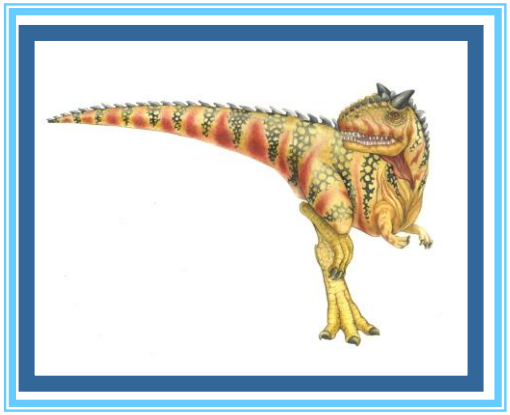
Bölüm 8: Bellek Yönetim Stratejileri



Silberschatz, Galvin and Gagne ©2009

Bölüm 8: Bellek Yönetim Stratejileri

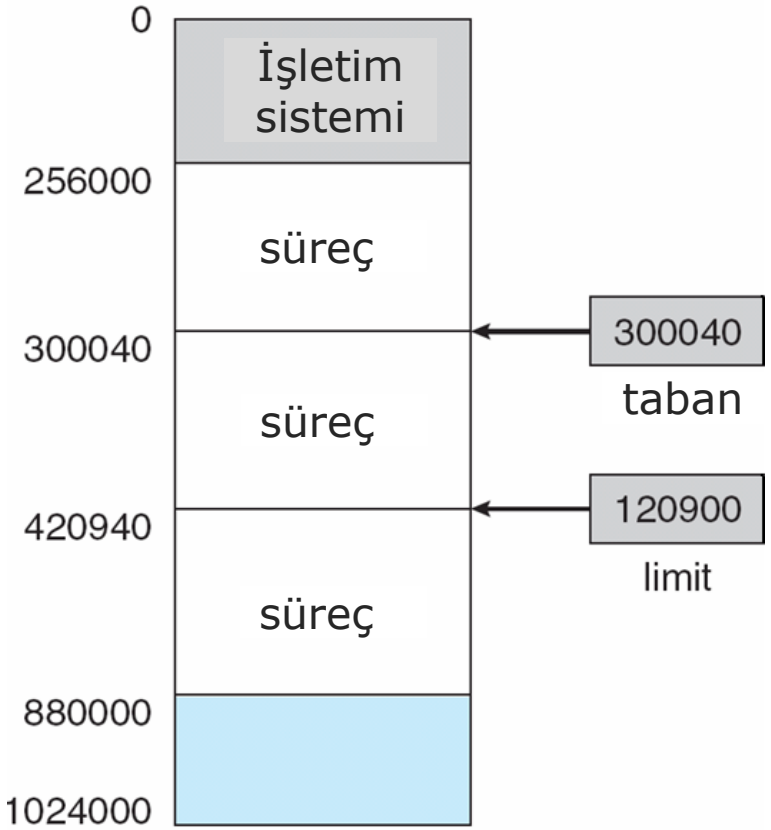
- Altyapı
- Değiş tokuş (Swapping)
- Ardışık Bellek Tahsisi (Contiguous Memory Allocation)
- Sayfalama (Paging)
- Sayfa Tablosunun Yapısı
- Parçalama - Segmentasyon (Segmentation)
- Örnek: The Intel Pentium

Altyapı

- Program, belleğe (diskten) getirilmiş ve çalıştırılmak için bir sürecin içine yerleştirilmiş olmalı
- Ana bellek ve Kayıtçılar (register) CPU'nun direk olarak erişebildiği tek depolama alanlarıdır
- Kayıtçıya bir CPU saat çevriminde (veya daha az sürede) erişilir
- Ana belleğe erişim ise daha uzun çevrimler sürebilir
- Önbellek (Cache), erişim süresi açısından ana bellek ve CPU kayıtçılarının arasında yer alır
- Belleğin korunması için doğru işlemenin garanti edilmesi gereklidir

Taban (Base) ve Limit Kayıtçıları

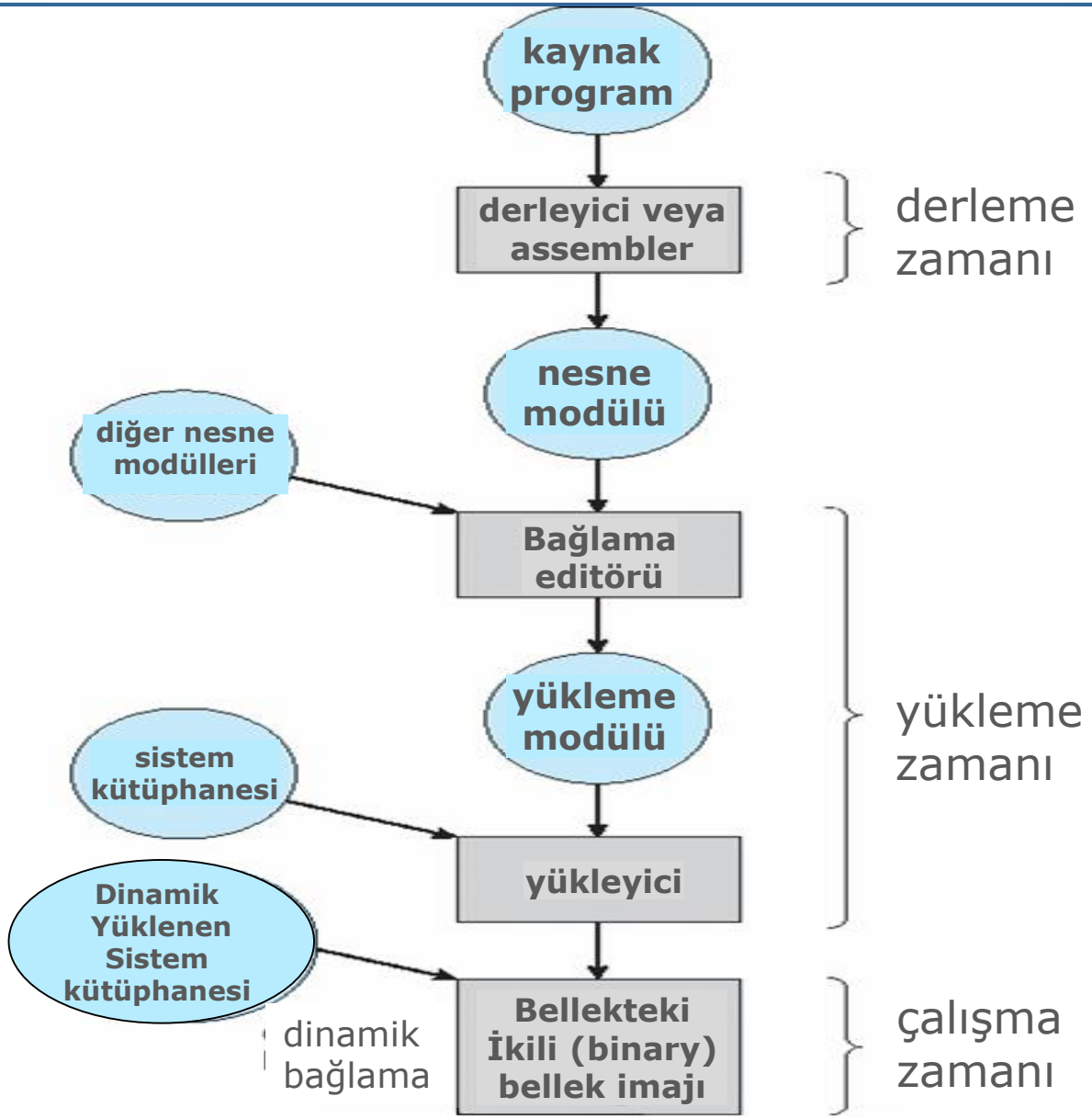
- Mantıksal adres alanını tanımlamak için bir çift taban (base) ve limit kayıtçısı kullanılır



Belleğe Veri ve Komutların Bağlanması

- Komut ve verilerin bellek adreslerine bağlanması üç aşamada gerçekleşebilir
  - **Derleme Zamanında (Compile time)**: Eğer bellek yeri önceden biliniyorsa, **kesin kod (absolute code)** üretilmelidir; ancak başlangıç yeri değişirse kod yeniden derlenmelidir.
  - **Yükleme Zamanında (Load time)**: Bellek yeri derleme zamanında biliniyorsa **yer değiştirebilir kod (relocatable code)** üretilmelidir
  - **Çalışma Zamanında (Execution time)**: Eğer süreç çalışma anında bir bellek segmentinden diğerine tanışabiliyorsa bağlama işlemi çalıştırma zamanına kadar ertelenmelidir. Adres haritaları için donanım desteğine ihtiyaç vardır (ör., taban ve limit kayıtçıları)

Bir Kullanıcı Programının Adım Adım İşlenmesi



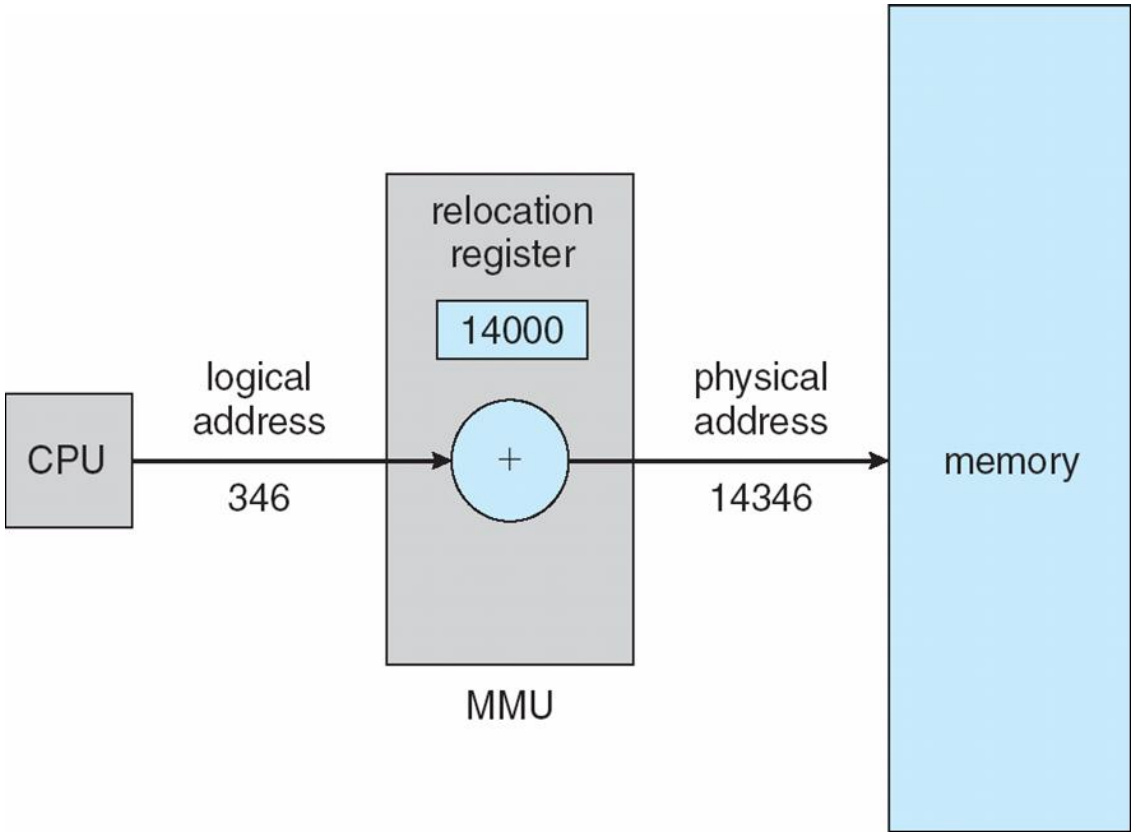
Mantıksal – Fiziksel Adres Alanı

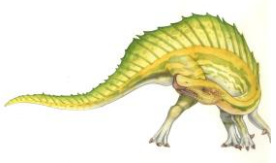
- Bellek yönetiminin merkezinde, ayrı **fiziksel adres alanı**na bağlanmış bir mantıksal adres alanı kavramı bulunmaktadır.
  - **Mantıksal adres (Logical address)** – CPU tarafından yaratılır; genellikle **sanal adres (virtual address)** olarak tanımlanır.
  - **Fiziksel adres (Physical address)** – bellek ünitesi tarafından görülen adrestir.
- Mantıksal ve fiziksel adresler derleme zamanı ve yükleme zamanı adres bağlama şemalarında aynıdır. Mantıksal (sanal) ve fiziksel adresler çalışma zamanı adres bağlama şemasında farklıdırlar.

