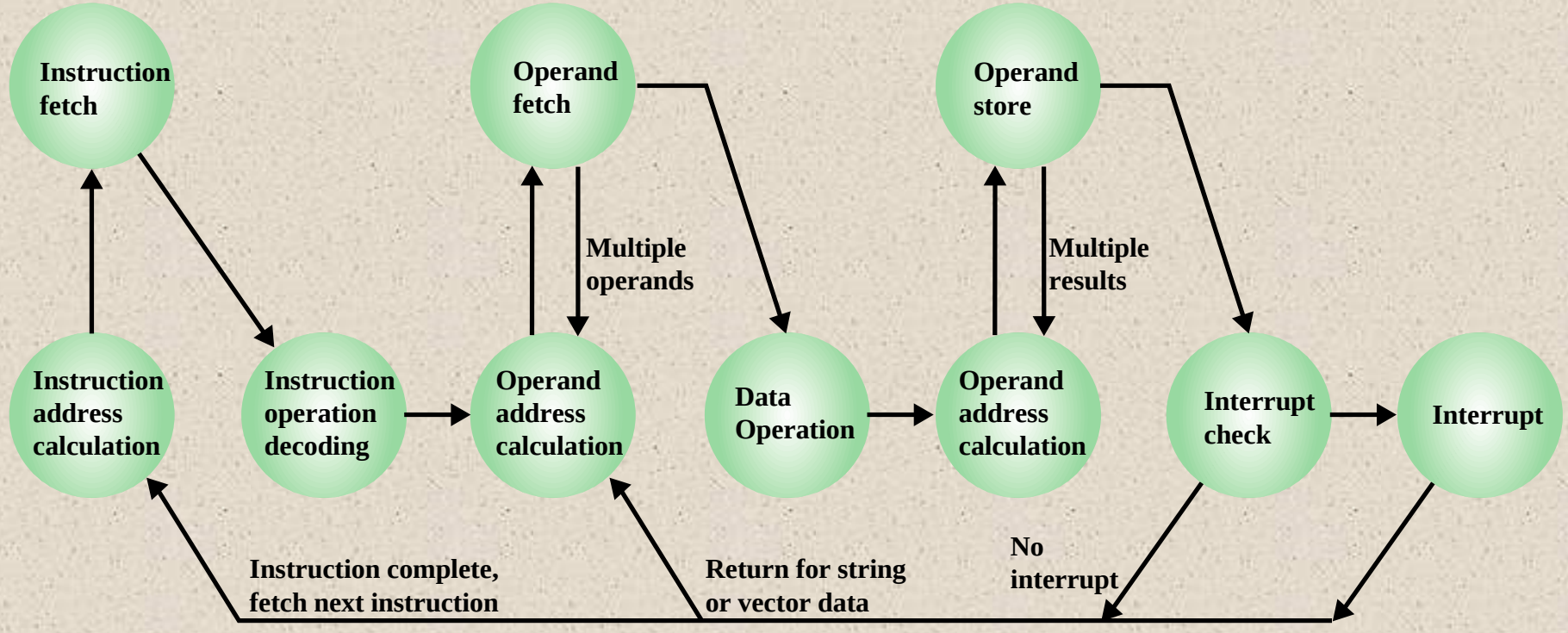
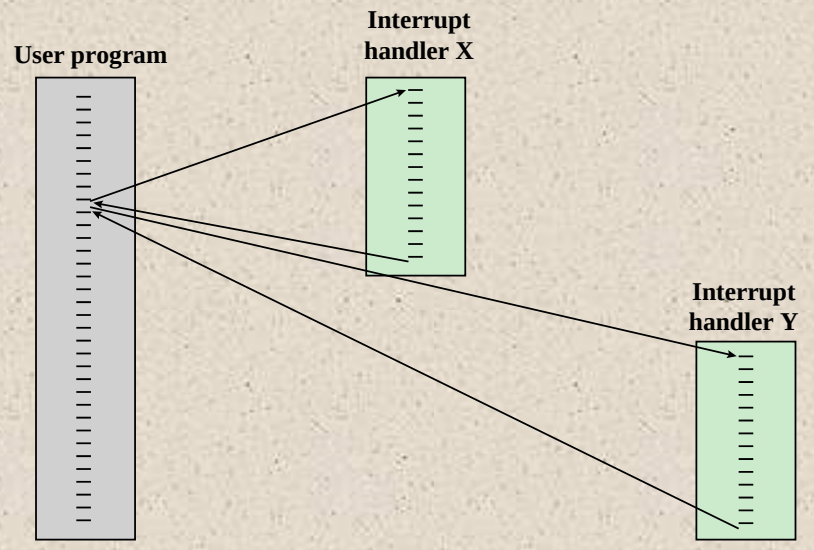


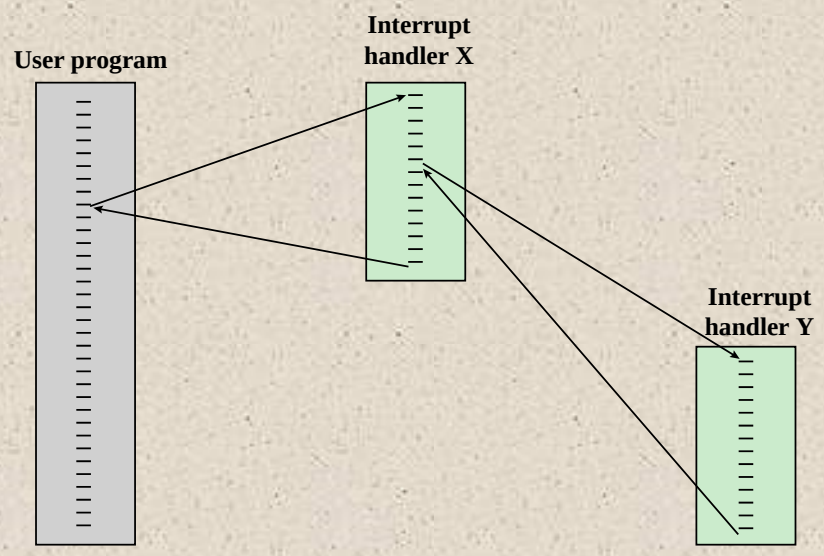
Figure 3.12 Instruction Cycle State Diagram, With Interrupts

Bir kullanıcı programı yürütülürken ve bir kesme meydana geldiğinde, kesmeler hemen devre dışı (*disable*) bırakılır. Kesme işleyici rutini tamamlandıktan sonra, kullanıcı programını sürdürmeden önce kesmeler etkinleştirilir ve işlemci ek kesmelerin olup olmadığını kontrol eder. Kesmeler kesin sırayla işlendiğinden, bu yaklaşım güzel ve basittir.(Şekil 3.13a).



(a) Sequential interrupt processing *(Sıralı kesme işleme)*

İkinci bir yaklaşım, kesmeler için öncelikleri (*priorities*) tanımlamak ve daha düşük öncelikli bir kesme işleyicisinin kendi başına kesmeye uğramasını sağlayıp daha yüksek öncelikli bir kesmeye izin vermektir (Şekil 3.13b).

