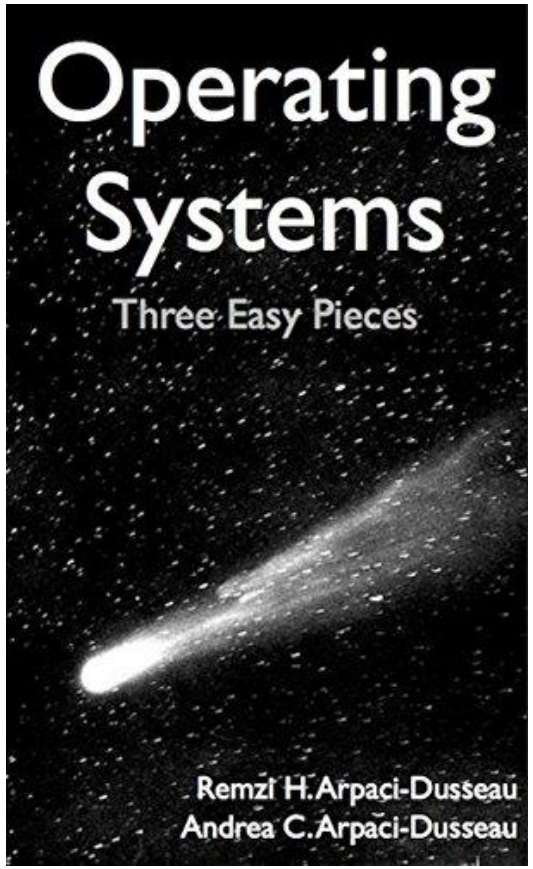
İşletim Sistemleri



10. Ders

Prof. Dr. Kemal Bıçakcı

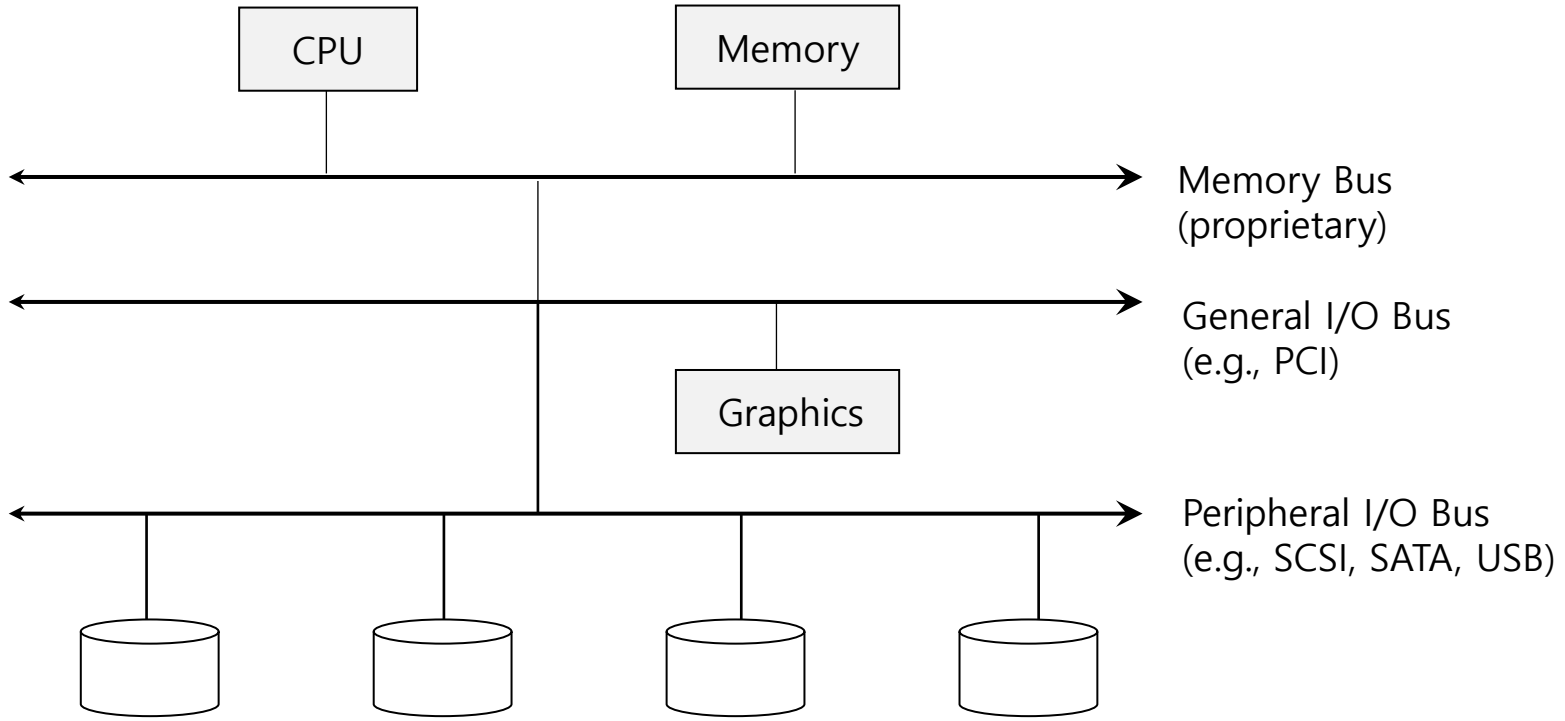
36. I/O Cihazları

Operating System: Three Easy Pieces

I/O Cihazları

- I/O (Girdi/Çıktı - G/Ç), bilgisayar sistemleri için kritik öneme sahiptir.
- Bilgisayar sistemleri ve yazdığımız programların anlam ifade edebilmesi için hem girdi hem de çıktı gereklidir.
- Sorunlar:
  - I/O, sistemlere nasıl entegre edilmelidir?
  - Genel mekanizmalar nelerdir?
  - Nasıl verimli hale getirebiliriz?

Sistem Mimarisi



Prototip bir Sistem Mimarisi

CPU, sistemin ana belleğine bir tür bellek veriyolu aracılığıyla bağlanır.

Bazı cihazlar sisteme genel bir G/Ç veri yolu üzerinden bağlanır.

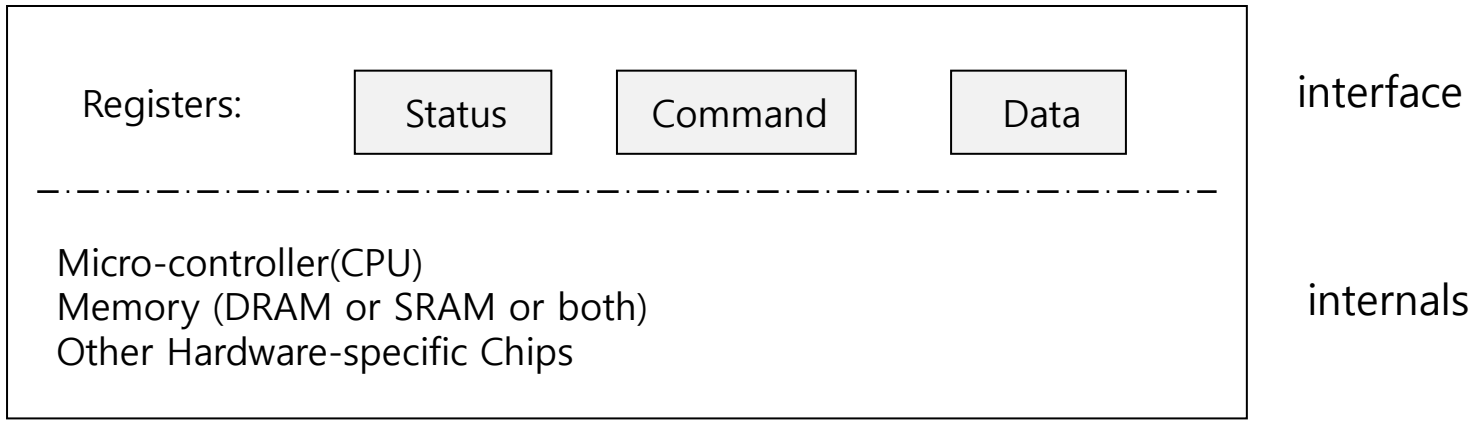
Daha da aşağıda, çevresel (peripheral) veri yolu dediğimiz bir veya daha fazla yol vardır.

