



## BTEP205 - İşletim Sistemleri

### Bellek Yönetimi

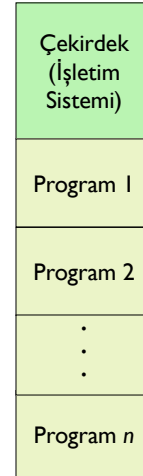
2

## Bellek Yönetimi

- ▶ İşletim sistemlerinde, gerçekleştirilen işlemlerin geçici olarak saklandığı merkeze **ana bellek** (RAM) denir.
- ▶ Ana bellek, giriş-çıkış aygıtlarının kolaylıkla ulaşabildiği bir bilgi deposudur.
- ▶ Merkezi işlem birimi (MİB), işlemleri ana bellekten alır ya da ana belleğe koyar.

## Bellek Yönetimi

- Çok görevli sistemlerde, ana belleğin işletim sistemi kullanımı dışında olan kısım, farklı programların kullanımı için değişik parçalara (bitişken alanlara) bölünür.



## Bellek Yönetimi

- Ana belleğin işlemler arasında paylaştırılmasına ana bellek yönetimi ya da **bellek yönetimi** (memory management) adı verilir.
- İşletim sisteminin bu amaçla oluşturulan kesimine de **bellek yöneticisi** (memory manager) adı verilir.

## Bellek Yönetimi

- ▶ Bellek yöneticisinin başlıca görevleri:
  - ▶ Belleğin hangi parçalarının kullanımda olduğunu, hangi parçalarının (bitişken alanların) kullanılmadığını izlemek,
  - ▶ Süreçlere (işlemlere) bellek tahsis etmek (allocate), tahsis edilen belleği geri almak (deallocate),
  - ▶ Bellek ile disk arasındaki yer değiştirme (swap) işlemlerini gerçekleştirmektir.

## Bellek Yönetimi

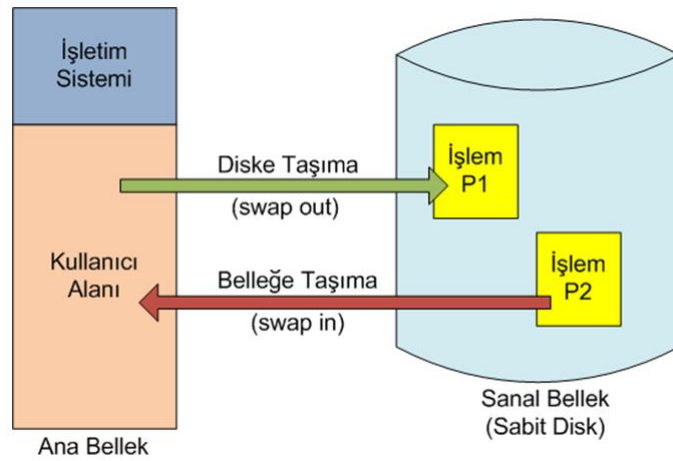
- ▶ **Yer Değiştirme (Swap)**
  - ▶ Çalışma zamanında süreçler bellek ile disk arasında sürekli yer değiştirir.
  - ▶ Süreçlerin bu şekilde disk ile bellek arasında yer değiştirilmesinin nedeni, belleğin boyutunun **yetersiz** olmasıdır.
  - ▶ Sisteme sunulan iş, o an sistemde işletimde olan işlerden daha öncelikli ise hemen işleme alınması gerekir.
  - ▶ Bu durumda, eğer ana bellekte yeterli alan yoksa, daha az öncelikli görevlerin, işletimlerini sonradan tamamlanmak üzere geçici olarak diske taşınması ve boş bellek alanı oluşturulması gerekir.

## Bellek Yönetimi

### ► Yer Değiştirme (Swap)

- Bu yolla açılan boş bellek alanları yeni görevlerin tanımlanabilmesine ve öncelikli işlerin çalışmasına olanak sağlar.
- Bellekte boş alan oluşturulunca işlemin buraya taşınmasına **belleğe taşıma (swap in)** denir.
- İşletimi tamamlanmamış bir görevin, daha öncelikli görevlere ana bellekte yer açmak üzere geçici olarak diskteki özel alanlara (sanal bellek) taşınmasına **diske taşıma (swap out)** denir.

## Bellek Yönetimi



## Bellek Yönetimi

- ▶ Yer Değiştirme yöntemi kullanıldığında ve ana bellekte yer açmak gerektiğinde, hangi görevin diske taşınması gerektiğine karar verilmelidir.
- ▶ Bekleyen görevler arasında, bellek alanı elinden alınacak görevi seçmek için aşağıdaki kriterleri göz önünde bulunduran değişik algoritmalar kullanılır.
  - ▶ Görev önceliği
  - ▶ En uzun kaynak bekleme süresi kalan
  - ▶ Ana belleği o ana kadar en çok kullanan
- ▶ Bazı görevler, yerine getirdikleri hizmetlerin önemi nedeniyle sistemden uzaklaştırılmazlar.

