# 7- SANAL BELLEK

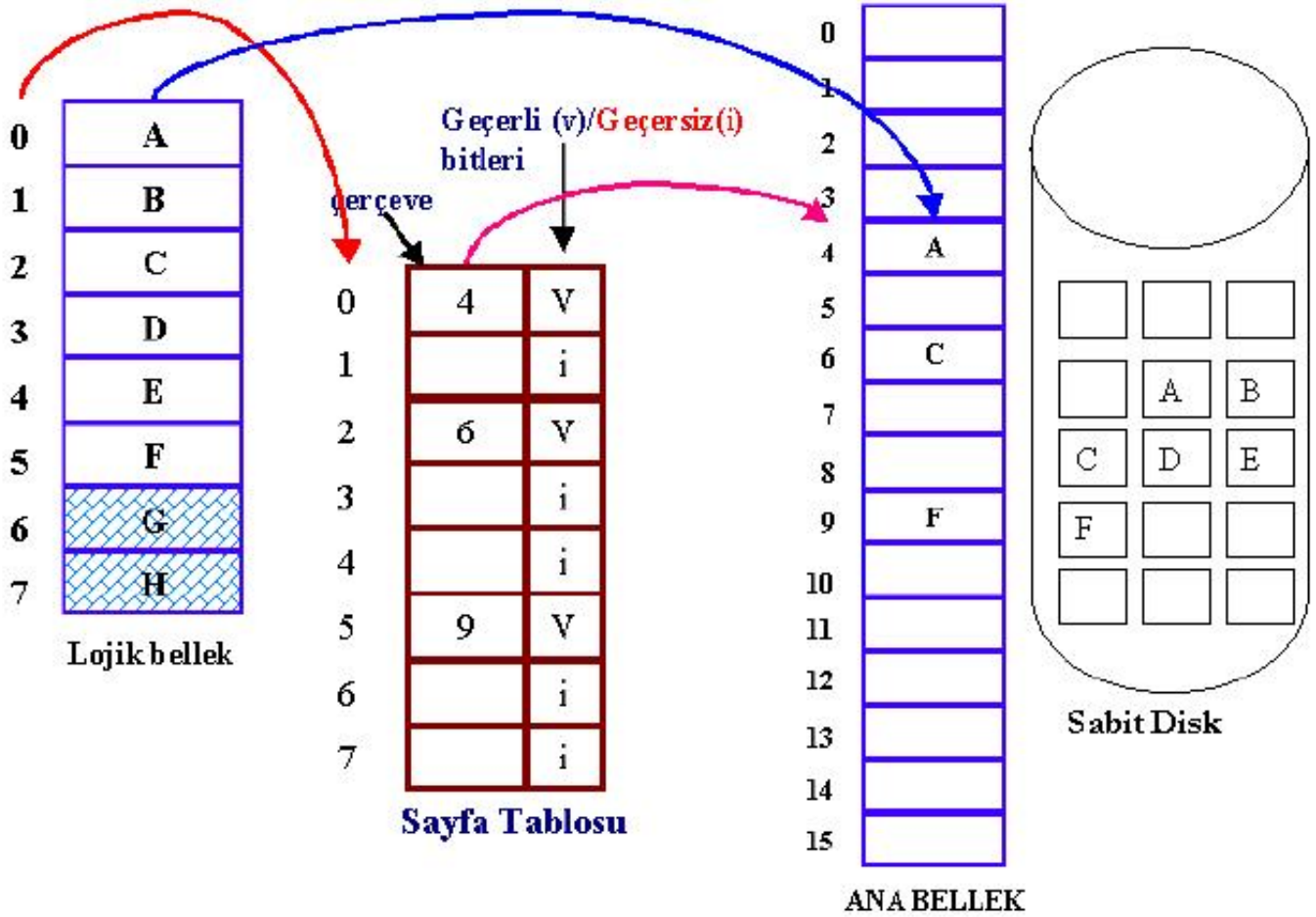
## 7.1 - Giriş

Sanal bellek yönetimi, ana belleğe tam olarak sığmayan işlemlerin koşturulmasına olanak veren bir bellek yönetim tekniğidir. En büyük avantajı fiziksel bellekten büyük programların çalıştırılabilmesidir. Sanal bellek genelde *İstemsel Sayfalama* ve bölümlendirme teknikleri ile gerçekleştirilir.

### 7.1.1 - İstemsel Sayfalama

İstemsel sayfalama, sayfalama işlemine benzerdir. İşlemler ikincil bellekte bulunurlar. Bir işlemin koşturulması istenildiğinde, işlem ikincil bellekten ana belleğe getirilir. Bu yer değiştirme için "tembel yerdeğiştiririci (lazy swapper)" denilen bir teknik kullanılır. Tembel yerdeğiştirici bir sayfayı ihtiyaç olmadan, kesinlikle belleğe getirmez. Bu sisteme kısaca "yerdeğiştirici (swapper)" de denir.
İşlemler birbirini belli sırayla takip eden sayfalardan oluşurlar. Bu sayfalar ile ilgili sisteme "sayfalayıcı(pager)" denir.

Ana belleğe yerleştirilen bir işlemin hangi sayfasının kullanılacağına sayfalayıcı karar verir. Sayfalayıcı, ana belleğe işlemin kullanılacak olan sayfasını getirir. Böylece sayfalayıcı kullanılmayacak olan sayfaları ana belleğe getirmez. Tüm sayfalar yerine sadece işlenecek olan sayfaların ana belleğe getirilmesi yerdeğiştirme zamanını düşürdüğü gibi, ana bellekte daha az yer kullanılmasını sağlar. Örneğin ana belleği 32 MB olan bir bilgisayarda, 290 MB'lık bir program koşturulmak istenirse, bu programın tümünün ana belleğe yerleşmesi mümkün değildir. Bu durumda istemsel sayfalama yöntemi kullanılarak, örneğin ana bellek 4 KB'lık sayfalara bölünerek, koşturulmak istenen programın sadece gerekli işlemlerine ait sayfalar ana belleğe yüklenir. Geri kalanı ise ikincil bellekte bekler. İkincil bellekteki kullanılmayan işlemlere ait sayfalar, gerekli olduğunda ana belleğe getirilir.

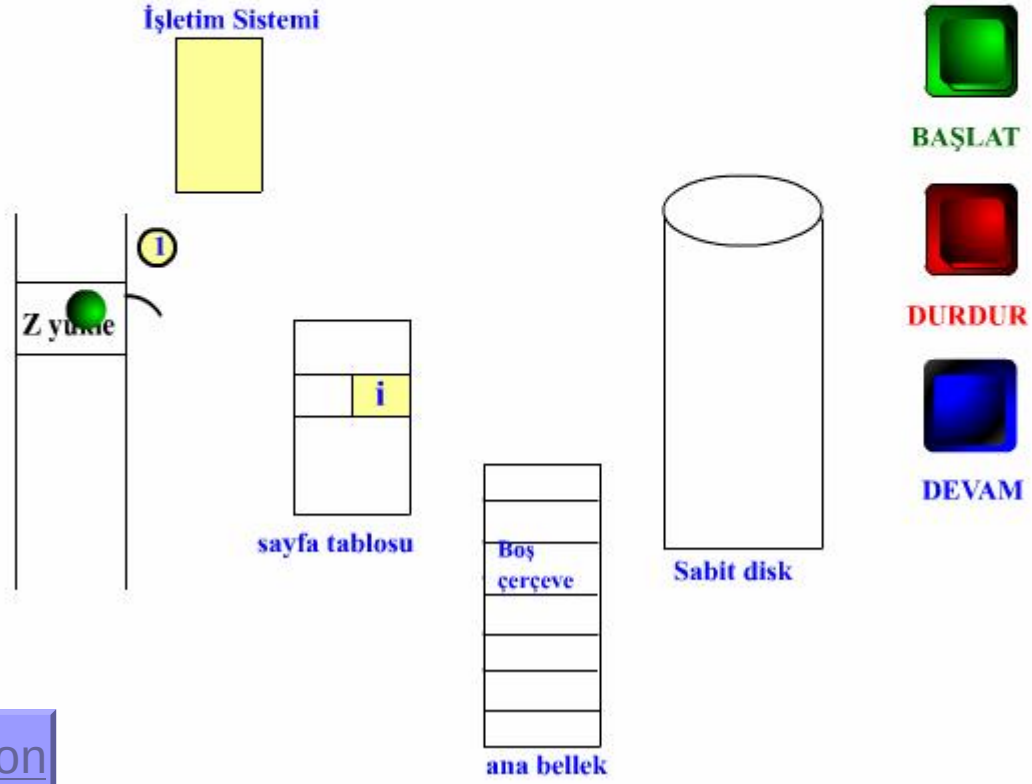



Şekil 7.1 Sayfa tablosu ve ana bellekteki görünümü.

Hangi sayfanın bellekte olduğu hangisinin olmadığı donanımsal olarak "geçerli-geçersiz(valid-invalid)" bitleri sayfa tablosunda kullanılarak gösterilir. "Geçerli(Valid)" durumunda sayfa ana bellekte, "geçersiz(invalid)" durumunda sayfa ana bellekte değil, ikincil bellek(sabit disk ya da disket) üzerinde demektir. Bu durum Şekil 7.1'de görülmektedir.

## 7.1.2 - Sayfa Hatası

Bir işlemin koşturulması sırasında, işlem için gerekli olan sayfa ya da sayfaların ana bellek üzerinde olmaması sonucu oluşan hataya "sayfa hatası (page fault)" denir. Bu durum Şekil 7.2'deki animasyonda görülmektedir.




