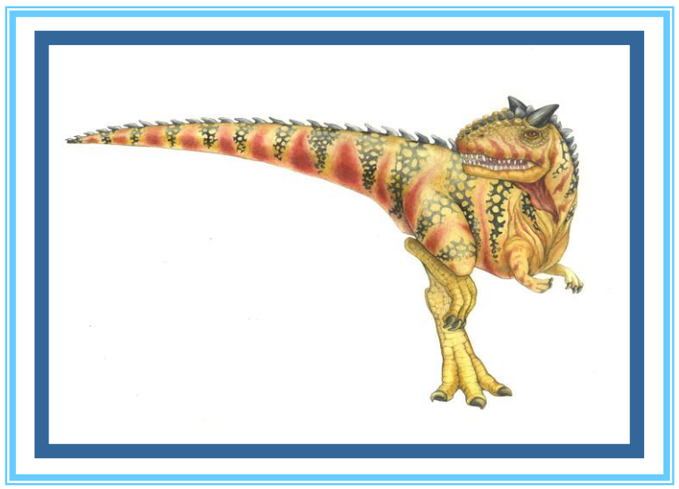
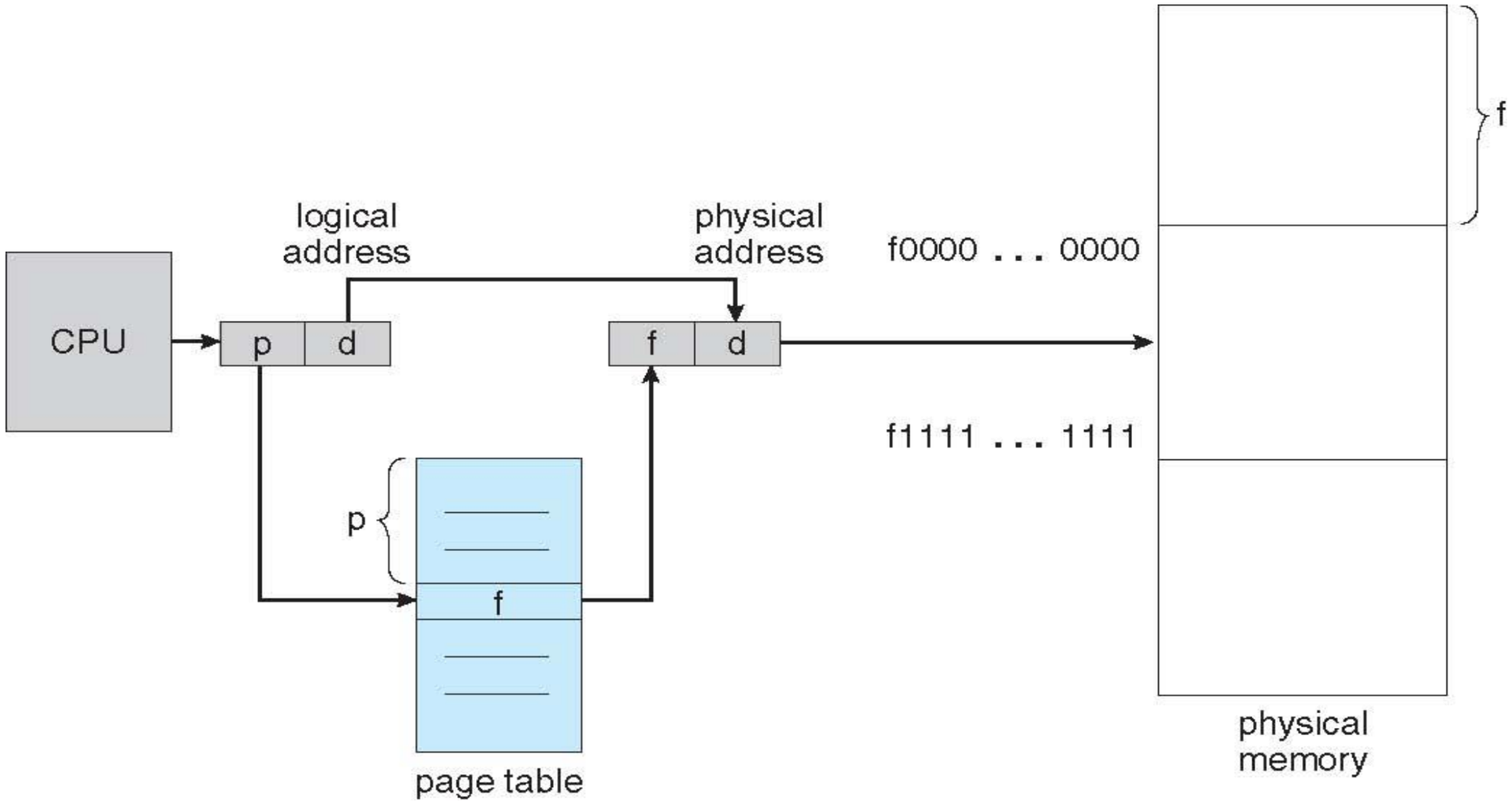
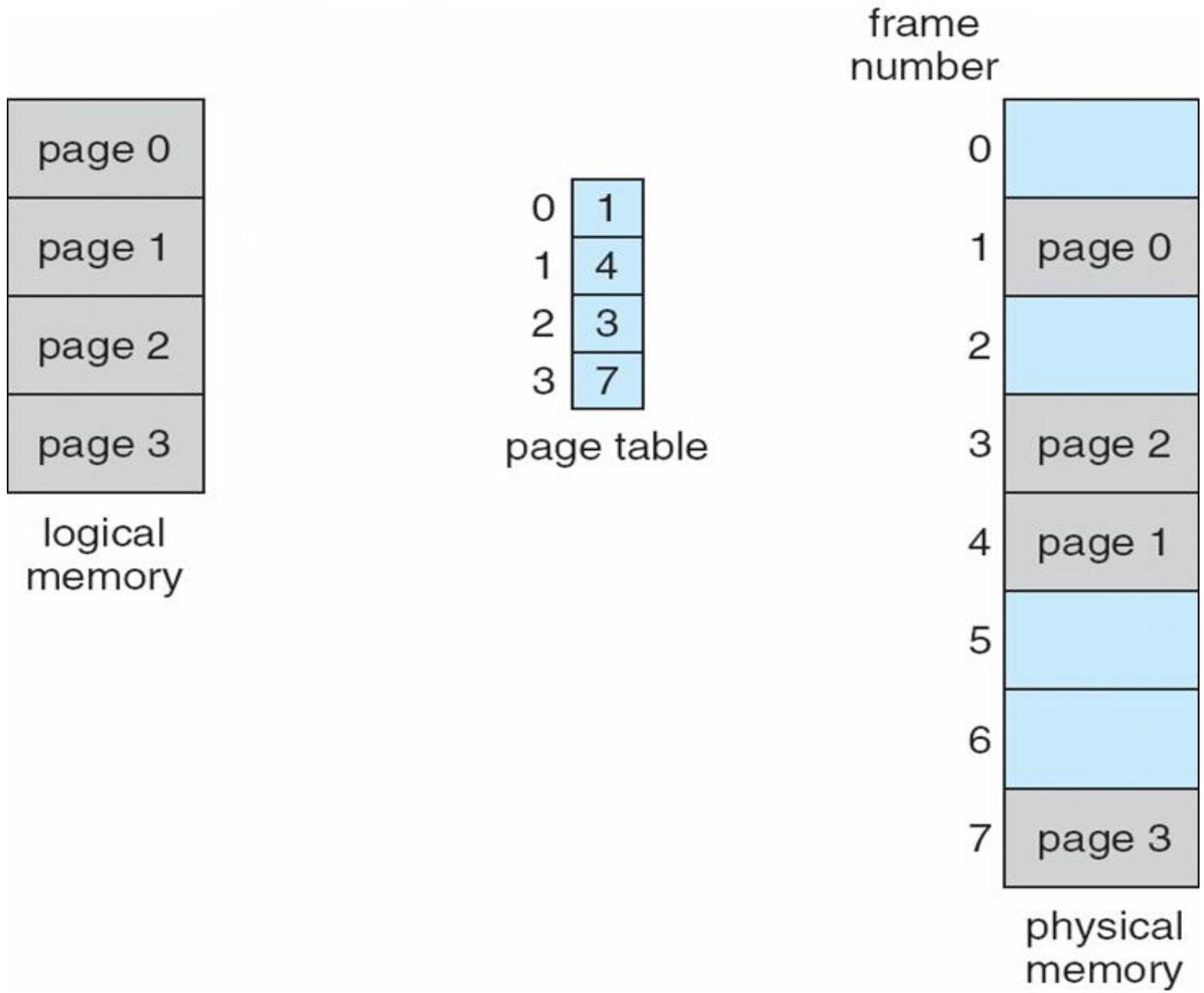
Bölüm 8: Ana Bellek (Main Memory)