Bellek Yönetim Ünitesi (BYU)
Memory-Management Unit (MMU)

- Fiziksel adreslerle, sanal adresleri birbirine eşleyen donanımdır.
- BYU (MMU) şemasında, yer değiştirme kayıtçısındaki (relocation register) değer, her kullanıcı süreci tarafından belleğe gönderildiği anda üretilen adrese eklenir.
- Kullanıcı programları *mantıksal* adreslerle ilgilenir, asla gerçek *fiziksel* adresi görmezler.

Yer Değiştirme Kayıtçısı Kullanarak Dinamik Yer Değiştirme





Dinamik Yükleme (Dynamic Loading)

- Rutin, çağrılana kadar yüklenmez
- Daha iyi bellek alanı kullanımı - kullanılmayan rutinler asla yüklenmez
- Düzensiz ortaya çıkan durumları yönetmek için büyük miktarda kod kullanımına ihtiyaç duyulduğu zamanlarda faydalıdır.
- Program tasarımı tarafından uygulanır, işletim sisteminden özel bir desteğe ihtiyaç duyulmaz.

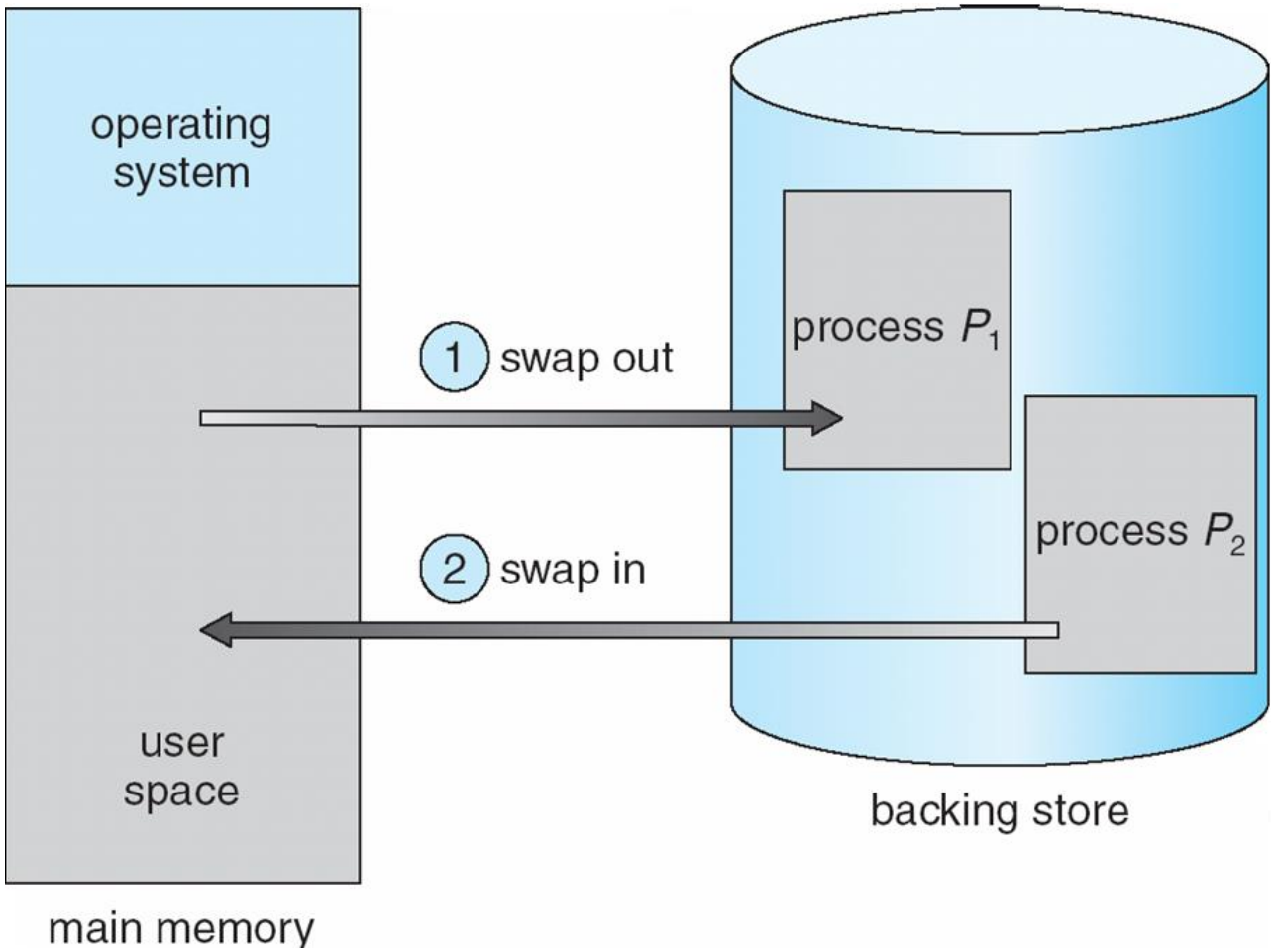
Dinamik Bağlama (Dynamic Linking)

- Bağlama işlemi çalışma zamanına kadar ertelenir
- Bellekte kalıcı uygun kütüphane rutininin yerini belirlemek için Küçük bir kod parçası, koçan (*stub),* kullanılır.
- Koçan kendisini rutinin adresi ile değiştirir ve rutini çalıştırır
- Rutinin süreçlerin bellek adreslerinde yer alıp almadığının kontrolü için işletim sistemi gereklidir
- Dinamik bağlama kısmen kütüphaneler için faydalıdır
- Bu sistem aynı zamanda **paylaşılan kütüphaneler (shared libraries)** olarak da bilinir

Değiş Tokuş (Swapping)

- Bir süreç geçici olarak bellekten bir yardımcı belleğe (backing store) ve daha sonra tekrar belleğe alınarak çalışmasına devam ettirilebilir
- **Yardımcı Bellek (Backing store)** – tüm kullanıcıların bellek imajlarının kopyalarını tutmaya yeterli büyüklükte hızlı disktir. Bu bellek imajlarına direk erişim sağlamalıdır.
- **Bellekten yollamak (Roll out), Belleğe getirmek (Roll in)** – öncelik tabanlı iş planlama algoritmaları için kullanılan değiş tokuş varyantıdır. Düşük öncelikli süreç bellekten yollanırken, daha yüksek öncelikli süreç yüklenir ve çalıştırılır
- Değiş tokuş zamanının ana parçası transfer zamanıdır. Toplam transfer zamanı, değiş tokuş edilecek bellek miktarı ile doğru orantılıdır.
- Birçok sistemde (ör., UNIX, Linux, ve Windows) değiş tokuşun değiştirilmiş versiyonları bulunur
- Sistem, disk üzerinde bellek imajları bulunan koşturulmaya hazır süreçlerin bulunduğu bir **hazır kuyruğu** bulundurur.

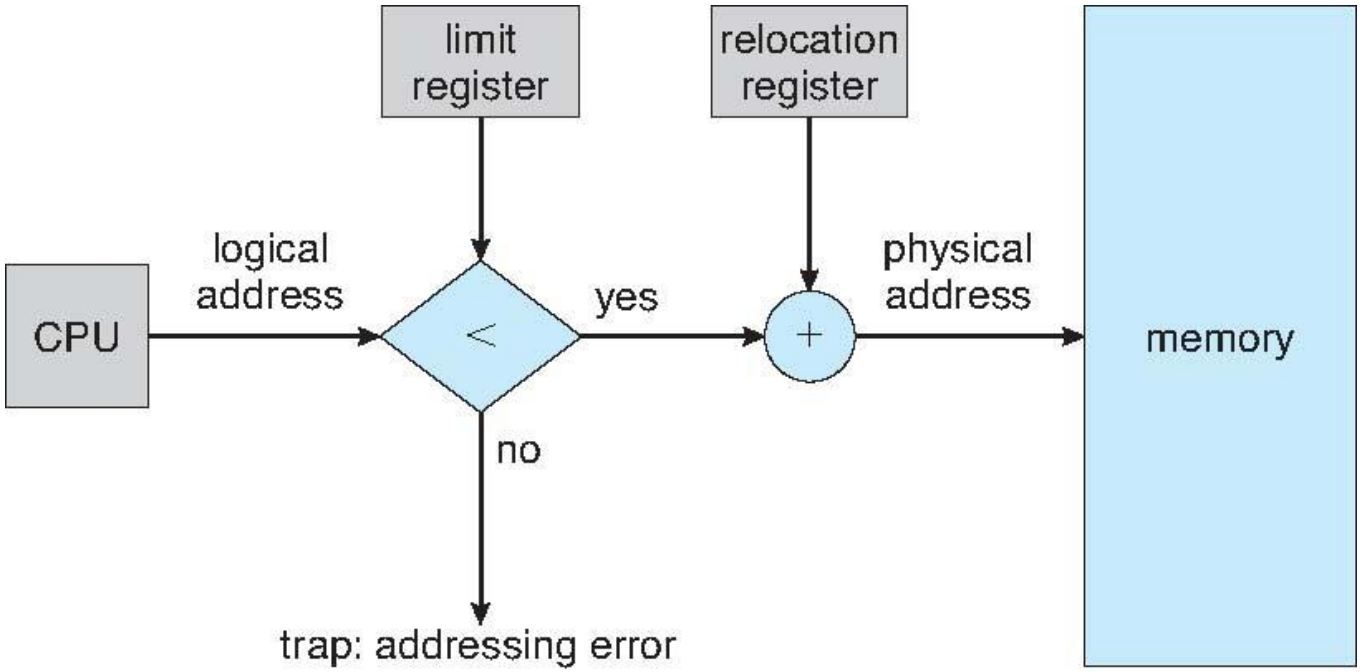
Değiş tokuşun (Swapping) Şematik Gösterimi

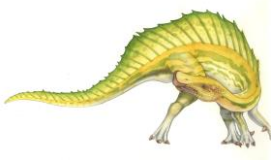


Ardışık Tahsis (Contiguous Allocation)