Animasyon

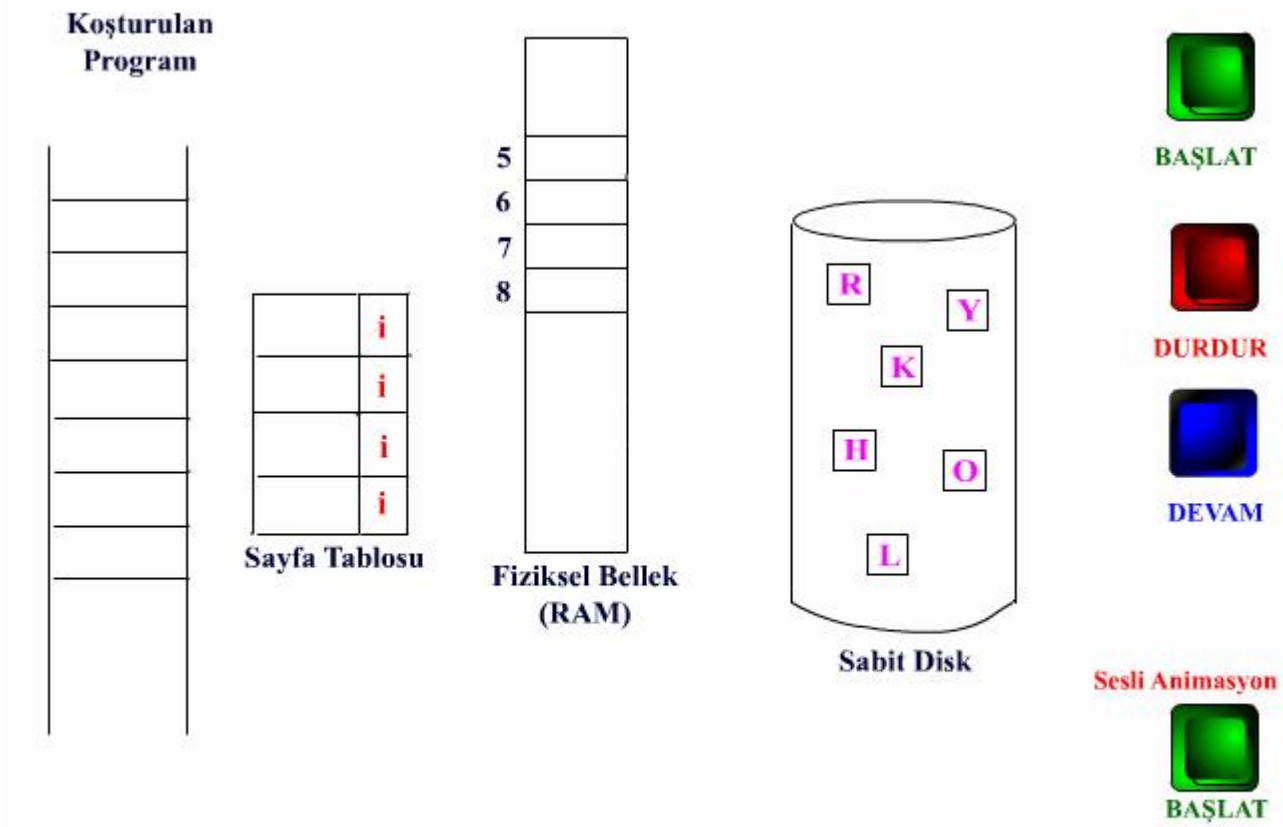
Şekil 7.2 Sayfa hatası oluşumu

## 7.2 Sayfa Yeniden Yerleştirilme

Bir program koşturulmak istediğinde, MİB tarafından koşulacak olan işlemlerin ana bellekte bulunması gerekir. Bu programın koşturulacak işlemlerine ait sayfalar ana bellekte değilse, bu sayfaların ikincil bellekten ana belleğe getirilmesi gerekir. Koşturulacak sayfaların ana belleğe getirilmesinde aşağıdaki adımlar sırayla gerçekleştirilir (Bu adımlar Şekil 7.3'deki animasyonda da görülmektedir):

1) İstenilen sayfa ikincil bellek(sabit disk gibi...) üzerinde bulunur.
2) Ana bellek üzerinde boş çerçeve(frame) bulunur.
   a) Eğer boş çerçeve varsa o kullanılır.
   b) Eğer boş çerçeve yoksa, sayfa yerleştirme algoritmalarına göre bir kurban çerçeve (victim frame ) seçilir. *(Sayfa yerleştirme algoritmaları bir sonraki kısımda açıklanacaktır.)*
   c) Seçilen kurban sayfa ikincil bellek üzerine yazılır. Sayfa ve çerçeve tablosu uygun şekilde değiştirilir.
3) İstenilen sayfa, boşaltılan çerçeve içine yerleştirilir. Sayfa ve çerçeve tablosu uygun şekilde değiştirilir.
4) İşlem yeniden başlatılır.

Görüldüğü gibi, eğer boş çerçeve yok ise iki sayfa yerdeğiştirme işlemine giriyor.




Animasyon

**Şekil 7.3 Sayfa yeniden yerleştirme.**

Özet olarak, ana bellek üzerinde boş çerçeve yoksa, bir kurban çerçeve seçilerek, bu çerçevenin ikincil bellek üzerine yerleştirilmesi, ikincil bellek üzerinden alınan sayfanın da boşaltılan boş çerçevenin yerine yerleştirilmesi işlemine sayfa yeniden yerleştirme denir.

# 7.3 Sayfa Yeniden Yerleştirilme Algoritmaları

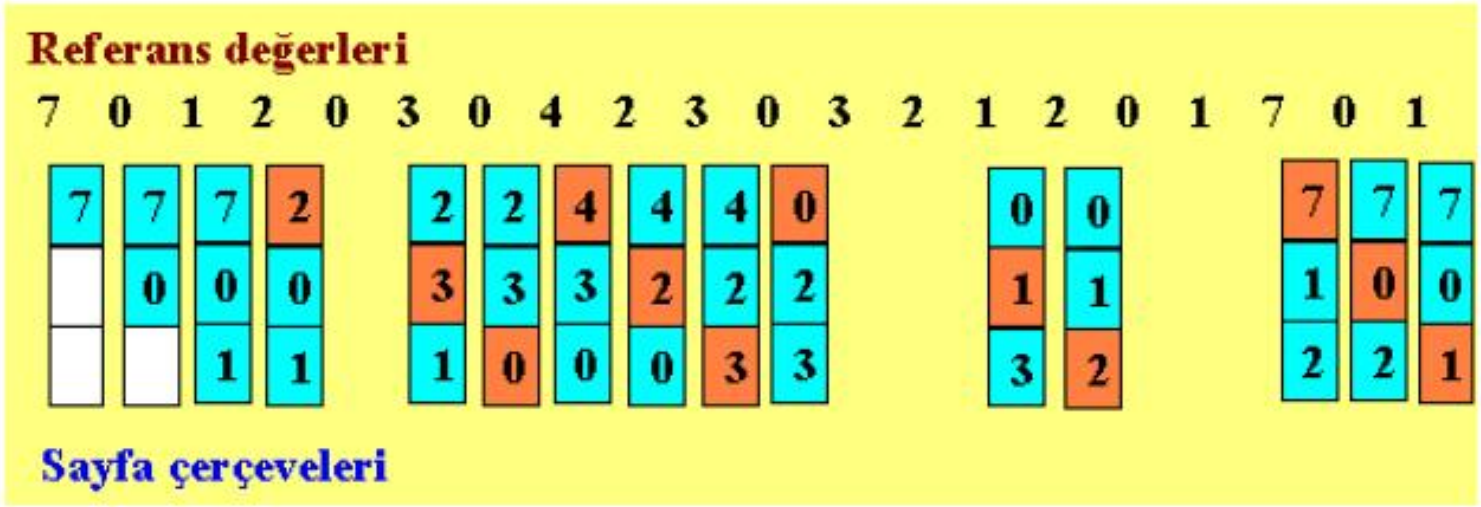
Farklı işletim sistemlerinin, değişik sayfa yerleştirme algoritmaları vardır. Sayfa yerleştirme algoritmalarının performansını gösteren kıstas sayfa hata sayısının ve sayfa yerdeğiştirme sayısının en düşük değerde olmasıdır.
Sanal bellek yönetiminde kullanılan sayfa yeniden yerleştirilmesindeki aşağıdaki algoritmalar incelenecektir.

- İlk Gelen İlk Gider(FIFO) Algoritması
- Optimal Algoritması
- LRU Algoritması

## 7.3.1 İlk Gelen İlk Gider Algoritması

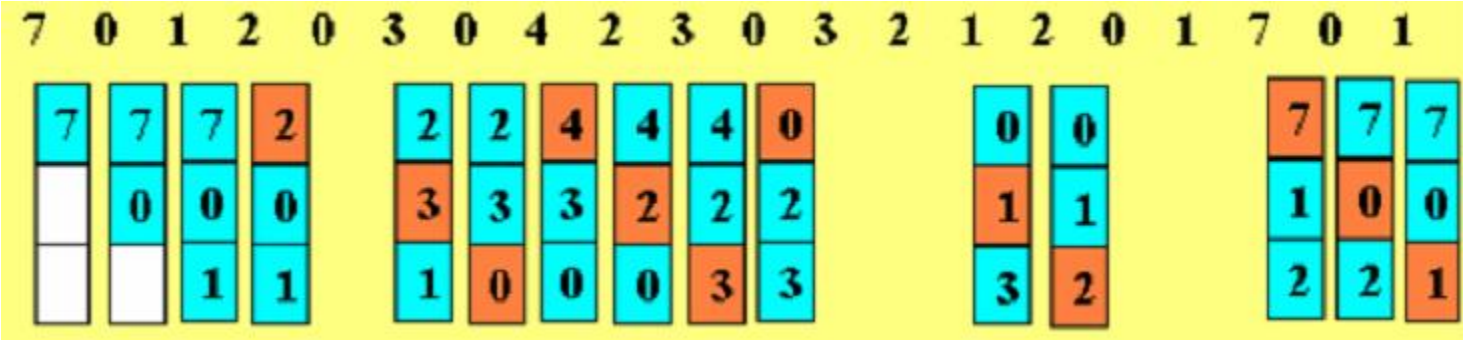
En basit sayfa yerleştirme algoritmasıdır. Çalışma yapısı, ana bellekte en uzun süre kalan sayfa, ana bellekten ilk olarak çıkartılır. Bu bir kuyruğa benzetilebilir. En son gelen sayfa, kuyruğun en arkasına gider. Eğer ana bellek üzerinde boş çerçeve yoksa, sıranın en önündeki sayfa ana bellekten çıkartılır.

# Örnek:




Şekil 7.4 FIFO Sayfa yeniden yerleştirme algoritması.

Şekil 7.4'de görülen örnekte,ana bellekte üç adet çerçeve bulunmaktadır. Başlangıçta bu üç çerçeveninde boş olduğu kabul edilmiştir. Gelen sayfa numaları da referans değeri olarak şekil üzerinde belirtilmiştir. Başlangıçta üç çerçevenin boş olması nedeni ile sırasıyla 7, 0 ve 1 numaralı sayfalar bu çerçevelere yerleşir. Bu nedenle ilk üç sayfanın yerleştirilmesinde 3 sayfa hatası oluşur.




Daha sonra gelen 2 nolu sayfa çerçeveler içerisinde bulunmadığından bir sayfa hatası oluşur. Fakat bu sayfanın yerleştirilebilmesi için, bir kurban sayfası seçilmesi gereklidir. Bu algoritmaya göre ilk olarak 7 nolu sayfa geldiği için, kurban sayfa olarak seçilir. 7 nolu sayfa ikincil belleğe gönderilirken, boşalan çerçeveye de 2 nolu sayfa yerleştirilir.
Daha sonra gelen 0 nolu sayfa çerçeveler içerisinde bulunduğundan dolayı sayfa hatası oluşmaz.
Bir sonra gelen sayfa 3 nolu sayfadır. Çerçeveler içerisinde bulunmadığından bir sayfa hatası oluşur. 3nolu sayfanın yerleştirilebilmesi için, bir kurban sayfası seçilmesi gereklidir. Bu algoritmaya göre çerçeveler içerisinde bulunan sayfalar içinde ilk olarak gelen sayfa 0 nolu sayfadır. Bu nedenle kurban sayfa olarak seçilir. 0 nolu sayfa ikincil belleğe gönderilirken, boşalan çerçeveye de 3 nolu sayfa yerleştirilir.
Bu işlemler diğer sayfalar için sırayla gerçekleştirilir.