I/O Mimarisi

- Veri yolları:
  - CPU(lar), RAM ve I/O cihazları arasında veri akışı sağlayan yollar.
- G/Ç veriyolu:
  - CPU'yu G/Ç aygıtına bağlayan veri yolu.
- G/Ç veri yolu, G/Ç aygıtına üç donanım bileşeniyle bağlanır:
  - G/Ç bağlantı noktaları, arayüzler ve cihaz denetleyicileri (device controllers).

Standart (Kanonik) bir Cihaz

- Standart bir I/O cihazının iki önemli bileşeni vardır:
  - Donanım arayüzü: sistem yazılımının cihazın çalışmasını kontrol etmesini sağlar.
  - İç öğeler: Cihaza özgü kısım.



Canonical Device

Donanım Arayüzü

- **Durum yazmacı (status register)**
  - Cihazın mevcut durumunu görmek için
- **Komut yazmacı (command register)**
  - Cihaza belirli bir görevi gerçekleştirmesini söylemek için
- **Veri yazmacı (data register)**
  - Cihaza veri aktarmak veya cihazdan veri almak için

Üç yazmaçta okuma ve yazma işlemleri yaparak işletim sistemi cihaz davranışını denetleyebilir.

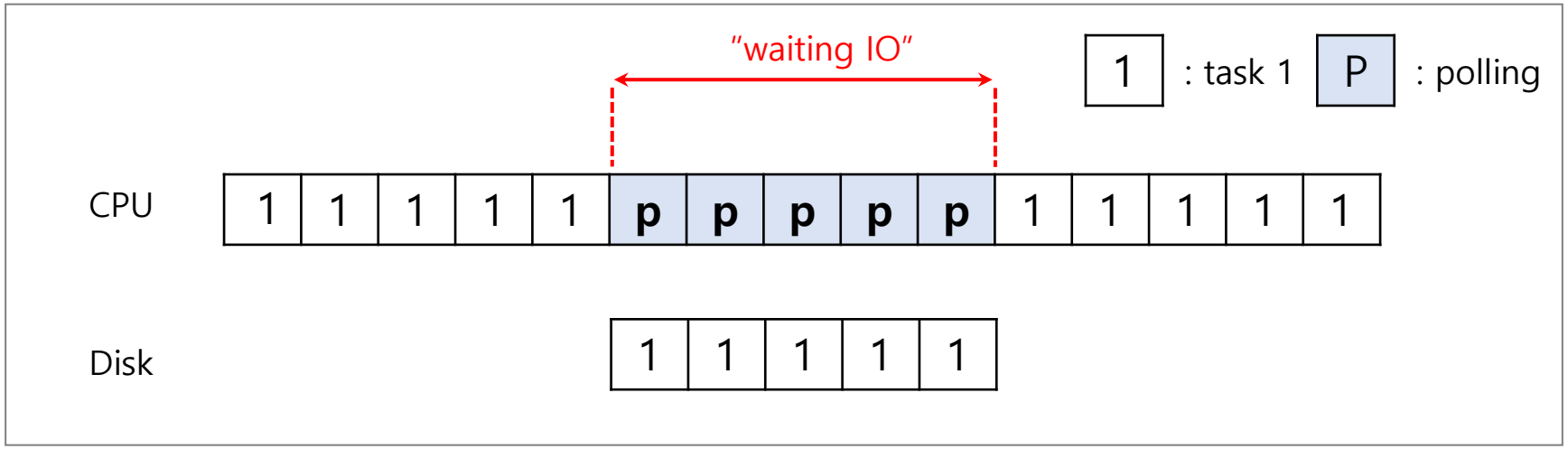
Donanım Arayüzü (Devam)

Tipik bir etkileşim örneği:

```
while ( STATUS == BUSY)
    ; //wait until device is not busy
write data to data register
write command to command register
    Doing so starts the device and executes the command
while ( STATUS == BUSY)
    ; //wait until device is done with your request
```

Yoklama (Polling)

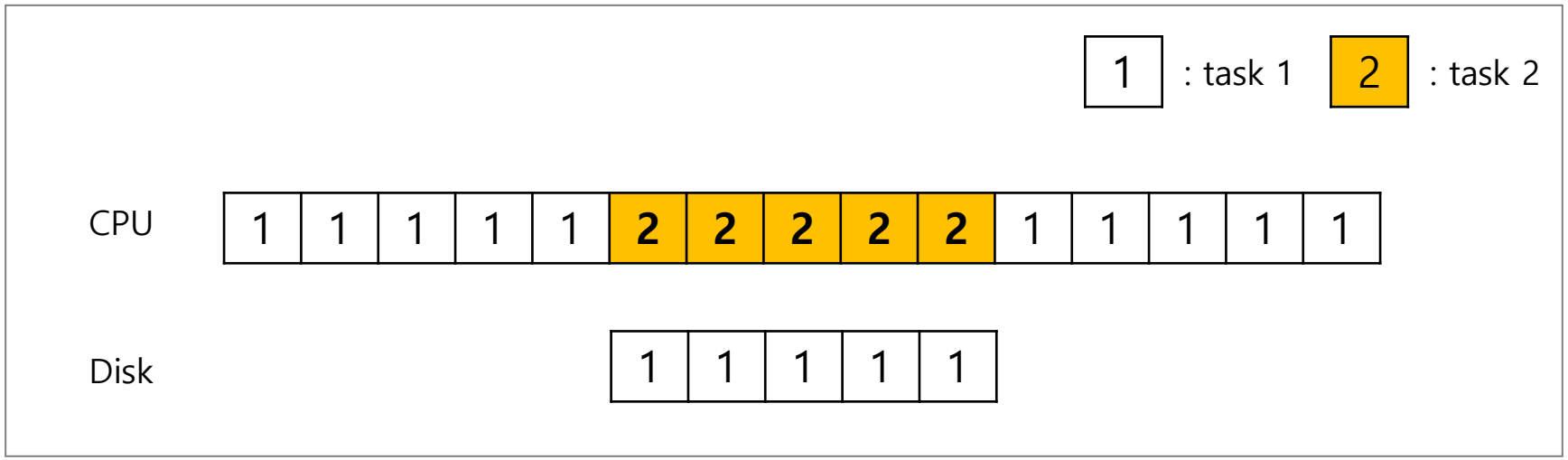
- İşletim sistemi, durum yazmacını tekrar tekrar okuyarak cihazın hazır olmasını bekler.
  - Yöntemin artısı: basittir ve çalışır.
  - Yöntemin eksisi: Ancak, cihazı beklerken CPU zamanını boşa harcar. Bu açıdan başka bir hazır işleme geçmek daha iyi olabilir.