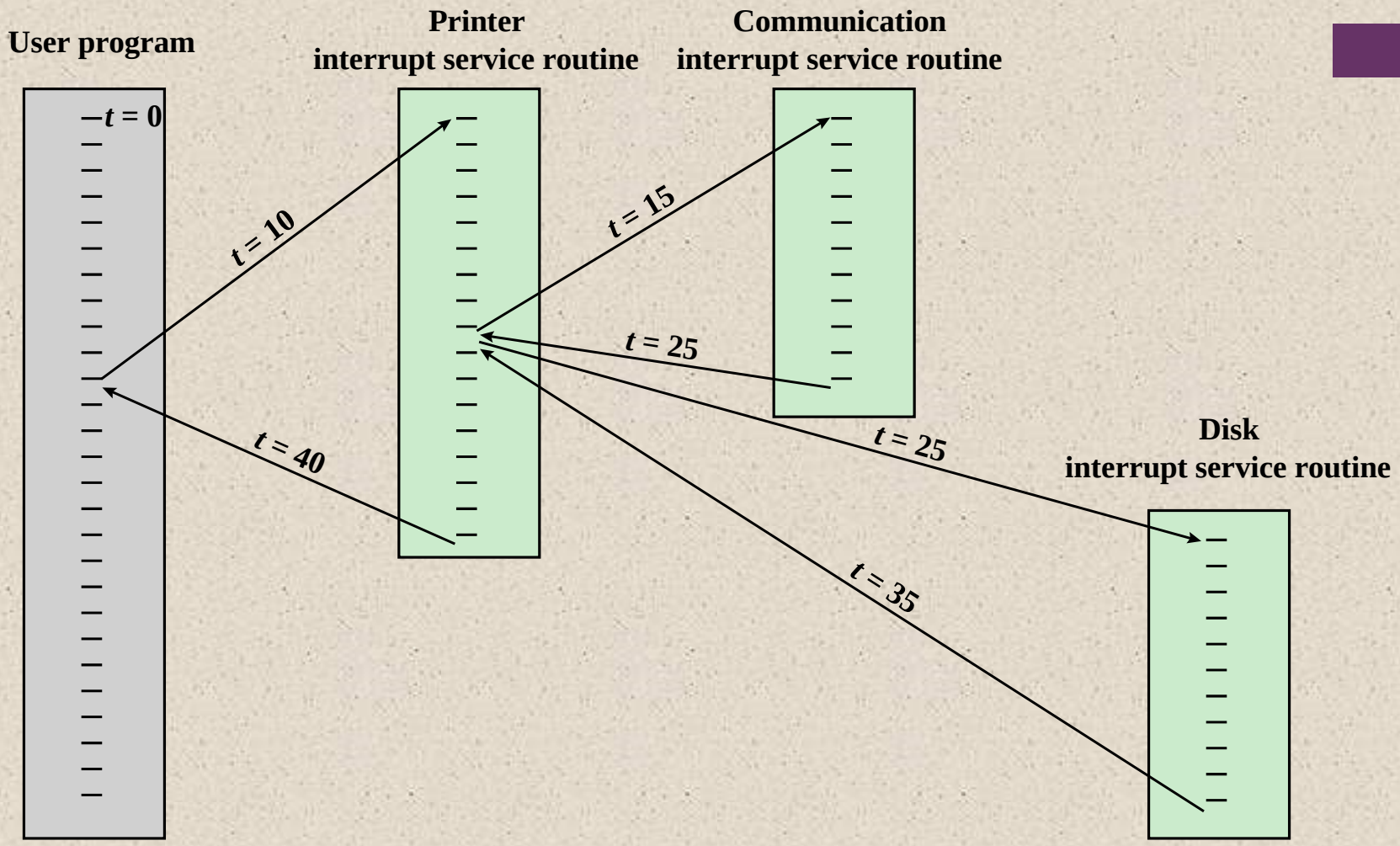


(b) Nested interrupt processing *(İç-içe kesme işleme)*

Figure 3.13 Transfer of Control with Multiple Interrupts

Bu yaklaşımın dezavantajı, **göreceli önceliği** veya **zaman açısından kritik ihtiyaçları hesaba katmamasıdır**.

Örneğin, iletişim hattından girdi geldiğinde, daha fazla girdiye yer açmak için hızla işlenmesi gerekebilir. İlk girdi grubu, ikinci grup gelmeden önce işlenmemişse, veriler kaybolabilir.



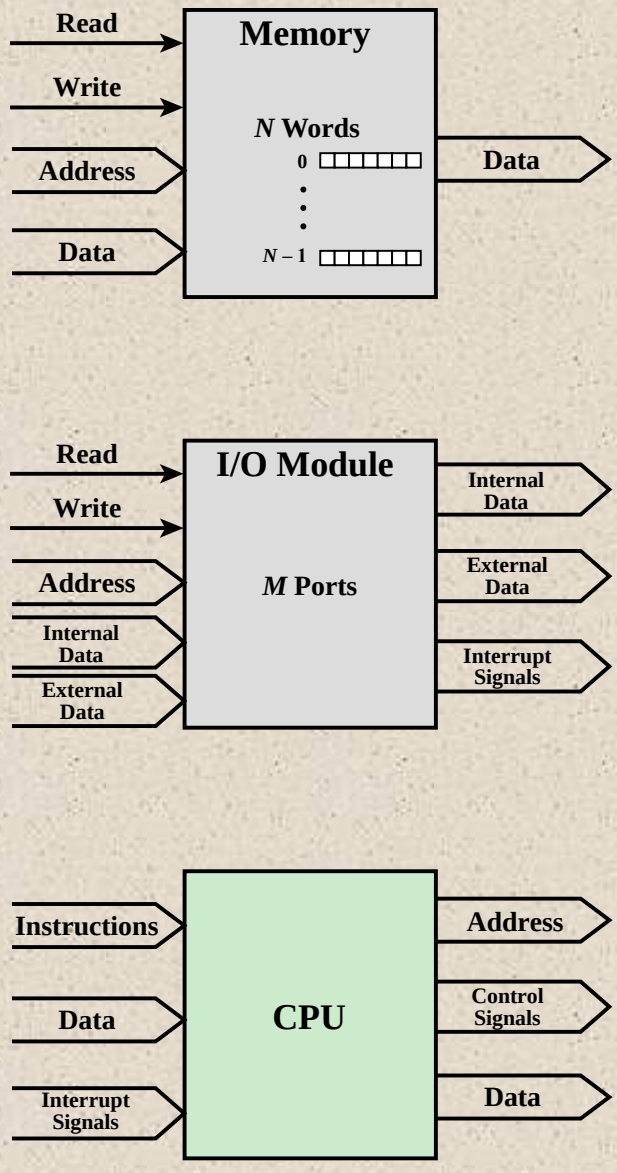
Bu ikinci yaklaşıma örnek olarak, üç I/O cihazına sahip bir sistemi düşünün: bir yazıcı, bir disk ve bir iletişim hattı (artan önceliklere sahip)

Figure 3.14 Example Time Sequence of Multiple Interrupts

I/O Function

- I/O modülü, doğrudan işlemci ile veri alışverişi yapabilir
- İşlemci bir I/O modülünden veri okuyabilir veya veri yazabilir
  - İşlemci, belirli bir I/O modülü tarafından kontrol edilen belirli bir cihazı tanımlar
  - Belleğe referans veren komutlar yerine I / O komutları
- Bazı durumlarda, I/O veri alışverişinin (*exchange*) doğrudan bellekle gerçekleşmesine izin verilmesi istenebilir.
  - İşlemci, bir I/O modülüne bellekten okuma veya belleğe yazma yetkisi verir, böylece I/O bellek aktarımı işlemciyi bağlamadan gerçekleşebilir
  - I/O modülü, işlemcinin veri alışverişi (*exchange*) sorumluluğunu ortadan kaldırarak belleğe okuma veya yazma komutları verir
  - Bu işlem, doğrudan bellek erişimi (DMA- **direct memory access**) olarak bilinir

Bir bilgisayar, birbiriyle iletişim kuran üç temel türden (işlemci, bellek, I/O) oluşan bir dizi bileşen veya modülden oluşur. Aslında, bilgisayar temel modüllerden oluşan bir ağdır. Bu nedenle, modülleri bağlamak için yollar olmalıdır.



Tipik olarak, bir bellek modülü eşit uzunlukta N kelimeden oluşur. Her kelimeye benzersiz bir sayısal adres (0, 1,…, N-1) atanır. Bir veri kelimesi bellekten okunabilir veya belleğe yazılabilir. İşlemin doğası, okuma ve yazma kontrol sinyalleri ile gösterilir. İşlemin konumu bir adresle belirtilir.

Dahili (bilgisayar sistemine) bakış açısından, I/O işlevsel olarak belleğe benzer. Okuma ve yazma olmak üzere iki işlem vardır.

Arayüzlerin her birini harici bir cihaza bağlantı noktası olarak adlandırabiliriz ve her birine benzersiz bir adres verebiliriz (örneğin, 0, 1,…, M-1).

Ek olarak, harici bir cihazla veri girişi ve çıkışı için harici veri (***external data***) yolları vardır. Son olarak, bir I/O modülü işlemciye kesme sinyalleri gönderebilir.

Figure 3.15 Computer Modules

Ara bağlantı (*interconnection*) yapısı aşağıdaki aktarım türlerini desteklemelidir:

Memory to processor	Processor to memory	I/O to processor	Processor to I/O	I/O to or from memory
İşlemci, bellekten bir komut veya bir veri birimi okur	İşlemci bir veri birimini belleğe yazar	İşlemci, bir I/O modülü aracılığıyla bir I/O cihazındaki verileri okur	İşlemci verileri I / O cihazına gönderir	Bir I / O modülünün, doğrudan bellek erişimi kullanarak işlemciden geçmeden doğrudan bellekle veri alışverişi yapmasına izin verilir

Yıllar boyunca, bir dizi ara bağlantı yapısı denenmiştir. Açık farkla en yaygın olanları (1) veri yolu (*bus*) ve çeşitli çoklu veri yolu yapıları ve (2) paketlenmiş veri aktarımına sahip noktadan noktaya ara bağlantı (*point-to-point interconnection*) yapılarıdır.

Bus Interconnection

İki veya daha fazla cihazı birbirine bağlayan bir iletişim yolu

- Anahtar özelliği, paylaşılan bir aktarım ortamı olmasıdır

Herhangi bir cihaz tarafından iletilen sinyaller, veri yoluna bağlı diğer tüm cihazlar tarafından alınabilir.

- Aynı zaman diliminde iki cihaz iletim/aktarım yaparsa, sinyalleri üst üste biner ve bozulur

Tipik olarak birden fazla iletişim hattından oluşur

- Her hat, binary 1 ve binary 0'ı temsil eden sinyalleri iletebilir.

Bilgisayar sistemleri, bilgisayar sistemi hiyerarşisinin çeşitli düzeylerindeki bileşenler arasında yollar sağlayan bir dizi farklı veri yolu içerir.