Aşağıda bu algoritmayı gösteren bir simülasyon olayı gerçekleştirilmiştir.Ana bellekte 3 adet çerçeve bulunmaktadır. Simülatör üzerinde koşulacak olan sayfa numaralarını giriniz.

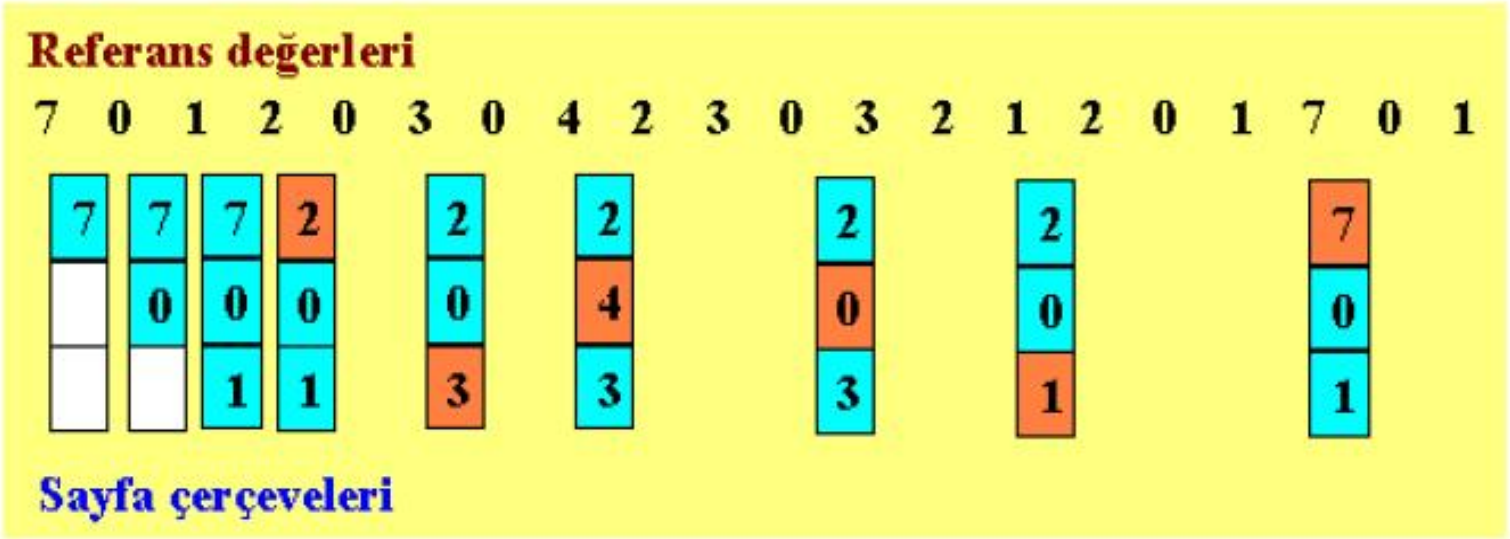

Simülatör 7.1 - İlk gelen ilk gider (FIFO) sayfa yeniden yerleştirme algoritması simülatörü.

Animasyon

# 7.3.2 Optimal Algoritması

Optimal algoritması, tüm algoritmalar içerisinde en düşük sayfa hatası sayısına sahiptir. Çalışma sistemi, mevcut sayfalar içerisinde en uzun süre kullanılmayacak olan sayfa, kurban sayfa olarak seçilerek ikincil belleğe gönderilir. Boşalan bu çerçeve yerine de yeni gelen sayfa yerleştirilir.

Örnek:




Şekil 7.5 Optimal Sayfa yeniden yerleştirme algoritması.

Şekil 7.5'de görülen örnekte,ana bellekte üç adet çerçeve bulunmaktadır. Başlangıçta bu üç çerçeveninde boş olduğu kabul edilmiştir. Gelen sayfa numaraları da referans değeri olarak şekil üzerinde belirtilmiştir. Başlangıçta üç çerçevenin boş olması nedeni ile sırasıyla 7, 0 ve 1 numaralı sayfalar bu çerçevelere yerleşir. Bu nedenle ilk üç sayfanın yerleştirilmesinde 3 sayfa hatası oluşur.

Daha sonra gelen 2 nolu sayfa çerçeveler içerisinde bulunmadığından bir sayfa hatası oluşur. Fakat bu sayfanın yerleştirilebilmesi için, bir kurban sayfası seçilmesi gereklidir. Bu algoritmaya göre en uzun süre kullanılmayacak olan sayfa kurban sayfa olarak seçilir. Çerçeveler içerisinde bulunan sayfalardan 7 nolu sayfa bir daha 18 sırada, 0 nolu sayfa 5 sırada, 1 nolu sayfa da 14 sırada kullanılacaktır. Bu nedenle 7 nolu sayfa kurban sayfa olarak seçilir. 7 nolu sayfa ikincil belleğe gönderilirken, boşalan çerçeveye de 2 nolu sayfa yerleştirilir.

Daha sonra gelen 0 nolu sayfa çerçeveler içerisinde bulunduğundan dolayı sayfa hatası oluşmaz.

Bir sonra gelen sayfa 3 nolu sayfadır. Çerçeveler içerisinde bulunmadığından bir sayfa hatası oluşur. Çerçeveler içerisinde bulunan sayfalardan 2 nolu sayfa bir daha 9 sırada, 0 nolu sayfa 7 sırada, 1 nolu sayfa da 14 sırada kullanılacaktır. Bu nedenle 1 nolu sayfa kurban sayfa olarak seçilir. 1 nolu sayfa ikincil belleğe gönderilirken, boşalan çerçeveye de 3 nolu sayfa yerleştirilir.

Bu işlemler diğer sayfalar için benzer şekilde gerçekleştirilir.

Aşağıda bu algoritmayı gösteren bir simülasyon olayı gerçekleştirilmiştir.Ana bellekte 3 adet çerçeve bulunmaktadır. Simülatör üzerinde koşulacak olan sayfa numaralarını giriniz.

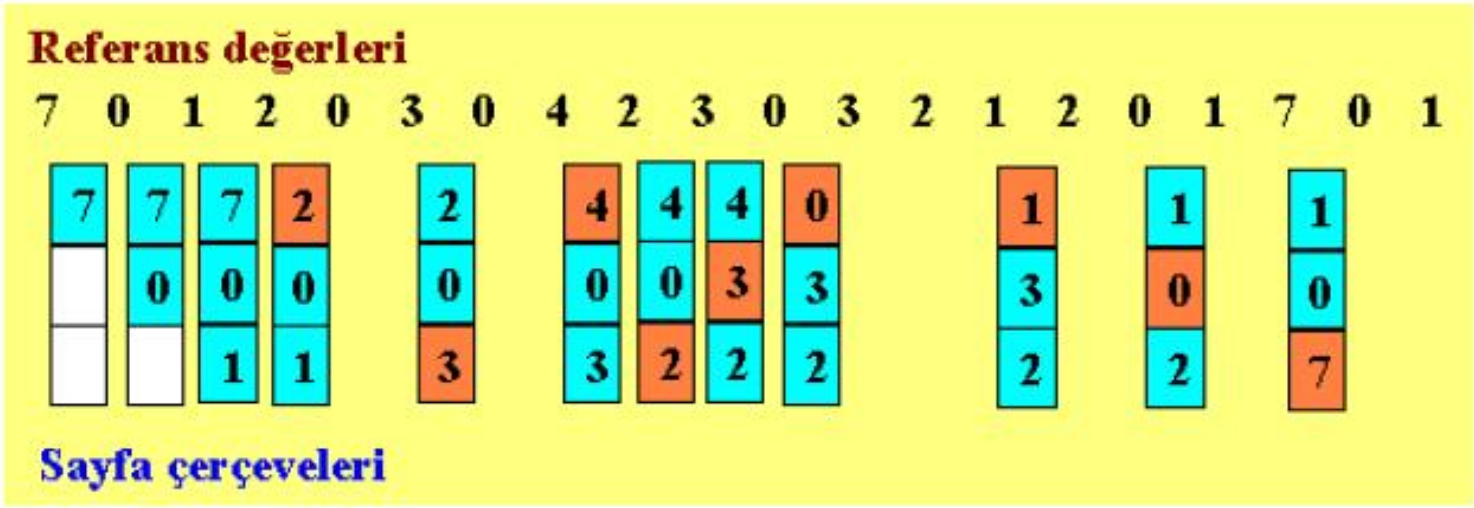

Simülatör 7.2 - Optimal sayfa yeniden yerleştirme algoritması simülatörü.

Animasyon

# 7.3.3 LRU Algoritması

LRU algoritması performans bakımından İlk gelen ilk gider algoritması ile Optimal algoritması arasında yer alır. En uzun süre kullanılmayan sayfa kurban olarak seçilerek ikincil belleğe gönderilir. Boşalan bu çerçeve yerine de yeni gelen sayfa yerleştirilir.

Örnek:

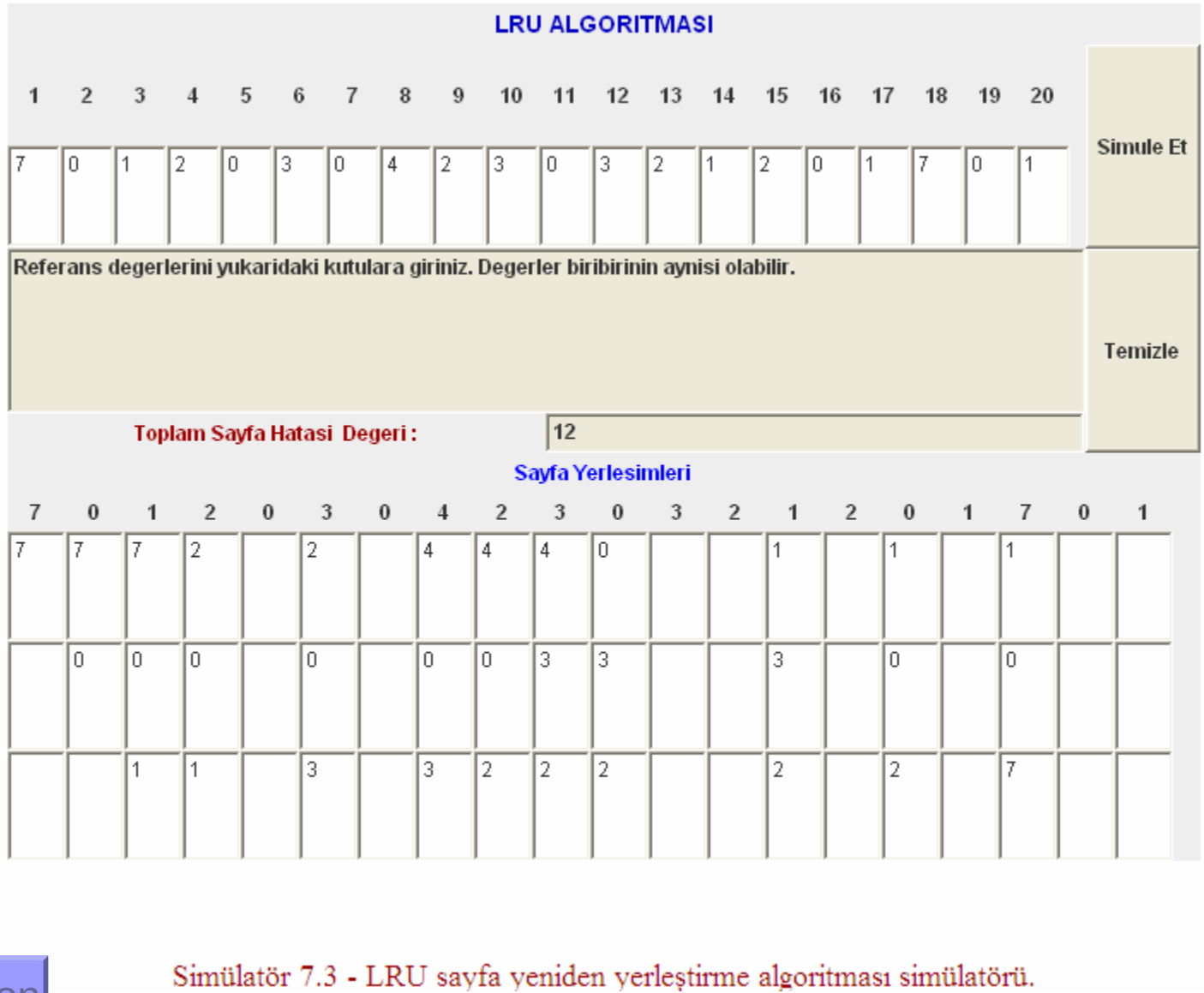



Şekil 7.6 LRU Sayfa yeniden yerleştirme algoritması.