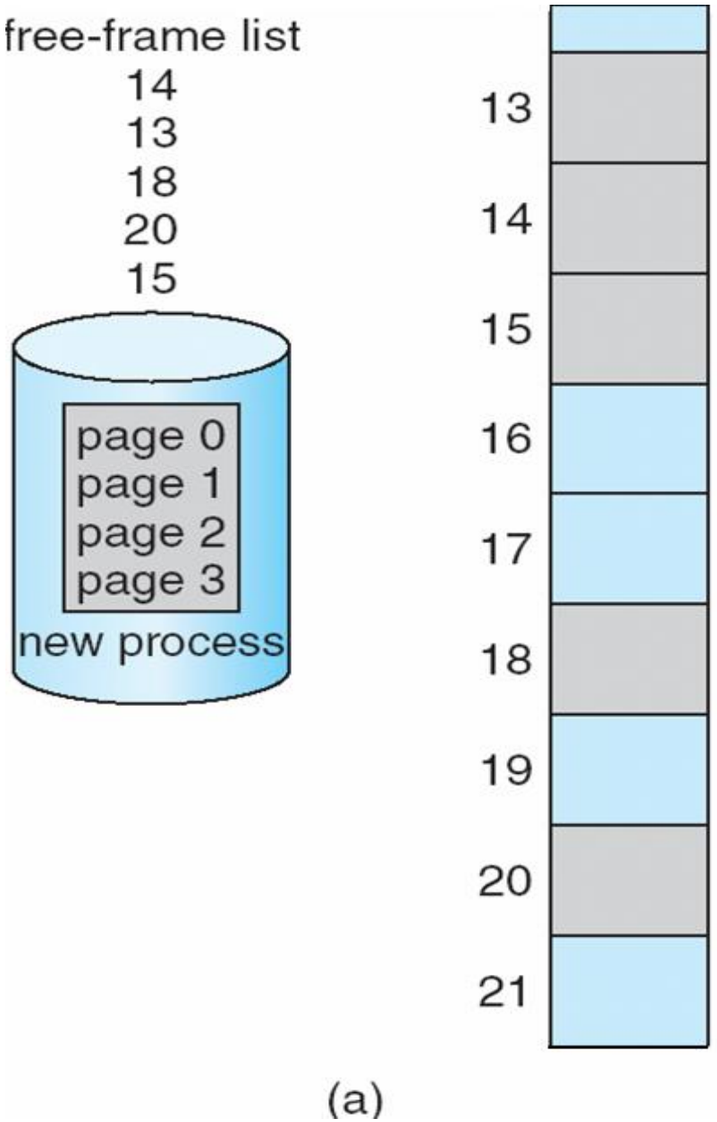
Donanım Sayfalama



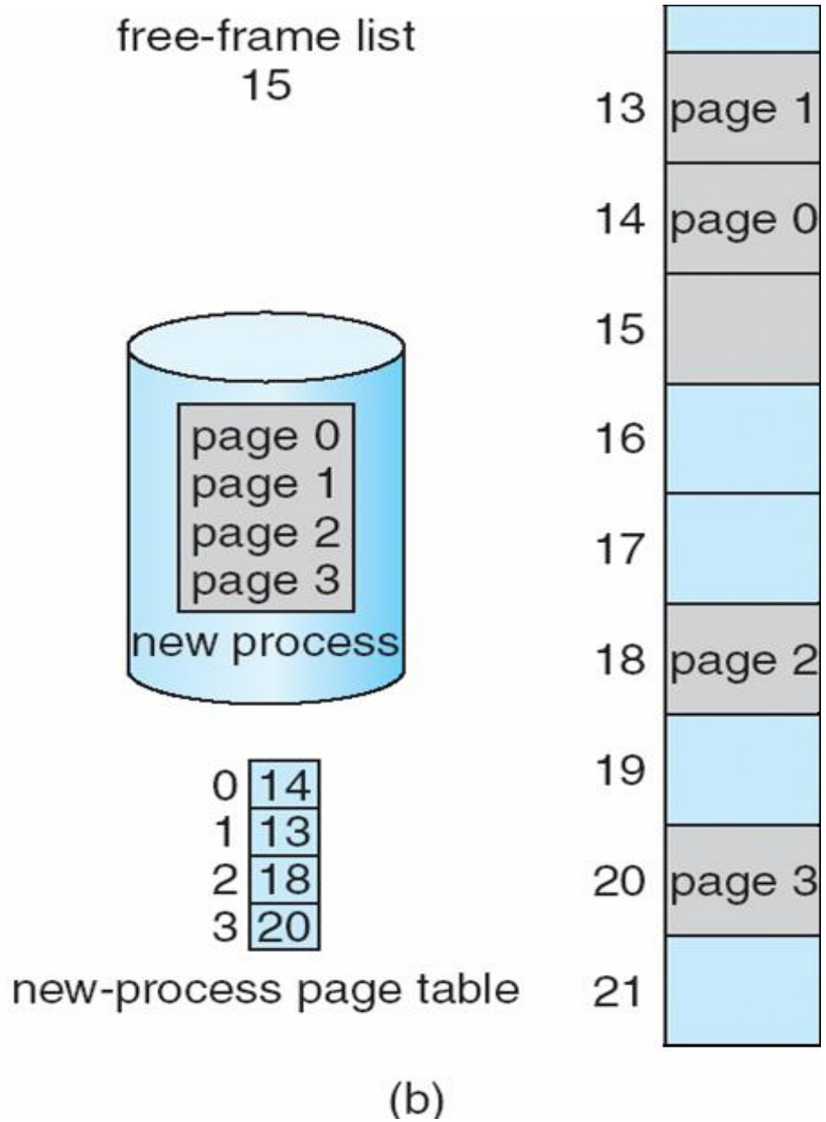
Mantıksal ve Fiziksel Bellek Sayfalama Modeli



Serbest Frame'ler



Before allocation



After allocation

Sayfa Tablosu Uygulaması

- Herbir işletim sist. Kendine ait sayfa tablosunu saklama metodu vardır. Bazıları proses kontrol bloğunda bu tabloyu saklar. Sayfa tablosu ana bellekte tutulur.
- **Page-table base register (PTBR)** sayfa tablosunu işaret eder.
- **Page-table length register (PTLR)** sayfa tablosunun boyutunu gösterir.
- Bu düzende her veri/komut iki bellek erişimine ihtiyaç duyar.
  - Sayfa tablosu için bir tane ve bir tane de veri/komut için.
- İki bellek erişimi problemi **ilişkisel bellek (associative memory)** ya da **translation look-aside buffers (TLBs)** olarak isimlendirilen özel hızlı-arama donanım önbelleği ile çözülebilir. Çünkü belleğe her erişim bu aşamadan geçmek zorunda.

Sayfa Tablosu Uygulaması

- TLB'ler adres alanı tanımlayıcılarını (**address-space identifiers-ASIDs)** depolarlar - bu işlem için adres alanı koruması sağlamak için her işlemi benzersiz/tekil olarak tanımlar
- TLB'ler tipik olarak küçük bellekler için (64 ila 1,024 kayıt tutar)
- Bir TLB hatası durumunda (TLB'de adres yoksa), bir dahaki sefere daha hızlı erişmek için TLB'ye değer yüklenir
  - TLB dolduysa, Değiştirme durumları göz önünde bulundurulmalıdır.(en son kullanılan veya random gibi)
  - Kalıcı hızlı erişim için bazı kayıtlar bağlanabilir (**wired down). TLB' ye sabitlenebilir**