## Bellek Bölümleme (Memory Partitioning)

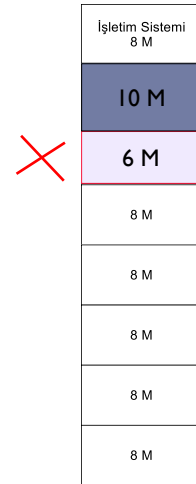
- ▶ Ana bellek, birden fazla program arasında paylaştırılmalı ve bölünmelidir.
- ▶ Bunu yapabilmek için kullanılan yöntemler:
  - ▶ Değişmez Bölümlü Bellek Yönetimi
  - ▶ Devingen Bölümlü Bellek Yönetimi
  - ▶ Sayfalı Bellek Yönetimi
  - ▶ Kesimli Bellek Yönetimi

## Değişmez Bölümlü Bellek Yönetimi

- ▶ Bu yöntemde ana bellek, işletim sistemi ile birden çok kullanıcı programı arasında paylaşılır.
- ▶ Ana bellek, bitişken, irili ufaklı, birden çok bölüm olarak düzenlenir.
- ▶ Bir iş, işleme alınmadan önce, ana bellekte kendisine, boyuyla uyumlu bir bölüm atanır.
- ▶ İş, kendisine atanan bu bölümü, işletimi tümüyle sonlanana kadar korur.
- ▶ Bölümler, bilgisayar sisteminin işletimi başlamadan önce belirlenen boylarını sistem işleme kapanana kadar korurlar.

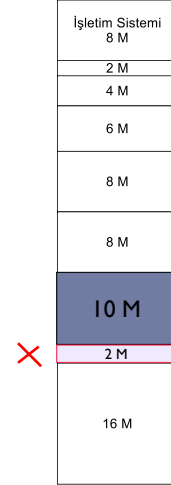
## Değişmez Bölümlü Bellek Yönetimi

- ▶ Değişmez bölümlü bellek yönetiminde bölümler **sabit** bir büyüklükte olabilir.
- ▶ Toplam büyüklüğü 64M, bölüm boyutları da 8M olan bir bellek yan tarafta görünmektedir.
- ▶ 10M bir işlem istek yolladığında 2 bölüm birleştirilecek ve bu istek kaydedilince 6M'lık bir **iç parçalanma** (internal fragmentation) oluşacaktır.



## Değişmez Bölümlü Bellek Yönetimi

- ▶ Değişmez bölümlü bellek yönetiminde bölümler **farklı** büyüklükte olabilir.
- ▶ Toplam büyüklüğü 64M, bölüm boyutları da farklı olan bir ana bellek yan tarafta görünmektedir.
- ▶ 10M bir işlem istek yolladığında en az iç parçalanmanın oluşacağı bölüm bulunacak istek bu bölüme konulacaktır.



## Devingen Bölümlü Bellek Yönetimi

- ▶ Bu yöntemin temel ilkesi, bölümlerin, işlerin görevlere dönüştürülüp sisteme sunulmaları aşamasında, devingen olarak yaratılmasıdır.
- ▶ Bu yöntemde, ana bellekte kullanıcı işlemleri boşluklara bloklar halinde yerleştirilmektedir.
- ▶ Bu yöntemle bir işlem ana belleğe yükleneceği zaman, işlem için gerekli miktarda yeni bir bölüm oluşturularak, işlem oluşturulan yeni bölüme yüklenir.

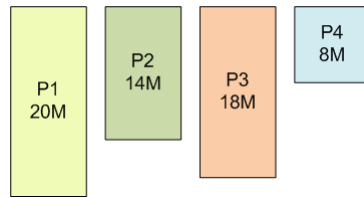
## Devingen Bölümlü Bellek Yönetimi

- ▶ Yeni bir işlem geldiğinde bu belleğe işlemin yeteceği kadar bir alan ayrılarak konumlandırılmakta ve işletim sistemi boş alanlar ve dolu alanlar ile ilgili bilgileri kayıt altına almaktadır.
- ▶ Bölümlerle ilgili bölüm başlangıç adresi, boyu gibi bilgiler tutulmaktadır.
- ▶ Bu yöntemde görevlere atanan bölümlerin yanı sıra, bu bölümler arasında kalan boş alanların da izlenmesi gereklidir.

## Devingen Bölümlü Bellek Yönetimi

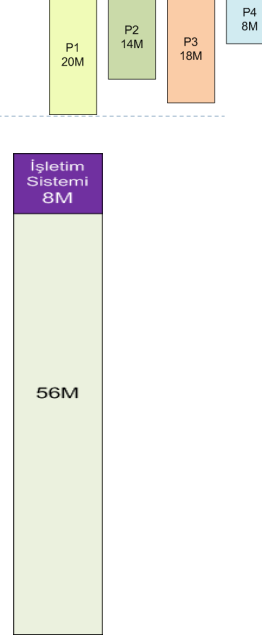
### Örnek:

- ▶ 64M ana belleğin aşağıdaki dört program için kullanılacağını varsayınız.