Şekil 7.6'de görülen örnekte,ana bellekte üç adet çerçeve bulunmaktadır. Başlangıçta bu üç çerçeveninde boş olduğu kabul edilmiştir. Gelen sayfa numaraları da referans değeri olarak şekil üzerinde belirtilmiştir. Başlangıçta üç çerçevenin boş olması nedeni ile sırasıyla 7, 0 ve 1 numaralı sayfalar bu çerçevelere yerleşir. Bu nedenle ilk üç sayfanın yerleştirilmesinde 3 sayfa hatası oluşur.
Daha sonra gelen 2 nolu sayfa çerçeveler içerisinde bulunmadığından bir sayfa hatası oluşur. Fakat bu sayfanın yerleştirilebilmesi için, bir kurban sayfası seçilmesi gereklidir. Bu algoritmaya göre en uzun süre kullanılmayan sayfa kurban sayfa olarak seçilir. Çerçeveler içerisinde bulunan sayfalardan 7 nolu sayfa bellekte en uzun süre kullanılmadan bekleyen sayfadır. Bu nedenle 7 nolu sayfa kurban sayfa olarak seçilir. 7 nolu sayfa ikincil belleğe gönderilirken, boşalan çerçeveye de 2 nolu sayfa yerleştirilir.
Daha sonra gelen 0 nolu sayfa çerçeveler içerisinde bulunduğundan dolayı sayfa hatası oluşmaz.

Bir sonra gelen sayfa 3 nolu sayfadır. Çerçeveler içerisinde bulunmadığından bir sayfa hatası oluşur. Çerçeveler içerisinde bulunan sayfalardan 1 nolu sayfa bellekte en uzun süre kullanılmadan bekleyen sayfadır.Bu nedenle 1 nolu sayfa kurban sayfa olarak seçilir. 1 nolu sayfa ikincil belleğe gönderilirken, boşalan çerçeveye de 3 nolu sayfa yerleştirilir.
Bu işlemler diğer sayfalar için benzer şekilde gerçekleştirilir.

Aşağıda bu algoritmayı gösteren bir simülasyon olayı gerçekleştirilmiştir.Ana bellekte 3 adet çerçeve bulunmaktadır. Simülatör üzerinde koşulacak olan sayfa numaralarını giriniz.




Animasyon

Simülatör 7.3 - LRU sayfa yeniden yerleştirme algoritması simülatörü.

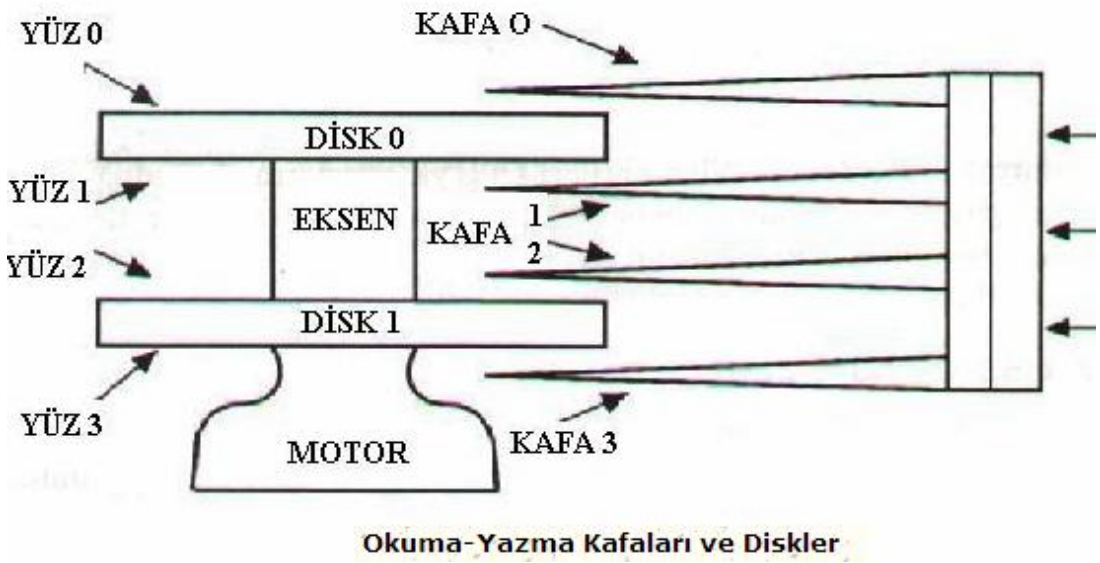
# 8 - İKİNCİL BELLEK YÖNETİMİ

## 8.1 - Giriş

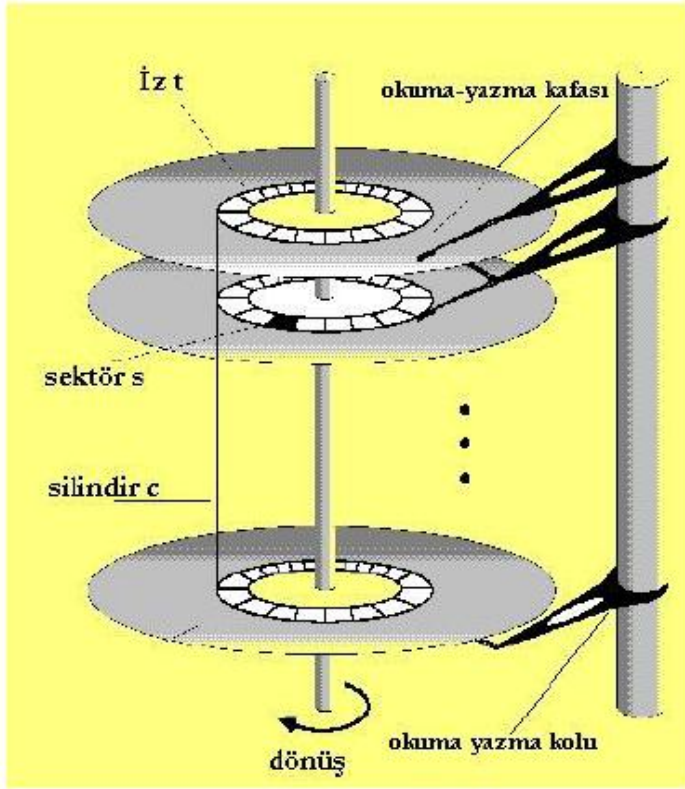
Bilgisayar sistemleri ana belleğin yanında bilgilerin ve programların saklanmasına olanak veren ikinci bir saklama aygıtına sahiptir. İlk saklama aygıtları delikli kartlar ve manyetik teypler olmuştur. Bu aygıtların sıralı bilgi saklaması ve hızlarının düşük olması modern saklama aygıtlarının ortaya çıkmasına neden olmuştur. Diskler, disketler ve optik diskler modern saklama aygıtlarıdır.

### 8.1.1 - Disk Yapısı

Bilgisayarlarda kullanılan diskler fiziksel olarak basit yapıdadırlar. Bilgi, her iki yüzü manyetik bir materyalle kaplı olan diskler üzerine kaydedilir. Disk yapısı Şekil 8.1'de basit olarak görülmektedir. Yüksek hızlı bir motor diskleri okuma yazma kafası önünde döndürür. Okuma yazma kafaları da bu diskler üzerinde bir kol yardımıyla hareket ederek istenilen bilgiye ulaşır.