Yoklamalı durumda CPU kullanım şeması

Kesmeler (Interrupts)

- G/Ç isteği yapan işlemi uyku moduna geçir ve «context switch» ile bir başka işleme geç.
- Cihaz ilgili görevi bitirdiğinde I/O bekleyen işlemi kesme ile uyandır.
  - Olumlu yön: CPU ve diskin verimli şekilde kullanılması mümkün olur.



Kesmeli durumda CPU kullanım şeması

Kesme Adımları

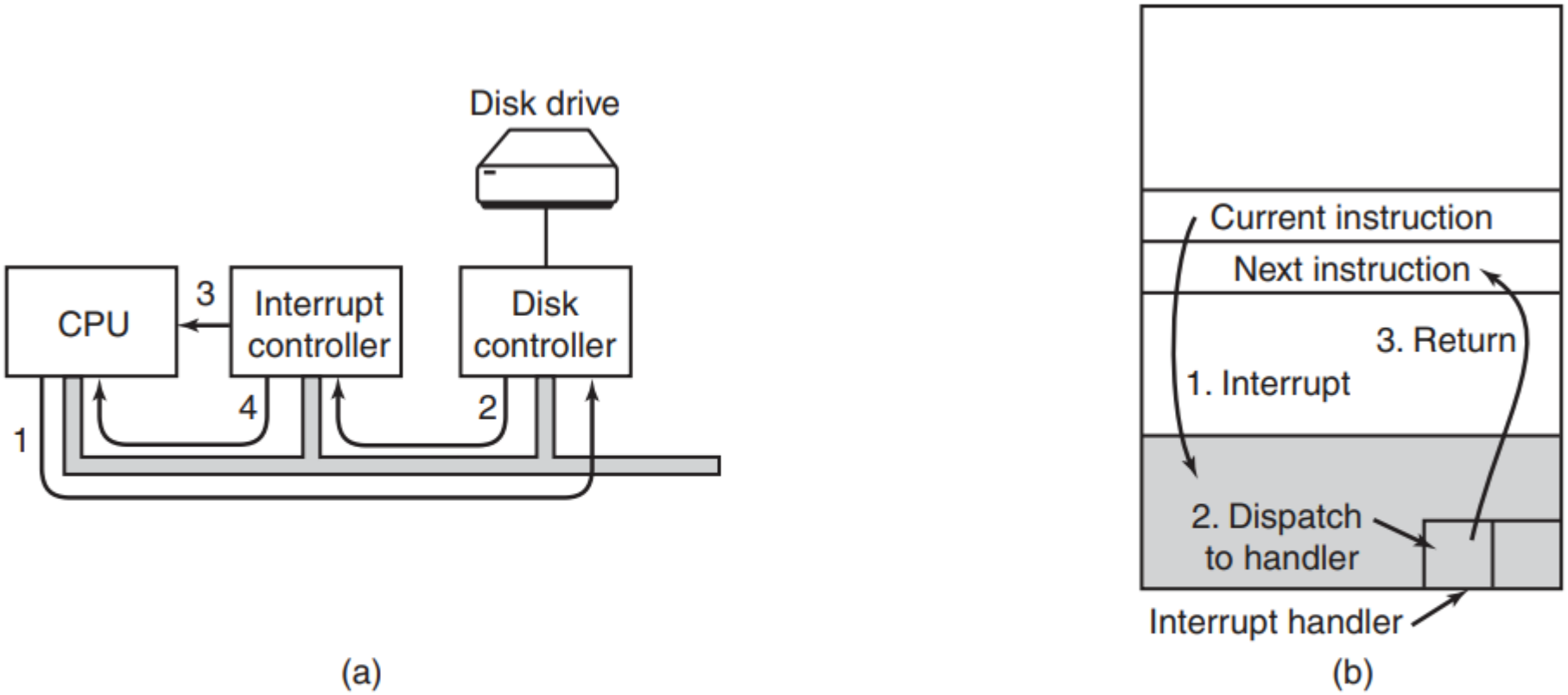


Figure 1-11. (a) The steps in starting an I/O device and getting an interrupt. (b) Interrupt processing involves taking the interrupt, running the interrupt handler, and returning to the user program.

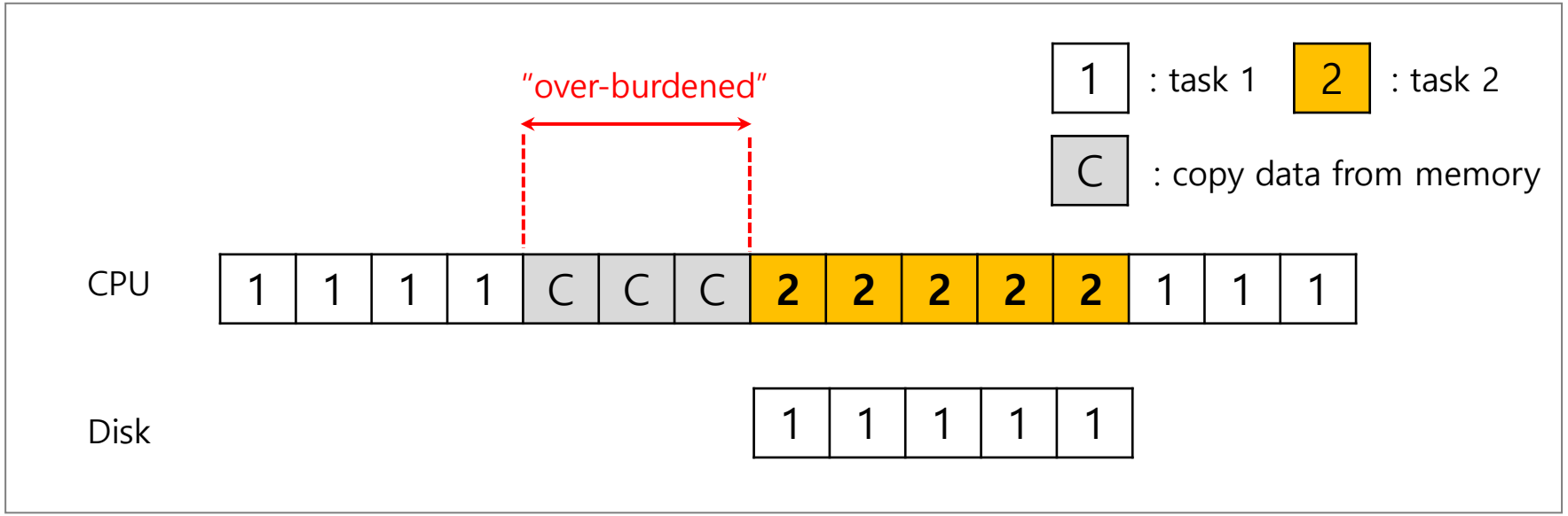
Yoklama veya Kesme Tercihi

- Ancak, kesme, her zaman en iyi çözüm değildir.
- Cihaz çok hızlı çalışıyorsa, kesme, sistemi yavaşlatır.
  - «Context switch» ile başka bir işleme geçiş maliyetli olduğu için

Hızlı cihaz → yoklama (polling) daha iyi
Yavaş cihaz → kesme (interrupt) daha iyi

Başka bir sorun daha var!