System bus

- Başlıca bilgisayar bileşenlerini (işlemci, bellek, I/O) bağlayan bir veri yolu

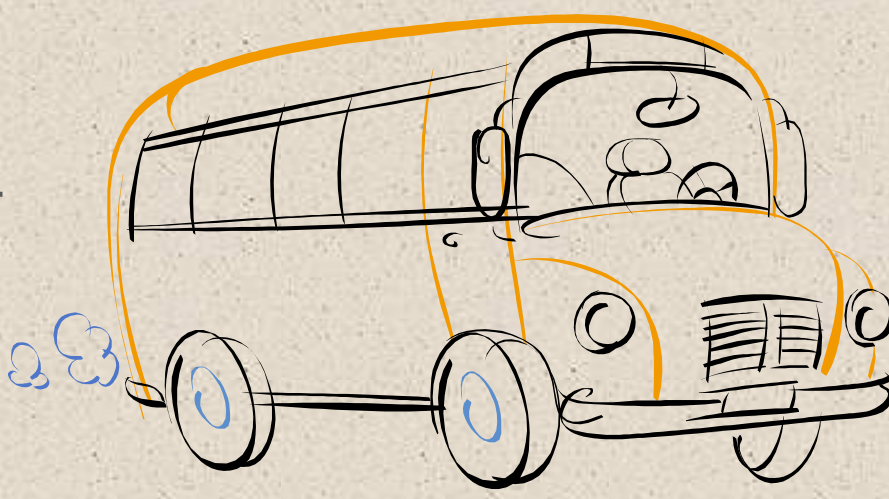
En yaygın bilgisayar ara bağlantı (*interconnection*) yapıları, bir veya daha fazla sistem veriyolunun kullanımına dayanır

Data Bus

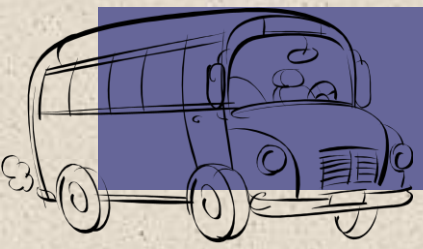
- Sistem modülleri arasında veri taşımak için bir yol sağlayan veri hatları
- 32, 64, 128 veya daha fazla sayıda ayrı hattan oluşabilir
- Hat/satır sayısı, veri yolunun genişliği (*width*) olarak adlandırılır
- Hat sayısı, bir seferde kaç bitin aktarılabileceğini belirler
- Veri yolunun genişliği,

  genel sistem performansını

  belirlemede önemli bir faktördür

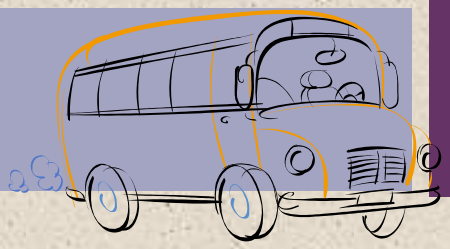


Address Bus



- Veri yolundaki verinin kaynağını veya hedefini belirlemek için kullanılır
  - İşlemci bellekten bir veri kelimesini okumak isterse, istenen kelimenin adresini adres satırlarına koyar
- Genişlik (*Width*), sistemin mümkün olan maksimum bellek kapasitesini belirler
- I/O bağlantı noktalarını (*ports*) adreslemek için de kullanılır
  - Daha üst dereceli bitler, veriyolunda belirli bir modülü seçmek için kullanılır ve düşük dereceli bitler, modül içinde bir bellek konumu veya I/O bağlantı noktası seçer

Control Bus



- Veri ve adres hatlarının erişimini ve kullanımını kontrol etmek için kullanılır
- Veri ve adres hatları tüm bileşenler tarafından paylaşıldığından, kullanımlarını kontrol etmenin bir yolu olmalıdır.
- Kontrol sinyalleri, sistem modülleri arasında hem komut hem de zamanlama bilgilerini (*command and timing information*) iletir
- Zamanlama sinyalleri, verilerin ve adres bilgilerinin geçerliliğini gösterir
- Komut sinyalleri gerçekleştirilecek işlemleri belirtir

Her hatta belirli bir anlam veya işlev atanır. Birçok farklı veri yolu tasarımı olmasına rağmen, herhangi bir veri yolunda hatlar üç işlevsel gruba ayrılabilir (Şekil 3.16): veri, adres ve kontrol hatları. İlave olarak, bağlı modüllere güç sağlayan güç dağıtım hatları olabilir.

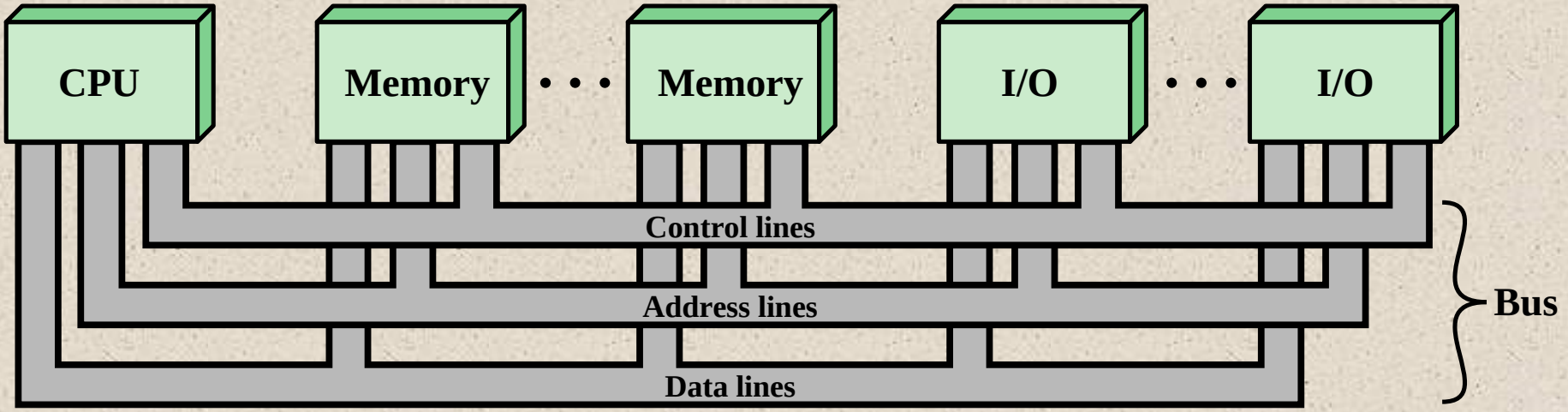


Figure 3.16 Bus Interconnection Scheme

Bus'ın çalışması aşağıdaki gibidir. Bir modül diğerine veri göndermek istiyorsa, iki şey yapmalıdır:
(1) veri yolu kullanımını elde etmek ve
(2) veri yolu üzerinden veri aktarmak.

Bir modül başka bir modülden veri talep etmek isterse,
(1) veri yolu kullanımını elde etmeli ve
(2) uygun kontrol ve adres hatları üzerinden diğer modüle bir talep aktarmalıdır. Daha sonra ikinci modülün (yani karşı tarafın) verileri göndermesini beklemesi gerekir.

Paylaşılan veri yolu mimarisi, on yıllardır işlemci ve diğer bileşenler (bellek, I / O vb.) arasındaki ara bağlantı için standart yaklaşımdı. Ancak modern sistemler, paylaşılan bus'lar yerine giderek daha fazla noktadan noktaya bağlantıya güveniyor.

Point-to-Point Interconnect

Değişimin temel nedeni, geniş eşzamanlı bus'ların frekansının artmasıyla karşılaşılan elektriksel kısıtlamalar

Daha yüksek ve daha yüksek veri hızlarında, senkronizasyon ve yol hakemliği işlevlerini (***synchronization and arbitration functions***) zamanında gerçekleştirmek giderek zorlaşıyor

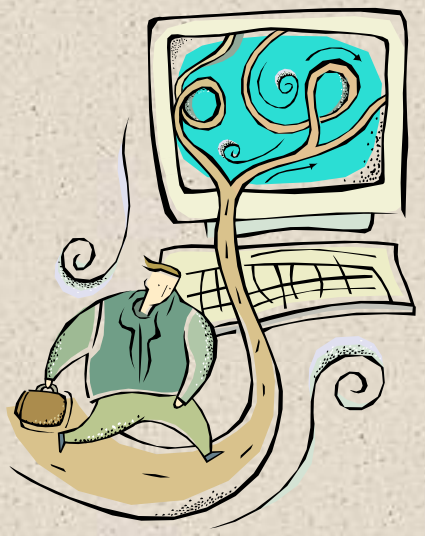
Aynı çip üzerindeki geleneksel bir paylaşılan veri yolu, *bus* veri hızını artırmanın ve işlemcilere ayak uydurmak için *bus* gecikmesini azaltmanın zorluklarını artırdı

PPI, daha düşük gecikme, daha yüksek veri hızı ve daha iyi ölçeklenebilirliğe sahiptir

Quick Path Interconnect

- 2008'de tanıtıldı
- Birden çok doğrudan bağlantı (***Multiple direct connections***)
  - Paylaşılan iletim sistemlerinde bulunan yol hakemliği ihtiyacını ortadan kaldıran diğer bileşenlere doğrudan çift yönlü bağlantılar
- Katmanlı protokol mimarisi(***Layered protocol architecture***)
  - Bu işlemci seviyesindeki ara bağlantılar, paylaşılan veriyolu düzenlemelerinde bulunan kontrol sinyallerinin basit kullanımı yerine katmanlı bir protokol mimarisi kullanır.
- Paketlenmiş veri aktarımı (***Packetized data transfer:***)
  - Veriler, her biri kontrol başlıklarını ve hata kontrol kodlarını içeren bir dizi paket olarak gönderilir