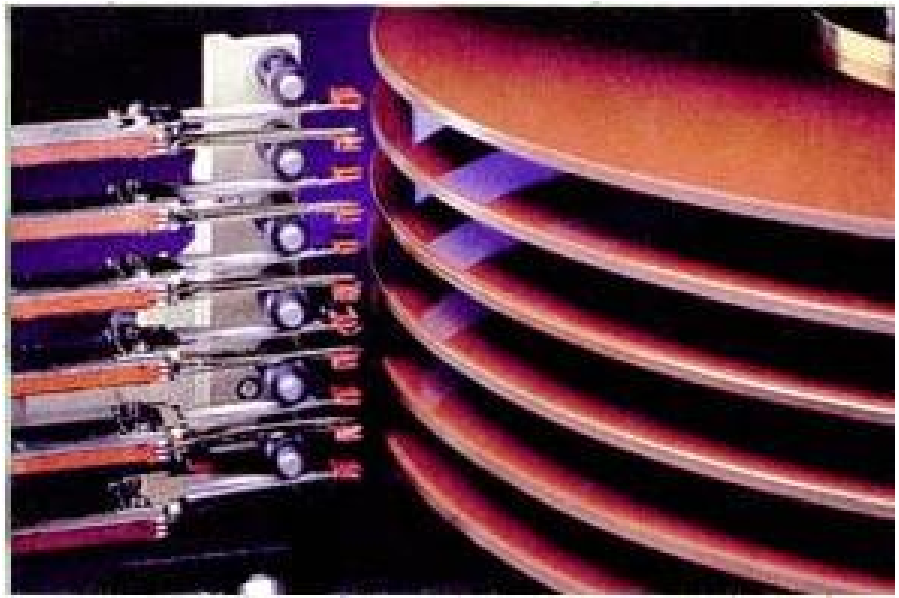


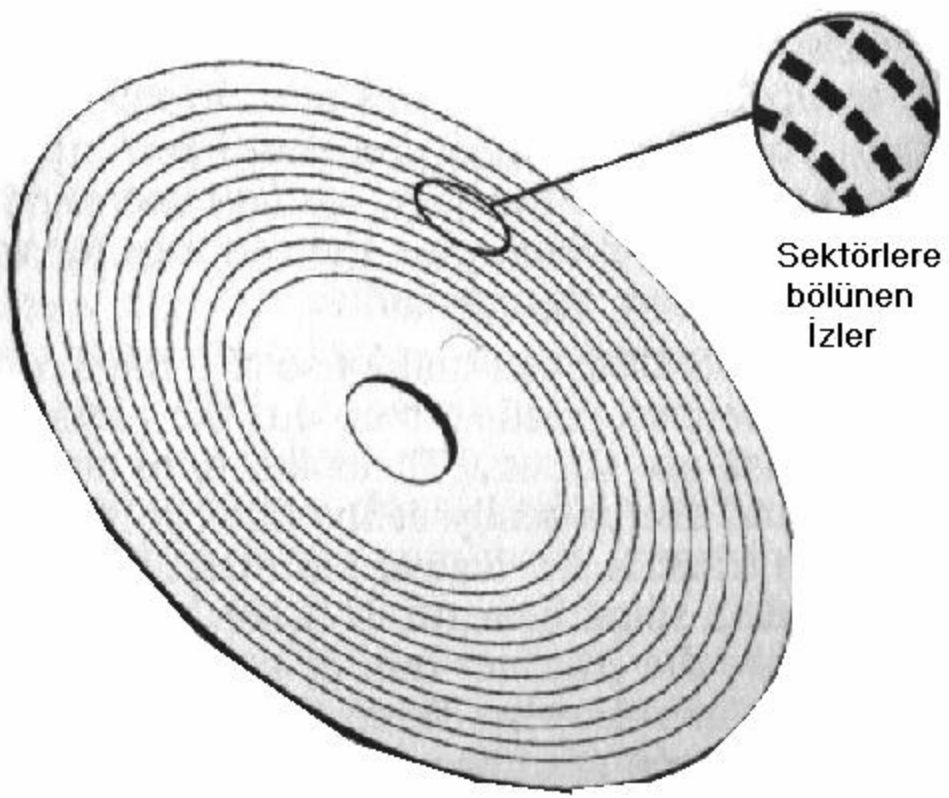

Okuma-Yazma Kafaları ve Diskler

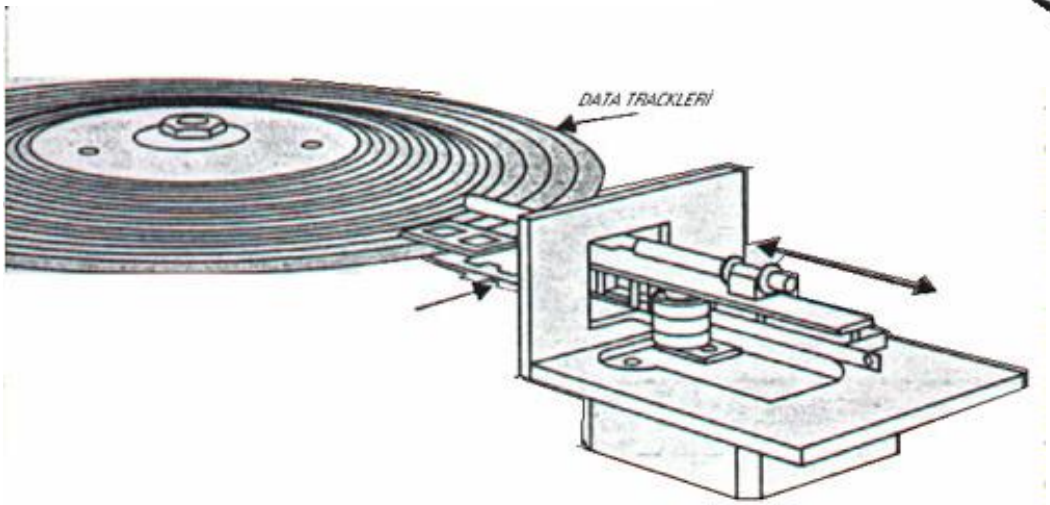


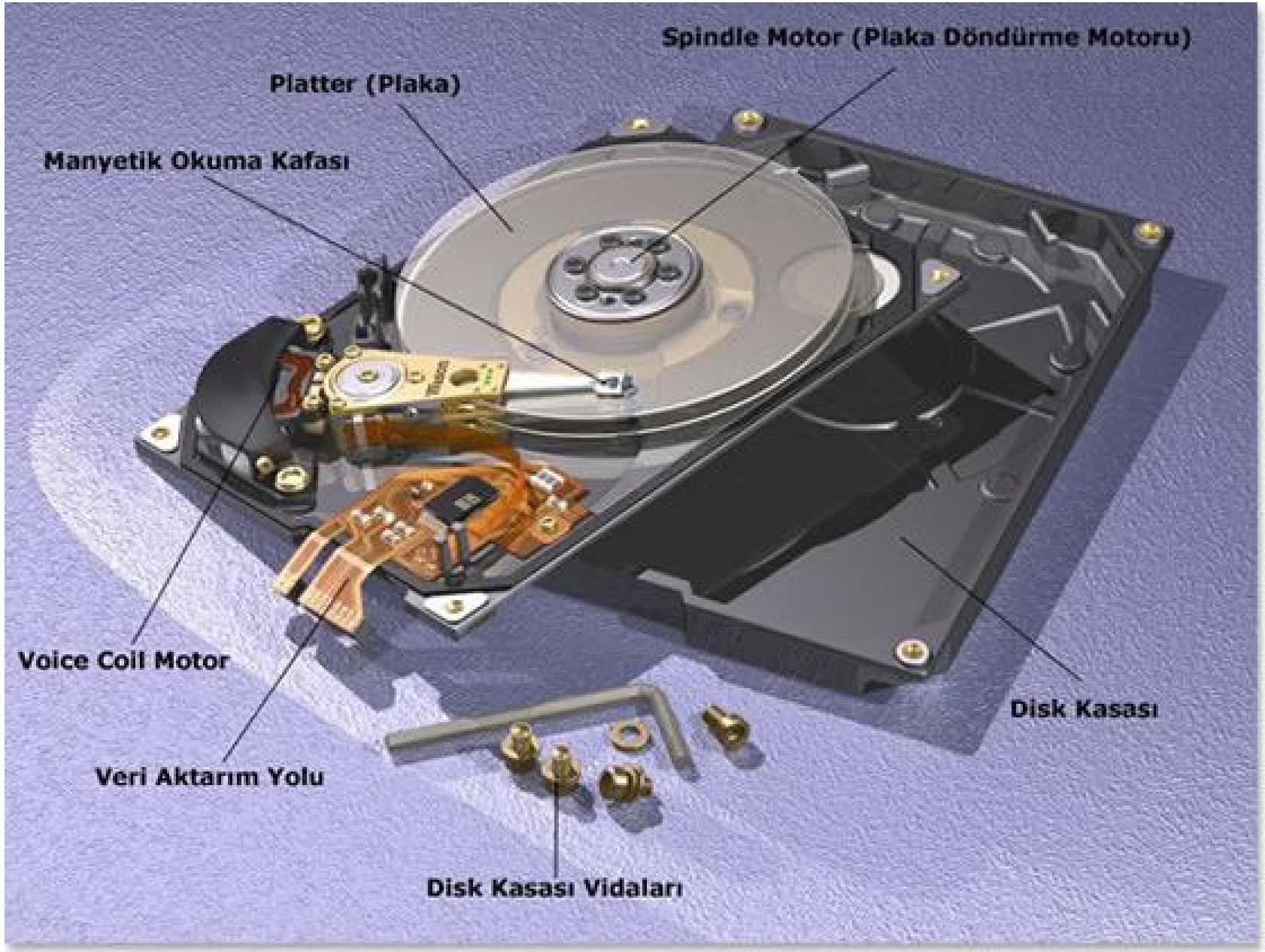

Şekil 8.1 Hareketli kafalı disk mekanizması.

Disk yüzeyi mantıksal olarak izlere(track) bölünmüştür. Bilgi okuma yazma kafasının altında bulunan ize otomatik olarak kaydedilir ya da o iz üzerinden okunur. Çeşitli yapıda disk mekanizmaları mevcuttur. Örneğin, sabit kafalı diskler ya da hareketli kafalı diskler gibi. Sabit kafalı disklerde her disk için bir kafa mevcuttur. Diskler bu kafaların önünde hareketlidir. Böylece izden ize daha hızlı hareket edilir. Ancak kafa sayısı arttığı için bu diskler oldukça pahalıdır. Hareketli kafalı disklerde ise , her disk üzerinde hareket edebilen bir tek okuma-yazma kafası bulunur. Okuma-yazma kafasının az oması sebebiyle bu diskler sabit kafalı disklere göre daha ucuzdur.




Sektörlere
bölünen
İzler


DATA TRACKLERİ


Spindle Motor (Plaka Döndürme Motoru)
Platter (Plaka)
Manyetik Okuma Kafası
Voice Coil Motor
Disk Kasası
Veri Aktarım Yolu
Disk Kasası Vidaları


ESMT
88i6522-LG01
FH05451.2
0448 A3P
Taiwan
JVC
R2
2D

Diskin çalışma mantığını anlatabilmek için bir örnek verelim. Diyelim ki bir Word dosyasını Word'de açıyoruz:
1. İlk aşama aranan bilginin nerede yer aldığını bulmaya yöneliktir. Bu iş için kullandığınız uygulama, işletim sistemi, BIOS ve muhtemelen işletim sisteminin disk için yüklediği sürücü diskin hangi bölümünün okunacağını belirlerler.
2. Aranan bilginin hangi silindirde, hangi kafa tarafında, hangi sektörde yer aldığının adresi çıkarılır; diske bu adres verilerek, o bölgeyi okuması istenir.
3. Bir güç koruma işlemi devreye girmemişse, disk zaten normal hızında döner durumdadır. Aksi halde diskin normal devir hızına ulaşması sağlanır.
4. Disk denetçisi aldığı adresi yorumlar ve istenen silindire bakar. Bu silindirde aranan bilginin yer aldığı iz bulunur ve kafa bu iz üzerine oturur.
5. Kafa izi okumaya başlar ve disk dönerken istenen bilginin yer aldığı sektörün altından geçmesini bekler. Sonra da bu sektörün içeriğini okur.
6. Disk denetçisi diskten okunan bilginin disk üzerinde geçici bir alana (tam-pon bellek) akışını kontrol eder. Daha sonra bu bilgiyi disk kablosu (arayüzü) üzerinden belleğe ileterek PC 'nin o bilgi için verdiği emri yerine getirmiş olur.

# 8.2 Boş Alan Yönetimi

Disk üzerinde bulunan dosyalar silindiğinde, silinen dosyaların boşalttığı alanları tekrar kullanılması gerekmektedir. Bu nedenle, işletim sistemleri disk üstündeki boş alanların yönetimiyle ilgili bir "boş alan listesi" tutar. Bir dosya, kaydedilmek istendiğinde işletim sistemi bu liste üzerinden dosyanın yerleşeceği en uygun boşluğu bulur ve dosya buraya kaydedilir. Daha sonra bu alan boş alan listesinden çıkartılır. Bir dosya silindiğinde ise, silinen dosyanın boşalttığı alan boş alan listesine eklenir. İkincil bellek üzerindeki boş alanların listesini tutmak için işletim sistemlerinde kullanılan çeşitli yöntemler vardır. Bu yöntemler aşağıda açıklanmıştır.