- Ana bellek genellikle iki bölümdür:
  - Kalıcı işletim sistemi, genellikle kesme vektörü ile alt bellekte tutulur
  - Kullanıcı süreçleri üst bellekte tutulur
- Yer değiştirme kayıtçıları (Relocation registers) kullanıcı süreçlerinin birbirinden korumak ve işletim sistemi kod ve verilerini değiştirmelerini önlemek için kullanılırlar
  - Taban kayıtçısı (Base register) en küçük fiziksel adres değerini içerir
  - Limit kayıtçısı (Limit register) mantıksal adres sınırını içerir – her mantıksal adres limit kayıtçısından küçük olmalıdır
  - MMU, mantıksal adresleri dinamik olarak kapsar

Yer Değiştirme ve Limit Kayıtçıları İçim Donanım Desteği

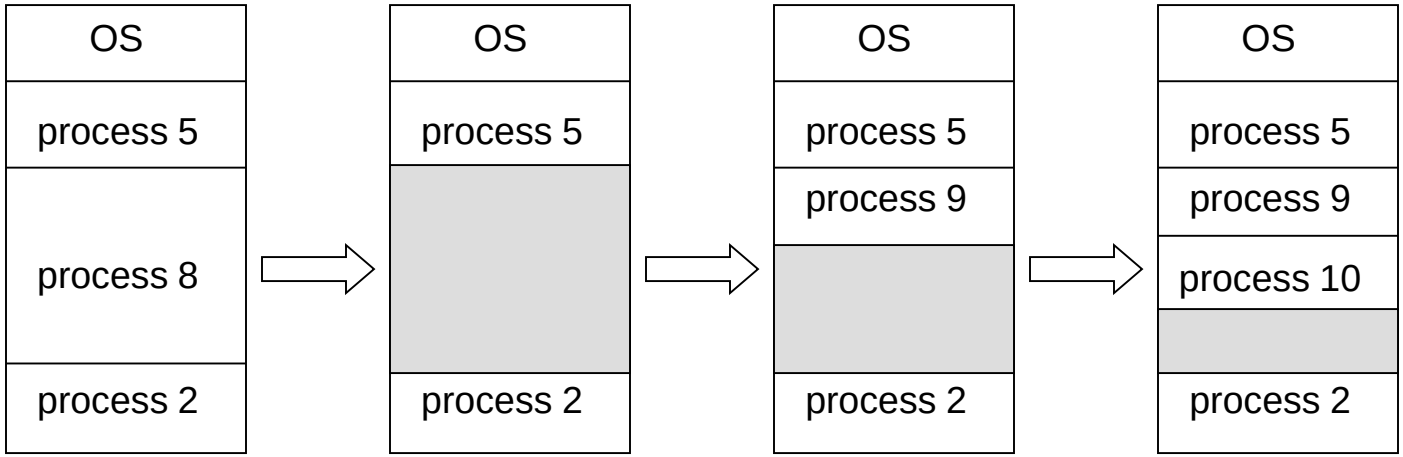


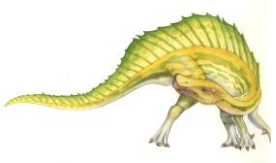


Ardışık Tahsis (Contiguous Allocation) (Devam)

- Çoklu-bölüm tahsisi (Multiple-partition allocation)
  - Delik (Hole) – mevcut bellek bloğu, değişik boyutlardaki delikler tarafından parçalanmıştır
  - Bir süreç geldiğinde, bellekte bunu karşılayacak büyüklükte bir deliğe yerleştirilir
  - İşletim sistemi aşağıdakiler hakkında bilgi tutar:
    a) tahsis edilmiş bölümler (allocated partitions)

    b) boş bölümler (delik(hole))





Dinamik Depolama-Tahsis Problemi

n boyutundaki bir isteği serbest delik listesinden nasıl karşılarız

- **İlk-uyan (First-fit)**: Yeterli büyüklükteki ilk deliğe tahsis et
- **En iyi-uyan (Best-fit)**: Yeterli büyüklükteki en küçük deliğe tahsis et; boyuta göre sıralanmamış olsa bile tüm listenin aranması gerekmektedir
  - En küçük kalan deliği üretir
- **En kötü-uyan (Worst-fit)**: En büyük deliği tahsis et; bunda da tüm listenin aranması gerekir
  - En büyük kalan deliği üretir

İlk-uyan ve En iyi-uyan hız ve depolama kullanımı açısından En kötü-uyan'dan daha iyidir

Parçalanma (Fragmentation)

- **Dış Parçalanma (External Fragmentation)** – isteği karşılamak için toplam bellek alanı mevcuttur, ancak ardışık değildir
- **İç Parçalanma (Internal Fragmentation)** – tahsis edilen bellek istenen bellekten biraz daha büyüktür; bu boyut farkı bellekteki bölümün içindedir ancak kullanılmamaktadır.
- Dış parçalanma **sıkıştırma (compaction)** kullanarak azaltılabilir
  - Bellek içeriklerini tüm serbest bellek büyük bir blok oluşturacak şekilde yer değiştirmek
  - Sıkıştırma sadece yer değiştirme dinamik ise mümkündür ve çalışma zamanında gerçekleştirilir
  - G/Ç problemi
    - İşi G/Ç durumunda iken bellekte tuttur
    - G/Ç'ı sadece İşletim Sisteminin tamponlarına gerçekleştir

Sayfalama (Paging)

- Bir sürecin mantıksal adres alanı bitişik olmayabilir; bir sonraki mevcut olduğu zaman süreç fiziksel belleğe yerleştirilir
- Fiziksel bellek **çerçeve (frames)** adı verilen sabit boyutlu bloklara (boyut 2'nin üsleri şeklindedir, 512 byte ile 16 MB arasında olabilir) bölünür.
- Mantıksal bellek ise **sayfa (pages)** adı verilen aynı boyutta bloklara bölünür.
- Tüm serbest (boş) çerçevelerin (frames) kaydı tutulur
- ***n*** sayfa boyutundaki bir programı çalıştırmak için, ***n*** adet boş çerçeve bulmaya ve programı yüklemeye ihtiyaç vardır
- Mantıksal adresleri fiziksel adreslere dönüştürmek için bir **sayfa tablosu** kurulur
- İç parçalanma

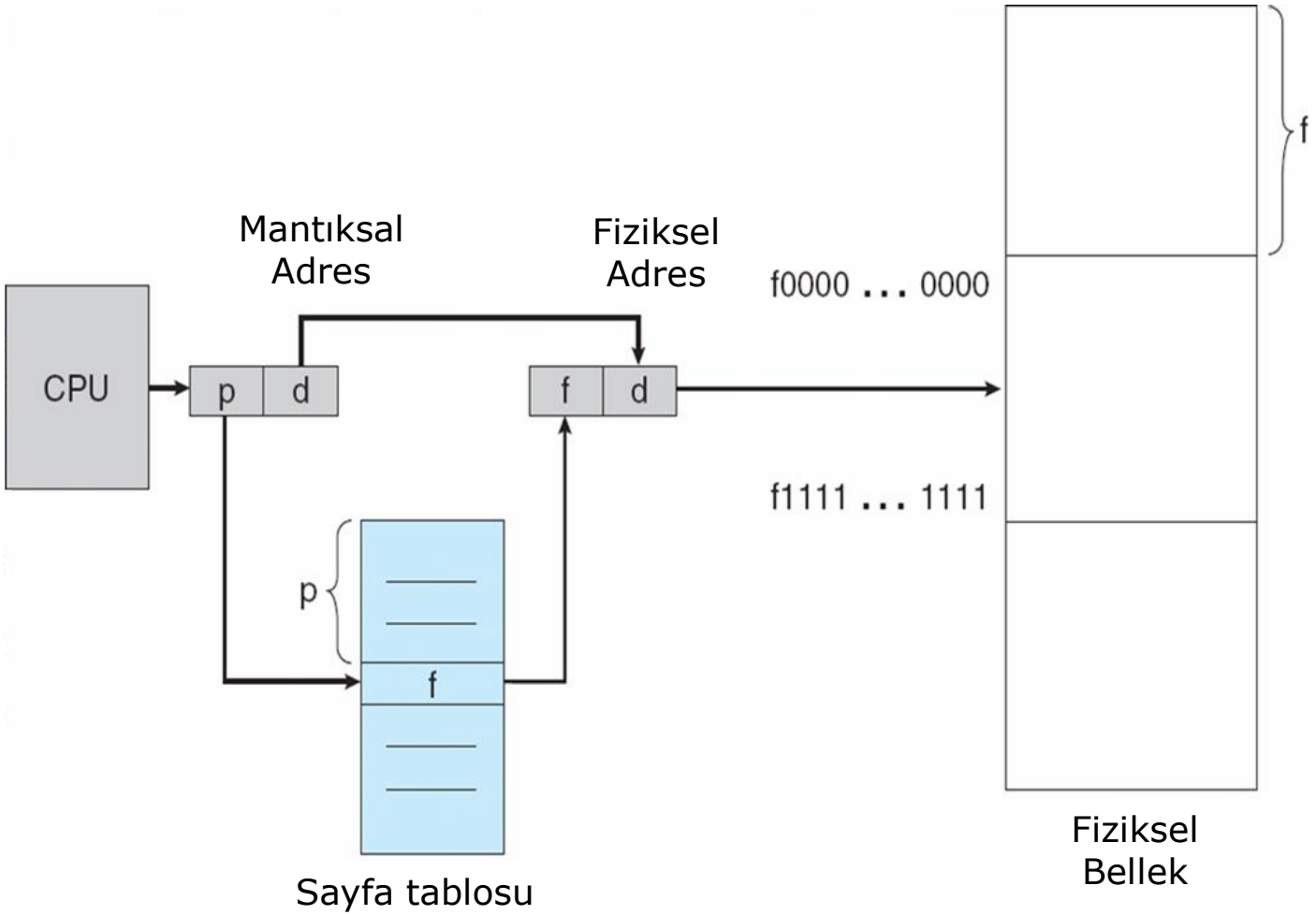
Adres Dönüşüm Şeması

- CPU tarafından üretilen adresler:
  - **Sayfa numarası (Page number) (*p*)** – fiziksel bellekteki he sayfanın (page) taban adresini tutmak için sayfa tablosunda bir endeks olarak kullanılır.
  - **Sayfa ofseti (Page offset) (d)** – taban adresi ile birlikte bellek ünitesine gönderilecek fiziksel bellek adresini tanımlamak için kullanılır