## Devingen Bölümlü Bellek Yönetimi

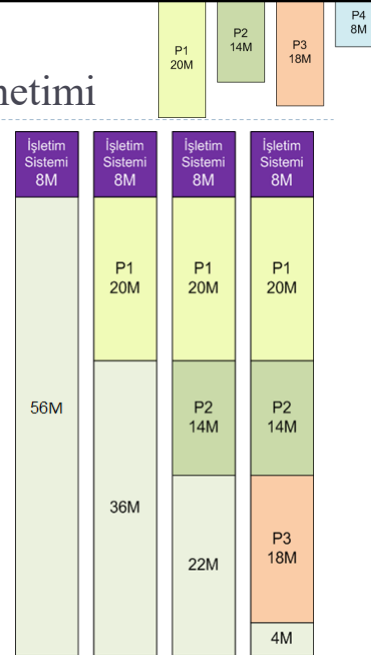
- ▶ Başlangıçta ana belleğin işletim sistemi dışında boş olduğunu varsayınız.
- ▶ Programlar birbirleri ardına gerekli bölümleri oluşturarak ana belleğe yüklenirler.



▶ BTEP205 - İşletim Sistemleri

## Devingen Bölümlü Bellek Yönetimi

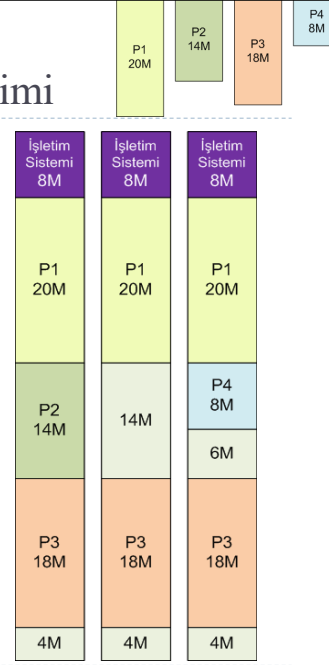
- ▶ Bu durumda P4 işlemi için bellekte yeterli yer yoktur.
- ▶ Yer değiştirme (swap) işlemiyle bir işlem sanal belleğe taşınır.
- ▶ P2'nin yer değiştireceğini varsayalım.



▶ BTEP205 - İşletim Sistemleri

## Devingen Bölümlü Bellek Yönetimi

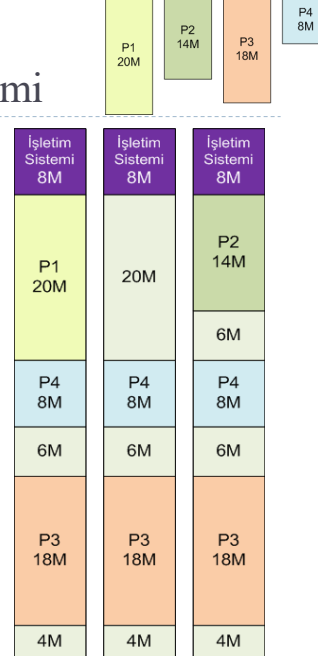
- P2'nin yer değiştirmesinden sonra bellekte 14M'lık bir bölüm serbest kalır.
- Bu bölüme P4 yüklenebilir.



► BTEP205 - İşletim Sistemleri

## Devingen Bölümlü Bellek Yönetimi

- P2'nin çalışması için tekrar ana belleğe yüklenmesi gerektiğini varsayalım.
- Bu sefer P1'in bekleme durumunda olduğunu ve yer değiştireceğini varsayalım.
- Yer değiştirme olayından sonra P2 tekrardan ana belleğe yüklenebilir.

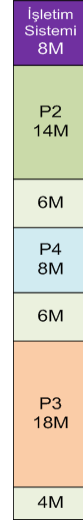


► BTEP205 - İşletim Sistemleri

## Devingen Bölümlü Bellek Yönetimi

### Ana Belleğin Parçalanması Sorunu

- ▶ Ana belleğin parçalanması, bitişken alanların görevlere atanan bölümlerle, zaman içinde ufalanması olarak tanımlanır.
- ▶ Bu sorun, kullanılan bölümler arasına sıkışmış, işletim için bekleyen görevlerin gereksinimini karşılayamayan boş alanların varlığıyla ortaya çıkar.
- ▶ Belleğin parçalanması sonucu, bellekteki boş alanların toplamı, gerekli sığmaları karşılıyor olmasına karşın yeni görevlere yer sağlanamaz durumlarla karşılaşılır.



## Devingen Bölümlü Bellek Yönetimi

- ▶ Görevlere sağlanan alanların konumlarının işletim sırasında değiştirilememesi parçalanma sorununun temel nedenleridir.

- ▶ Bölüm içi yararlanılamayan boş alanlar **iç parçalanma**, bölümler arasında kalan boş alanlar ise **dış parçalanma** olarak adlandırılır.

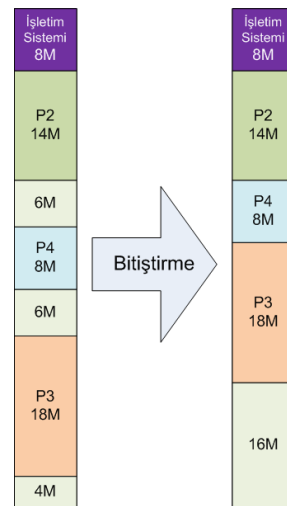
- ▶ Devingen bölümlü bellek yönetiminde bölümler arasında boş alanlar kalmakta, yani dış parçalanma oluşmaktadır.