## 8.2.1 - Bit Vektör Yöntemi

Bit vektör yönteminde boş alan listesi bir bit haritası şeklinde tutulur. Bilgi bulunan yerler 1, boş olan yerler 0ile gösterilir (*Bazı işletim sistemlerinde bunun tersi de olabilir*). Örneğin disk üstünde 2,3,4,5,8,9,10,11,17 nolu alanların boş olduğunu kabul edilirse, bu yönteme göre bit vektörü şu şekilde olur.

1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 0 1 1 .....

Bu yöntemin temel avantajı basit olması ve ardarda gelen boş alanların kolaylıkla bulunabilmesidir.

## 8.2.2 - Bağlantılı Liste Yöntemi

Boş alanların tutulması ile ilgili işletim sistemlerinin kullandığı diğer bir yöntem ise bağlantılı liste yöntemidir. Bu yönteme göre boş alan kendisinden sonra gelen boş alanı gösteren bir işaretleyici(pointer) tutar.

## 8.2.3 - Gruplama Yöntemi

Kullanılan bir diğer yöntem de gruplamadır. Bu yöntemde birbirlerini takip eden boş alanlar gruplanır ve grubun ilk elemanın adresi bir önceki boş alan grubunun son elamanında tutulur. Grubun ilk elemanı kendisinden sonra kaç tane boş alan olduğu bilgisini tutar.

## 8.2.4 - Sayma Yöntemi

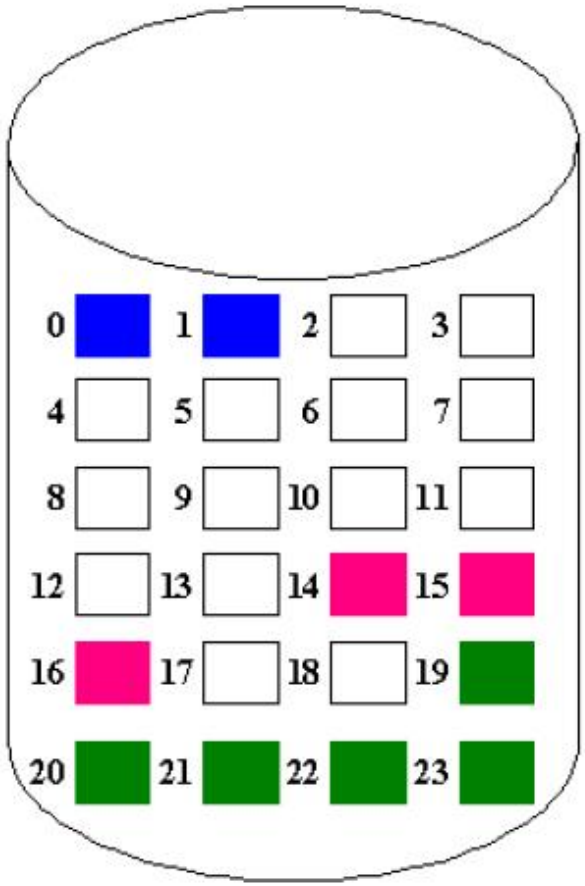
İkinci bellek üzerinde bulunan ilk boşluk alanda, tüm boş alanların ve bunları takip eden boşlukların adres ve sayıları tutulur.

## 8.3 Yerleşim Metodları

İkincil bellek üzerine bir çok dosya kaydedilir. Bu nedenle disk üzerindeki boş alanların yönetimi ne kadar önemliyse, kaydedilecek dosyaların disk üzerine en verimli şekilde yerleştirilmesi de o derece önemlidir. İşletim sistemlerinin ikincil bellek üzerine bilgi yerleştirmek için kullandıkları üç çeşit metod bulunmaktadır.
Bu metodlar aşağıda açıklanmaktadır.

### 8.3.1 - Yanyana Yerleşim

Yanyan yerleşim metodunda, her dosya disk üzerinde birbirini takip eden adreslerde bulunur. Bu yerleşim yöntemiyle yerleştirilmiş olan dosya, ilk blok adresi ve uzunluğuyla tanımlanır. Örneğin bir dosya n blok uzunluğunda ve b adresinden başlayarak yerleştirilmişse, dosyanın yerleştiği alan b, b+1, b+2, …… ….b+(n-1). Bu durum Şekil 8.2'de görülmektedir. Örneğin, listedeki dene dosyası 14 nolu bloktan başlayarak, sıra ile 14,15,16 numaralı bloklarda yerleştirilmiştir. dene dosyası 14 nolu blok adresi ile başlamaktadır ve toplam uzunluğu üç bloktur.



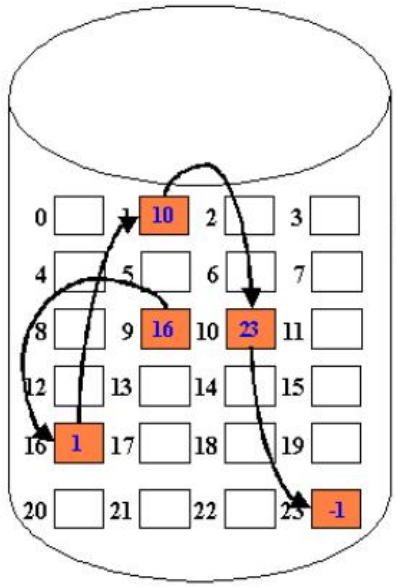

**Dizin**

| Dosya Adı | Baslangıcı | Uzunluğu |
|---|---|---|
| sayac | 0 | 2 |
| dene | 14 | 3 |
| posta | 19 | 5 |

Şekil 8.3 Sıralı yerleşim.

## 8.3.2 - Bağlantılı Yerleşim

Bağlantılı yerleşim tekniğinde, her dosya parçalara ayrılarak değişik yerlerdeki disk bloklarına yerleştirilebilir. Dizinler dosyanın ilk ve son bloğunun bulunduğu adresleri gösteren bir işaretliyiciye(pointer) sahiptir. Aynı zamanda, dosyanın yerleştiği her blok dosya bilgisinin yanında kendisinden sonra gelen bloğu gösteren bir işaretliyiciye de sahiptir. Şekil 8.3'de görülen araba dosyası 9 nolu bloktan başlayıp 23 nolu blokta sona ermektedir. 9 nolu blok kendinden sonraki 16 nolu bloğun numarasını tutar. Benzer şekilde, diğer bloklar da kendisiden sonra gelen bloğun numarasını tutarlar. MS-DOS, OS/2 ve MS Windows tabanlı işletim sistemleri bu yöntemi kullanırlar.



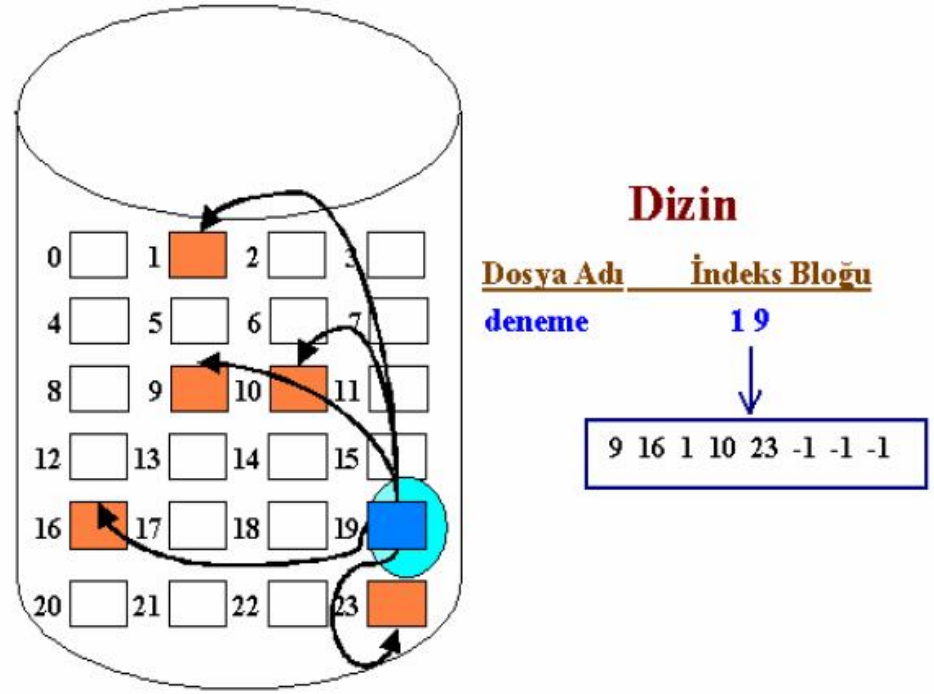

**Dizin**

| Dosya Adı | Baslangıcı | Uzunluğu |
|---|---|---|
| araba | 9 | 23 |

Şekil 8.4 Bağlantılı yerleşim.

# 8.3.3 - İndeksli Yerleşim

İndekli yerleşimde her dosya için bir indeks bloğu oluşturulmuştur. Dizinler dosyaların indeks blok adresini tutar. Bu indeks bloğunda dosyanın yerleşmiş olduğu her bloğun adresi vardır. Böylece her bloğa direkt olarak ulaşabilir. Şekil 8.4'de görülen deneme dosyasının indeks bloğu 19 nolu bloktur. 19 nolu blokta deneme dosyasının yerleşmiş olduğu blokların numaraları bulunmaktadır.




Şekil 8.5 İndeksli yerleşim.

İndeks blokları üç tür yöntemle tutulabilir. Bu yöntemler sırasıyla :

Bağlantılı Şema : İndeks bloğu bir disk bloğundan oluşur. Böylece dosyaya direkt olarak bu disk bloğundan ulaşılır. Eğer dosya bir disk bloğuna sığmayacak kadar büyükse ikinci bir indeks bloğu kullanılır. İlk indeks bloğunun son adres boşluğunda ikinci indeks bloğunun adresi bulunur.

Çok-seviyeli Şema : Bağlantılı şemanın farklı bir gösterimidir. Bu yöntemde çeşitli seviyelerde indeks blokları bulunur. Dosya veri bloklarının adresleri ilk seviyedeki indeks bloğundan başlayarak yerleştirilir. Eğer ikinci bir indeks bloğu kullanılacaksa bu indeks bloğu ikinci seviye indeks bloğu olur. İlk seviye indeks bloğu bu ikinci seviye indeks bloğunun adresini de içerir.