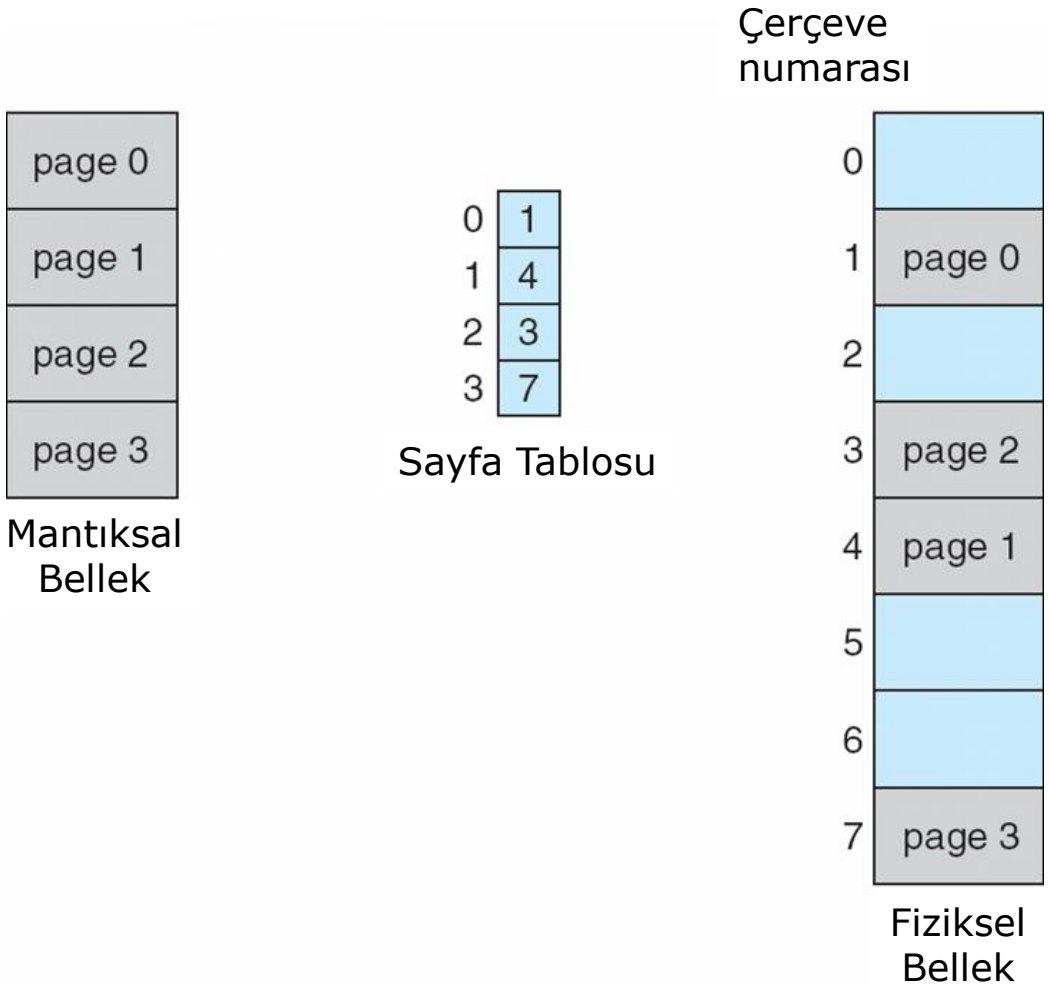
Sayfa numarası	Sayfa ofseti
p	*d*
m - n	*n*

  - Verilen mantıksal adres alanı 2^m *ve sayfa boyutu* 2^n

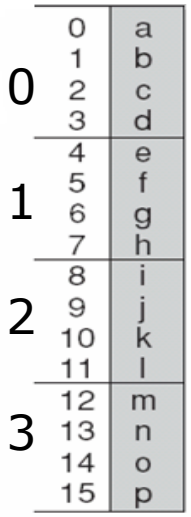
Sayfalama Donanımı



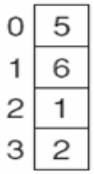
Mantıksal ve Fiziksel Belleğin Sayfalama Modeli



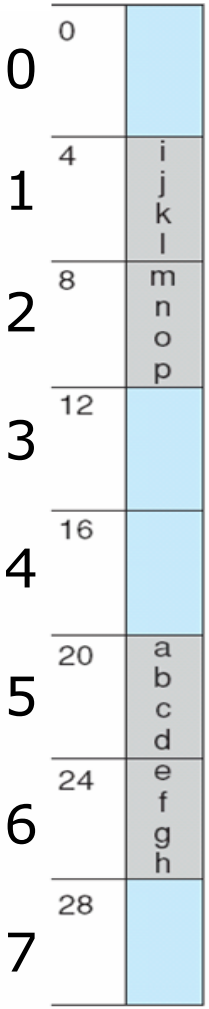
Sayfalama Örneği



Mantıksal Bellek



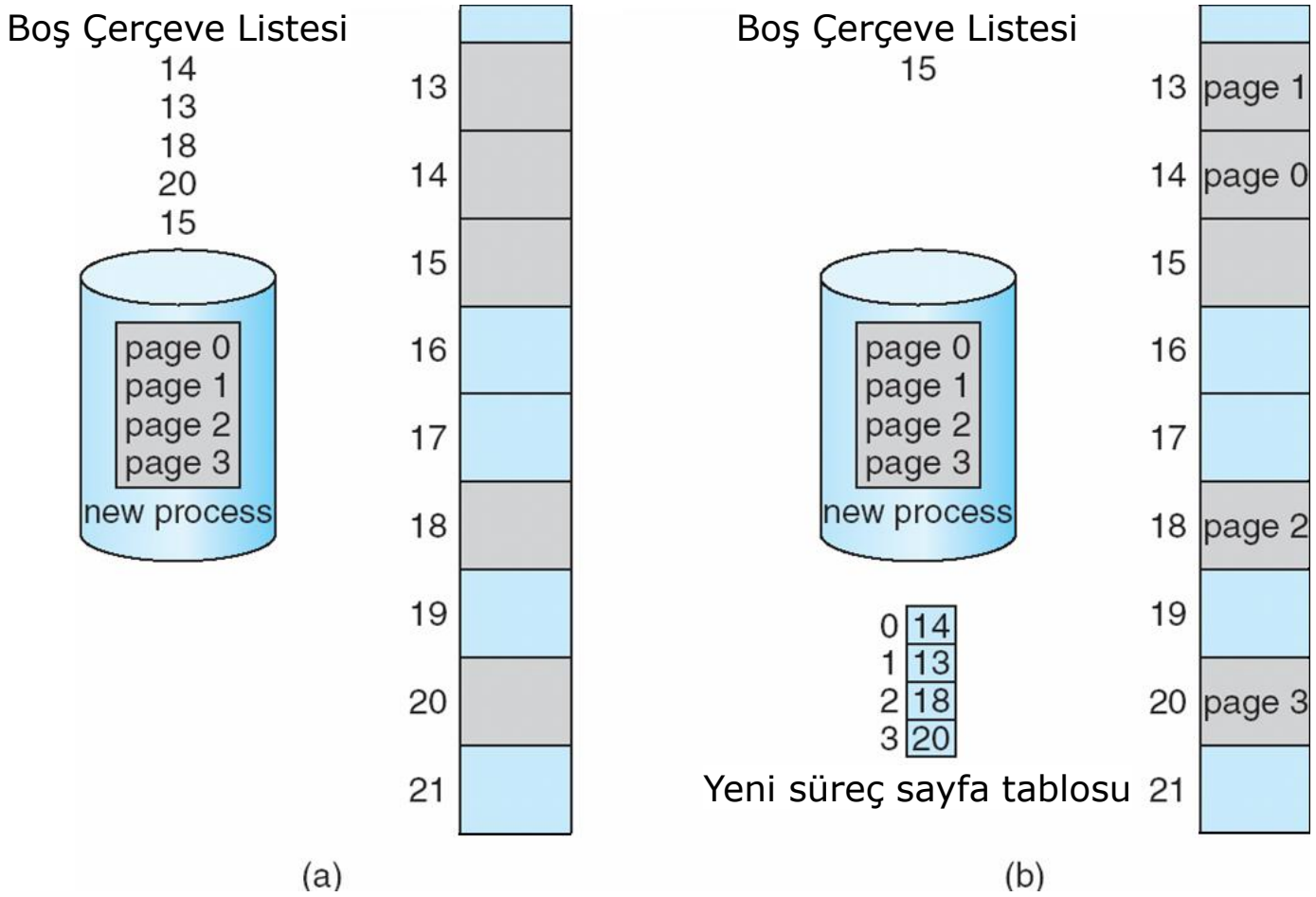
Sayfa Tablosu



Fiziksel Bellek

32-byte bellek ve 4-byte sayfalar

Boş (Serbest) Çerçeveler



Tahsisten önce

Tahsisten sonra

Sayfa Tablosu Uygulaması

- Sayfa Tablosu ana bellekte tutulur
- **Sayfa Tablosu Taban Kayıtçısı (Page-table base register) (PTBR)** sayfa tablosunu gösterir
- **Sayfa Tablosu Uzunluk Kayıtçısı (Page-table length register) (PRLR)** ise sayfa tablosunun boyutunu belirtir
- Bu şemada her veri/komut erişimi iki bellek erişimine ihtiyaç duyar. Bir tanesi sayfa tablosu, diğeri de veri/komut için.
- İki bellek erişim problemi **ilişkili bellek (associative memory)** veya **dönüşüm bakış tamponları (translation look-aside buffers (TLBs))** adı verilen özel bir hızlı bulma donanım önbelleği kullanımı ile çözülebilir
- Bazı TLB'ler her TLB girişinde **adres alanı belirleyiciler (address-space identifiers (ASIDs))** tutar. Bunlar her süreç için adres alanı koruması sağlayarak süreçleri ayırt ederler.

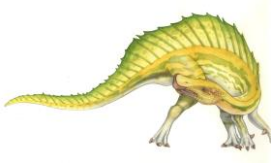
İlişkili Bellek (Associative Memory)

- İlişkili bellek – paralel arama

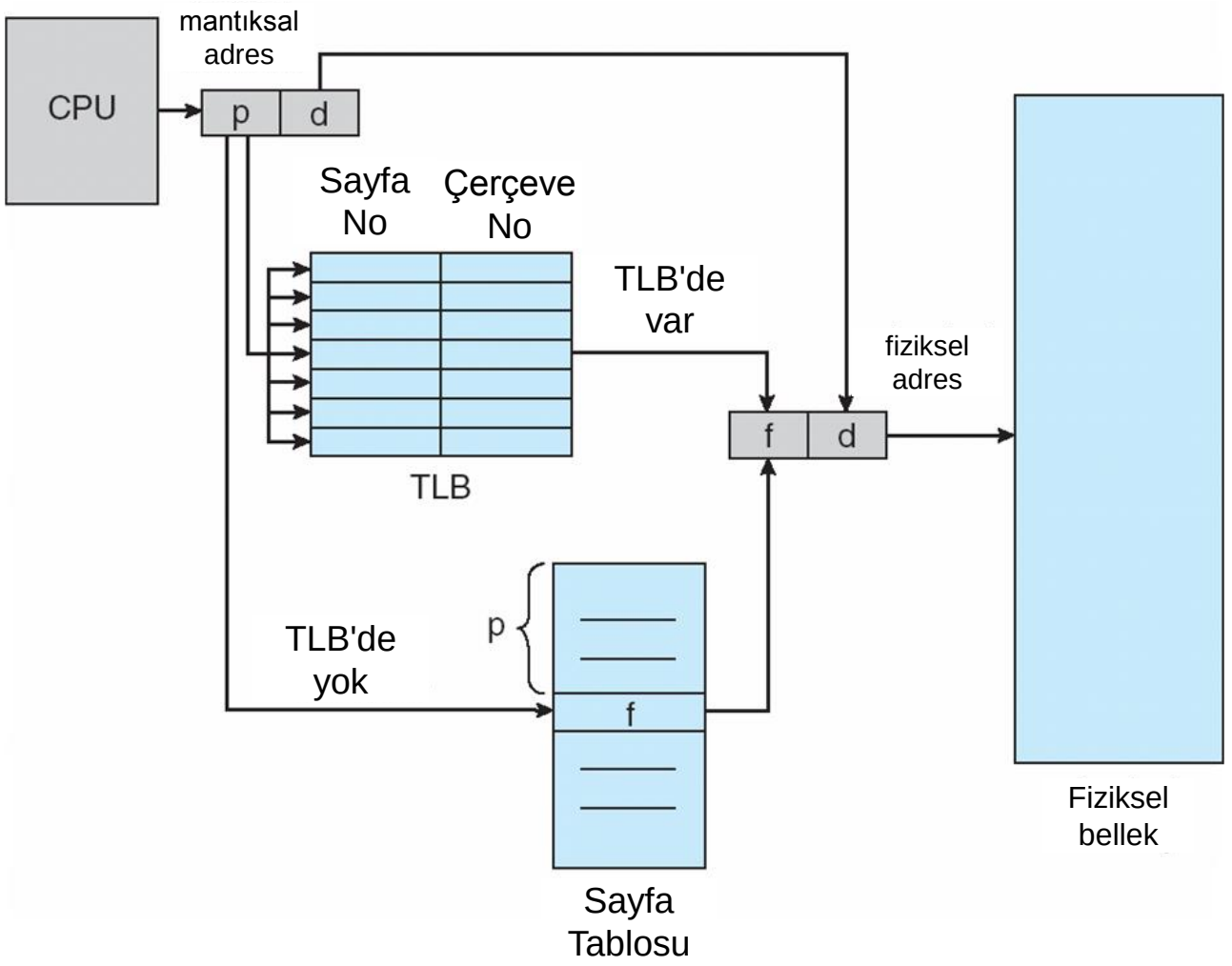
Sayfa no	Çerçeve no

Adres dönüşümü (p, d)