## Bitiştirme (Compaction)

- ▶ Tüm bellek alanına dağılmış durumdaki bölümleri, yerlerini değiştirerek yan yana yerleştirip tek bir bitişken boş alan yaratma işlemine **bitiştirme işlemi (compaction)** denir.
- ▶ Verilen örnekte, görevlere bellek ataması yapılması ve bir göreve atanan bellek konumunun işletim sırasında değiştirilememesi sonucu bitişkin alanların parçalandığı görülmektedir.
- ▶ Bu sakıncalı durumun yok edilebilmesi için bitiştirme işlemi kullanılabilir.

## Bitiştirme (Compaction)

- ▶ Bitiştirme işlemiyle tüm kullanılmayan parçalar bir yerde toplanır.
- ▶ Bitiştirme işlemi, işlemciyi meşgul eden, zaman alan ve dolayısıyla bilgisayarı yavaşlatan bir işlemdir.
- ▶ Bitiştirme ihtiyacını azaltmak için farklı **yerleştirme algoritmaları** kullanılır.



## Yerleştirme Algoritmaları

- ▶ İşletim sistemi, bellekteki boşlukların bulunduğu listeden işlem için en uygun boşluğu belirlemede 3 başlıca strateji kullanır:
  - ▶ İlk Uygun Yer Algoritması (First Fit)
  - ▶ Sonraki Uygun Yer Algoritması (Next Fit)
  - ▶ En Uygun Yer Algoritması (Best Fit)

## Yerleştirme Algoritmaları

### İlk Uygun Yer Algoritması

- ▶ Bir süreç (işlem) bellek isteğinde bulunduğu anda, bellek baştan sona taranır ve süreç için gerekli olan bellek boyutunu sağlayan ilk boş alana yerleştirilir.
- ▶ Gerçekleştirilmesi kolay ve en hızlı çalışan algoritmadır.

## Yerleştirme Algoritmaları

### Sonraki Uygun Yer Algoritması

- ▶ İlk uygun yer algoritması ile aynı mantıkta çalışır, fakat bu algoritma en son bulduğu uygun yer bilgisini saklar.
- ▶ Bir sonraki aramada belleğin en başından değil de, saklamış olduğu noktadan itibaren aramaya başlar.
- ▶ İlk uygun yer algoritmasına göre biraz daha yavaş çalışır.

## Yerleştirme Algoritmaları

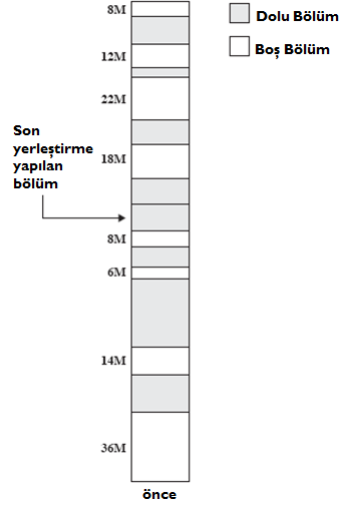
### En Uygun Yer Algoritması

- ▶ Bu algoritma belleğin başından sonuna kadar tüm boş alanları tarar ve süreç için gerekli olan bellek boyutuna en uygun olan (en yakın boyuttaki) boş bellek alanına süreci yerleştirir.
- ▶ En yavaş algoritma çalışan algoritmadır.
- ▶ Çok küçük ve kullanışsız parçalanmalara sebep olur.

## Yerleştirme Algoritmaları

### Örnek 1

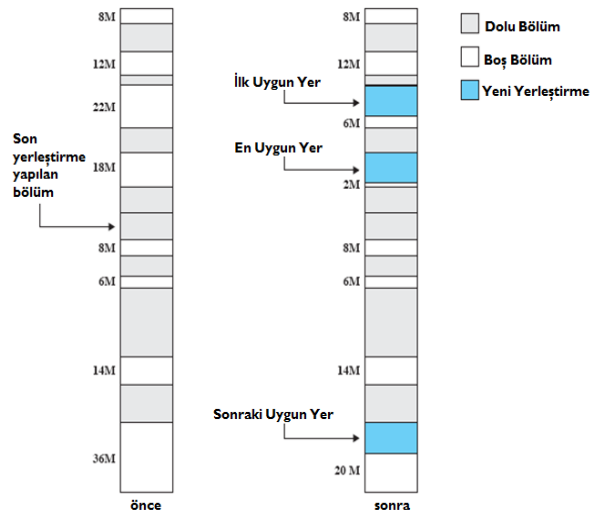
- ▶ Ana bellekteki boş ve dolu bölümlerin yan tarafta görüldüğü gibi olduğunu varsayınız.
- ▶ 16M'lık bir yerleştirme isteğini ilk uygun, sonraki uygun ve en uygun yer algoritmasına göre belleğe yerleştiriniz.