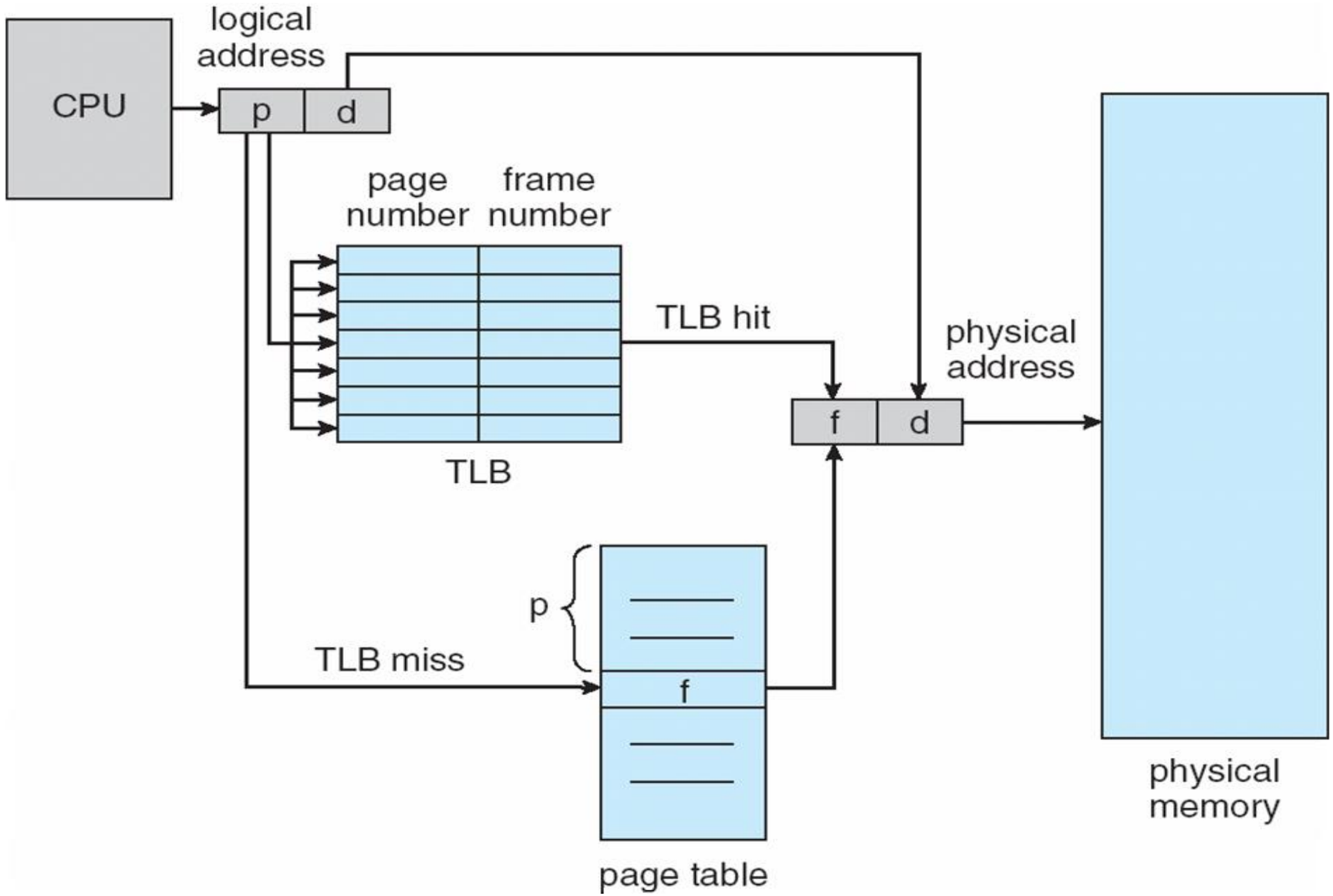
İlişkisel Bellek

- İlişkisel bellek – paralel arama

Page #	Frame #

- Adres dönüştürme (p, d)
  - Eğer p ilişkisel kaydedicini içinde ise, frame # çıkışını alın
  - Aksi taktirde, bellekteki sayfa tablosundan frame # al

TLB İle Donanım Sayfalama



Etkin Erişim Süresi

- İlişkisel arama = ε zaman birimi
  - Bellek erişim süresinin %10'undan az olabilir.
- İsabet (hit) oranı (TLB'de istenilen sayfa numarasını bulma oranı) = α
  - İsabet (hit) oranı – ilişkisel kayıtlar içerisinde bir sayfa bulunma süresinin yüzdesi; oran , ilişkili kaydedicilerin sayısı ile ilişkili
- α = 80% olsun , ε = 20ns TLB aramaları için, 100ns bellek erişimi için

Etkin Erişim Süresi

- **Etkin Erişim Süresi (Effective Access Time - EAT)**

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

- α = 80% olsun , ε = 20ns TLB aramaları için, 100ns bellek erişimi için
  - EAT = 0.80 x 120 + 0.20 x 220 = 140ns
- Daha yavaş bellek düşünün ancak daha iyi isabet oranı -> α = 98%, ε = 20ns TLB aramaları için, 140ns bellek erişimi için
  - EAT = 0.98 x 160 + 0.02 x 300 = 162.8ns

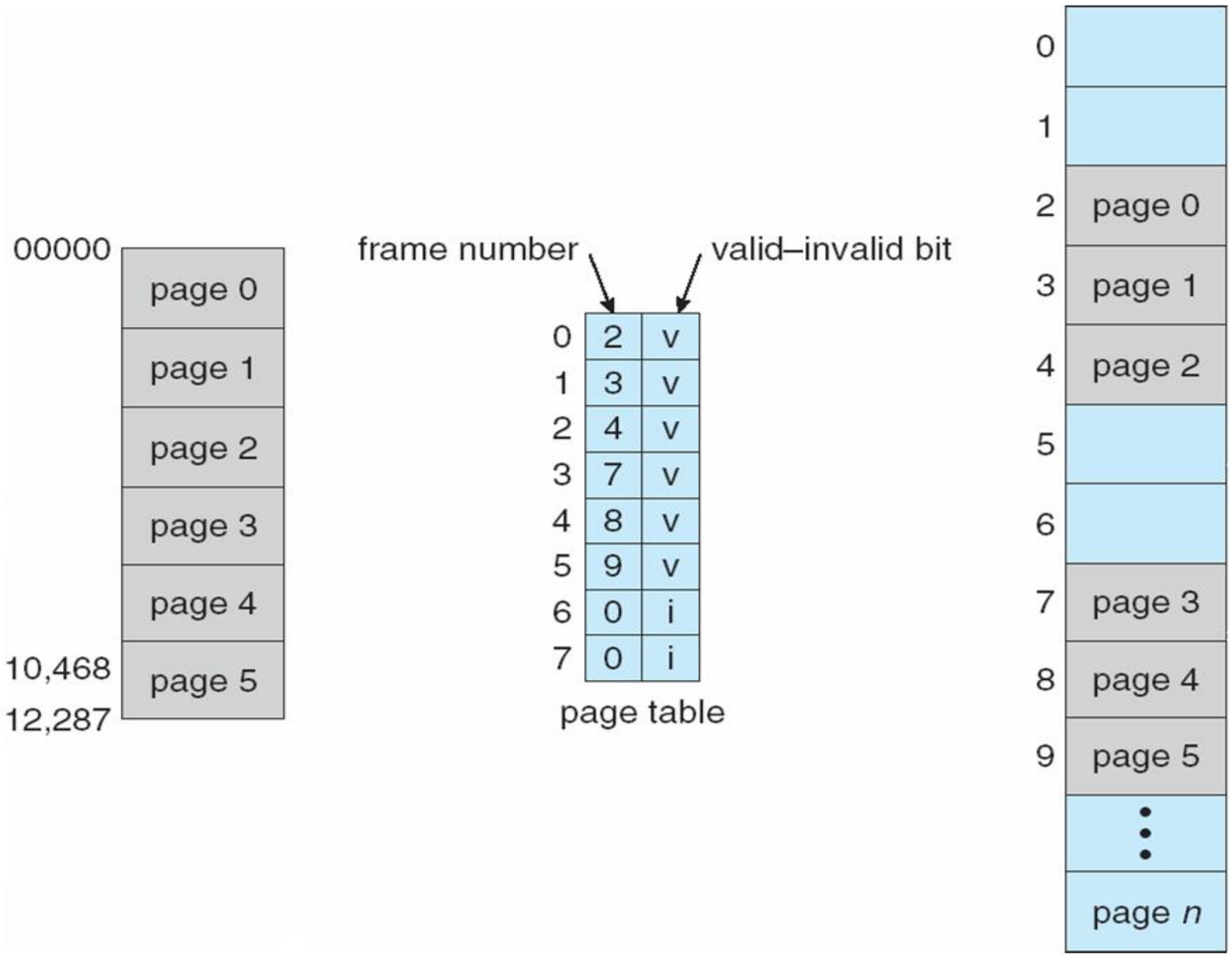
Bellek Koruması

- Okuma ya da okuma-yazma izninin olup olmadığını göstermek için her frame ile koruma biti ilişkilendirilerek bellek koruması uygulanır.
  - Ayrıca yalnızca-çalıştırılabilir (execute-only) sayfa göstermek için daha fazla bit eklenebilir, vb.

- Sayfa tablosunda her kayıt için geçerli-geçersiz (**Valid-invalid**) biti eklenir:
  - "Geçerli", ilişkili sayfanın işlemin mantıksal adres alanında olduğunu ve dolayısıyla yasal bir sayfa olduğunu gösterir
  - "Geçersiz", sayfanın işlemin mantıksal adres alanına girmediğini belirtir
  - Ya da PTLR(**page-table length register**) kullanılır.