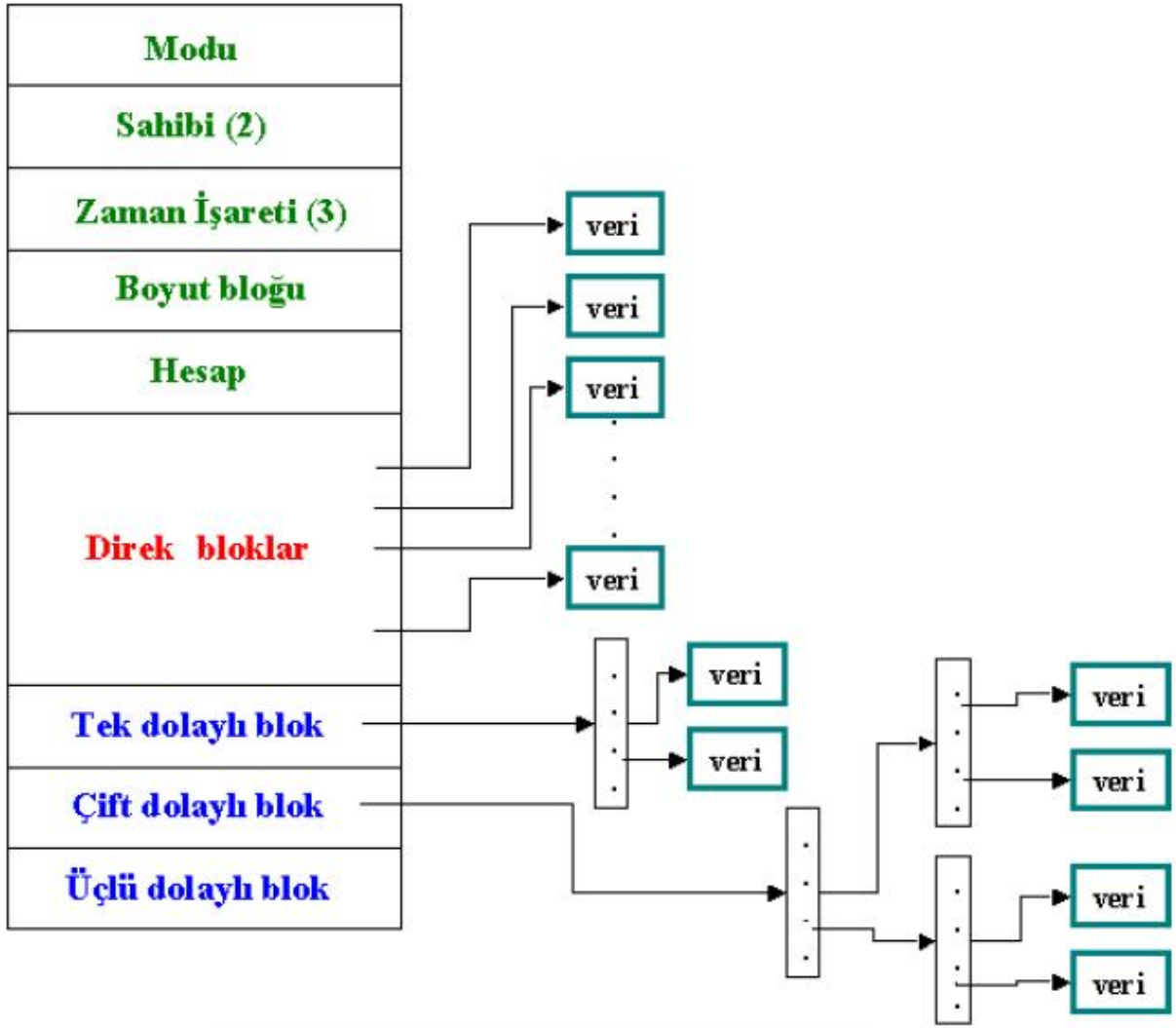
Birleştirilmiş Şema : BSD UNIX sisteminde kullanılan bir yaklaşımdır. Bu yöntemde dosya indeks bloğu içindeki ilk 15 işaretleyici, dolaylı blok ve direk blok olmak üzere iki gruba ayrılmıştır. Şekil 8.6'da bu yöntem görülmektedir. İlk 12 işaretleyici direkt blok bölümüne aittir. Dosya eğer 12 bloktan küçük ise dosya yerinin belirtilmesi için bu işaretleyiciler kullanılır.

Geri kalan 3 işaretleyi ise dolaylı blok bölümüdür. Bu bölüm tek dolaylı blok, çift dolaylı blok ve üçlü dolaylı blok olmak üzere üçe ayrılır.

- Tek dolaylı blok : Bir indeks bloğunun adresini tutar.

- Çift dolaylı blok : Dosya eğer birden fazla indeks bloğuna sahipse bu işaretleyici kullanılır. Bu işaretleyici, dosyanın sahip olduğu indeks bloklarının adresini tutan bir indeks bloğunun adresini tutar.

- Üçlü dolaylı blok : Dosyanın sahip olduğu indeks bloklarının sayısı eğer bir indeks bloğuyla gösterilemiyor ve ikinci ya da üçüncü bir indeks bloğu kullanılıyorsa, indeks bloklarının yerini gösterecek bu indeks blokları için bir indeks bloğu oluşturulur. Bu indeks bloğunun adresi bu işaretleyicide tutulur.




Şekil 8.6 Birleştirilmiş Şema yöntemi.

## 8.4 Disk Çizelgelemeleri

İkincil bellek üzerinde çok sayıda okuma/yazma işlemi yapılmaktadır. Okuma/yazma işlemlerinin çok hızlı gerçekleştirilmesi gerekmektedir. Disk hızını belirleyen üç etken vardır. Bunlar;

- Sistem öncelikle kafayı uygun iz ya da silindir üzerine hareket ettirir. Bu harekete " ARAMA " geçen süreyede " ARAMA SÜRESİ " denir
- Kafa doğru iz üzerinde, fakat iz üzerinde blok kafanın bulunduğu yere uzak ise, bu iz üzerindeki bloğun okuma yazma kafasına gelinceye kadar geçen süreye " GİZLİLİK ZAMANI " denir.
- Son olarak bilgi diskten okunarak ana belleğe transfer edilinceye kadar geçen süreye de "TRANSFER SÜRESİ " denir.

Bir diskin okuma/yazma isteğine cevap verme süresi bu üç sürenin toplamıyla bulunur. Arama süresinin kısa olması ve kafanın istenilen bloğa hızlı ulaşabilmesi için, çeşitli disk çizelgeleme algoritmaları gerçekleştirilmiştir. Bu algoritmalardan bazıları aşağıda verilmiştir.

- İlk gelen ilk hizmet disk çizelgeleme algoritması
- SCAN disk çizelgeleme algoritması
- C-LOOKdisk çizelgeleme algoritması
- C-SCAN disk çizelgeleme algoritması
- SSTF disk çizelgeleme algoritması

# 8.4.1 İlk Gelen İlk Hizmet Disk Çizelgelemesi

İlk gelen ilk hizmet disk çizelgelemesi en basit disk çizelgelemesidir. Disk kuyruğundaki sıraya göre okuma-yazma kafası izler üzerinde hareket ettirilir.

Örnek :

Disk kuyruğu : 68,123, 57, 128, 44, 184, 95, 32
Okuma-Yazma Kafası başlama konumu: 78 izde.
Bu algoritmaya göre gerçekleştirilecek olan işlem şu sırayla olacaktır.

78. izde bulunan okuma-yazma kafası önce 68. ize sonra sırayla 123, 57, 128, 44, 184, 95 ve 32 izlere hareket edecektir.

Aşağıda bu algoritmayı gösteren bir simülasyon olayı gerçekleştirilmiştir. Simülasyonda okuma-yazma kafası başlangıçta 90 izde bulunmaktadır. Kutulara okuma-yazma kafasının gitmesini istediğiniz izlerin numaralarını girin.

Simülasyon

# 8.4.2 Scan Disk Çizelgelemesi

SCAN disk çizelgeleme algoritmasında okuma yazma kafası bulunduğu konumdan (başlama konumu) sıfırıncı iz yönüne doğru hareket ederek, bu alan içinde bulunan işlemleri gerçekleştirir Sıfırıncı ize ulaştıktan sonra son iz yönünde hareket etmeye başlar. Son iz yönüne doğru hareket ederken başlama konumu ile son iz arasındaki izler üzerinde hareket eder.

Örnek :

Disk kuyruğu : 68,123, 57, 128, 44, 184, 95, 32
Okuma-Yazma Kafası başlama konumu: 78 iz.
Bu algoritmaya göre gerçekleştirilecek olan işlem şu sırayla olacaktır.

78. izde bulunan okuma-yazma kafası önce 68. ize sonra sırayla 57, 44, 32 ve ordan 0 ize hareket eder. Daha sonra sırasıyla başlama konumu olan 78. izden sonraki 95. 123. 128 ve 184'üncü izler üzerine hareket eder.

Simülasyon

## 8.4.3 C-Scan Disk Çizelgelemesi

C-Scan disk çizelgeleme algoritmasında okuma-yazma kafası bulunduğu konumdan son iz yönünde hareket ederek bu alan içinde bulunan işlemleri gerçekleştirir. Daha sonra, son ize ulaştığında buradan hızlı bir şekilde ilk ize dönerek, ilk iz ile ilk konumu arasında kalan işlemleri yapar.

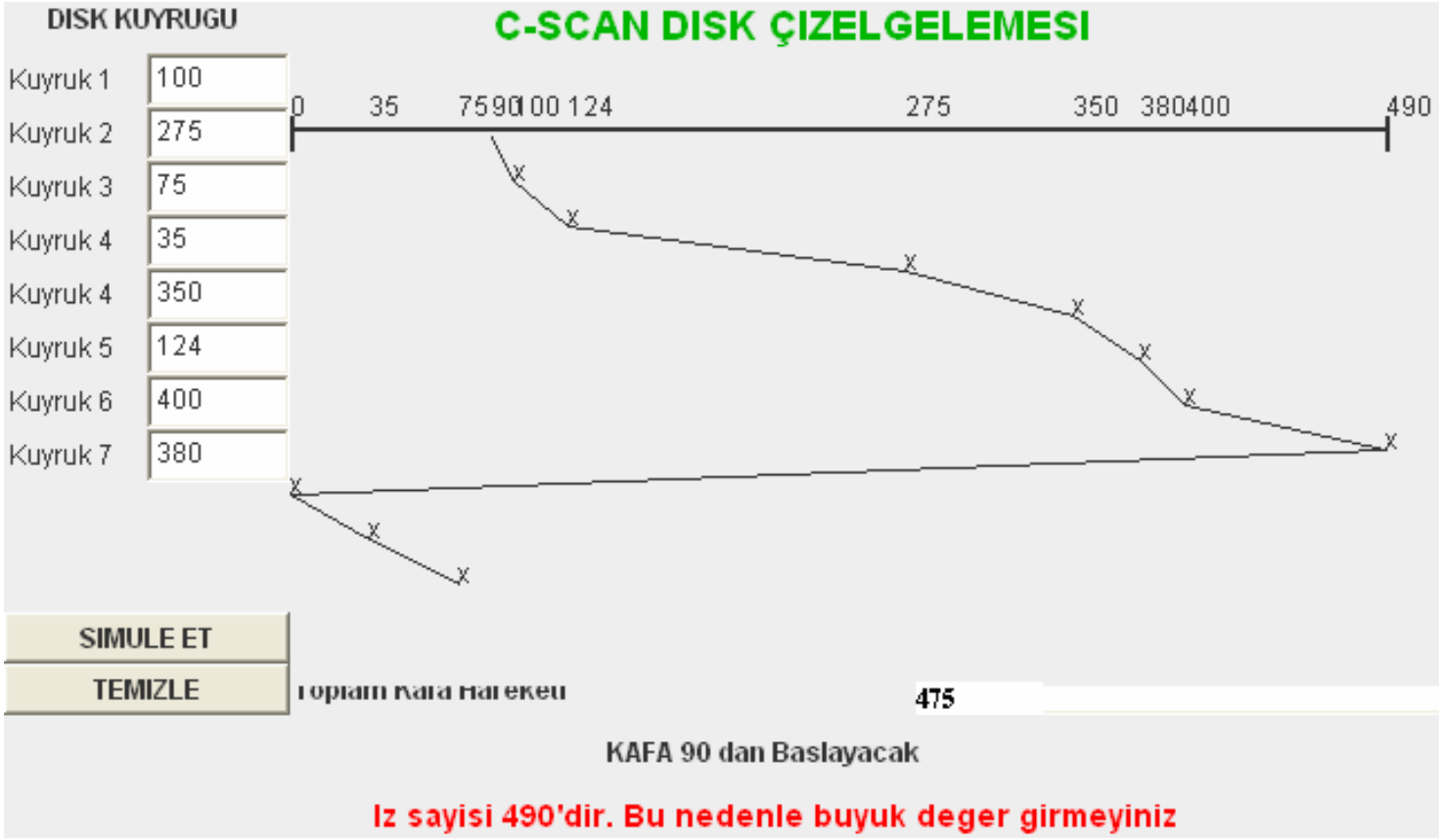
Örnek :