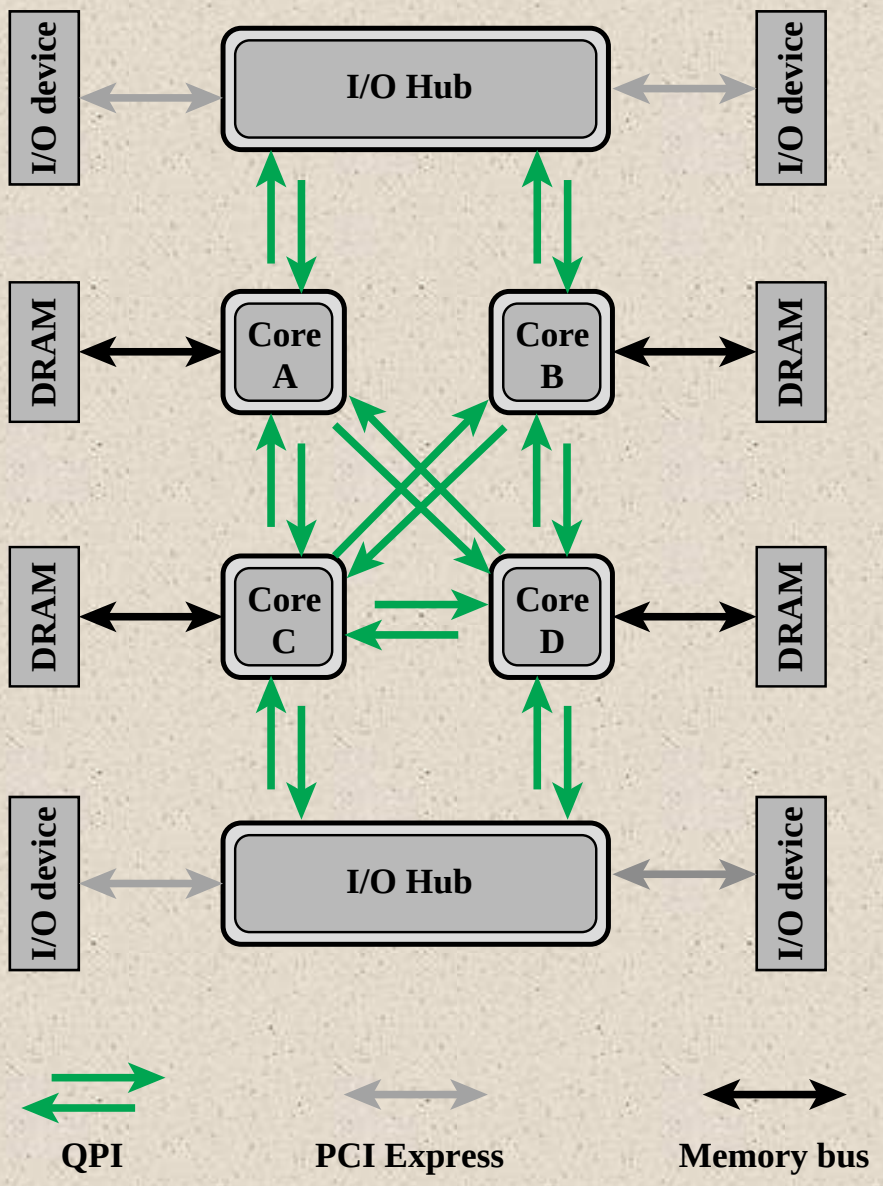
QPI



Her bir çekirdek işlemci çifti arasında doğrudan QPI bağlantıları kurulabilir.

Şekil 3.17'deki çekirdek A'nın çekirdek D'deki bellek denetleyicisine erişmesi gerekiyorsa, isteğini B veya C çekirdeği aracılığıyla gönderir ve bu da bu isteği çekirdek D'deki bellek denetleyicisine iletmelidir.

Benzer şekilde, sekiz veya daha fazla işlemciye sahip daha büyük sistemler, üç bağlantılı (*with three links*) işlemciler ve trafiği ara işlemciler üzerinden yönlendirerek oluşturulabilir.



Ek olarak, QPI, I/O hub (IOH) adı verilen bir I/O modülüne bağlanmak için kullanılır. IOH, trafiği I/O cihazlarına yönlendiren bir anahtar görevi görür.

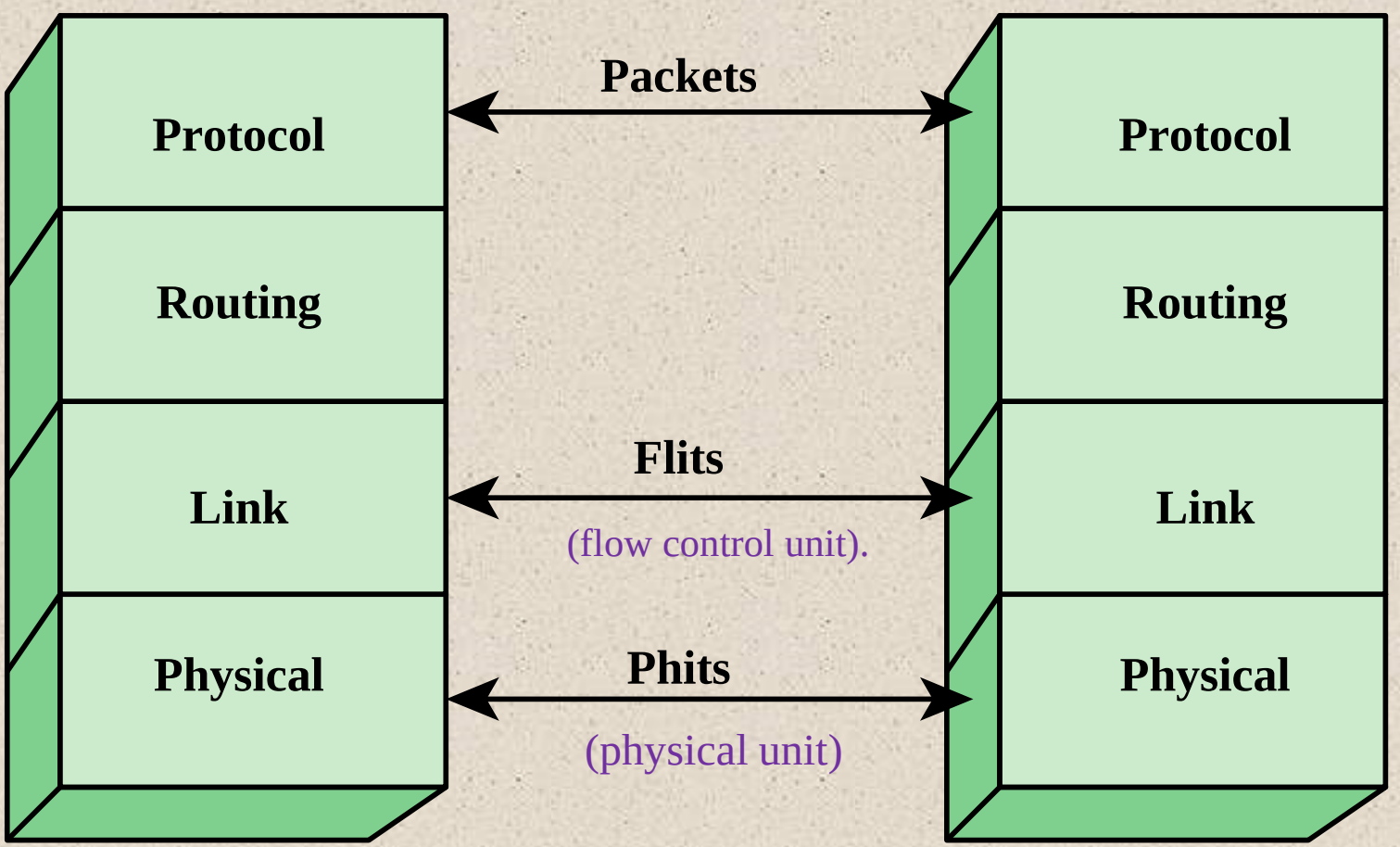
Tipik olarak daha yeni sistemlerde, IOH'den I/O aygıt denetleyicisine olan bağlantı, **PCI Express (PCIe)** adı verilen bir ara bağlantı teknolojisini kullanır.

IOH, QPI protokolleri ve formatları ile PCIe protokolleri ve formatları arasında dönüşüm yapar.

Bir çekirdek aynı zamanda bir ana bellek modülüne (tipik olarak bellek, dinamik erişimli rastgele bellek (DRAM) teknolojisi kullanır) özel bir bellek veri yolu kullanarak bağlanır.

Figure 3.17 Multicore Configuration Using QPI

QPI, dört katmanlı bir protokol mimarisi olarak tanımlanır:



Protocol: Cihazlar arasında veri paketlerinin değiş tokuşu için üst düzey kurallar dizisi. Bir paket, çok sayıda flit'ten oluşur.