▶ BTEP205 - İşletim Sistemleri

## Yerleştirme Algoritmaları

- ▶ **İlk uygun** yer algoritması 22MB'lık bölümü kullanarak 2MB'lık dış parçalanma, **sonraki uygun** yer algoritması son 36MB'lık bölümü kullanarak 20MB'lık dış parçalanma, **en uygun** yer algoritması da tüm yerlere bakarak 18M'lık bölümü kullanarak, 2M'lık dış parçalanma oluşturur.



▶ BTEP205 - İşletim Sistemleri

## Yerleştirme Algoritmaları

### Örnek 2

- Ana bellekte sırayla aşağıdaki boş bölümlerin olduğunu varsayınız.

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 40 K     | 36 K     | 14 K     | 18 K     | 24 K     | 30 K     |

- 24K, 20K ve 18K'lık bellek kullanım istekleri için sırasıyla hangi boşlukların kullanılacağını ve oluşacak parçalanmaları aşağıdaki algoritmaları kullanarak belirtiniz.
- İlk Uygun Yer Algoritması
  - Sonraki Uygun Yer Algoritması
  - En Uygun Yer Algoritması

► BTEP205 - İşletim Sistemleri

## Yerleştirme Algoritmaları

### İlk Uygun Yer Algoritması

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 40 K     | 36 K     | 14 K     | 18 K     | 24 K     | 30 K     |

**24 K Boşluk 3 (16K parçalanma)**

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 24 K     | 16 K     | 36 K     | 14 K     | 18 K     | 24 K     |
|          |          |          | 30 K     |          |          |          |          |

**20 K Boşluk 1 (0K parçalanma)**

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 24 K     | 16 K     | 36 K     | 14 K     | 18 K     | 24 K     |
|          |          |          | 30 K     |          |          |          |          |

**18 K Boşluk 4 (18K parçalanma)**

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 24 K     | 18 K     | 18 K     | 14 K     | 18 K     | 24 K     |
|          |          |          | 30 K     |          |          |          |          |

► BTEP205 - İşletim Sistemleri

## Yerleştirme Algoritmaları

### Sonraki Uygun Yer Algoritması

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 40 K     | 36 K     | 14 K     | 18 K     | 24 K     | 30 K     |

#### 24 K Boşluk 3 (16K parçalanma)

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 24 K     | 16 K     | 36 K     | 14 K     | 18 K     | 24 K     |

#### 20 K Boşluk 4 (16K parçalanma)

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 24 K     | 16 K     | 20 K     | 16 K     | 14 K     | 18 K     |

#### 18 K Boşluk 6 (0K parçalanma)

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 24 K     | 16 K     | 20 K     | 16 K     | 14 K     | 18 K     |

► BTEP205 - İşletim Sistemleri

## Yerleştirme Algoritmaları

### En Uygun Yer Algoritması

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 40 K     | 36 K     | 14 K     | 18 K     | 24 K     | 30 K     |

#### 24 K Boşluk 7 (0K parçalanma)

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 40 K     | 36 K     | 14 K     | 18 K     | 24 K     | 30 K     |

#### 20 K Boşluk 1 (0K parçalanma)

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 40 K     | 36 K     | 14 K     | 18 K     | 24 K     | 30 K     |

#### 18 K Boşluk 6 (0K parçalanma)

| BOŞLUK 1 | BOŞLUK 2 | BOŞLUK 3 | BOŞLUK 4 | BOŞLUK 5 | BOŞLUK 6 | BOŞLUK 7 | BOŞLUK 8 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 20 K     | 8 K      | 40 K     | 36 K     | 14 K     | 18 K     | 24 K     | 30 K     |

► BTEP205 - İşletim Sistemleri

## Sayfalı Bellek Yönetimi

- ▶ İç ve dış parçalanma sorununun üstesinden gelmek için **sayfalı bellek yönetimi (paging)** kullanılabilir.
- ▶ Fiziksel bellek üzerindeki aynı uzunluktaki bloklara **çerçeve (frame)**, mantıksal bellek üzerindeki aynı uzunluktaki bloklara ise **sayfa (page)** denilmektedir.
- ▶ Bir sistemde çerçeve ve sayfa uzunlukları eşittir. Sayfaların uzunlukları donanım tarafından belirlenmektedir.

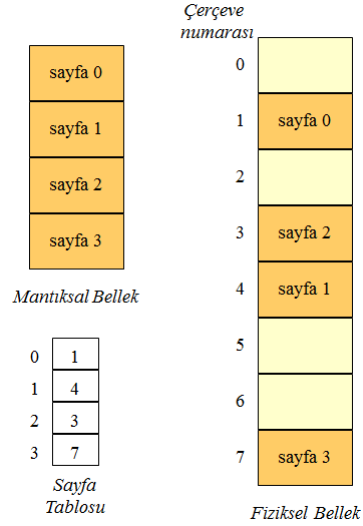
## Sayfalı Bellek Yönetimi

- ▶ Sayfalı bellek yönetimi olan sistemlerde bellek adresi iki bölümden oluşmaktadır: sayfa numarası ve sayfa ofseti.
  - ▶ **Sayfa numarası (p)** fiziksel bellekteki her bir sayfanın taban adresini tutan sayfa tablosundaki göstergedir.
  - ▶ **Sayfa ofseti (d)** taban adresi ile birleştirilerek fiziksel bellekte sayfanın içerisindeki yerin belirlenmesinde kullanılır.