- Eğer p, ilişkili kayıtçıda (associative register) ise çerçeve numarasını al
- Diğer durumda çerçeve numarasını bellekteki sayfa tablosundan al



TLB ile Sayfalama Donanımı



Verimli Erişim Süresi

- İlişkili Arama (Associative Lookup) = ε zaman birimi
- Bellek çevrim süresini ise "t" mikrosaniye olarak düşünelim
- Bulma oranı (Hit ratio) – sayfa numarasının ilişkili kayıtçılarda bulunma oranı; oran ilişkili kayıtçı sayısına bağımlıdır
- Bulma oranı = α
- **Verimli Erişim Süresi (Effective Access Time)** (EAT)

$$EAT = (t + \varepsilon)\,\alpha + (2t + \varepsilon)(1 - \alpha)$$

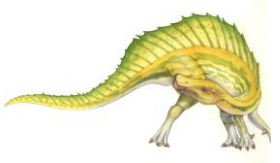
$$= (2t + \varepsilon - \alpha t)$$

Örnek(TLB)

- %80 bulma oranına sahip demek istenen sayfanın TLB'de %80 oranında bulunabildiğini gösterir. Örneğin TLB'yi aramak 20 nanosaniye, belleğe erişim de 100 nanosaniye olsun, bu durumda sayfa numarası TLB'de mevcut ise erişim süresi 120 nanosaniye olur.
- Eğer sayfa numarası TLB'de bulunamazsa (20 nanosaniye) bu durumda önce belleğe sayfa tablosuna ve çerçeve numarasına erişip (100 ns) sonra da istenen bellek parçasına erişilebilir (100 ns), dolayısıyla toplamda 220 ns geçer.
- Verimli Bellek erişim Süresi= 0.80 x (100+20) + 0.20 x (2*100+20) = 140 ns

Dolayısıyla %40'lık bir yavaşlama olur

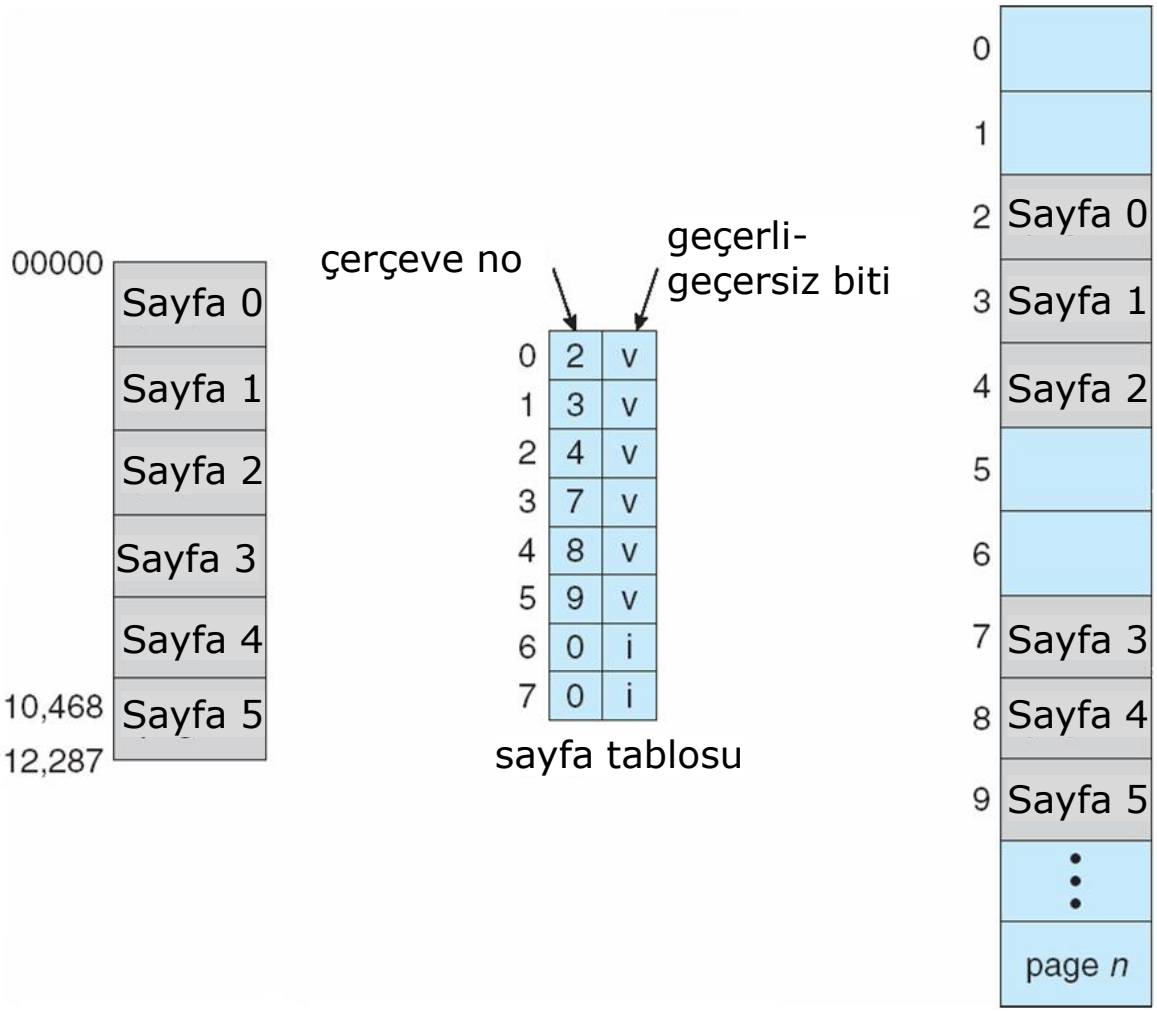
- %98 bulma oranı ile:
- VBES = 0.98 x (100+20) + 0.02 x (2*100+20) = 122 ns (%22'lik yavaşlama)



Bellek Koruma (Memory Protection)

- Bellek koruma her çerçeve ile bir koruma bit'i ilişkilendirerek uygulanır
- **Geçerli- Geçersiz (Valid-invalid)** biti sayfa tablosundaki her girdiye eklenir:
  - "Geçerli" (“valid”) ilişkili sayfanın sürecin mantıksal adres alanında bulunduğunu ve geçerli bir sayfa olduğunu gösterir
  - "Geçersiz" (“invalid”) ise sayfanın sürecin mantıksal adres alanında olmadığını gösterir

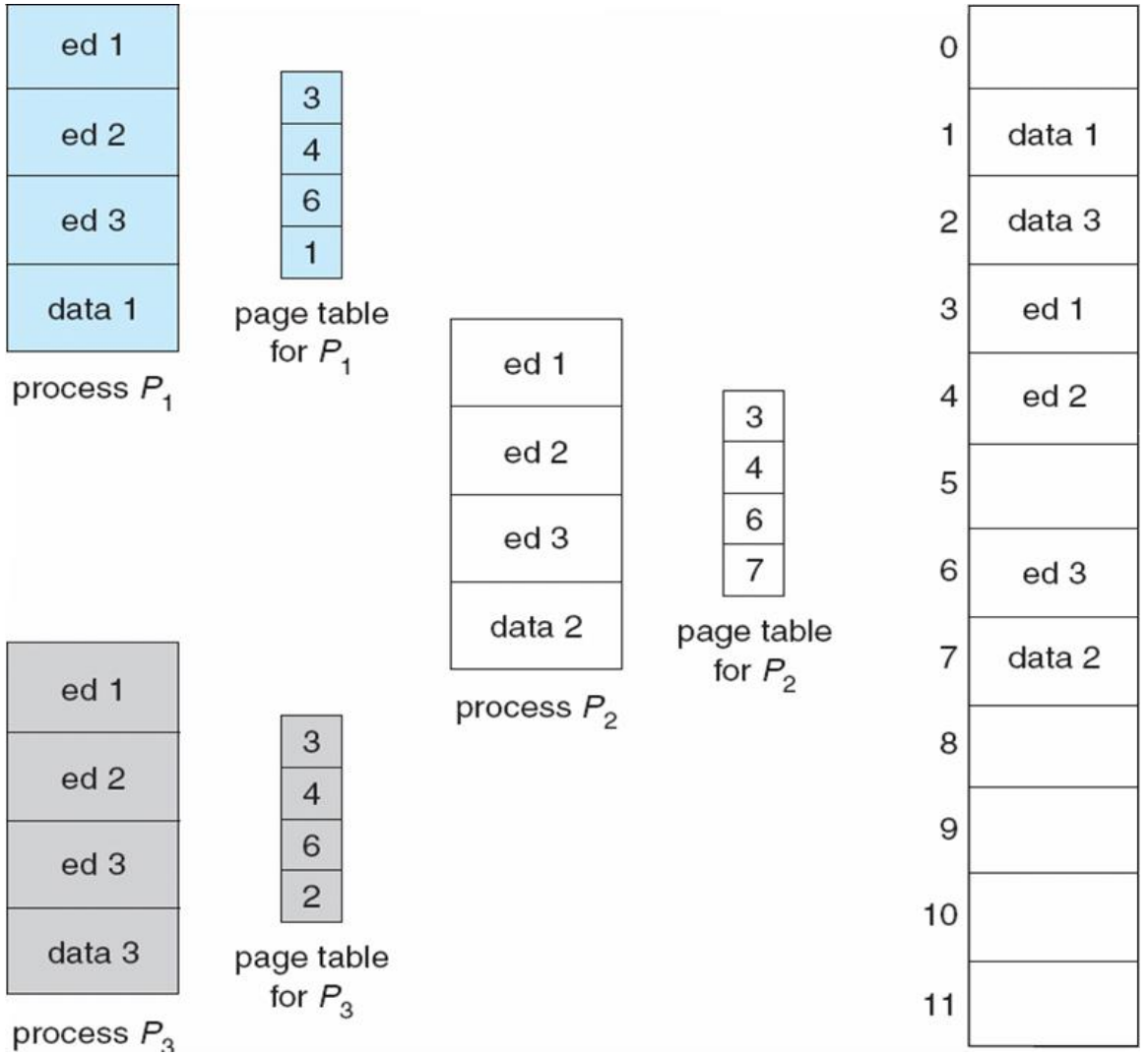
Sayfa Tablosundaki Geçerli (Valid) (v) veya Geçersiz (Invalid (i)) Biti



Paylaşılan Sayfalar (Shared Pages)