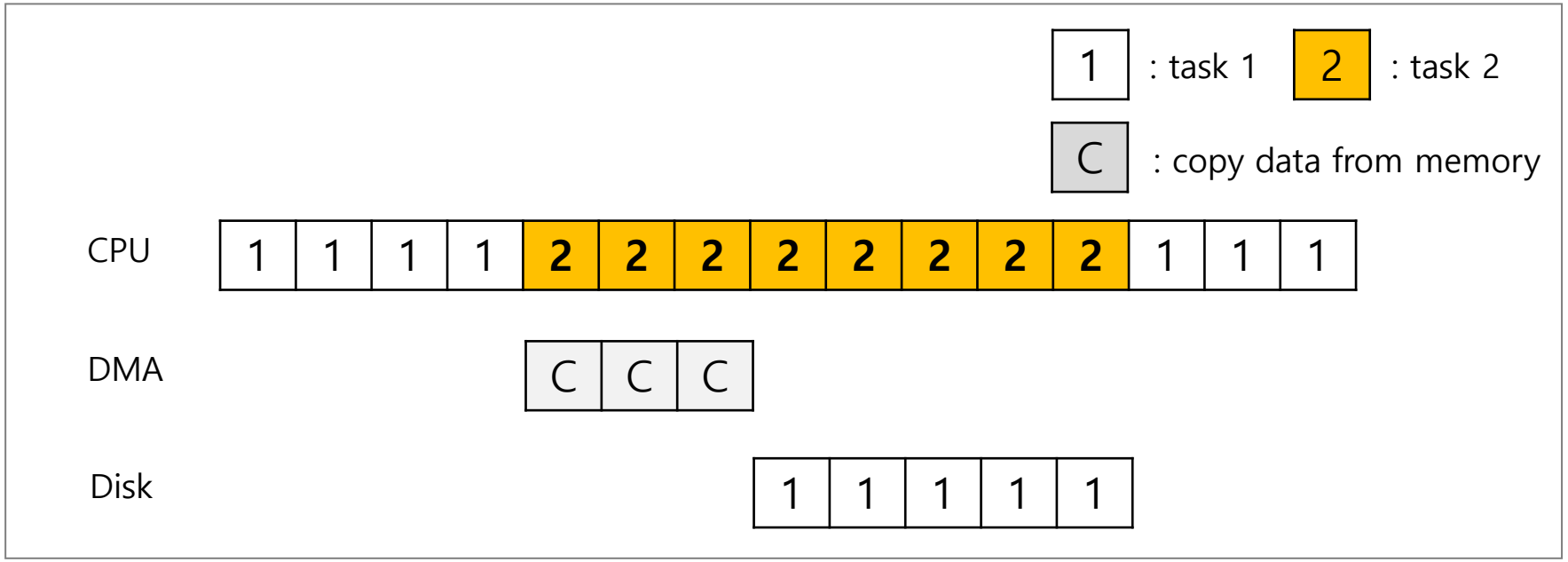
- CPU, büyük miktardaki veriyi bellekten cihaza kopyalamak için de çok zaman harcar.



CPU kullanım şeması

DMA (Direct Memory Access)

- Verilerin bellekte nerede tutulduğunu, ne kadar veri kopyalanacağını bilerek, fazla CPU müdahalesi olmadan bellekteki verileri kopyalama işlemi.
- Tamamlandığında, DMA bir kesme başlatır, Diskte G/Ç başlar.



DMA ile CPU kullanım şeması

Cihaz Etkileşimi

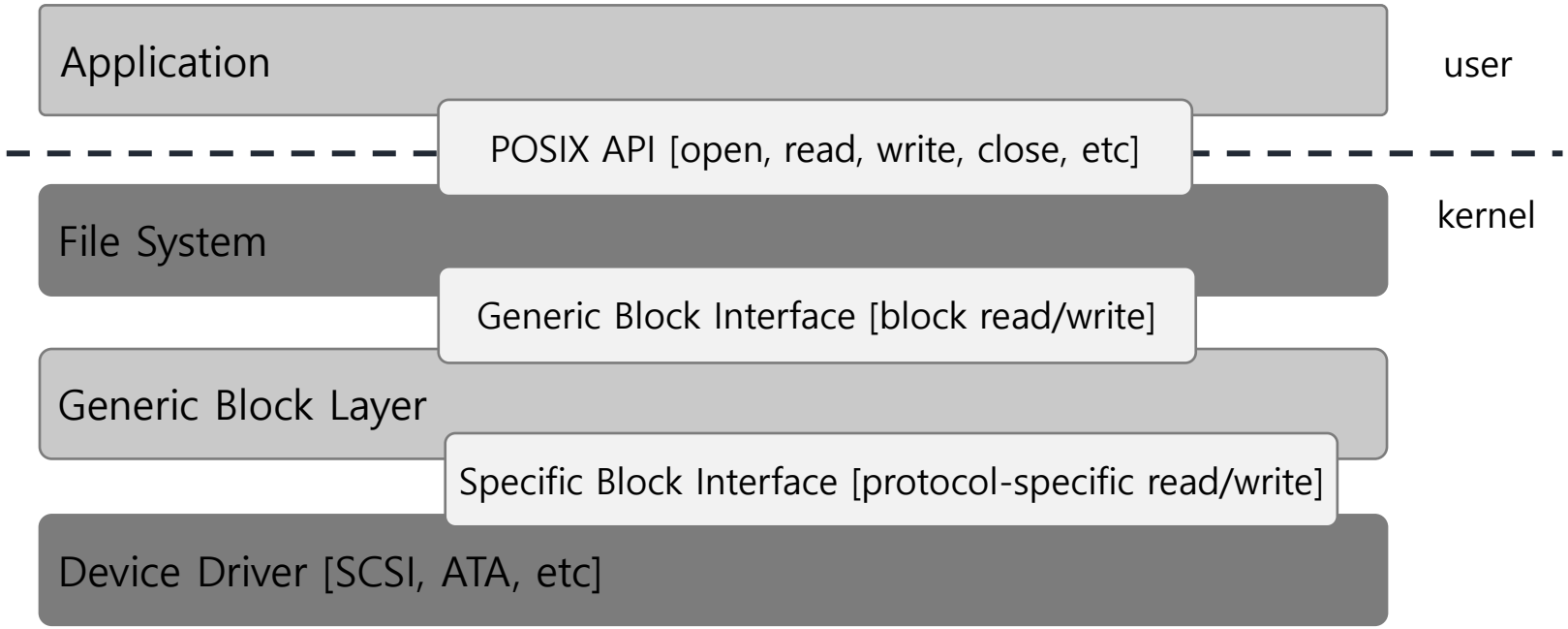
- İşletim sistemi cihazla nasıl iletişim kurar?
- Çözümler:
  - I/O buyrukları: işletim sisteminin belirli cihaz yazmaçlarına veri göndermesi için özel bir yol.
    - Örnek: x86'da `in` ve `out` buyrukları
  - Bellek eşlemeli G/Ç (memory-mapped I/O)
    - Cihaz yazmaçları, bellek konumlarıymış gibi kullanılabilir.
    - İşletim Sistemi, `load` (cihazdan okumak için) veya `store` (cihaza yazmak için) buyruklarını ana bellekte olduğu gibi kullanabilir.

Cihaz Etkileşimi (Devam)

- İşletim sistemi farklı arayüzlerle nasıl etkileşime girer?
  - Örneğin, SCSI diskleri, IDE diskleri, USB sürücüleri vb. üzerinde çalışabilen bir dosya sistemi oluşturmak istediğimizi düşünelim.
  - Çözümler: Soyutlama (Abstraction)
- Soyutlama, cihaz etkileşiminin tüm ayrıntılarını saklar (encapsulation).