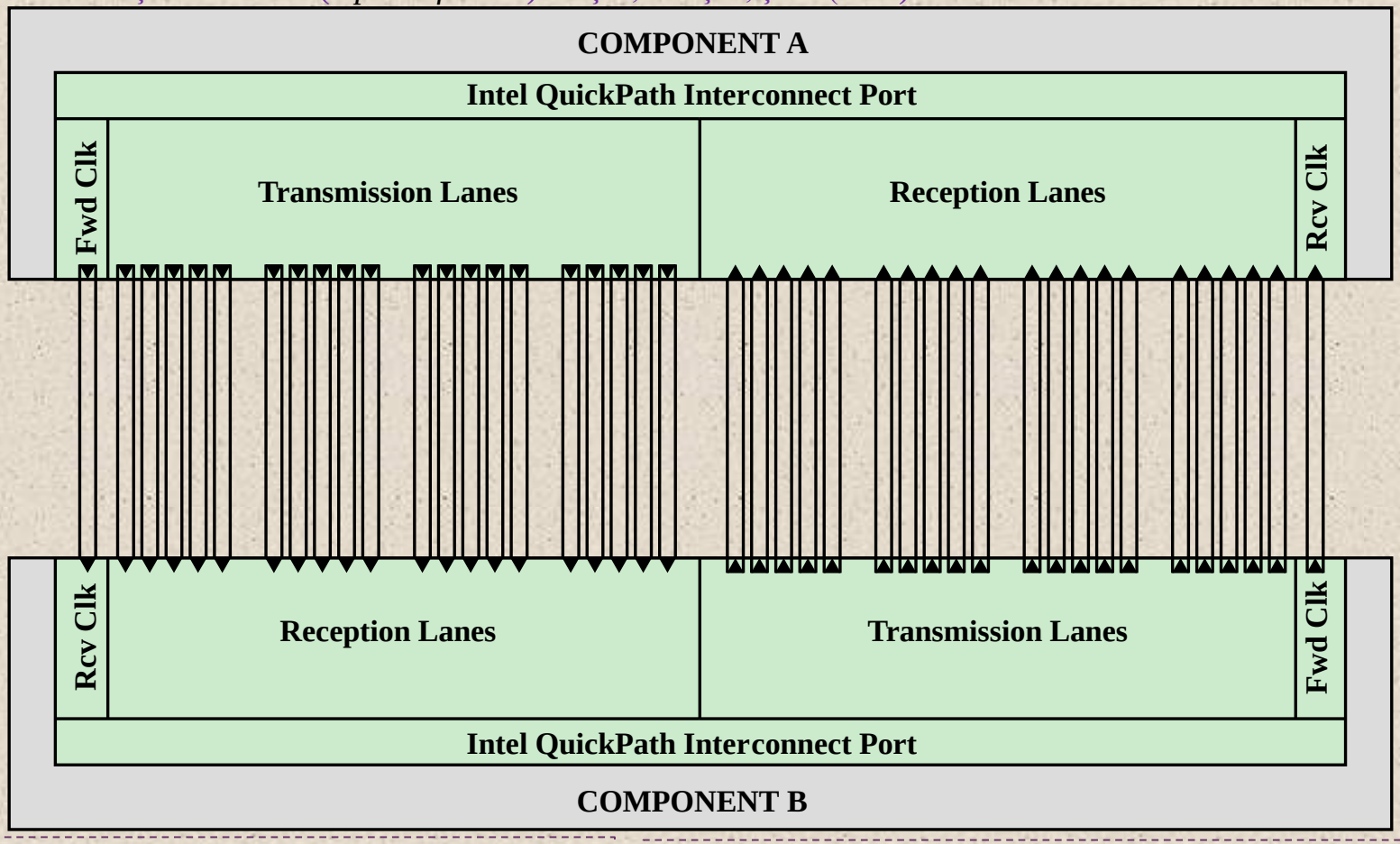
Routing: Paketleri yapı boyunca yönlendirmek için çerçeve/mekanizma sağlar.

Link: Güvenilir iletim ve akış kontrolünden sorumludur. Bağlantı katmanının aktarım birimi **80 bitlik** bir flittir (akış kontrol birimi).

Physical: Sinyalleri taşıyan gerçek kabloların yanı sıra 1'lerin ve 0'ların iletimi ve alınmasında gerekli olan yardımcı özellikleri desteklemek için devre ve mantıktan oluşur. Fiziksel katmandaki aktarım birimi, Phit (fiziksel birim) olarak adlandırılan, **20 bittir**.

Figure 3.18 QPI Layers

QPI bağlantı noktası, aşağıdaki şekilde gruplandırılmış 84 ayrı bağlantıdan oluşur. Her veri yolu, her seferinde bir bit veri ileten bir çift kablodan (*a pair of wires*) oluşur; her çift, şerit (***lane***) olarak adlandırılır.



Her yöndeki şeritler, her biri 5 şeritli dört bölüm halinde gruplandırılmış tır.

Bazı uygulamalarda, bağlantı; güç tüketimini azaltmak veya arızaların giderilmesi için yarım veya çeyrek genişliklerde de çalışabilir.

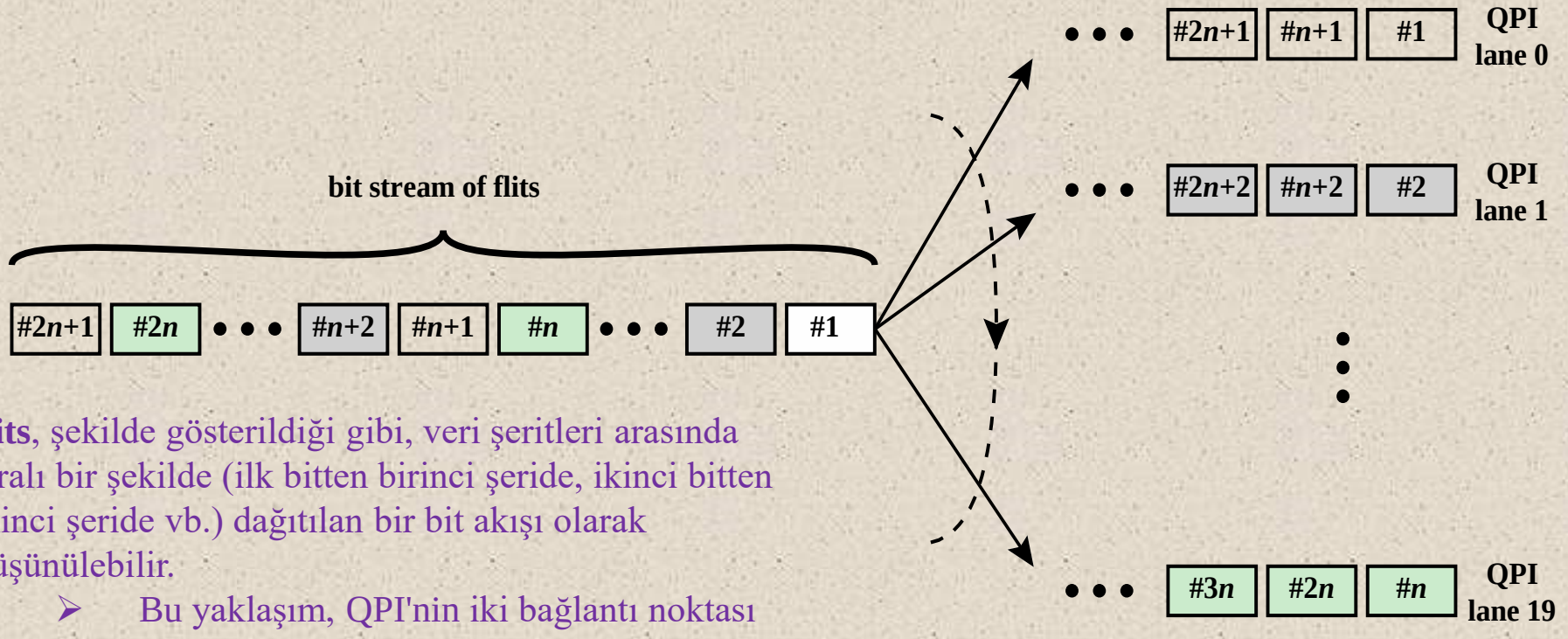
Her yönde (gönderme ve alma) 20 veri şeridi ve ayrıca her yönde bir saat şeridi vardır. Böylece QPI, her yönde paralel olarak 20 bit iletebilir. 20 bitlik birim, phit olarak adlandırılır.

Mevcut ürünlerdeki bağlantının tipik sinyal hızları, **6,4 GT/s** (saniye başına aktarım) ile çalışmayı gerektirir. Aktarım başına 20 bit'te bu toplam **16 GB/sn** ve QPI bağlantıları ayrılmış iki yönlü (*bidirectional*) çiftler içerdiğinden, toplam **kapasite 32 GB/sn**'dir.

Figure 3.19 Physical Interface of the Intel QPI Interconnect

Şekil 3.19, bir QPI bağlantı noktasının fiziksel mimarisini gösterir.

Fiziksel katman tarafından gerçekleştirilen diğer bir fonksiyon, çok şeritli dağıtım (***multilane distribution***) olarak bilinen bir teknik kullanarak 80 bitlik flit'ler ve 20 bitlik phit'ler arasındaki dönüşümü yönetmesidir.



flits, şekilde gösterildiği gibi, veri şeritleri arasında sıralı bir şekilde (ilk bitten birinci şeride, ikinci bitten ikinci şeride vb.) dağıtılan bir bit akışı olarak düşünülebilir.

- Bu yaklaşım, QPI'nin iki bağlantı noktası arasındaki fiziksel bağlantıyı çok sayıda paralel kanal olarak uygulayarak çok yüksek veri hızları elde etmesini sağlar.