- Herhangi bir ihlal, çekirdeği tuzağa düşürür

Sayfa Tablosundaki Geçerli (v) ya da Geçersiz (i) Bit

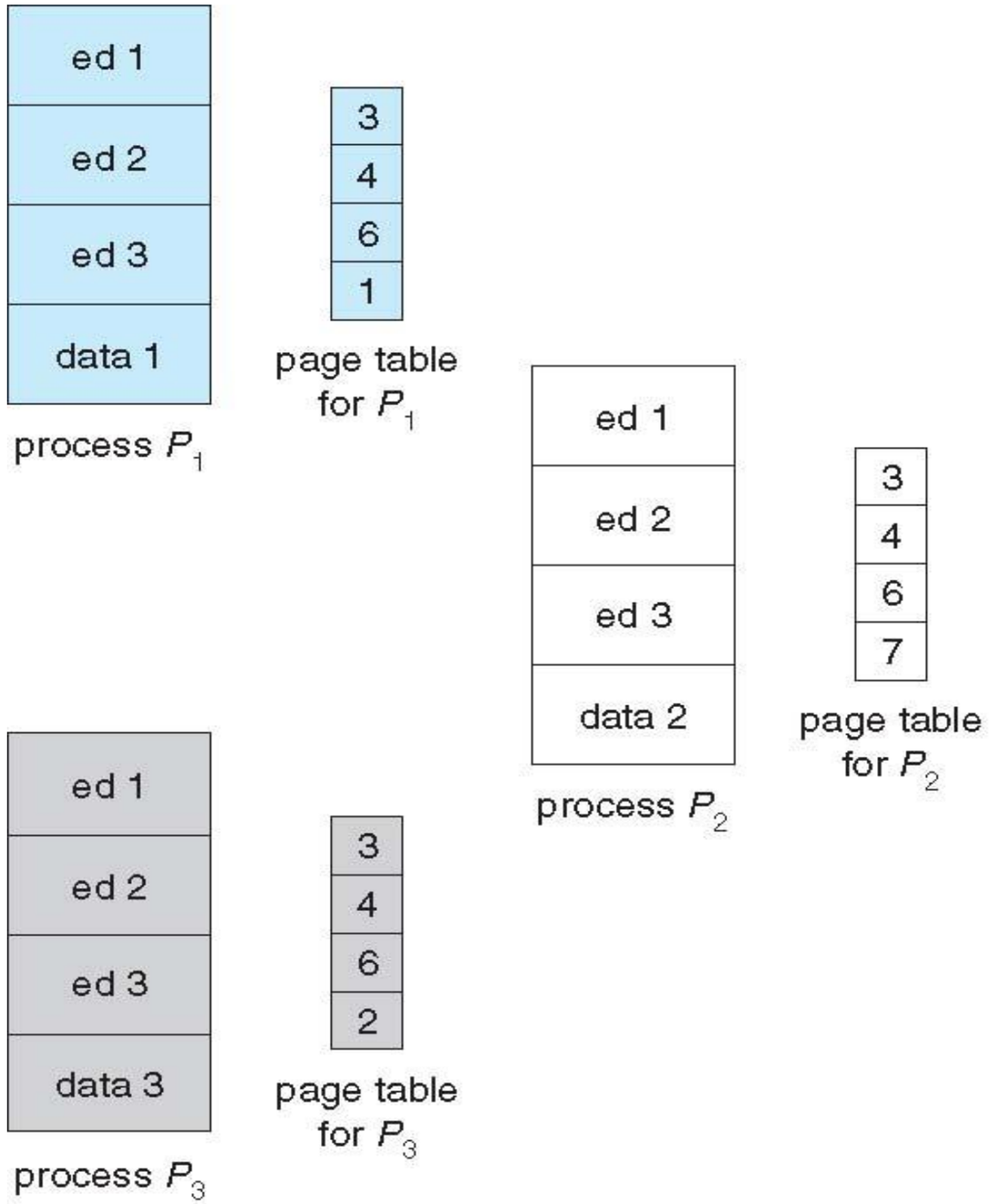


Paylaşımlı Sayfalar

- **Paylaşımlı kod**
  - Salt okunur kodun bir kopyası processler arasında paylaşılabilirır. (i.e., metin editörleri, derleyiciler, windows sistemleri)
  - Birden çok iş parçacığının aynı process alanını paylaşması gibi
  - Ayrıca, prosesler-arası iletişim için yararlı olur eğer ki okuma-yazma sayfalarının paylaşılmasına izin verilirse

- **Özel kod ve veri**
  - Her process kod ve verinin ayrı bir kopyasını tutar.
  - Özel kod ve veri için sayfalar, mantıksal adres alanı içindeki herhangi bir yerde görülebilirdir.

Paylaşımlı Sayfa Örneği



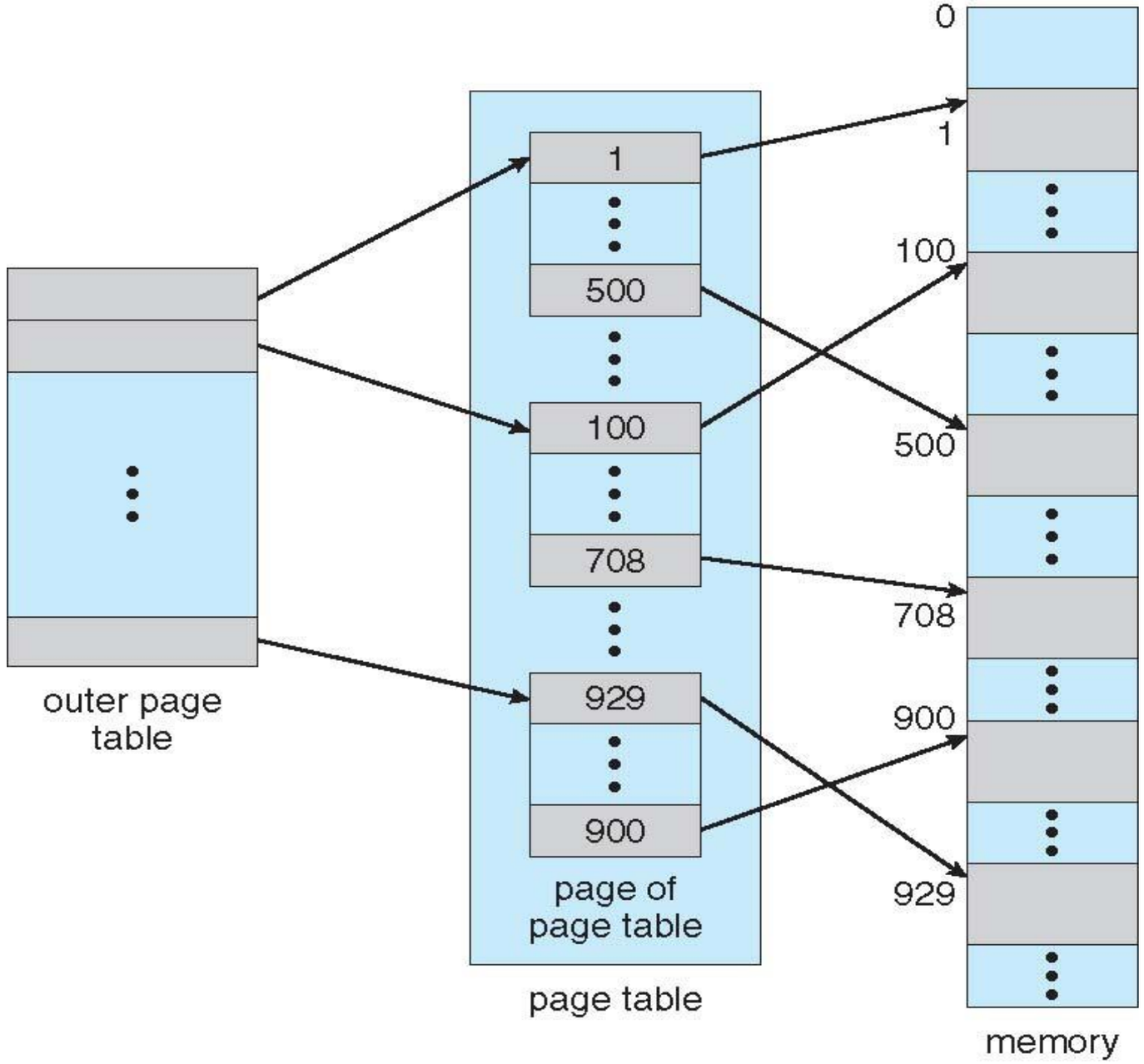
Sayfa Tablosunun Yapısı

- Sayfalama için bellek yapısı, devasa boyutlarda büyük olabilir.
  - Modern bilgisayarlarda 32-bit'lik mantıksal adresler (4 GB) olduğunu göz önüne alın.
  - Sayfanın boyutu 4 KB (2^{12})
  - Sayfa tablosunun 1 milyon kayıt olurdu (2^{32} / 2^{12})
  - Eğer Herbir kayıt 4 bytes ise-> 4 MB fiziksel adres alanı gerekir / sadece sayfa tablosu için alan
    - Çok fazla olan bu bellek miktarı
    - Ana bellekte o bitişik tahsis etmek istemeyiz.
- Hiyerarşik Sayfalama
- Hashed Sayfa Tabloları
- Inverted (Terslenmiş) Page Tables

Hiyerarşik Sayfa Tablolama

- Mantıksal adres alanını birden çok sayfa tablosuna bölün
- İki aşamalı sayfa tablosu basit bir tekniktir.
- Yani, sayfa tablosunu sayfalayacağız

İki Aşamalı Sayfa-Tablo Şeması



İki Aşamalı Sayfalama Örneği

- Bir mantıksal adres (32-bit'lik bir makine üzerinde 1K'lık sayfa boyutu ile şu şekilde ayrılıştır:
  - 22 bit'i sayfa numarasını oluşturur.
  - 10 bit'i sayfa ofsetini oluşturur.