Disk kuyruğu : 68,123, 57, 128, 44, 184, 95, 32
Okuma-Yazma Kafası: 78 izde.
Bu algoritmaya göre gerçekleştirilecek olan işlem şu sırayla olacaktır.

78. izde bulunan okuma-yazma kafası önce 95 . ize sonra sırayla 123, 128 ve 184'üncü oradanda en son ize hareket edecektir. Daha sonra hızlı bir şekilde 0. ize geri dönerek sırayla 32, 44,57 ve oradan 68 ize hareket ederek işlemlerini tamamlayacaktır.

Aşağıda bu algoritmayı gösteren bir simülasyon olayı gerçekleştirilmiştir. Simülasyonda okuma-yazma kafasının başlama konumu 90 izdir. Kutulara okuma-yazma kafasının gitmesini istediğiniz izlerin numaralarını girin.

Simülasyon




8.4 - C-Scan disk çizelgeleme algoritması simülatörü

# 8.4.4 C-Look Disk Çizelgelemesi

C-Look disk çizelgeleme algoritmasında, okuma yazma kafası başlama konumundan en son iz yönünde hareket eder. En son ize yakın en son işlem gerçekleştirildikten sonra, ilk ize en yakın işlemden başlayarak başlama konumundan önceki işlemleri yapar.

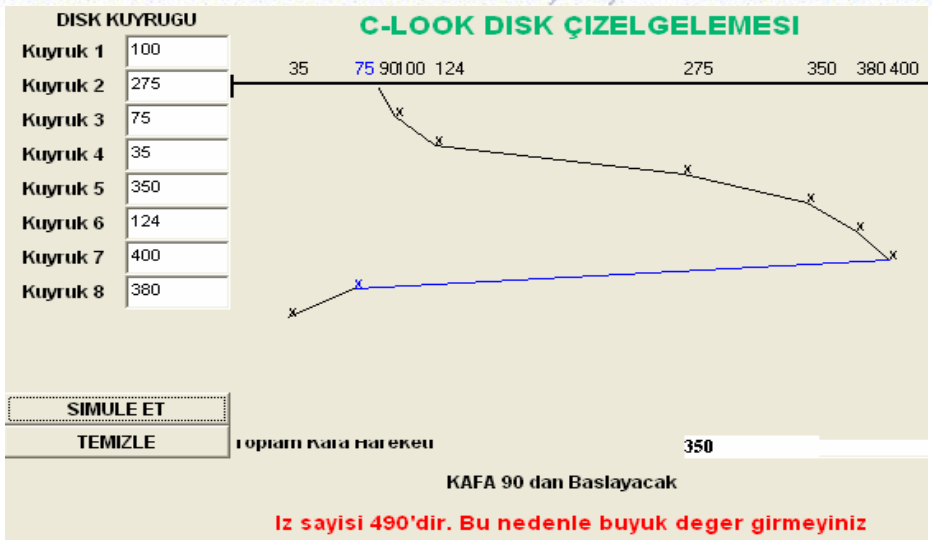
Örnek :

Disk kuyruğu : 68,123, 57, 128, 44, 184, 95, 32
Okuma-Yazma Kafası: 78 izde.

Bu algoritmaya göre gerçekleştirilecek olan işlem şu sırayla olacaktır.

78. izde bulunan okuma-yazma kafası önce 95. ize sonra sırayla 123, 128 ve 184. izdeki işlemleri gerçekleştirir. Buradan, ilk konumu olan 78. izden önce bulunan 32, 44, 57 ve 68'inci izlere hareket ederek işlemlerini gerçekleştirir.

Simülasyon



## 8.4.5 SSTF Disk Çizelgelemesi

SSTF disk çizelgeleme algoritmasında okuma yazma kafası bulunduğu konumdan (başlama konumu), arama zamanının en düşük olduğu iz üzerine hareket ederek işlemlerini gerçekleştirir.

Örnek :

Disk kuyruğu : 68,123, 57, 128, 44, 184, 95, 32
Okuma-Yazma Kafası başlama konumu: 78 iz.
Bu algoritmaya göre gerçekleştirilecek olan işlem şu sırayla olacaktır.

78. izde bulunan okuma-yazma kafası arama zamanı en düşük olan 68. ize hareket eder. Daha sonra sırasıyla 57, 44 ve 32 izler yönüde hareket eder. 32 izden sonra arama zamanı en düşük olan 95 ize hareket eder. Buradan da sırası ile 123, 128 ve 184'üncü izlere hareket eder.

# 9 - DOSYA YÖNETİMİ

## 9.1 - Giriş

***Dosya:*** Dosya, bilgilerin ve programların saklandığı yerdir.

***Dizin:*** Dizin, bilgiler ve programlarla birlikte dosyalarında saklanabildiği yerlerdir.

Dosya yönetimi işletim sisteminin sunduğu temel hizmetlerden biridir. Dosyalar, disk, disket, teyp gibi, çeşitli ikincil saklama alanlarında saklanır. Tüm bu saklama aygıtlarının kendine özel karakteristik yapısı vardır. İşletim sistemi, bu yapıya uygun dosya yönetimini gerekli aygıt üzerinde gerçekleştirir.

### 9.1.1 - Dosya İşlemleri

İşletim sisteminin, dosyayla ilgili sunduğu hizmetler :

1- Dosya yaratılması

2- Dosyanın yazılması

Dosyanın, disk üzerindeki bir dizin içine yazılabilmesi için, işletim sistemi dosyanın yerleştirileceği dizini bulur. Her dizin, içine yazılan dosya bilgilerini tutan bir yazma işaretleyicisine sahiptir. Bu işaretleyiciye göre dosyanın yerleşeceği adres bloğu bulunur ve dosya bu yere yazılır. İşaretleyicinin bilgileri bu işlemden sonra güncellenir.

3- Dosyanın okunması

Dosyanın disk üzerinde okunması için, işletim sistemi dosyanın yerleştiği dizini bulur. Her dizin, bir okuma işaretleyicisine sahiptir. Bu işaretleyici bilgisine göre dosya dizin üzerinden okunur.
Bir çok işletim sisteminde, dizinlerde bulunan okuma ve yazma işaretleyicileri yerine "aktif dosya pozisyon" işaretleyicisi bulunur. Bu işaretleyici her iki işaretleyicinin yaptığı işi tek başına yapar.

4- Dosyanın yerinin yeniden düzenlenmesi

Yeri yeniden düzenlenecek olan dosyaya ilişkin dizin bulunur. Bu dizinin aktif dosya işaretleyicisinde, belirtilen dosya ile konum bilgisi yeniden düzenlenir.

5- Dosyanın silinmesi

Silinecek olan dosyaya ilişkin dizin bulunur. Aktif dosya işaretleyicisinden bu dosya ile ilgili olan yer çıkartılır. Böylece dosyanın bulunduğu yer, diğer dosyalar için kullanılabilir hale gelir.

## 9.1.2 - Dosya Tipleri

İşletim sistemi değişik dosya tiplerini tanıyabilmelidir. Eğer bir işletim sistemi bir dosya tipini tanıyamıyorsa, o tipteki dosyalar ile ilgili çalıştırma, okuma-yazma gibi işlemleri gerçekleştiremez. Dosya tiplerinin belirtilmesi için kullanılan en yaygın yöntem dosya adının iki alana ayrılarak, son üç karakter ile dosyanın tipini gösterilmesidir. Tablo 9.1 değişik dosya tipleri, uzantıları ve işlevleri görülmektedir.

Tablo 9.1 Genel dosya tipleri

| Dosya Tipi | Uzantısı | İşlevi |
|---|---|---|
| Çalıştırılabilir dosya | exe,com,bin yada hiçbiri | makina dili programı olarak çalıştırılmaya hazır |
| Nesne dosyası | obj, o | makina diline çevrilmiş fakat bağlanmamış |
| Kaynak kodu dosyası | c, p, pas, f77, asm, a, java | çeşitli programlama dillerinin kaynak kodları |
| Toplu işlem dosyası | bat, sh | komut yorumlayıcılar için komutlar |
| Metin dosyası | txt, rtf | dokümalar ya da metin belgeleri |
| Kelime işlem dosyası | wp, tex, rrf, doc | çeşitli kelime işlemci formatındaki belgeler |
| Kütüphane dosyası | lib, a | programcılar için oluşturulmuş rutin kütüphaneleri |
| Yazıcı ya da görüntü dosyası | ps, dvi, gif | Yazıcı için oluşturulmuş binary ya da ASCII biçimindeki dosyalar |
| Arşiv dosyası | arc, zip, tar | bir ya da birden fazla dosyanın sıkıştırılarak bir tek dosya içinde toplanmasıyla oluşan dosyalar |

### 9.1.3 - Dizin İşlemleri

İşletim sistemi dizin yapısıyla ilgili aşağıdaki hizmetleri sunar.

1- Bir dizin içindeki dosyanın aranması.
2- Bir dizin içine yeni dosyaların yaratılması.
3- Dizin içinde gerekli olmayan dosyaların silinmesi.
4- Dizinin listelenmesi.
5- Dizinin yedeklenmesi.
6- Bir dizin içindeki dosya isminin değiştirilmesi.

## 9.2 Erişim Yöntemleri

Dosyalar verileri saklarlar. Bir dosyanın kullanılması istenildiğinde mutlaka o dosyaya erişilmeli ve dosyadan okunmalıdır. Dosya erişimi için çeşitli metodlar geliştirilmiştir. Bu metodlar sırayla aşağıda incelenecektir.