|                    |                  |
|--------------------|------------------|
| Sayfa Numarası (p) | Sayfa Ofseti (d) |
|--------------------|------------------|

## Sayfalı Bellek Yönetimi

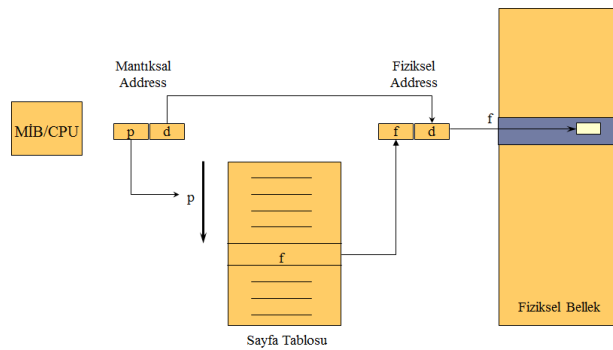
- Her işlem eşit uzunluktaki sayfalara, ana bellek de eşit uzunluktaki çerçevelere ayrılmıştır.
- İşlemlerin hangi sayfasının hangi çerçevede olduğu bilgisi işlemlerin **sayfa tablosunda** tutulmaktadır.
- Yandaki örnekte sayfa 0, çerçeve 1'de, sayfa 1 çerçeve 4'te, sayfa 2 çerçeve 3'te ve sayfa 3 de çerçeve 7'de yer almaktadır.



► BTEP205 - İşletim Sistemleri

## Sayfalı Bellek Yönetimi

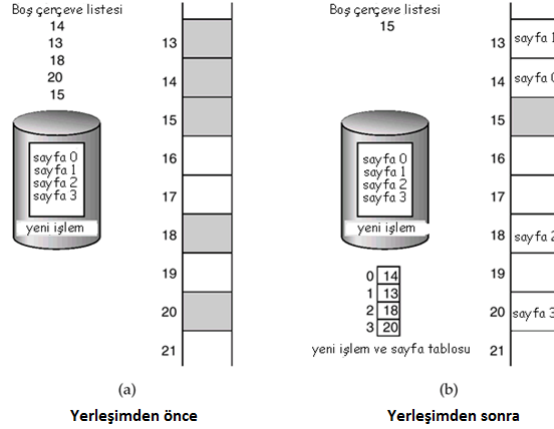
- Mantıksal bellekteki bir adres, sayfa numarası ve sayfa ofsetinden oluşmaktadır.
- Bu adresin fiziksel bellekteki karşılığının bulunması için sayfa tablosu kullanılır.



► BTEP205 - İşletim Sistemleri

## Sayfalı Bellek Yönetimi

- İşletim sistemi bu yöntemi kullanırken boş çerçevelerin de listesini tutar.



► BTEP205 - İşletim Sistemleri

## Sayfalı Bellek Yönetimi

### Örnek

- 4 bayt uzunluğunda sayfalardan oluşan 32 bayt'lık bir bellek olduğunu varsayınız. Sayfa tablosu yanda verilmiştir.
- Buna göre aşağıdaki mantıksal adresler için fiziksel adresleri bulunuz.
  - 0
  - 3
  - 4
  - 13

|   |   |
|---|---|
| 0 | 5 |
| 1 | 6 |
| 2 | 1 |
| 3 | 2 |

Sayfa tablosu

Fiziksel adresi hesaplamak için aşağıdaki formül kullanılabilir:

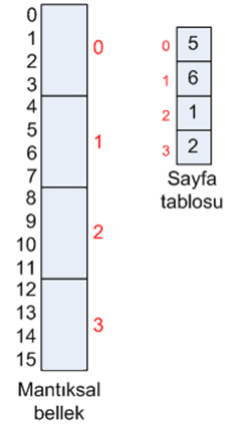
**fiziksel adres (FA)=çerçeve numarası (ÇN)\*çerçeve boyutu (ÇB)+ofset (O)**

► BTEP205 - İşletim Sistemleri

## Sayfalı Bellek Yönetimi

### Örnek

- ▶ İlk olarak mantıksal belleğin çizilmesi gerekir.
- ▶ Ardından sayfa boyutuna göre belleğin bölünmesi ve her bir mantıksal adresin sayfa numarası ve sayfa ofsetinin hesaplanması gerekir.
- ▶ Sayfa numarası, sayfa tablsundan çerçeve numarasını bulmak için kullanılacaktır.