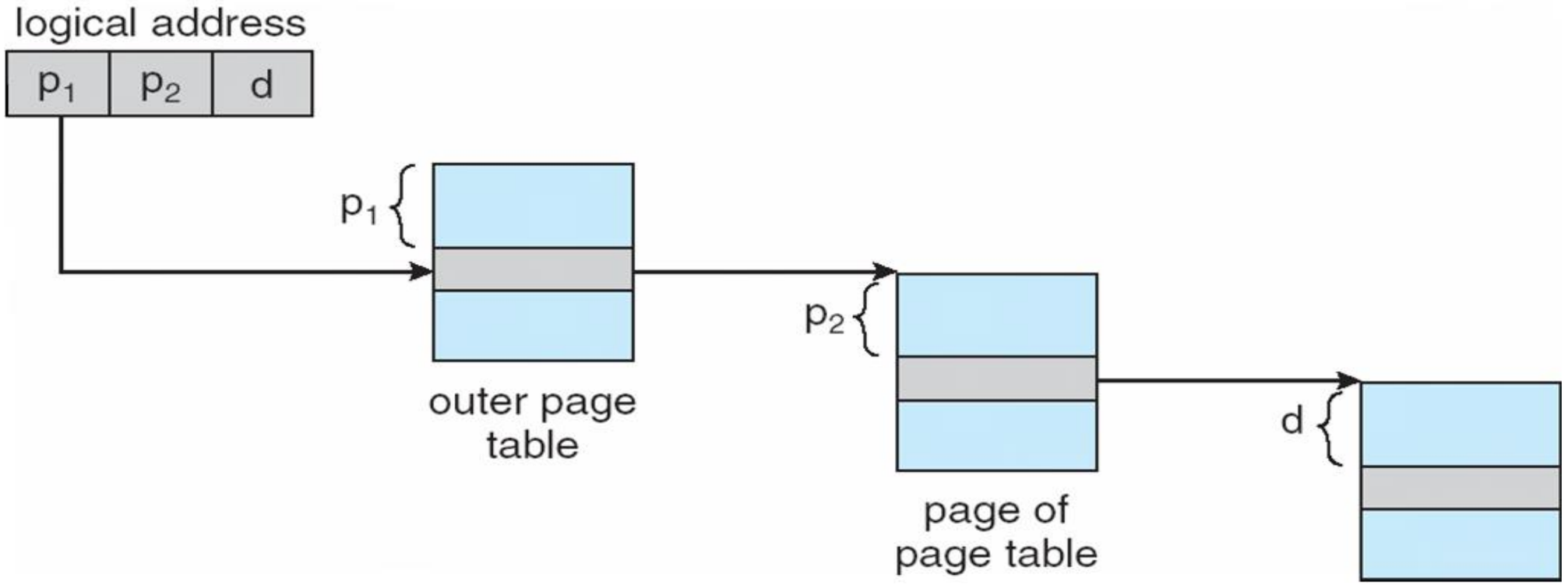
- Sayfa tablosu sayfalandırıldığından, sayfa numarası daha da bölünmüştür :
  - 12-bit'lik bir sayfa numarası
  - 10-bit'lik bir sayfa ofseti

- Böylece,bir mantıksal adres aşağıdaki gibi olur:

page number		page offset
p_1	p_2	d
12	10	10

- burada p_1 dış sayfa tablosundaki bir dizindir ve p_2 iç sayfa tablosunun sayfasındaki yer değiştirmedir
- **forward-mapped page table** olarak bilinir.

Adres Dönüşüm Şeması



64-bit Mantıksal Adres Alanı

- İki aşamalı sayfalama şeması da yeterli değildir.
- Eğer sayfa boyutu 4 KB (2^{12}) ise
  - Sayfa tablosunda 2^{52} kayıt vardır.
  - Eğer iki seviyeli şema olursa, iç sayfa tabloları 2^{10} 4-byte kayıt olabilir

outer page	inner page	page offset
p_1	p_2	d
42	10	12

  - Adres şunun gibi görünecektir:
  - Outer page table (dış sayfa tablosu) 2^{42} kayıt ya da 2^{44} byte'a sahip olabilir.
  - Bir çözüm ise ikinci bir dış sayfa tablosu eklemektir.
  - Fakat aşağıdaki örnekte ikinci dış sayfa tablosu hala 2^{34} byte boyutundadır.
    - Ve muhtemelen tek bir fiziksel bellek alanı almak için 4 bellek erişimi olacaktır.

Üç Aşamalı Sayfalama Şeması

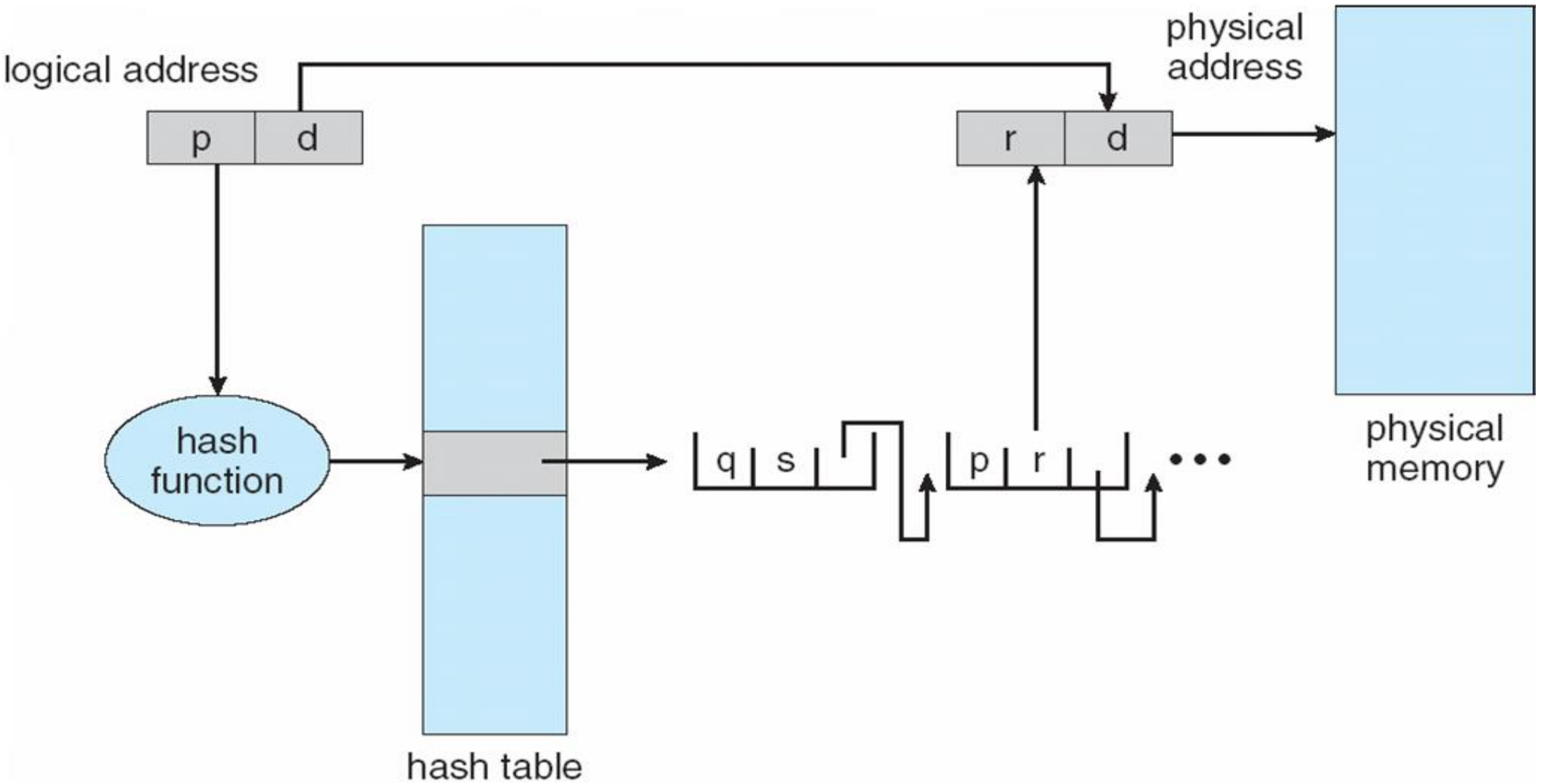
outer page	inner page	offset
p_1	p_2	d
42	10	12

2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

Hashed Sayfa Tabloları

- Genelde adres alanları > 32 bit
- Sanal sayfa numarası, bir sayfa tablosu içinde hashed(özetlenmiş-karılmış) durumdadır.
- Her eleman (1) sanal sayfa numarası (2) haritalanmış sayfa çerçeve değeri (3) sonraki eleman için bir işaretçi içerir.
- Sanal sayfa numarası bu dizin içinde bir eşleşme bulmak için karşılaştırılır.
  - Eğer eşleşme bulunduysa ilgili fiziksel frame elde edilir.