- **Paylaşılan kod (Shared code)**
  - Süreçler (ör., metin editörleri, derleyiciler, pencere sistemleri) arasında yalnızca okunabilir (read-only (reentrant)) kodun tek kopyası paylaşılır.
  - Paylaşılan kod tüm süreçlerin mantıksal adres alanlarında aynı yerde gözükmelidir
- **Özel Kod ve Veri (Private code and data)**
  - Her süreç ayrı bir kod ve veri kopyası tutar
  - Özel kod ve verinin sayfaları mantıksal adres alanında herhangi bir yerde olabilir

Paylaşılan Sayfalar Örneği



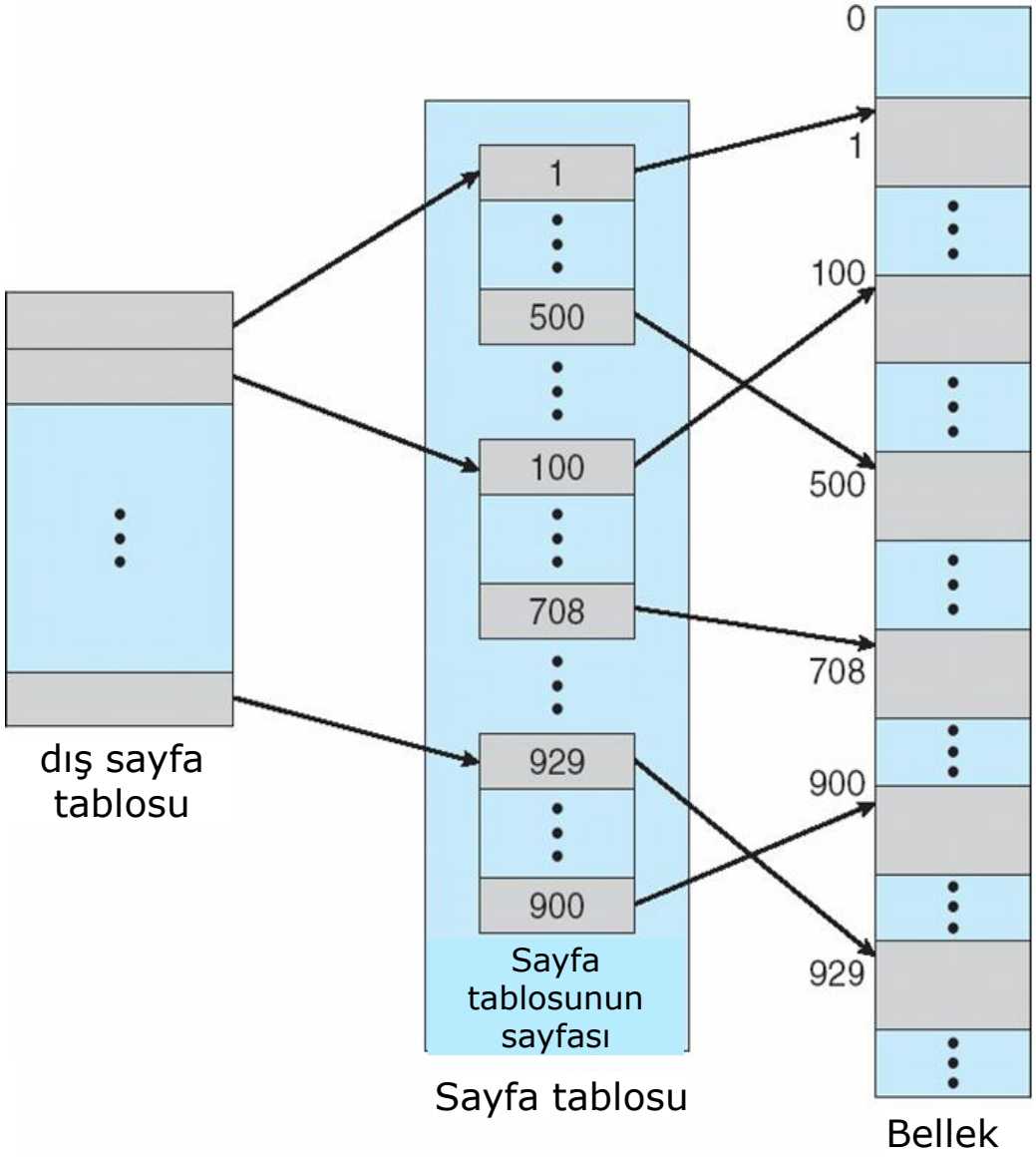
Sayfa Tablosu Yapısı

- Hiyerarşik Sayfalama (Hierarchical Paging)
- Hashed Sayfa Tabloları (Hashed Page Tables)
- Ters Sayfa Tabloları (Inverted Page Tables)

Hiyerarşik Sayfa Tabloları

- Mantıksal adres alanı birçok sayfa tablosuna bölünmüştür
- Temel tekniklerden biri çift-seviyeli sayfa tablosudur

Çift-Seviyeli Sayfa Tablosu Şeması



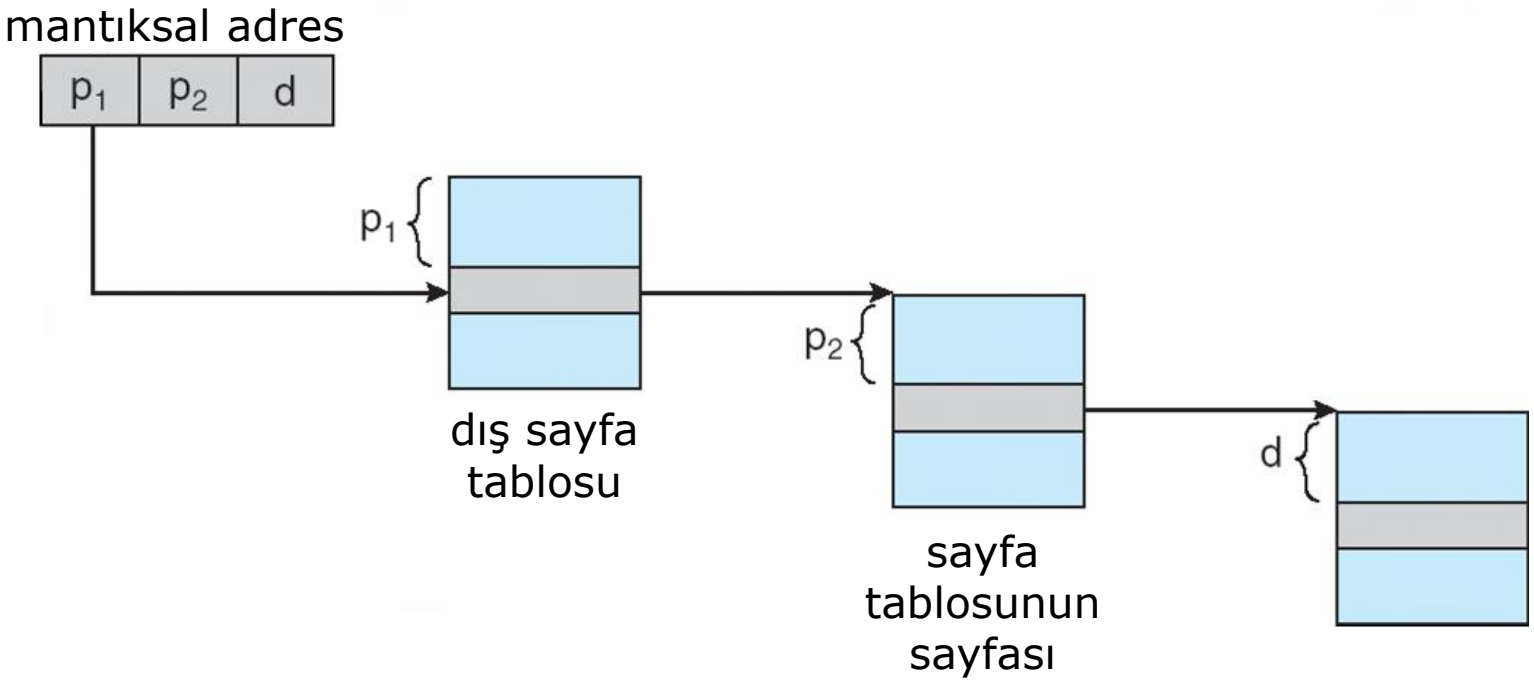
Çift-Seviyeli Sayfalama Örneği

- Mantıksal adres (32-bit bir makinada 1K sayfa boyutu ile) şunlara bölünmüştür:
  - 22 bitlik oluşan sayfa numarası
  - Ve 10 bitlik sayfa ofseti
- Sayfa tablosu sayfalandığından dolayı, sayfa numarası da:
  - 12-bitlik sayfa numarası
  - 10-bitlik sayfa ofsetine bölünmüştür
- Dolayısıyla mantıksal adres aşağıdaki gibidir:

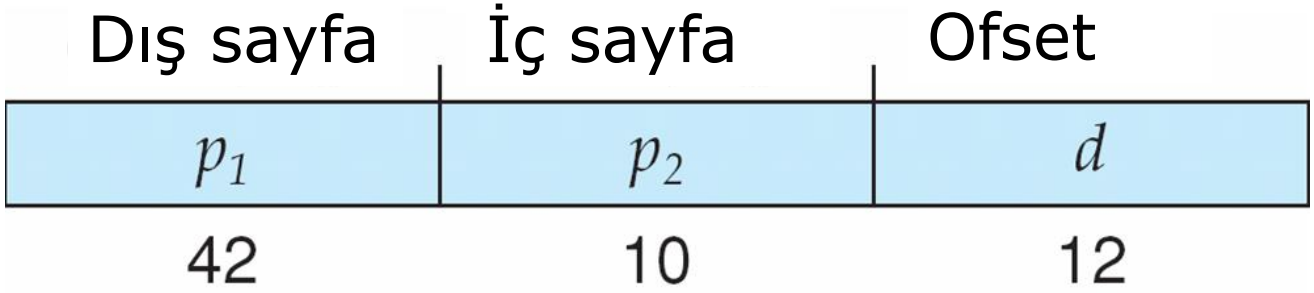
sayfa numarası		sayfa ofseti
p_i	p_2	d
12	10	10

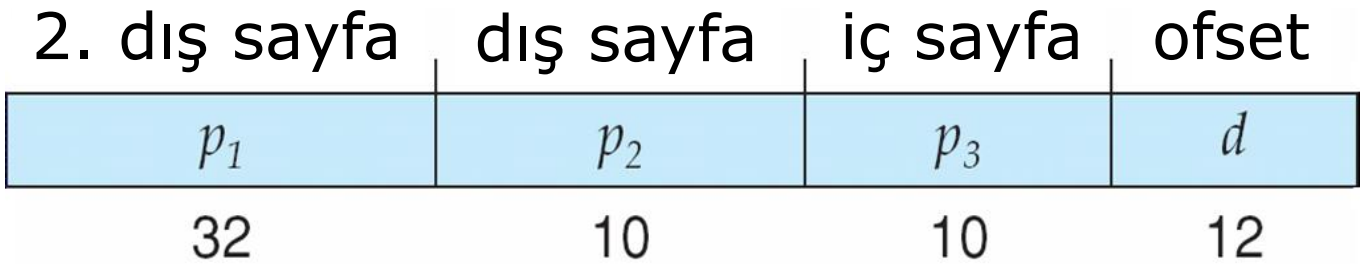
p_i dış sayfa tablosuna bir endeks, ve p_2 dış sayfa tablosundaki sayfalar arasında yerdeğişimdir