Figure 3.20 QPI Multilane Distribution

QPI Link Layer

Bir flit yükü, verilerden veya mesaj bilgilerinden oluşabilir. Veri flit'leri, gerçek veri bitlerini çekirdekler arasında veya bir çekirdek ile bir IOH arasında aktarır. Mesaj flit'leri; akış kontrolü (*flow control*), hata kontrolü (*error control*) ve önbellek tutarlılığı (*cache coherence*) gibi fonksiyonlar için kullanılır.

- İki fonksiyonu gerçekleştirir: akış kontrolü (*flow control*) ve hata kontrolü (*error control*)
  - «flit» düzeyinde çalışır (flow control unit)
  - Her flit 72 bitlik bir mesaj yükü ve CRC (*cyclic redundancy check*) adı verilen 8 bitlik bir hata kontrol kodundan oluşur.
- Akış kontrol fonksiyonu
  - Gönderen bir QPI biriminin, alıcının verileri işleyebileceğinden daha hızlı veri göndererek ve daha fazla gelen veri için arabellekleri temizleyerek alıcı QPI birimini bunaltmamasını sağlamak için gerekli
- Hata kontrol fonksiyonu
  - Bit hatalarını algılar ve kurtarır ve böylece daha yüksek katmanları bit hatalarından izole eder

QPI Routing and Protocol Layers

Routing Layer

- Bir paketin mevcut sistem ara bağlantılarından geçeceği seyri belirlemek için kullanılır
- Donanıma kaydedilmiş yazılım (*firmware*) tarafından tanımlanır ve bir paketin izleyebileceği olası yolları tanımlar

Protocol Layer

- Paket, transfer birimi olarak tanımlanır
- Bu seviyede gerçekleştirilen bir temel işlev, birden çok önbellekte tutulan ana bellek değerlerinin tutarlı olmasını sağlamayı amaçlayan bir önbellek tutarlılık protokolüdür.
- Tipik bir veri paketi yükü (*payload*), bir önbelleğe giden/gelen bir veri bloğudur.

Intel, Pentium tabanlı sistemleri için 1990 yılında PCI üzerinde çalışmaya başladı.

Peripheral Component Interconnect (PCI)

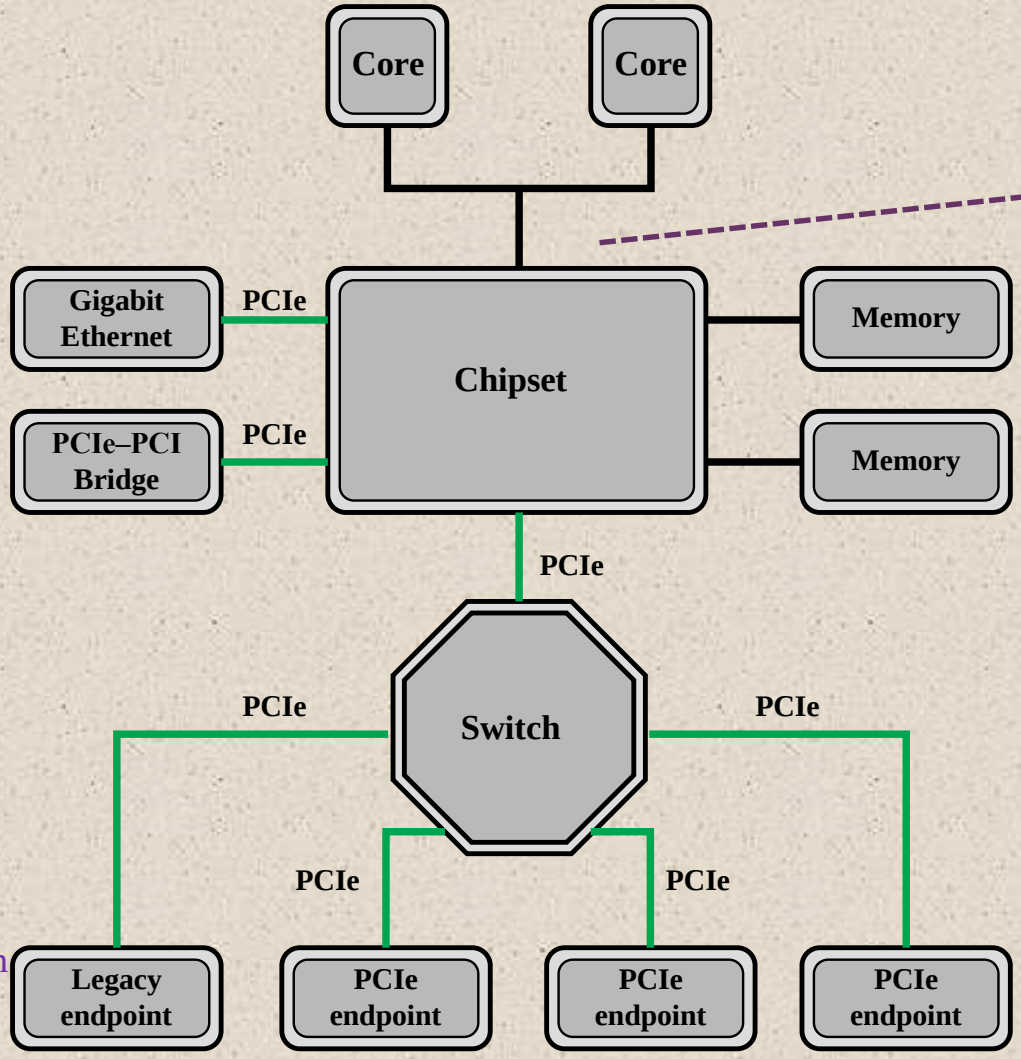
- Ara veya çevresel veri yolu olarak işlev görebilen popüler, yüksek bant genişliğine sahip, işlemciden bağımsız veri yolu
- Yüksek hızlı I/O alt sistemleri için daha iyi sistem performansı sunar

Önceki bölümlerde tartışılan sistem veriyolunda olduğu gibi, veri yolu tabanlı (bus-based) PCI düzeni, bağlı cihazların veri hızı taleplerine ayak uyduramadı. Dolayısıyla, PCI Express (PCIe) olarak bilinen yeni bir sürüm geliştirildi.

- PCI Express (PCIe)
  - PCI gibi veri yolu tabanlı (*bus-based*) düzenlerin yerini alması amaçlanan noktadan noktaya ara bağlantı düzeni (*point-to-point interconnect scheme*)
  - Temel gereksinim, Gigabit Ethernet gibi daha yüksek veri hızlı I/O cihazlarının ihtiyaçlarını desteklemek için yüksek kapasitedir *(high capacity*)
  - Diğer bir gereksinim, zamana bağlı veri akışlarını (*time-dependent data streams*) destekleme ihtiyacıyla ilgilenir

« Video-on-demand» ve «audio redistribution» gibi uygulamalar da sunuculara gerçek zamanlı kısıtlamalar getiriyor.

Chipset, tipik olarak, bazıları doğrudan bir PCIe cihazına bağlanan ve bir veya birden fazla PCIe akışını yöneten bir switch'e bağlanan birden çok PCIe bağlantı noktasını destekleyecektir.



A **root complex** device, also referred to as a *chipset* or a *host bridge*,

işlemci ve bellek alt sistemini, bir veya daha fazla PCIe ve PCIe switch aygıtını içeren PCI Express switch yapısına bağlar.

I/O denetleyicileri ile bellek ve işlemci bileşenleri arasındaki veri hızlarındaki farklılıkların üstesinden gelmek için bir arabellekleme aygıtı (*buffering device*) olarak işlev görür.

Switch : birden çok PCIe akışını yönetir.