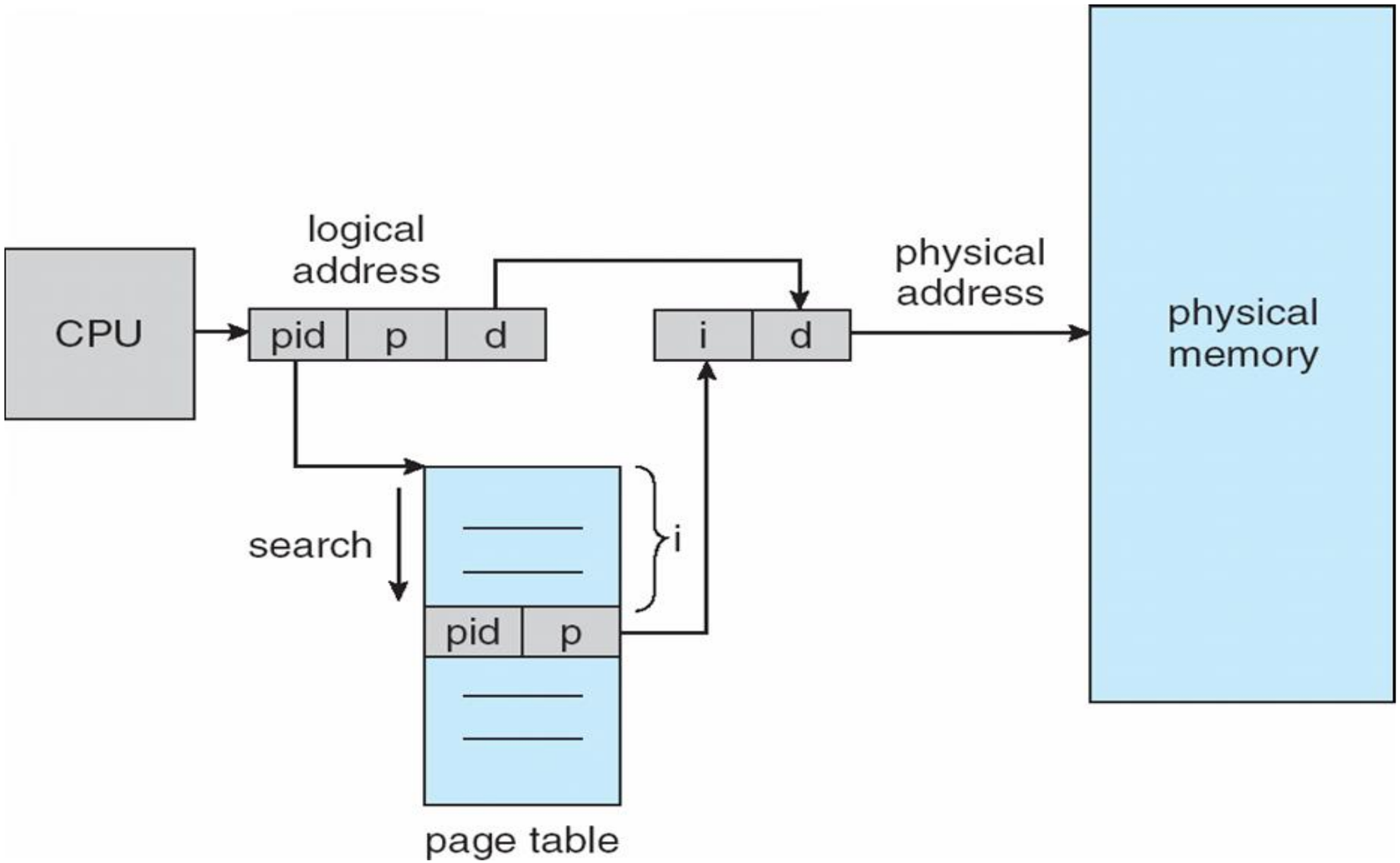
Hashed Sayfa Tablosu



Inverted (Terslenmiş) Sayfa Tablosu

- Her proses bir sayfa tablosuna sahip olmaktansa ve mümkün olan tüm mantıksal sayfaları takip etmek yerine, tüm fiziksel sayfalar takip edilir.
- Belleğin her gerçek sayfası için bir kayıt
- Kayıt, gerçek bellek konumunda saklanan sayfanın sanal adresini ve o sayfanın sahibi olan proses hakkında bilgi içerir
- Gerekli bellek miktarını azaltmak için her sayfa tablosu depolanmalıdır, fakat bir sayfa talebi olduğunda tablo aramak için gereken zaman artar.
- Aramayı sayfa tablosu kaydı ile sınırlandırmak için hash tablosu kullanılabilir
  - TLB erişimi hızlandırabilir.

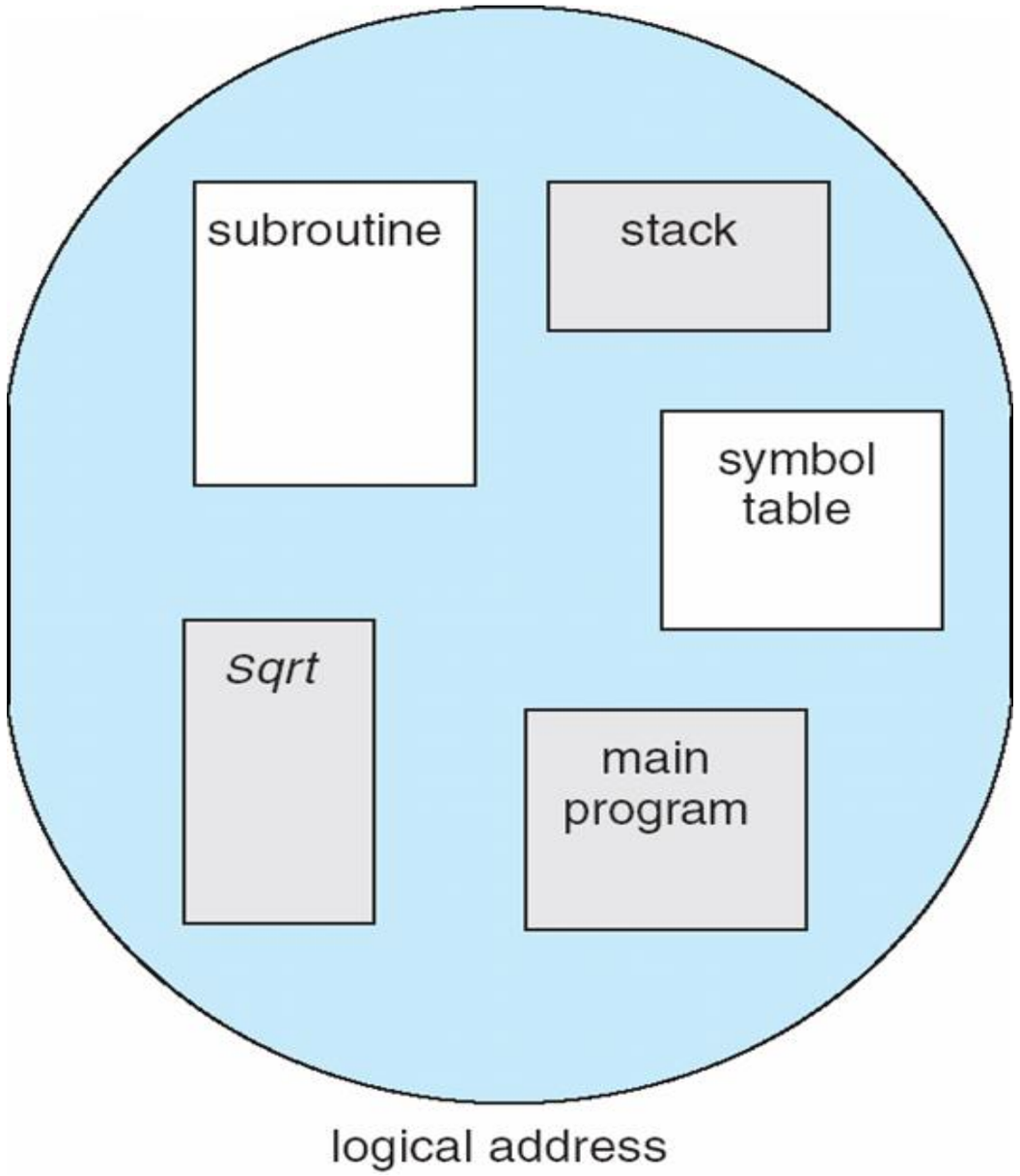
Inverted Page Table Architecture



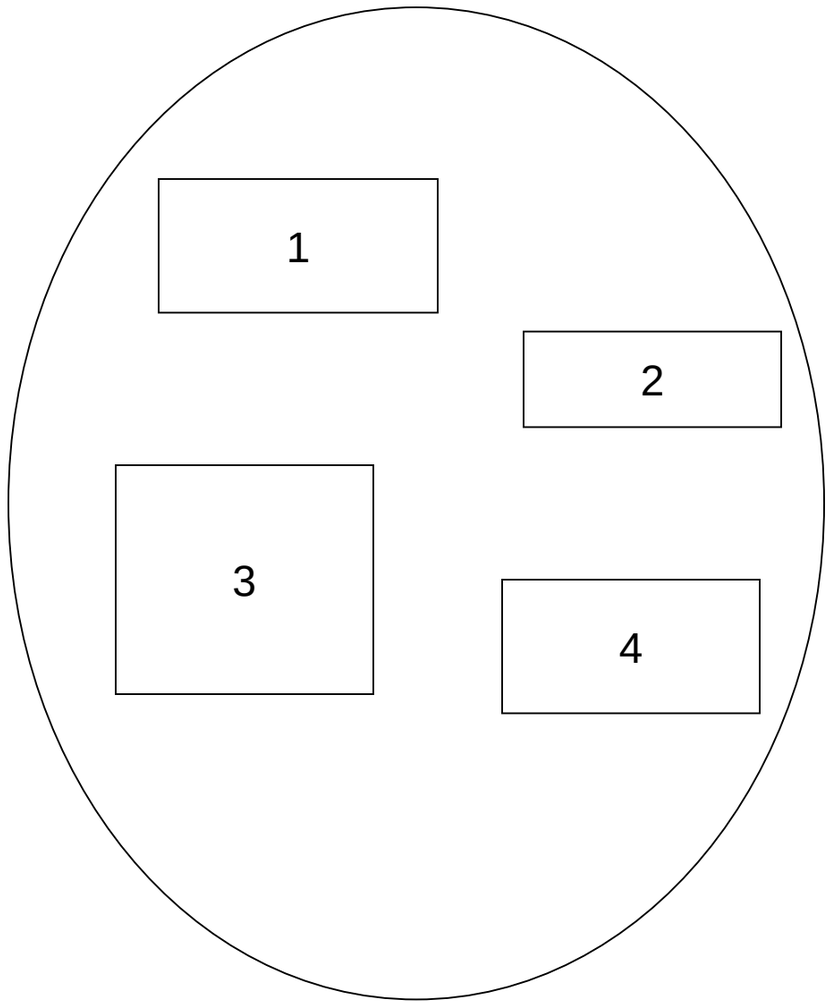
Segmentasyon

- Belleğin kullanıcı görünümünü destekleyen bellek yönetimi şeması.
- Program bir segmentler topluluğudur.
  - Bir segment, şunlar gibi mantıksal bir birimdir:
    - ana program
    - prosedür
    - fonksiyon
    - metot
    - nesne
    - yerel değişkenler, global değişkenler
    - ortak blok
    - yığın
    - sembol tablosu
    - diziler