Dosya Sistemi Soyutlaması

- Dosya sistemi, kullandığı disk sınıfının özelliklerinden tamamen habersizdir.
- Sadece, genel blok katmanına blok okuma ve yazma isteği gönderir.



Dosya Sistemi Yığını

Soyutlamanın Problemi

- Özel yeteneklere sahip bir cihaz varsa, bu yetenekler genel blok arayüzü katmanında kaybolur, dolayısıyla kullanılmaz.

Cihaz Sürücüleri

- İşletim sistemi kodunun %70'inden fazlası cihaz sürücülerinde bulunur.
- Sisteminize takabileceğiniz herhangi I/O cihaz için bir cihaz sürücüsü gereklidir.
- Cihaz Sürücüleri, çekirdek çökmelerine (kernel crash) birincil katkıda bulunurlar ve daha fazla hata (bug) içerirler.

Basit Bir IDE Disk Sürücüsü

- Dört tür yazmaç:
  - Kontrol, komut bloğu, durum ve hata
- `in` ve `out` I/O buyrukları

Basit Bir IDE Disk Sürücüsü (Devam)

- Kontrol Yazmacı (Control Register):

  Address 0x3F6 = 0x80 (0000 1RE0): R=reset, E=0 means "enable interrupt”

- Komut Bloğu Yazmaçları (Command Block Registers):

  Address 0x1F0 = Data Port
  Address 0x1F1 = Error
  Address 0x1F2 = Sector Count
  Address 0x1F3 = LBA low byte
  Address 0x1F4 = LBA mid byte
  Address 0x1F5 = LBA hi byte
  Address 0x1F6 = 1B1D TOP4LBA: B=LBA, D=drive
  Address 0x1F7 = Command/status

Basit Bir IDE Disk Sürücüsü (Devam)

- Durum Yazmacı (Status Register) (Address 0x1F7):

7	6	5	4	3	2	1	0
BUSY	READY	FAULT	SEEK	DRQ	CORR	IDDEX	ERROR

- Hata Yazmacı (Error Register) (Address 0x1F1): (check when Status ERROR==1):

7	6	5	4	3	2	1	0
BBK	UNC	MC	IDNF	MCR	ABRT	T0NF	AMNF

  - BBK = Bad Block
  - UNC = Uncorrectable data error
  - MC = Media Changed
  - IDNF = ID mark Not Found
  - MCR = Media Change Requested
  - ABRT = Command aborted
  - T0NF = Track 0 Not Found
  - AMNF = Address Mark Not Found

Basit Bir IDE Disk Sürücüsü (Devam)

- **Sürücünün hazır olmasını bekleyin**. Sürücü meşgul ise HAZIR olana kadar Durum Yazmacını (0x1F7) okuyun.
- **Komut yazmaçlarına parametreleri yazın**. Komut yazmaçlarına (0x1F2-0x1F6) sektör sayısını, erişilecek sektörlerin mantıksal blok adresini (LBA) ve sürücü numarasını (master=0x00 veya slave=0x10, IDE yalnızca iki sürücüye izin verdiği için) yazın.
- **G/Ç'yi başlatın**. komut yazmacına (0x1F7) okuma/yazma (READ —WRITE) komutunu yazarak.
- **Veri aktarımı (yazma için)**: Sürücü durumu HAZIR ve DRQ (sürücünün veri talebi) olana kadar bekleyin; veri portuna veri yazın.
- **Kesmeleri işleyin**. En basit durumda, aktarılan her sektör için bir kesme gerçekleştirin; daha karmaşık yaklaşımlar gruplamaya ve dolayısıyla tüm transfer tamamlandığında son nihai bir kesmeye izin verir.
- **Hata yönetimi**. Her işlemden sonra durum yazmacını okuyun. ERROR biti 1 ise, ayrıntılar için hata yazmacını okuyun.

Basit Bir IDE Disk Sürücüsü (xv6 İşletim Sistemi – Temel Fonksiyonlar)

```
void ide_rw(struct buf *b) {}
static void ide_start_request(struct buf *b) {}
static int ide_wait_ready() {}
void ide_intr() {}
```

Basit Bir IDE Disk Sürücüsü (xv6)

```
void ide_rw(struct buf *b) {
   acquire(&ide_lock);
   for (struct buf **pp = &ide_queue; *pp; pp=&(*pp)>qnext)
   ; // walk queue
   *pp = b; // add request to end
   if (ide_queue == b) // if q is empty
      ide_start_request(b); // send req to disk
   while ((b->flags & (B_VALID|B_DIRTY)) != B_VALID)
      sleep(b, &ide_lock); // wait for completion
   release(&ide_lock);
}
```

Basit Bir IDE Disk Sürücüsü (xv6)

```
static void ide_start_request(struct buf *b) {
  ide_wait_ready();
  outb(0x3f6, 0); // generate interrupt
  outb(0x1f2, 1); // how many sectors?
  outb(0x1f3, b->sector & 0xff); // LBA goes here ...
  outb(0x1f4, (b->sector >> 8) & 0xff); // ... and here
  outb(0x1f5, (b->sector >> 16) & 0xff); // ... and
  here!
  outb(0x1f6, 0xe0 | ((b->dev&1)<<4) | ((b-
  >sector>>24)&0x0f));
  if(b->flags & B_DIRTY){
    outb(0x1f7, IDE_CMD_WRITE); // this is a WRITE
    outsl(0x1f0, b->data, 512/4); // transfer data too!
  } else {
    outb(0x1f7, IDE_CMD_READ); // this is a READ (no
    data)
  }
```

Basit Bir IDE Disk Sürücüsü (xv6)

```
static int ide_wait_ready() {
      while (((int r = inb(0x1f7)) & IDE_BSY) ||
            !(r & IDE_DRDY))
      ; // loop until drive isn't busy
}
```

A Simple IDE Disk Driver (xv6)

```
void ide_intr() {
    struct buf *b;
    acquire(&ide_lock);
    if (!(b->flags & B_DIRTY) && ide_wait_ready(1) >= 0)
        insl(0x1f0, b->data, 512/4); // if READ: get data
    b->flags |= B_VALID;
    b->flags &= ~B_DIRTY;
    wakeup(b); // wake waiting process
    if ((ide_queue = b->qnext) != 0) // start next request
        ide_start_request(ide_queue); // (if one exists)
    release(&ide_lock);
}
```

Açıklamalar

- `ide_rw` bir isteği kuyruğa alır (bekleyen başka istekler varsa) veya doğrudan diske gönderir (`ide_start_request` kullanarak).
- Her iki durumda da rutin, isteğin tamamlanmasını bekler ve çağrıyı yapan işlem uyku moduna alınır.
- `ide_start_request` diske bir istek (ve yazma durumunda veri) göndermek için kullanılır; giriş ve çıkış x86 buyrukları, sırasıyla cihaz yazmaçlarını okumak ve yazmak için çağrılır.
- `ide_start_request`, bir istekte bulunmadan önce sürücünün hazır olduğundan emin olmak için üçüncü bir fonksiyon olan `ide_wait_ready`'yi kullanır.
- En sonunda, `ide_intr()` bir kesme gerçekleştiğinde çağrılır; cihazdan veri okur (istek yazma değil okuma ise), G/Ç'nin tamamlanmasını bekleyerek işlemi uyandırır ve (G/Ç kuyruğunda daha fazla istek varsa), sonraki I/O'yu `ide_start_request` aracılığıyla başlatır.