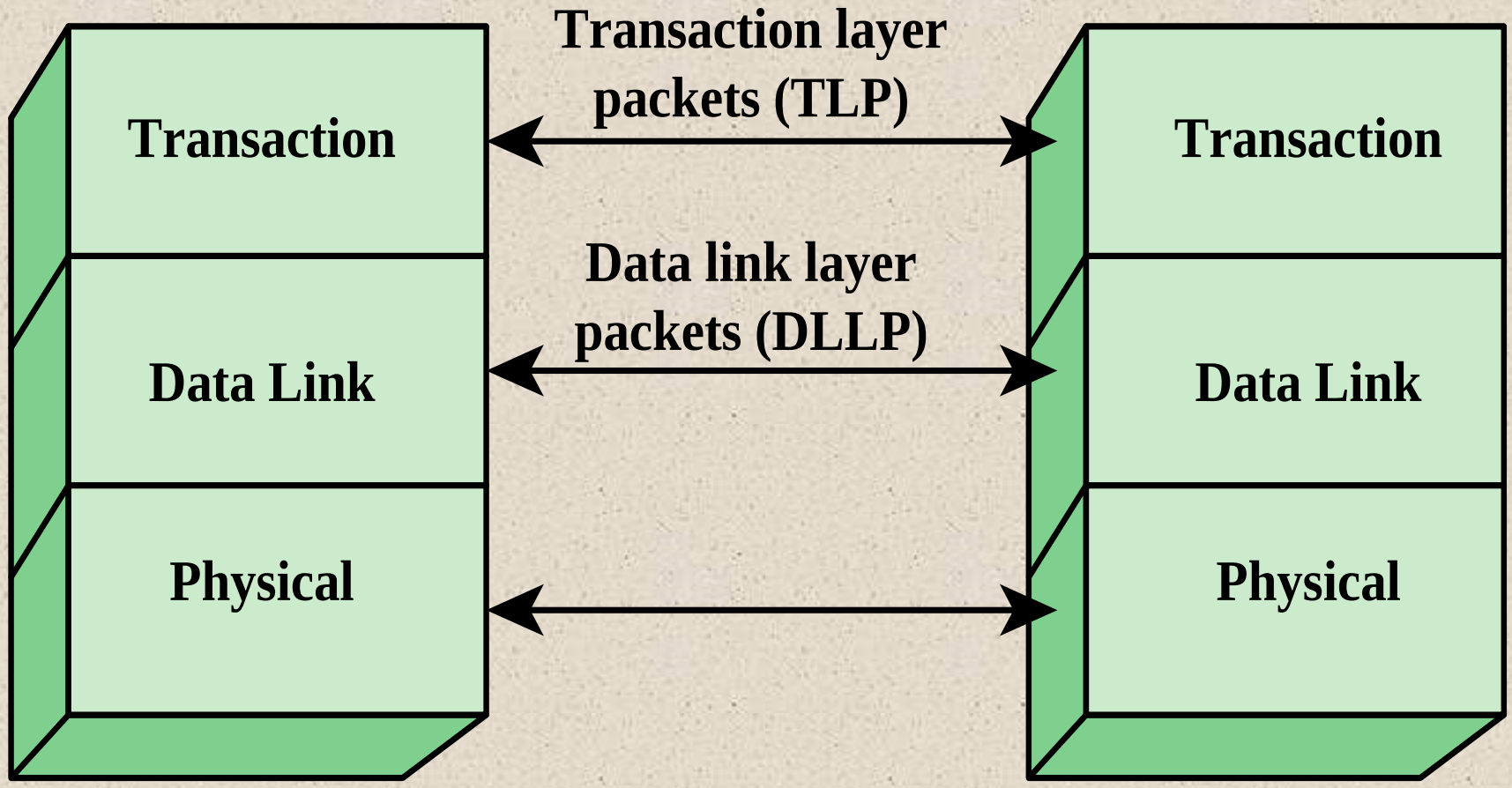
PCIe endpoint : Gigabit Ethernet switch, grafik veya video denetleyicisi, disk arabirimi veya iletişim denetleyicisi gibi PCIe uygulayan bir I/O aygıtı veya denetleyicisi.

Legacy endpoint: Eski uç nokta kategorisi, PCI Express'e taşınan mevcut tasarımlar için tasarlanmıştır ve I/O alanı kullanımı ve kilitli işlemler gibi eski davranışlara izin verir.

Figure 3.21 Typical Configuration Using PCIe

PCIe/PCI bridge: Eski PCI cihazlarının PCIe tabanlı sistemlere bağlanmasına izin verir.

QPI'da olduğu gibi, PCIe etkileşimleri bir protokol mimarisi kullanılarak tanımlanır.



Physical : Sinyalleri taşıyan gerçek kabloların yanı sıra 1'lerin ve 0'ların iletimi ve alınmasında gerekli olan yardımcı özellikleri desteklemek için devre ve mantıktan oluşur.

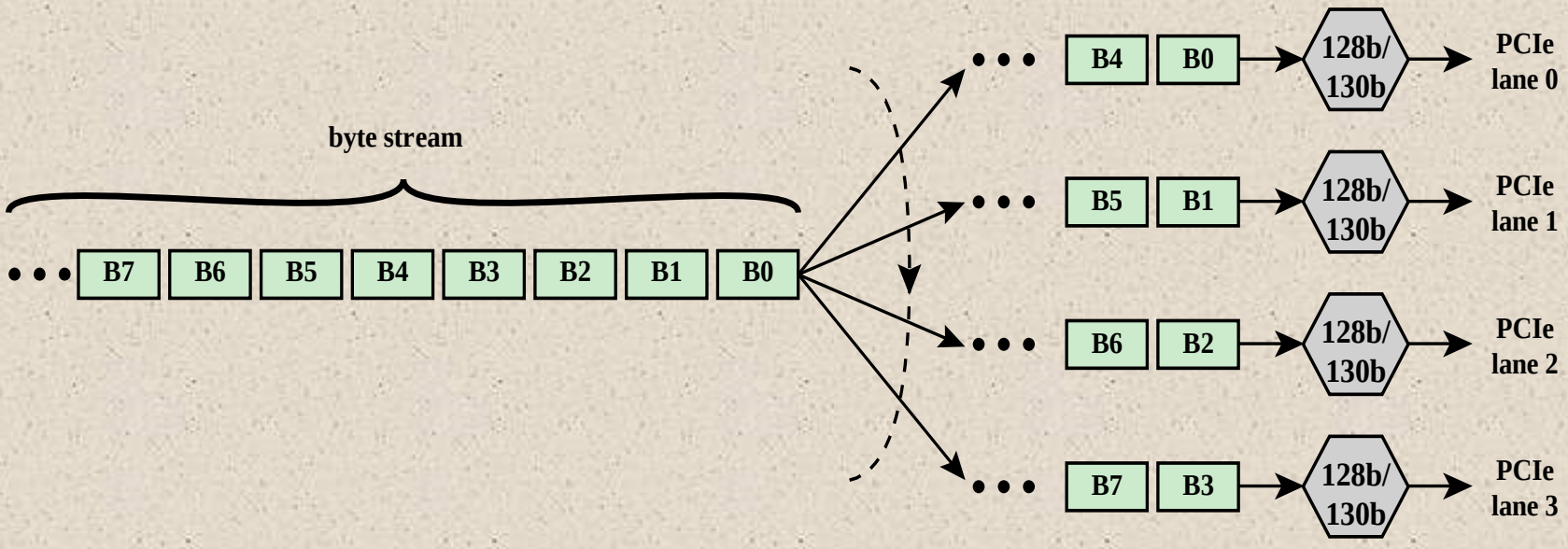
Data link: Güvenilir iletim ve akış kontrolünden sorumludur. DLL tarafından oluşturulan ve tüketilen veri paketlerine Veri Bağlantısı Katmanı Paketleri (DLLP'ler) denir.

Figure 3.22 PCIe Protocol Layers

Transaction : Veri aktarım mekanizmalarını yüklemek/depolamak için kullanılan veri paketlerini üretir ve tüketir ve ayrıca bir bağlantı üzerindeki iki bileşen arasındaki bu paketlerin akış kontrolünü yönetir. TL tarafından üretilen ve tüketilen veri paketlerine İşlem Katmanı Paketleri (TLP'ler) denir.

QPI'ye benzer şekilde, PCIe noktadan noktaya bir mimaridir. QPI'da olduğu gibi, PCIe çok şeritli bir dağıtım tekniği kullanır.

Veriler, basit bir Round Robin düzeni kullanılarak bir seferde 1 baytlık dört şeride dağıtılır



Her fiziksel şeritte, veriler arabelleğe alınır ve bir seferde 16 bayt (128 bit) işlenir. 128 bitlik her bir blok, iletim için benzersiz bir 130 bitlik kod sözcüğüne kodlanır; buna 128b/130b kodlama (*encoding*) denir. Böylelikle, tek bir şeridin etkin veri hızı 128/130 kat azaltılır.

Figure 3.23 PCIe Multilane Distribution

İletilecek bit sayısını artırmayan karıştırma (scrambling), verilerin daha rastgele görünmesini sağlayan bir eşleme tekniğidir. Karıştırma, alıcıda daha düzgün aralıklı görünmeleri için geçişlerin sayısını yayma eğilimindedir, bu da senkronizasyon için iyidir.