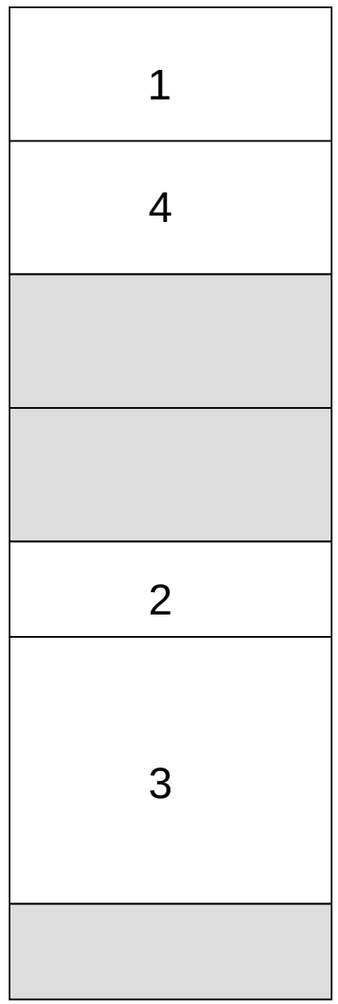
Bir Programın Kullanıcı Görünümü



Mantıksal Segmentasyon Görünümü



user space



physical memory space

Segmentasyon Mimarisi

- Mantıksal adres iki bölümden oluşur:

  <segment-numarası, ofset>,

- **Segment table (Segment Tablosu)**– iki boyutlu fiziksel adresleri haritalar; her tablo kaydında şunlar vardır:
  - **base** – segmentlerin fiziksel başlangıç adreslerini tutar.
  - **limit** – segmentin uzunluğunu belirtir.

- **Segment-table base register (STBR)** segment tablosunun bellekteki konumunu işaret eder.

- **Segment-table length register (STLR)** bir program tarafından kullanılan segment sayısını gösterir;

  s segment numarası olmak üzere, ***s*** < **STLR** olmalıdır.

Segmentasyon Mimarisi (Devam)

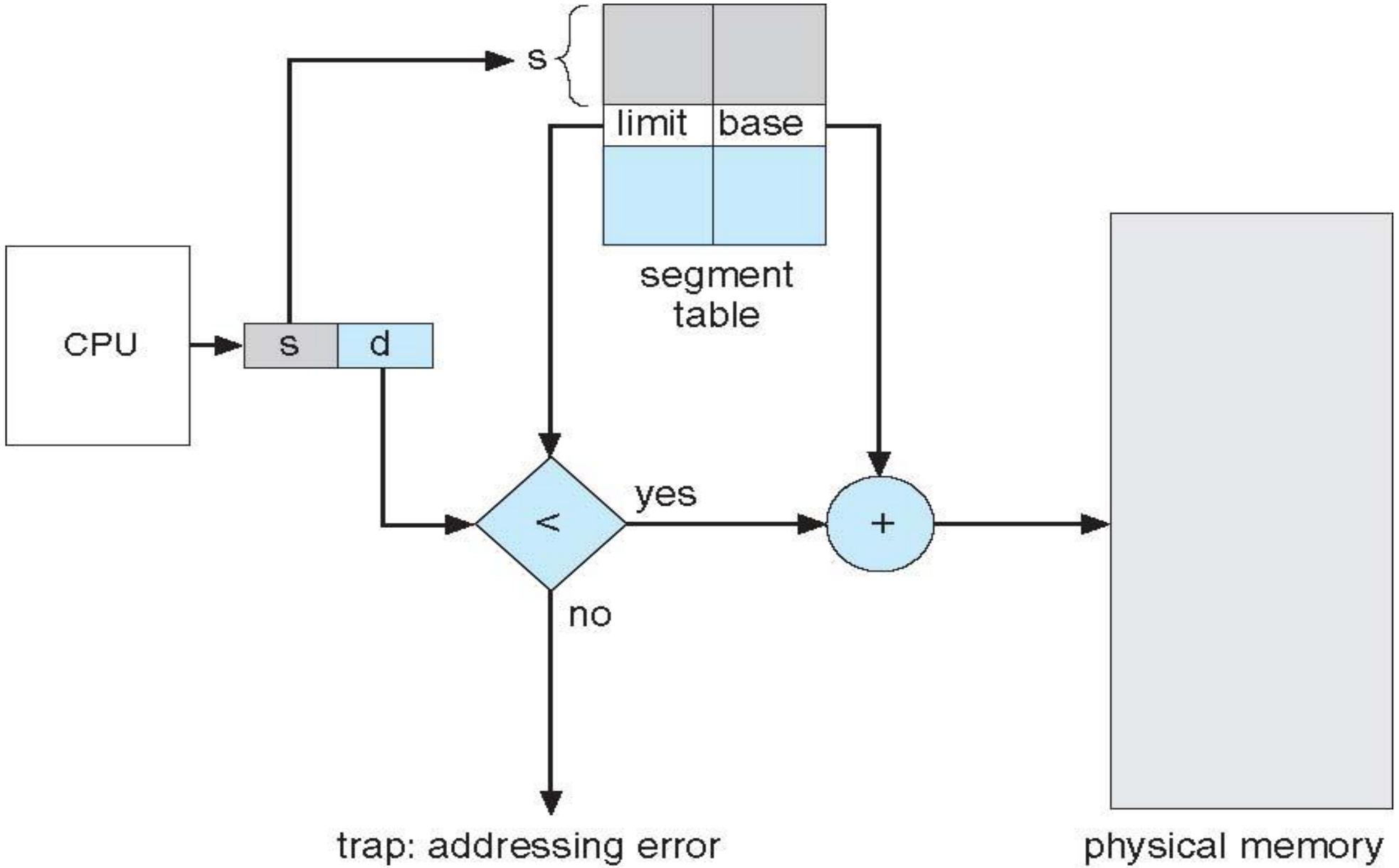
- Koruma
  - Segment tablosundaki her kayıt şunlar ile ilişkilendirilir:
    - Doğrulama bit = 0 ⇒ illegal segment
    - okuma/yazma/çalıştırma ayrıcalıkları

- Koruma biti segmentler ile ilişkilidir; kod paylaşımı segment düzeyinde gerçekleşir.

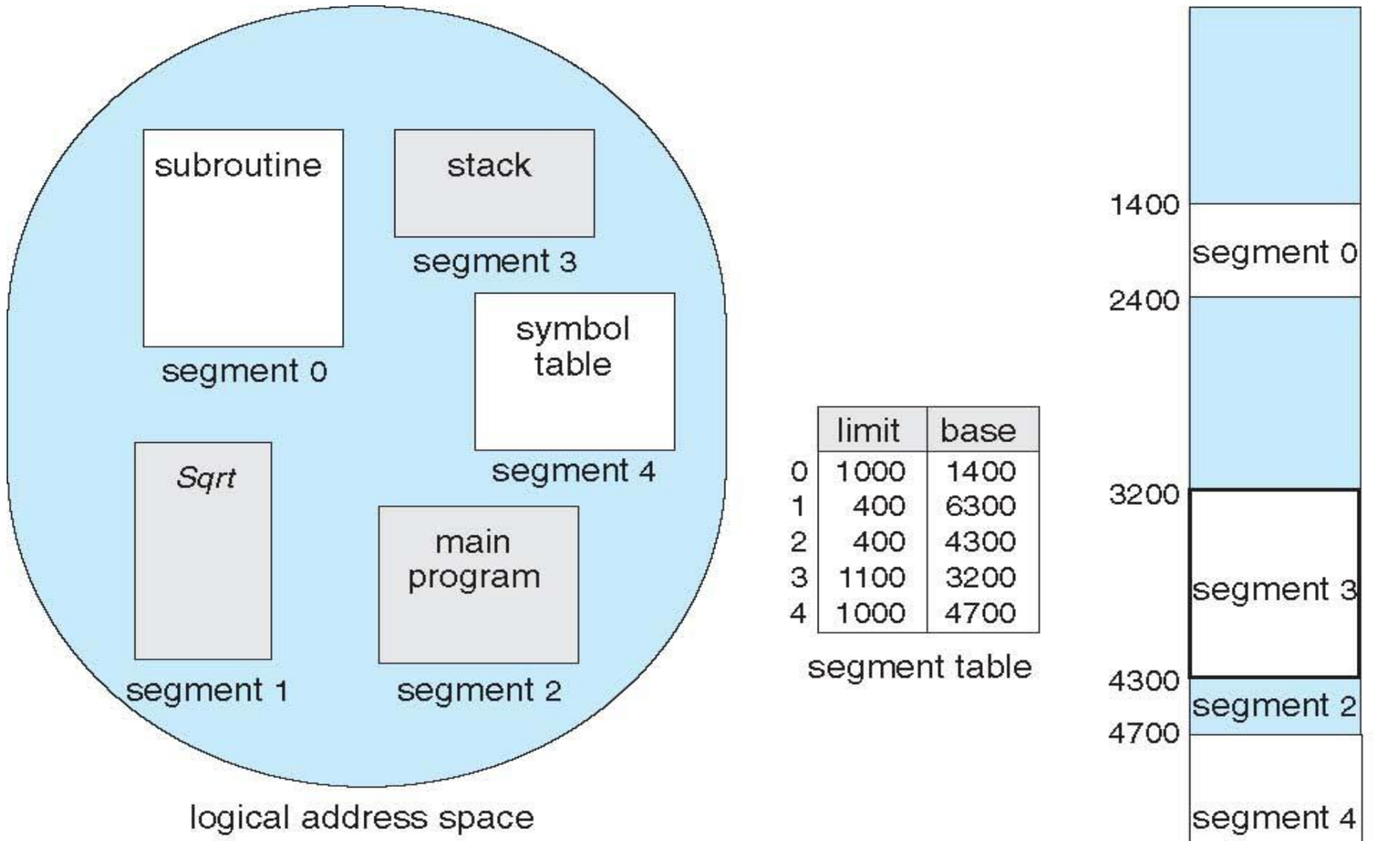
- Segment uzunlukları değişebildiğinden dolayı, bellek tahsisi bir dinamik depolama-tahsis problemidir.