37. Sabit Disk Sürücüleri (Hard Disk Drives)

Operating System: Three Easy Pieces

Sabit Disk Sürücüleri (HDD)

- Sabit disk sürücüleri, onlarca yıldır bilgisayar sistemlerinde kalıcı veri depolamanın en yaygın tercihi olmuştur.
  - HDD mi SSD mi?

- Sürücü, çok sayıda sektörden (512 baytlık bloklar) oluşur.
- Adres Uzayı:
  - n sektörlü bir diski 0 - n-1 aralığında bir sektör dizisi olarak düşünebiliriz.

SSD vs HDD

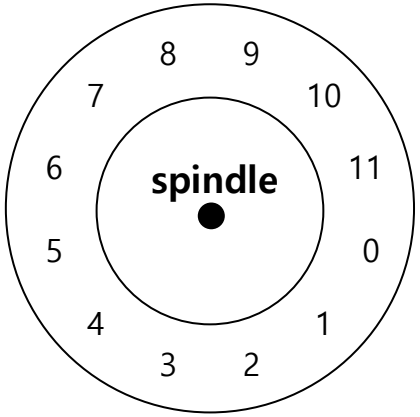
SSD		HDD
faster	✓ ✗	slower
shorter lifespan	✗ ✓	longer lifespan
more expensive	✗ ✓	cheaper
non-mechanical (flash)	✓ ✗	mechanical (moving parts)
shock-resistant	✓ ✗	fragile
best for storing operating systems, gaming apps, and frequently used files		best for storing extra data, such as movies, photos, and documents

Sürücü Arayüzü

- Verilen tek garanti, 512 baytlık tek bir yazmanın atomik olmasıdır.
- Çok sektörlü operasyonlar mümkündür. Birçok dosya sistemi aynı anda 4 KB okur veya yazar.
- Eksik yarım kalan yazma: Zamansız bir güç kaybı meydana gelirse, daha büyük bir yazmanın yalnızca bir kısmı tamamlanabilir.

- Bitişik bir yığındaki bloklara erişim, en hızlı erişim modudur.
  - Sıralı okuma veya yazma, herhangi bir rastgele erişim modelinden çok daha hızlıdır.

Temel Geometri

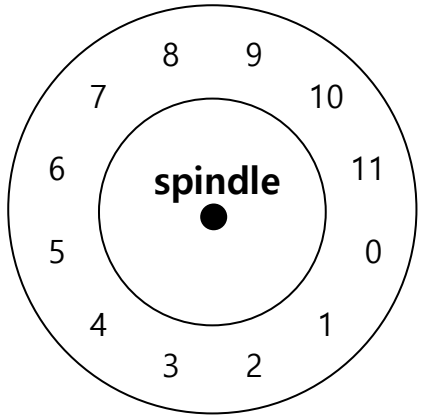


Tek izli (Track) Bir Disk (12 sektör)

- Plaka (Platter) (İnce manyetik tabaka ile kaplanmış alüminyum)
  - Dairesel sert bir yüzey
  - Veriler, üzerinde manyetik değişiklikler indüklenerek kalıcı olarak saklanır.
  - Her tablanın yüzey (surface) adı verilen iki tarafı vardır.

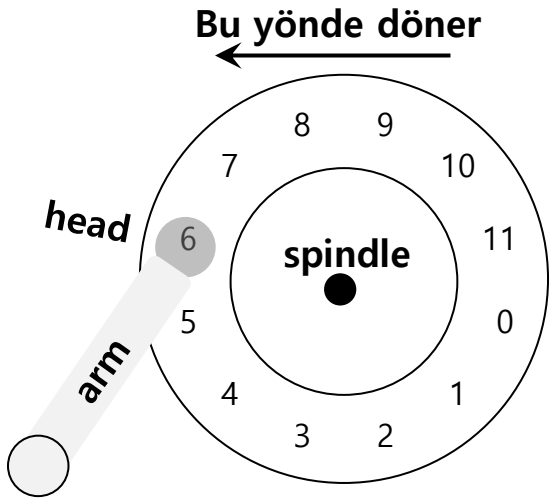
Temel Geometri (Devam)

- Mil (Spindle), plakaları etrafında döndüren bir motora bağlıdır.
- Dönme hızı, RPM (Dakika Başına Dönme) cinsinden ölçülür.
  - Tipik modern değerler: 7.200 RPM - 15.000 RPM.
  - Örneğin, 10000 RPM: Tek bir dönüş yaklaşık 6 ms sürer.
- İz (Track), sektörlerin eşmerkezli daireleri.
  - Veriler, bir izin her iki yüzeyinde de kodlanır.
  - Tek bir yüzey binlerce iz içerebilir.



Tek İzli (Track) Bir Disk (12 sektör)