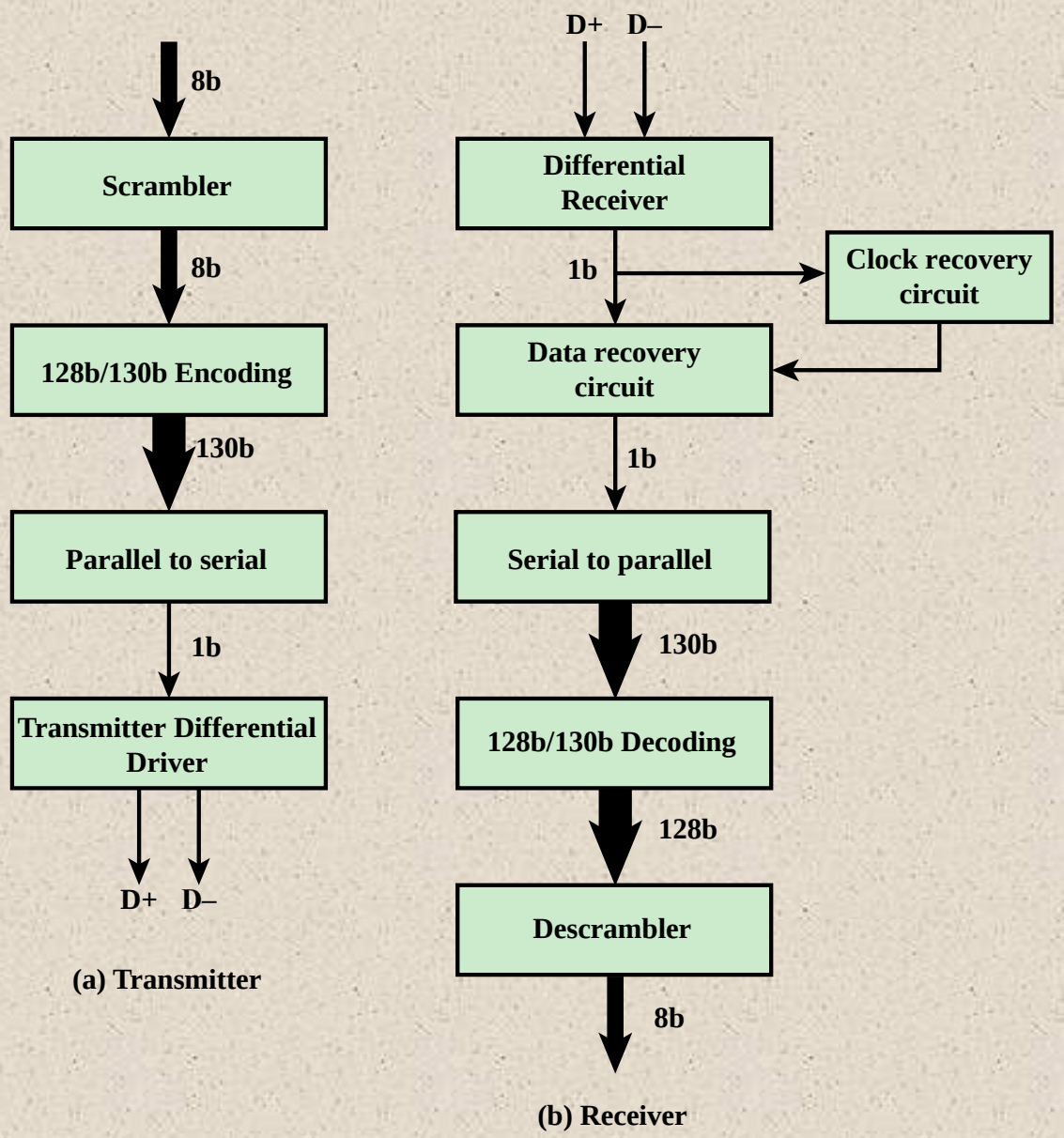


Figure 3.24 PCIe Transmit and Receive Block Diagrams

PCIe

Transaction Layer (TL)

- ***Transaction Layer (TL)*** üzerindeki yazılımdan okuma ve yazma isteklerini alır ve bağlantı katmanı (*datalink layer*) aracılığıyla bir hedefe aktarım için istek paketleri oluşturur
- Çoğu transaction bir bölünmüş transaction (*split transaction*) tekniği kullanır
  - Bir kaynak PCIe cihazı tarafından bir istek paketi gönderilir ve daha sonra tamamlama paketi adı verilen bir yanıtı bekler.
- ***TL*** mesajları ve bazı yazma işlemleri bilgilendirme/postalanmış işlemlerdir (yani yanıt beklenmez)
- ***TL*** paket formatı, 32 bit bellek adreslemeyi ve genişletilmiş 64 bit bellek adreslemeyi destekler

The TL supports four address spaces:

- Memory
  - Bellek alanı, sistem ana belleğini ve PCIe I/O aygıtlarını içerir
  - Belirli bellek adres aralıkları I/O cihazlarıyla eşleşir
- Configuration
  - Bu adres alanı, TL'nin I/O cihazlarıyla ilişkili konfigürasyon kayıtlarını okumasını/yazmasını sağlar.
- I/O
  - Bu adres alanı, eski I/O aygıtlarını adreslemek için kullanılan ayrılmış adres aralıklarıyla eski PCI aygıtları (*legacy PCI devices*) için kullanılır.
- Message
  - Bu adres alanı; kesme, hata işleme ve güç yönetimi ile ilgili kontrol sinyalleri içindir.

Table 3.2
PCIe TLP Transaction Types

Address Space	TLP Type	Purpose
Memory	Memory Read Request	Transfer data to or from a location in the system memory map.
	Memory Read Lock Request	
	Memory Write Request	
I/O	I/O Read Request	Transfer data to or from a location in the system memory map for legacy devices.
	I/O Write Request	
Configuration	Config Type 0 Read Request	Transfer data to or from a location in the configuration space of a PCIe device.
	Config Type 0 Write Request	
	Config Type 1 Read Request	
	Config Type 1 Write Request	
Message	Message Request	Provides in-band messaging and event reporting.
	Message Request with Data	
Memory, I/O, Configuration	Completion	Returned for certain requests.
	Completion with Data	
	Completion Locked	
	Completion Locked with Data	

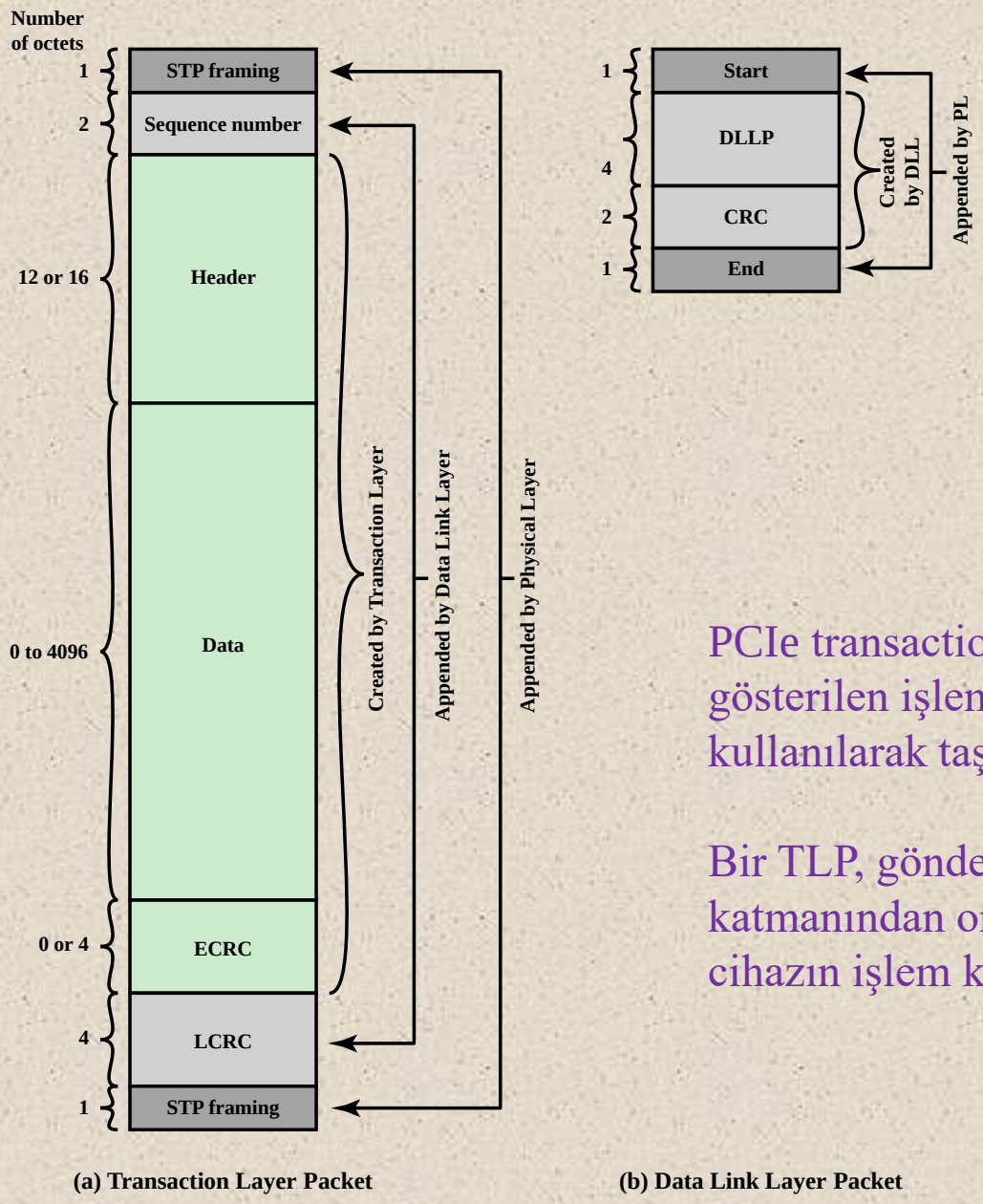


Figure 3.25 PCIe Protocol Data Unit Format

PCIe transaction'ları, Şekil 3.25a'da gösterilen işlem katmanı paketleri kullanılarak taşınır.

Bir TLP, gönderici cihazın işlem katmanından ortaya çıkar ve alıcı cihazın işlem katmanında sona erer.

Summary

Chapter 3

A Top-Level View of Computer Function and Interconnection

- Computer components
- Computer function
  - Instruction fetch and execute
  - Interrupts
  - I/O function
- Interconnection structures
- Bus interconnection
- Point-to-point interconnect
  - QPI physical layer
  - QPI link layer
  - QPI routing layer
  - QPI protocol layer
- PCI express
  - PCI physical and logical architecture
  - PCIe physical layer
  - PCIe transaction layer
  - PCIe data link layer