## Sayfalı Bellek Yönetimi

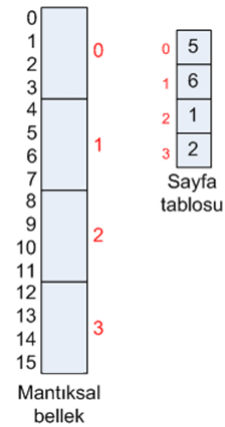
a) 0

sayfa numarası=0      mantıksal adres  $\xrightarrow{\text{Ofset=0}}$  0      Sayfa no =0  
 sayfa ofseti=0      1  
                                  2  
                                  3      (0,0)

0 numaralı sayfanın bulunduğu çerçeve numarası=5

$$FA = \text{ÇN} * \text{ÇB} + O$$

$$FA = 5 * 4 + 0 = 20$$

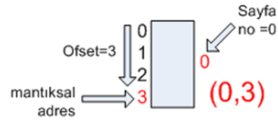


## Sayfalı Bellek Yönetimi

b) 3

sayfa numarası=0

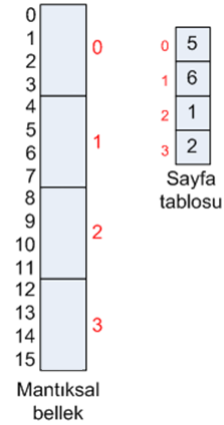
sayfa ofseti=3



0 numaralı sayfanın bulunduğu çerçeve numarası=5

$FA = \text{ÇN} * \text{ÇB} + O$

$FA = 5 * 4 + 3 = 23$

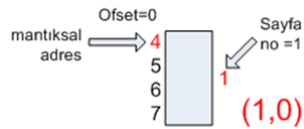


## Sayfalı Bellek Yönetimi

c) 4

sayfa numarası=1

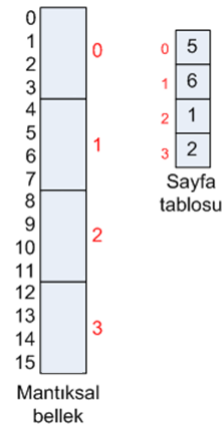
sayfa ofseti=0



1 numaralı sayfanın bulunduğu çerçeve numarası=6

$FA = \text{ÇN} * \text{ÇB} + O$

$FA = 6 * 4 + 0 = 24$



# Sayfalı Bellek Yönetimi

d) 13

sayfa numarası=3

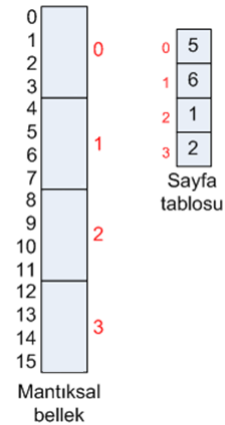
sayfa ofseti=1



3 numaralı sayfanın bulunduğu çerçeve  
numarası=2

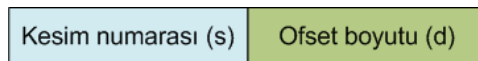
$$\text{FA}=\underset{3}{\text{C}}\text{N}^*\underset{3}{\text{C}}\text{B}+\text{O}$$

$$FA = 2 * 4 + 1 = 9$$



## Kesimli Bellek Yönetimi

- ▶ **Kesimli bellek yönetiminde (segmentation)** mantıksal adres alanı sayfalar yerine kesimlerden (bölümlerden) oluşmaktadır.
- ▶ Bir adres, hem **kesim (bölüm) numarası (s)** hem de kesim içerisindeki adresi belirten **ofset numarasından (d)** meydana gelmektedir.



## Kesimli Bellek Yönetimi

- ▶ Mantıksal adreslerin fiziksel adreslere dönüştürülmesi için **kesim tablosuna** ihtiyaç vardır.
- ▶ Kesim tablosu, her kesimin **başlangıç (taban) adresi** ve **uzunluğu** bilgisini tutmaktadır. Böylece her kesimin başlama ve bitiş (tavan adresi) noktaları hesaplanabilir.

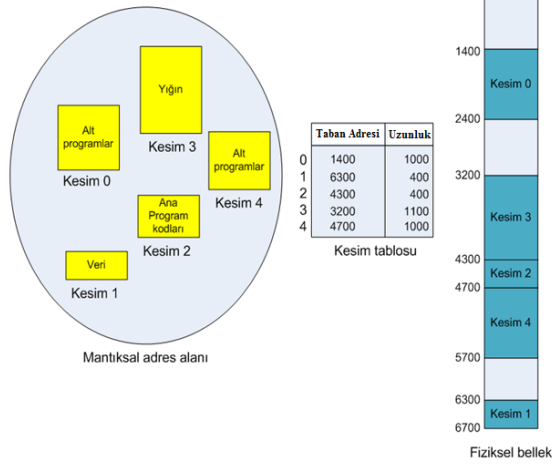
## Kesimli Bellek Yönetimi

- ▶ Bir mantıksal adresin ofset değeri, taban ve tavan değerleri arasında olmalıdır.
- ▶ Eğer ofset değeri fazla olursa mantıksal adresin kesimi aşması **bellek erişim hatası** (segment fault) oluşur.
- ▶ Eğer ofset değeri belirtilen sınırlar arasında ise kesim taban değeri ile ofset adresi toplanarak, fiziksel adres oluşturulmaktadır.

$$\text{Fiziksel Adres (FA)} = \text{Kesim Taban Adresi (TA)} + \text{Ofset (d)}$$

## Kesimli Bellek Yönetimi

- ▶ Mantıksal adres alanının yanda görüldüğü gibi olduğunu varsayınız.
- ▶ Kesim tablosu bu bilgiler kullanılarak elde edilebilir.
- ▶ Fiziksel bellek de kesim tablosu kullanılarak oluşturulabilir.