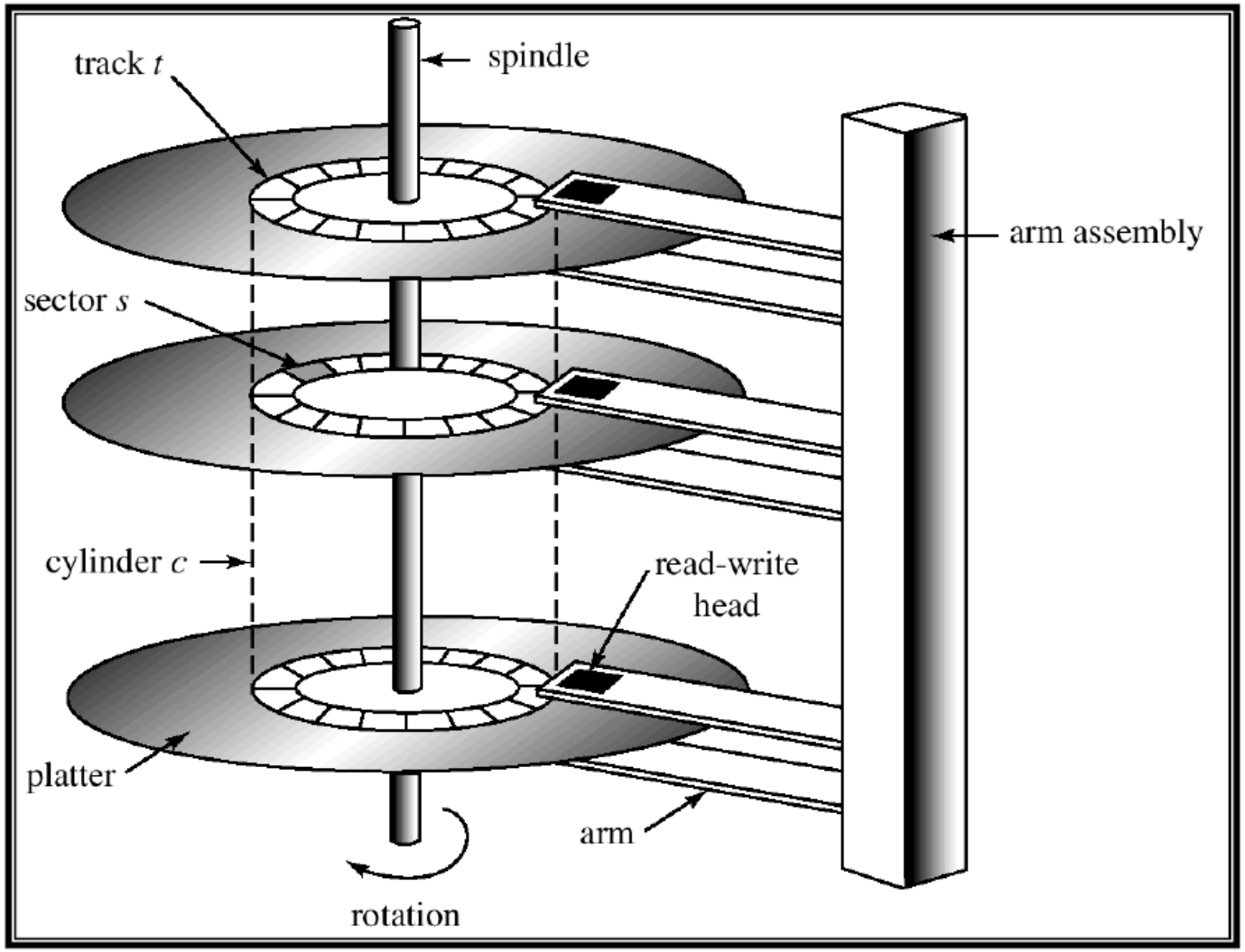
Basit bir Disk Sürücü



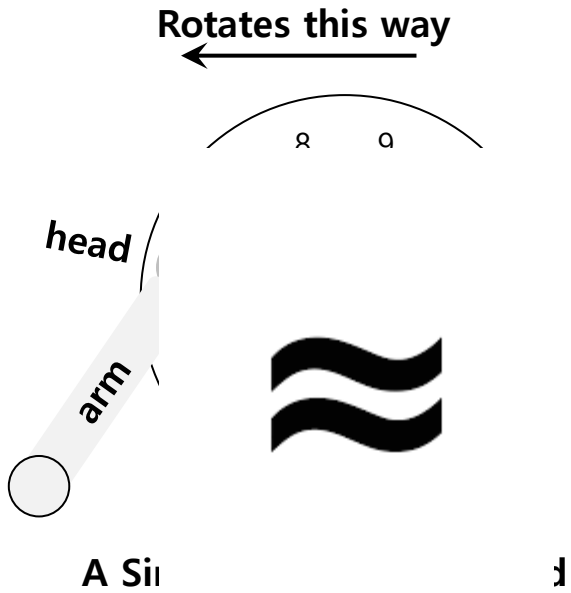
Kafa (Head) ve Tek Bir İz

- **Disk kafası** (Her bir yüzey için ayrı bir kafa bulunur)
  - Okuma ve yazma işlemi disk kafası tarafından gerçekleştirilir.
  - Kafa, yüzey boyunca hareket eden bir disk koluna bağlıdır.

Disk Örneği

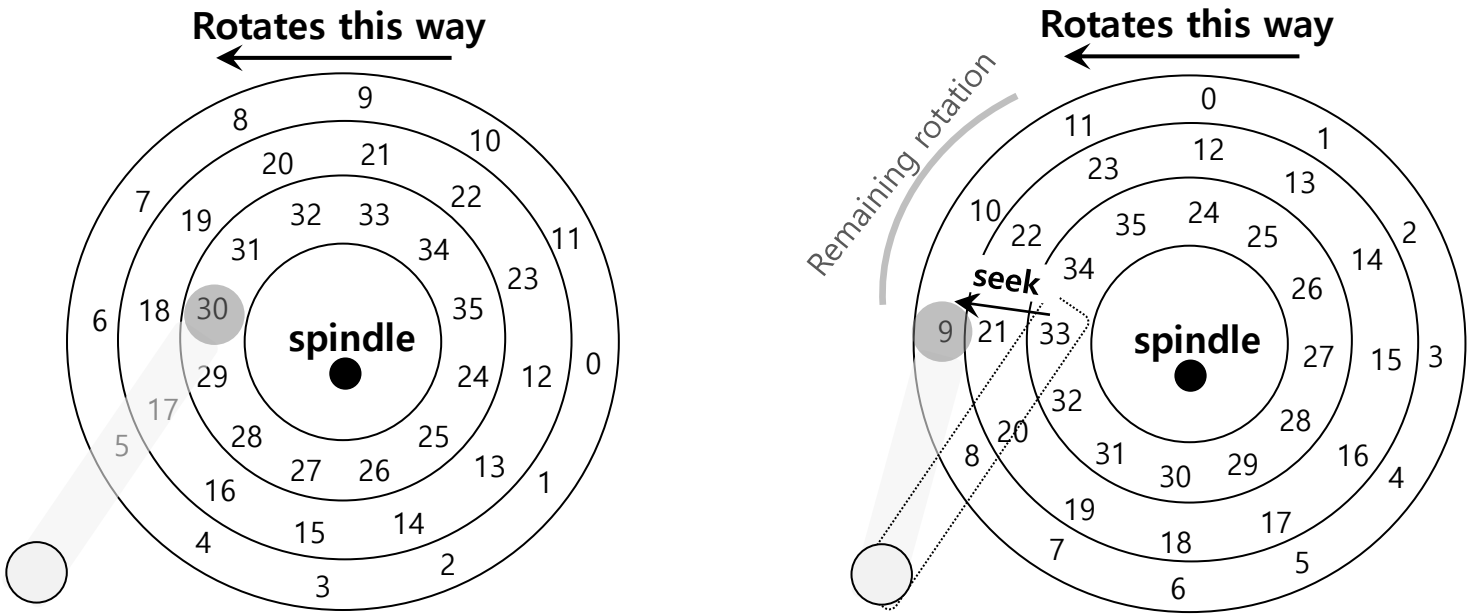

track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation

Tek İz Gecikmesi: Dönme Sebebiyle



- Dönme gecikmesi: İstenen sektörün dönmesi için geçen süre
- Örneğin, tam dönüş gecikmesi R'dir ve sektör 6'da başlar ise
  - Sektör 0'ı okumak için: Dönme gecikmesi = $R/2$
  - Sektör 5'i okumak için: Dönme gecikmesi = (11/12 ~ 1) R (en kötü durum)

Birden Fazla İz: Arama Zamanı



Three Tracks Plus A Head (Right: With Seek)
(e.g., read to sector 11)

- **Arama (Seek)**: Disk kolunu doğru ize hareket ettirmek için
- **Arama süresi (Seek Time)**: İstenen sektörü içeren ize kafayı hareket ettirme zamanı. En maliyetli disk işlemlerinden birisidir.

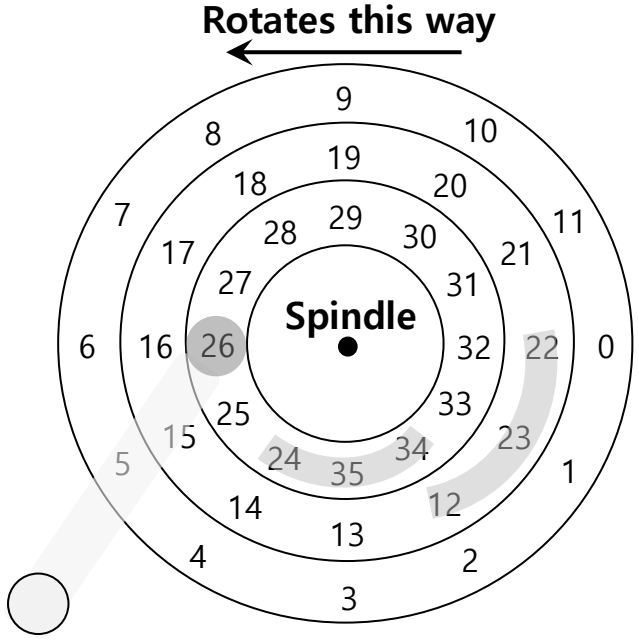
Transfer Süresi

- G/Ç'nin son aşaması
  - Veriler ya bir yüzeyden okunur ya da yüzeye yazılır.