Adres-Dönüşüm Şeması



Üç-Seviyeli Sayfalama Şeması

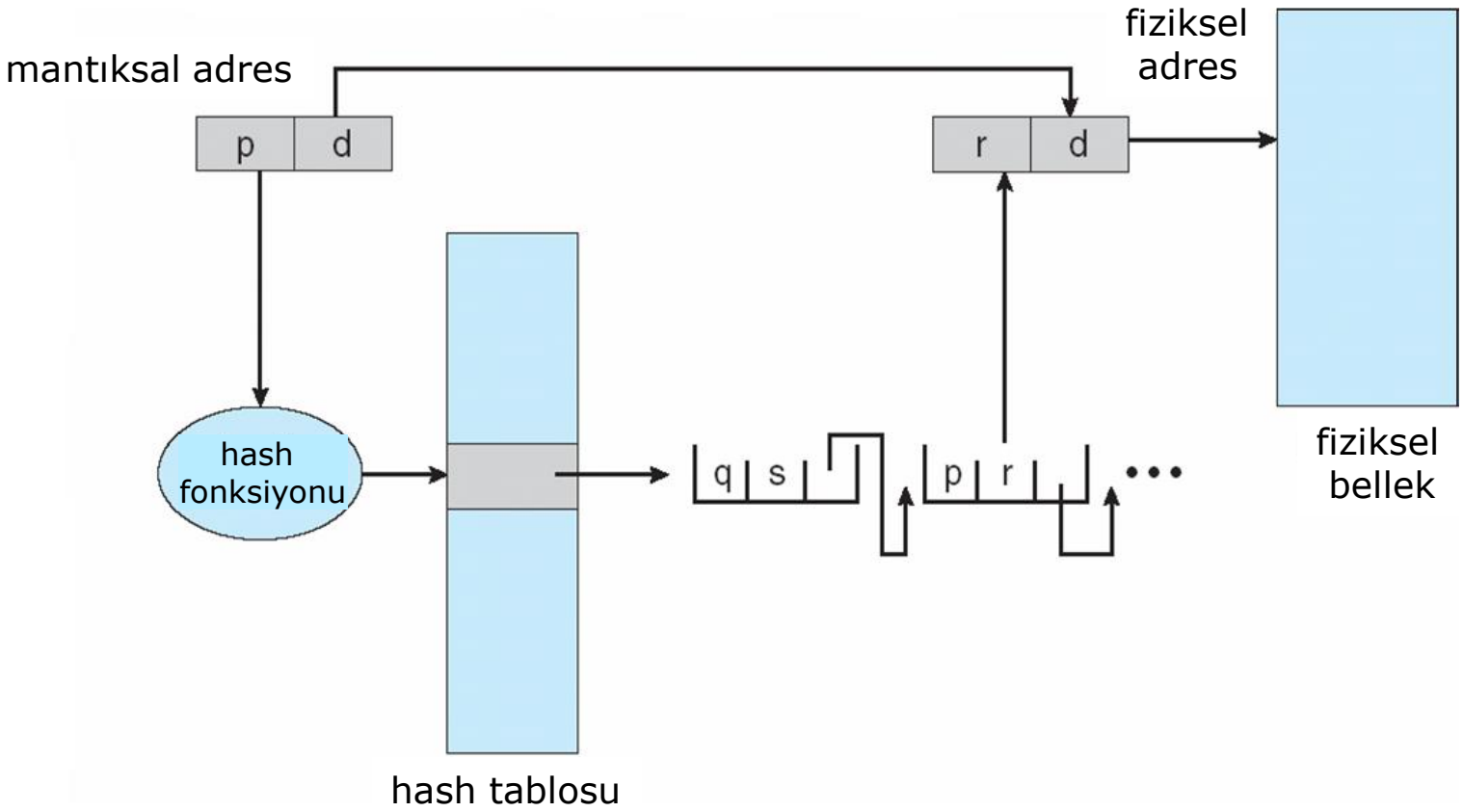




Hashed Sayfa Tabloları

- 32 bitten büyük adres alanlarında sık kullanılır
- Sanal sayfa numarası bir sayfa tablosunda hash'lenmiştir
  - Bu sayfa tablosunda aynı yere hashlenen bir elemanlar zinciri mevcuttur
- Sanal sayfa numaraları bu zincirde bir denklik bulunana kadar karşılaştırılır
  - Bir denklik bulunursa, karşılık gelen fiziksel çerçeve çıkartılır

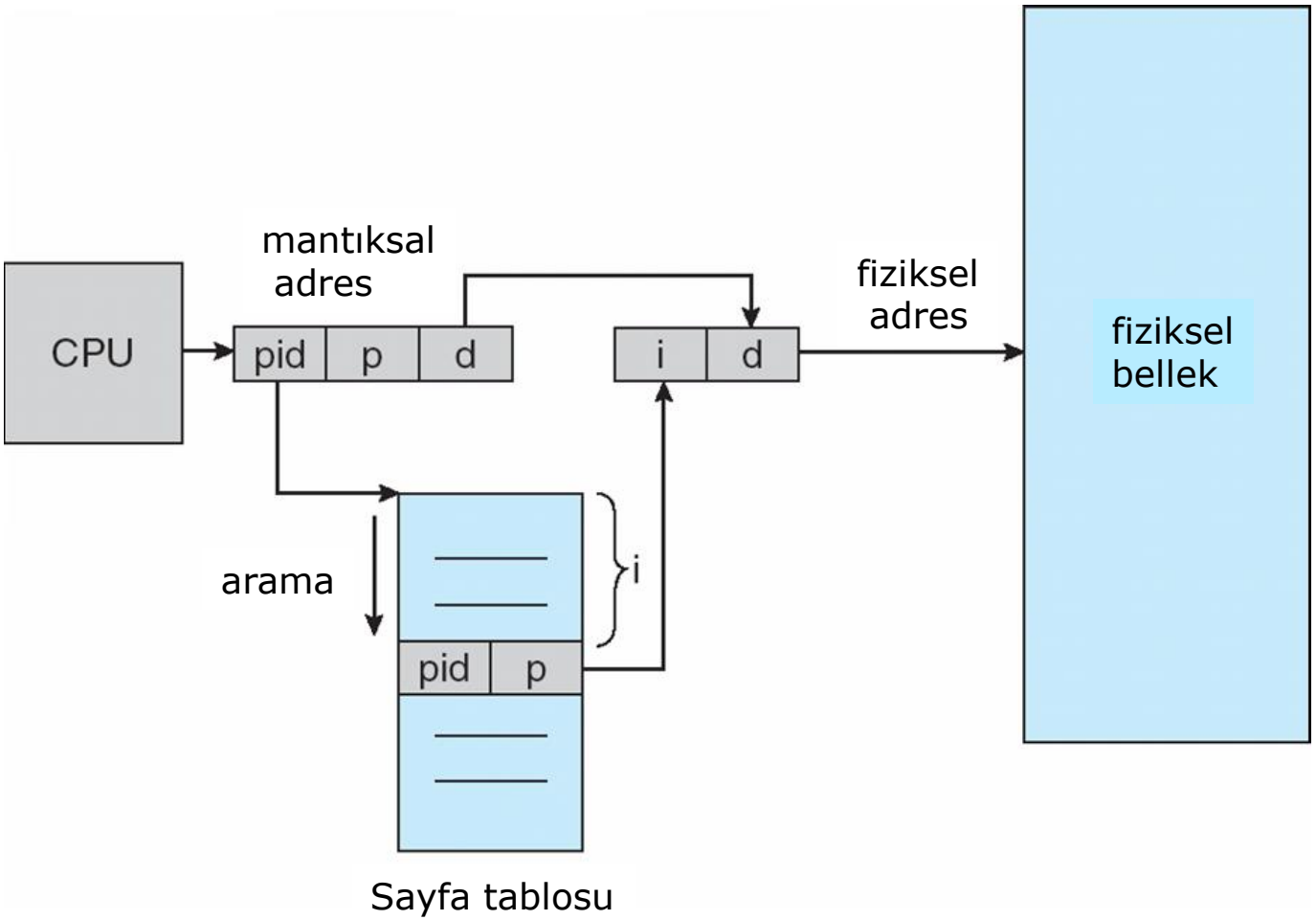
Hashed Sayfa Tablosu



Ters Sayfa Tablosu

- Bellekteki her gerçek sayfa için bir girdi
- Girdide sayfanın sahibi olan süreç bilgisi ile beraber gerçek bellek yerinde tutulan sayfanın sanal adresi tutulur
- Her sayfa tablosu için gereken belleği düşürür, ancak bir sayfa referansı geldiğinde tabloyu arama süresini uzatır
- Aramayı bir tane yada az sayıda sayfa tablosu girdisine sınırlandırmak için hash tablosunu kullanın

Ters Sayfa Tablosu Mimarisi



Segmentasyon (Parçalama)

- Belleğin kullanıcı görünümünü destekleyen bellek yönetim şemasıdır
- Bir program segmentler topluluğudur
  - Segment mantıksal bir ünitedir:

    ana program

    prosedür

    fonksiyon

    metot

    nesne

    yerel değişkenler, global değişkenler

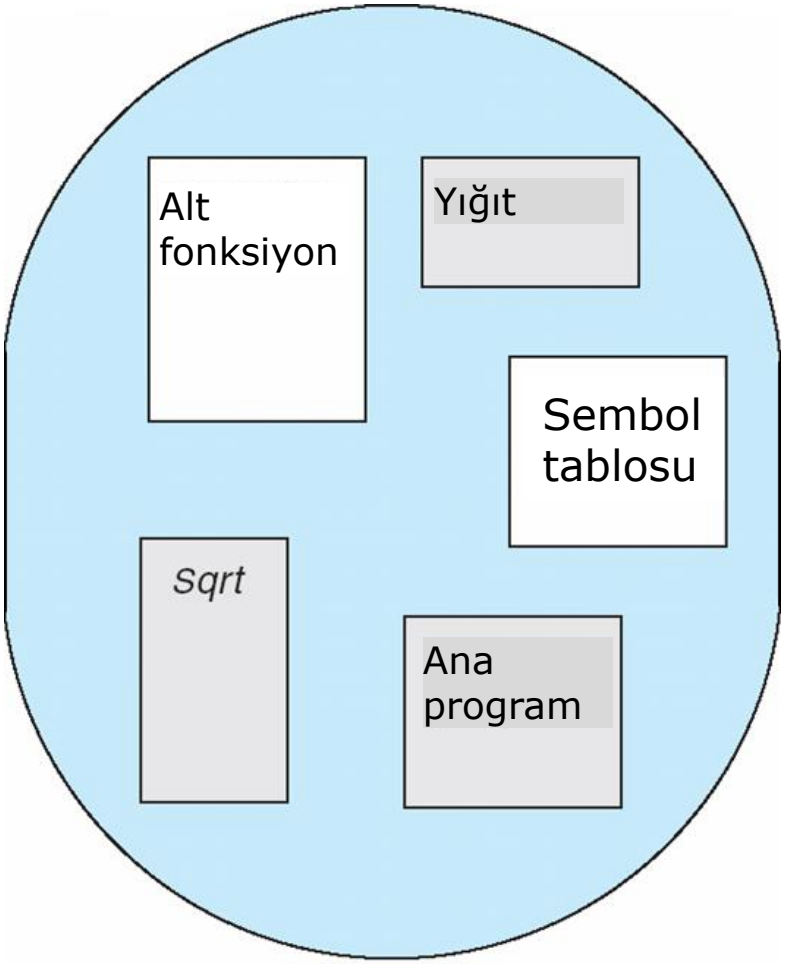
    ortak blok

    yığıt (stack)