▶ BTEP205 - İşletim Sistemleri

## Kesimli Bellek Yönetimi

### Örnek

- ▶ Kesim tablosunun yandaki gibi olduğunu varsayınız.
- ▶ Aşağıdaki mantıksal adreslerin (kesim no, ofset) fiziksel adres karşılıklarını bulunuz.
  - (0, 198)
  - (2, 156)
  - (1, 530)
  - (3, 455)
  - (0, 252)

| Kesim No | Taban Adresi | Uzunluk |
|----------|--------------|---------|
| 0        | 660          | 248     |
| 1        | 1752         | 422     |
| 2        | 222          | 198     |
| 3        | 996          | 604     |

Fiziksel adresi hesaplamak için aşağıdaki formül kullanılabilir:

**Fiziksel Adres (FA) = Kesim Taban Adresi (TA) + Ofset (d)**

▶ BTEP205 - İşletim Sistemleri

## Kesimli Bellek Yönetimi

| Kesim No | Taban Adresi | Uzunluk |
|----------|--------------|---------|
| 0        | 660          | 248     |
| 1        | 1752         | 422     |
| 2        | 222          | 198     |
| 3        | 996          | 604     |

- ▶ İlk olarak her mantıksal adresin hangi kesimde olduğu kesim tablosu kullanılarak bulunmalı ve ofset değeri bu kesimin uzunluğu ile karşılaştırılmalıdır.
- ▶ Eğer ofset küçükse fiziksel adres hesaplanmalı, değilse bellek erişim hatası oluşturulmalıdır.

|    | Mantıksal adres | Fiziksel adres   |                      |
|----|-----------------|------------------|----------------------|
| a) | (0, 198)        | $198 < 248$      | $660 + 198 = 858$    |
| b) | (2, 156)        | $156 < 198$      | $222 + 156 = 378$    |
| c) | (1, 530)        | $530 \nless 422$ | bellek erişim hatası |
| d) | (3, 455)        | $455 < 604$      | $996 + 455 = 1451$   |
| e) | (0, 252)        | $252 \nless 248$ | bellek erişim hatası |

▶ BTEP205 - İşletim Sistemleri

52

## Sayfalı ve Kesimli Bellek Yönetimleri Arasındaki Farklar

- ▶ Kesimli bellek yönetiminin amacı adres alanının **mantıksal** olarak dilimlenmesidir. Sayfalı bellek yönetiminde ise bellek **fiziksel** olarak dilimlenip oluşturulur.
- ▶ Sayfalar makine donanımına bağlı olarak **sabit** boyuttadır. Bölümler ise **değişken** boyuttadır.
- ▶ Program adresinin sayfa ve ofset numaralarına ayrılması donanımın bir işlevidir. Ofset numarasının sınırı aşması otomatik olarak sayfanın **çevrilmesine** sebep olur. Oysa kesim ve ofset numarasında bir sınır aşması söz konusu değildir. Bu durumda **bellek erişim hatası** oluşur.

▶ BTEP205 - İşletim Sistemleri

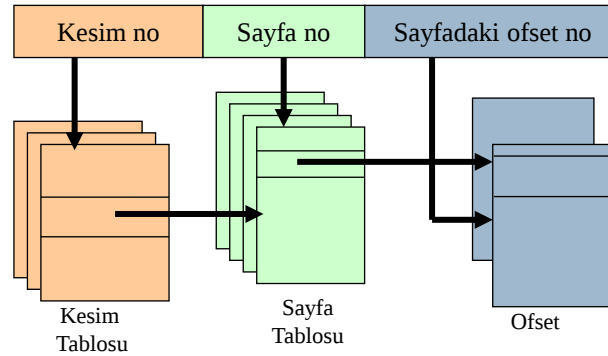
## Kesimlemenin Sayfalama ile Birlikte Kullanılması

- Kesimli bellek yönetimi ile sayfalı bellek yönetimi birlikte kullanılabilir.
- Bu yöntemde bölümler (kesim) kendi içerisinde sayfalara ayrılmaktadır.
- Mantıksal adres üç parçadan oluşmaktadır: **kesim numarası**, **sayfa numarası** ve **offset adresi**.

|                   |                   |                              |
|-------------------|-------------------|------------------------------|
| Kesim<br>numarası | Sayfa<br>numarası | Sayfadaki offset<br>numarası |
|-------------------|-------------------|------------------------------|

## Kesimlemenin Sayfalama ile Birlikte Kullanılması

- Fiziksel adres hesaplanırken kesim tablosu, sayfa tablosu ve ofsete ihtiyaç vardır.



## Bellek Yönetimi Konu Sonu