- Tüm G/Ç süresi:
  - Arama gecikmesi
  - Dönme gecikmesi
  - **Transfer gecikmesi**

İz Eğriliği (Track Skew)

- İz sınırlarını geçerken bile sıralı okumaların düzgün bir şekilde yapılabilmesi için



Three Tracks: Track Skew Of 2

- İz eğriliği yok ise, kafa bir sonraki ize taşındığında istenen bir sonraki blok zaten kafanın altından kaymış olur.

Önbellek - Cache (Track Buffer)

- Diskten okunan veya diske yazılan verileri tutar.
- Sürücünün isteklere hızla yanıt vermesini sağlar.
- Az miktarda bellek kapasitesi içerir (genellikle yaklaşık 8 veya 16 MB).

Önbelleğe Yazma

- Sonra yaz (Anında raporla):
  - Verileri belleğe koyduğunda yazmanın tamamlandığını bildir.
  - Daha hızlı ama tehlikeli
- Baştan sona yaz:
  - Ancak yazma gerçekten diske yazıldıktan sonra bir yazmanın tamamlandığını bildir.

I/O Süresi: İşin Matematiği

- I/O süresi ($T_{I/O}$): $$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

- I/O hızı ($R_{I/O}$): $$R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$$

	Cheetah 15K.5	**Barracuda**
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects Via	SCSI	SATA

Disk Drive Specs: SCSI Versus SATA

I/O Süresi Örneği

- **Rastgele iş yükü:** Diskteki rastgele konumlarda 4KB okuma yap
- **Sıralı iş yükü:** Diskten art arda 100 MB oku

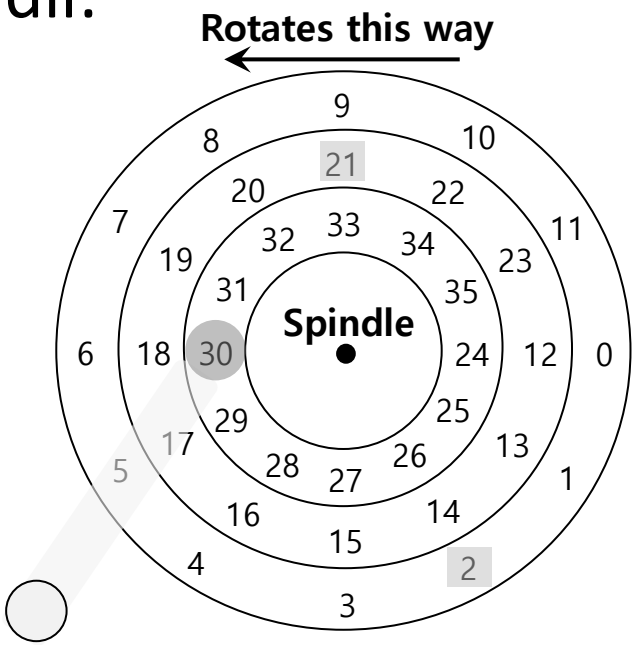
		Cheetah 15K.5	Barracuda
T_{seek}		4 ms	9 ms
$T_{rotation}$		2 ms	4.2 ms
Random	$T_{transfer}$	30 microsecs	38 microsecs
	$T_{I/O}$	6 ms	13.2 ms
	$R_{I/O}$	0.66 MB/s	0.31 MB/s
Sequential	$T_{transfer}$	800 ms	950 ms
	$T_{I/O}$	806 ms	963.2 ms
	$R_{I/O}$	125 MB/s	105 MB/s

Disk Drive Performance: SCSI Versus SATA

Rastgele ve sıralı iş yükleri arasında sürücü performansında büyük bir fark vardır.

Disk Zamanlayıcı (Scheduling)

- Disk Zamanlayıcı, bir sonraki hangi G/Ç isteğinin zamanlanacağına karar verir.
- SSTF (Önce En Kısa Arama Süresi): I/O istek kuyruğunu iz numarası bilgisine göre sıra ve önce tamamlamak için en yakın izdeki istekleri önceliklendir.



SSTF: Scheduling Request 21 and 2

issue the request to 21 → issue the request to 2

SSTF her zaman en iyisi midir?

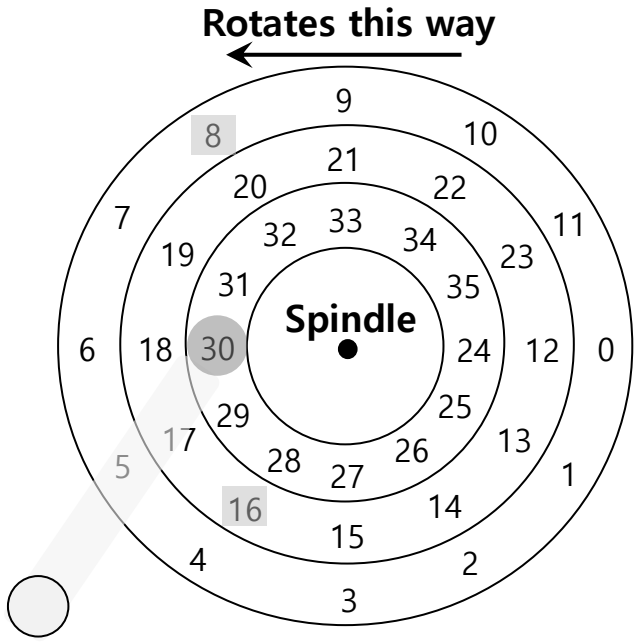
- Sorun 1: Sürücü geometrisi, ana bilgisayar işletim sisteminde mevcut değildir.
  - Çözüm: İşletim sistemi, En Yakın Blok Önceliği'ni (NBF) basitçe uygulayabilir.

- Sorun 2: Açlık problemi.
  - Eğer iç ize sabit bir istek akışı var ise, diğer izlere yönelik istek tamamen göz ardı edilebilir.

Asansör (Elevator) Yöntemi

- İzler arasında belli bir yönde hareket etme.
- Süpürme: Disk boyunca gezmelerden her bir tanesi
  - Diskin bu süpürmesinde (taramasında) daha önce hizmet verilmiş bir izde bir blok için istek gelirse, bir sonraki taramaya kadar kuyruğa alınır.
- F-TARAMA: Süpürme yaparken hizmet verilecek kuyruğu dondur.
  - Uzaktaki isteklerin açlık problemine karşın uygulanırç
- C-TARAMA (Dairesel TARAMA): Dıştan içe ve ardından içten dışa olacak şeklinde.

Disk Döndürme Maliyetleri?



SSTF: Sometimes Not Good Enough

- Döndürme aramadan daha hızlıysa: istek 16 → istek 8
- Arama, döndürmeden daha hızlıysa: istek 8 → istek 16

Modern sürücülerde, hem arama hem de döndürme kabaca eşdeğerdir: Bu nedenle, SPTF (Önce En Kısa Konumlandırma Süresi) daha kullanışlıdır.