    sembol tablosu

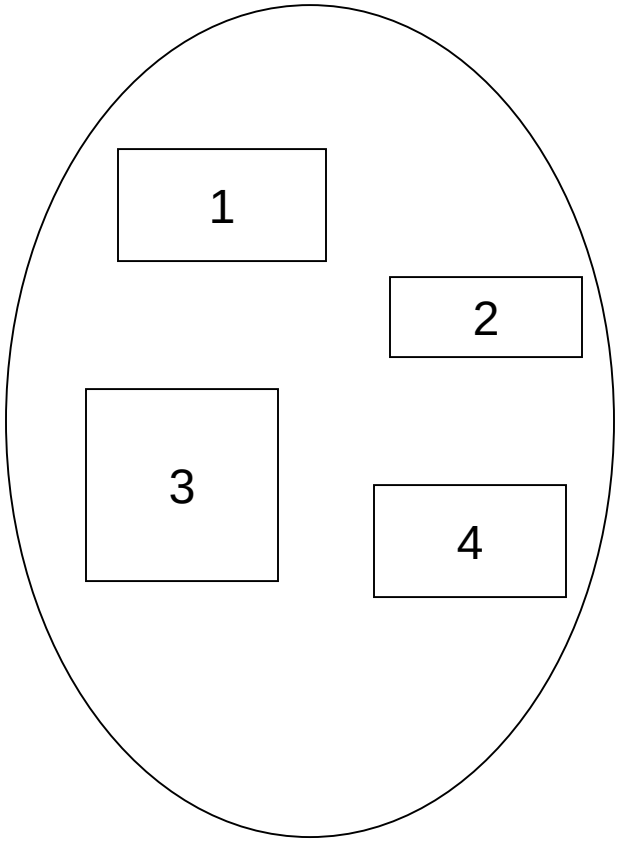
    diziler

Programın Kullanıcı Görünümü



mantıksal adres

Segmentasyonun Mantıksal Görünümü



Kullanıcı alanı

Fiziksel bellek alanı

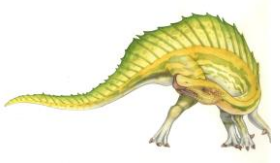
Segmentasyon Mimarisi

- Mantıksal adres bir ikiliden oluşur:

  <segment-numarası, ofset>,

- **Segment tablosu** – iki boyutlu fiziksel adresleri kapsar; her tablo girdisinde:
  - **taban (base)** – segmentlerin bellekte bulundukları fiziksel adresin başlangıcını içerir
  - **limit** – segmentin uzunluğunu belirtir
- **Segment tablosu taban kayıtçısı (Segment-table base register (STBR))** segment tablosunun bellekteki yerini gösterir
- **Segment tablosu uzunluk kayıtçısı (Segment-table length register (STLR))** ise bir program tarafından kullanılan segment sayısını belirtir;

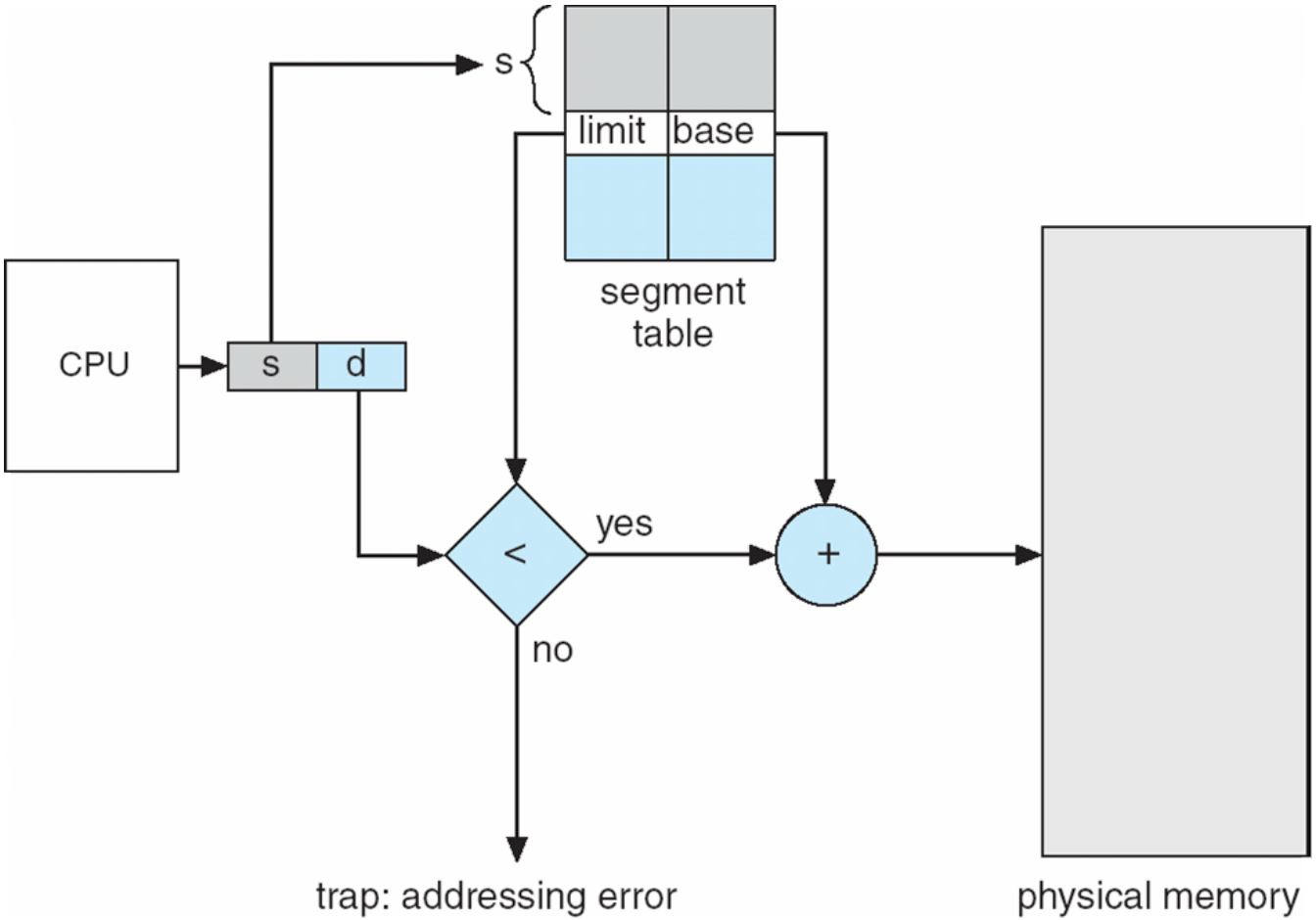
  segment numarası ***s***; eğer ***s*** < **STLR** ise geçerlidir



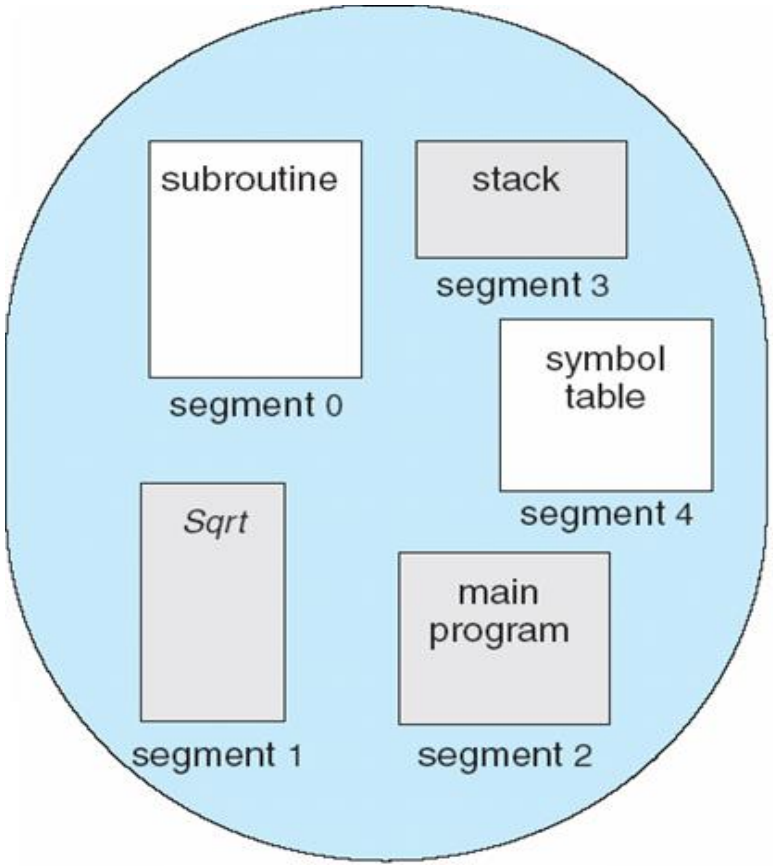
Segmentasyon Mimarisi

- Koruma
  - Segment tablosundaki her girdi için:
    - Geçerleme (validation) biti = 0 $\Rightarrow$ geçersiz (illegal) segment
    - okuma/yazma/çalıştırma (read/write/execute) yetkileri (privileges)
- Koruma bitleri segmentlerle ilişkilendirilir; kod paylaşımı segment seviyesinde oluşur
- Segmentlerin uzunlukları değiştiğinden dolayı, bellek tahsisi bir dinamik depolama tahsis problemidir

Segmentasyon Donanımı



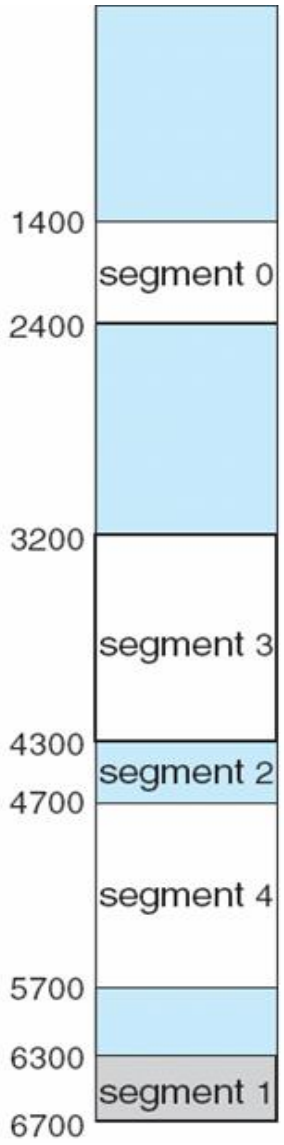
Segmentasyon Örneği



mantıksal adres alanı

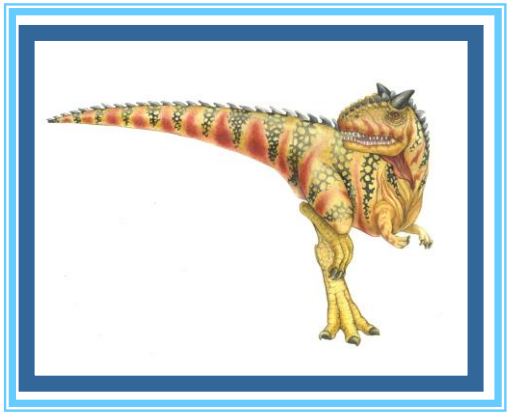
	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment tablosu



fiziksel bellek

Bölüm 8 Son



Silberschatz, Galvin and Gagne ©2009