- Bir segmentasyon örneği aşağıdaki diyagramda gösterilmiştir.

Donanım Segmentasyonu



Segmentasyon Örneği

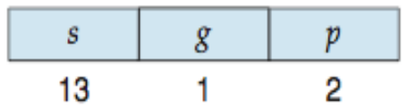


	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table

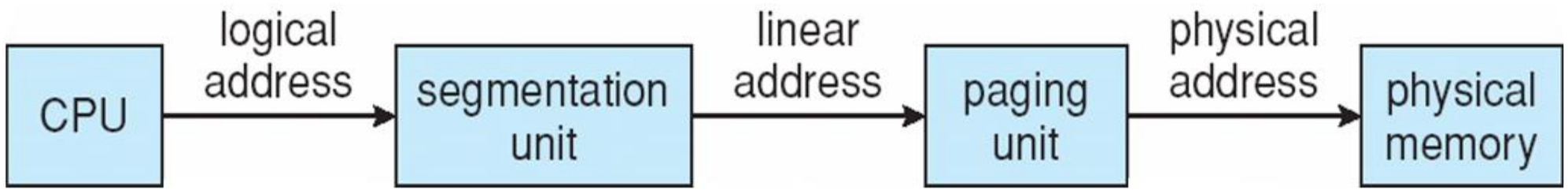
Örnek: The Intel Pentium

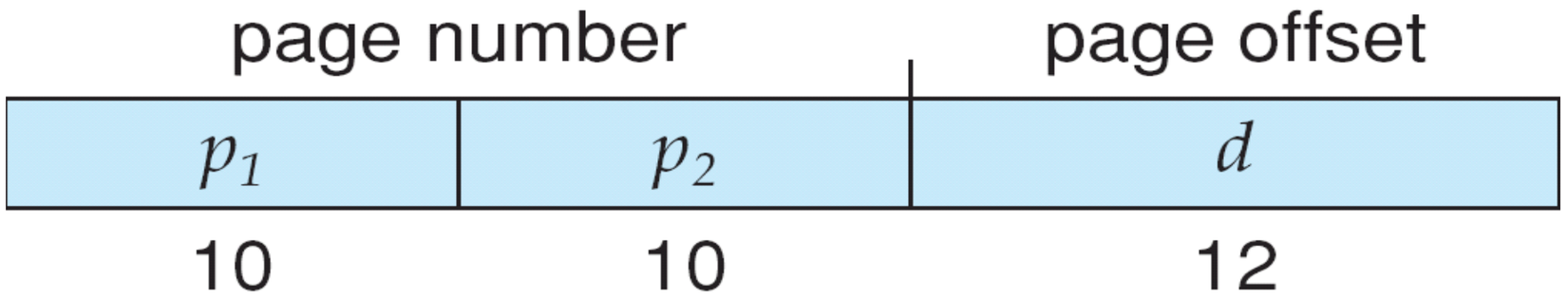
- Hem segmentasyonu hem de sayfalama ile segmentasyonu destekler.
  - Her segment 4 GB olabilir.
  - Process başına en fazla 16K segment olabilir
  - İki bölüme ayrılmıştır.
    - 8 K kadar olan ilk bölüm segmentleri prosese özeldir. (**local descriptor table LDT** ‘de tutulur.)
    - 8 K kadar olan ikinci bölüm tüm processler arasında paylaşılır.(**global descriptor table GDT** ‘de tutulur.)
- CPU mantıksal adresi oluşturur.
  - Segmentasyon birimine verir.
    - Doğrusal adresler üretilir.



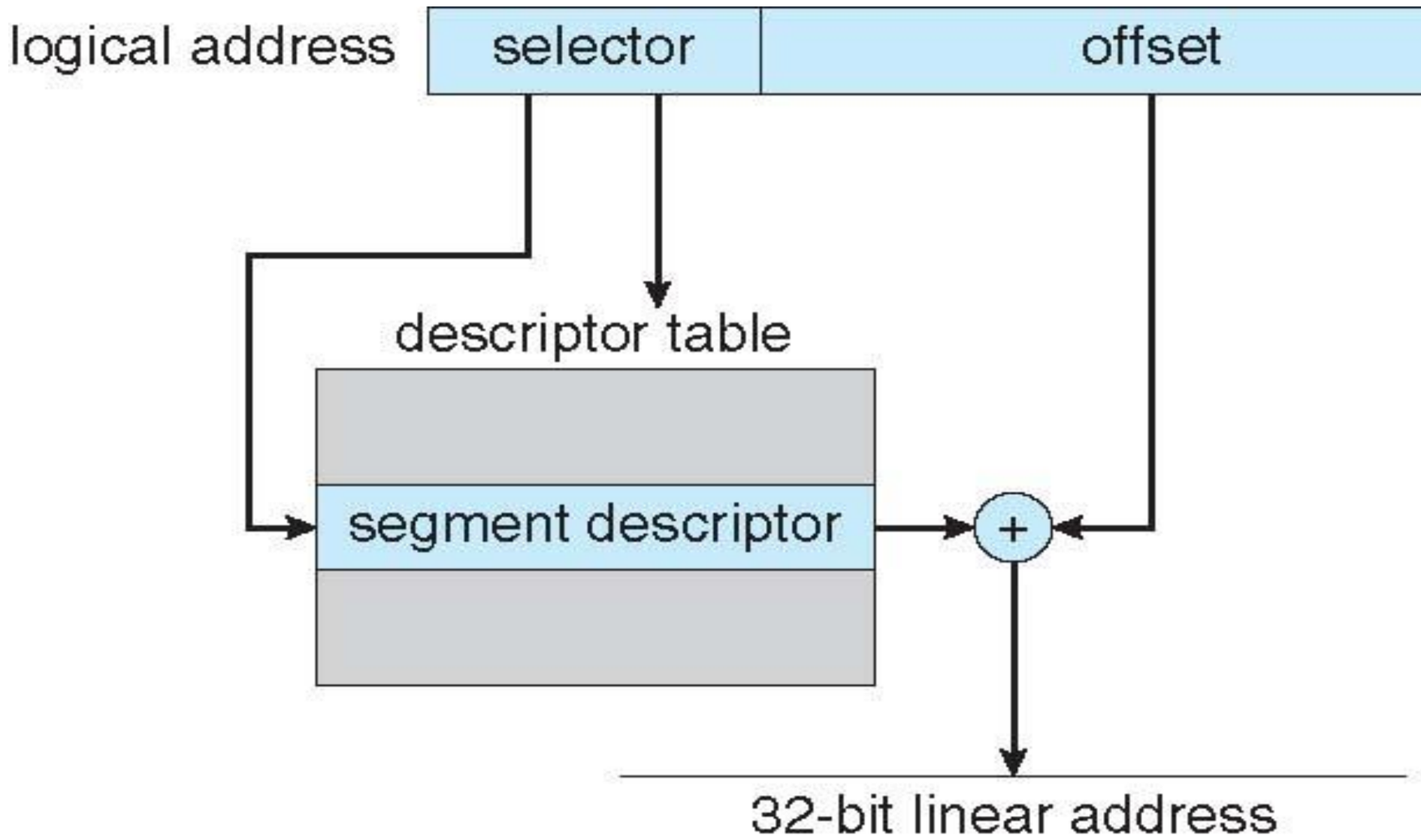
  - Doğrusal adres sayfalama birimine verilir.
    - Ana bellekte fiziksel adres üretir.
    - Sayfalama birimleri eşdeğer MMU oluşturur.
    - Sayfaların boyutları 4 KB ya da 4 MB olabilir.

Pentium'da Mantıksal Adresin Fiziksel Adrese Dönüştürülmesi

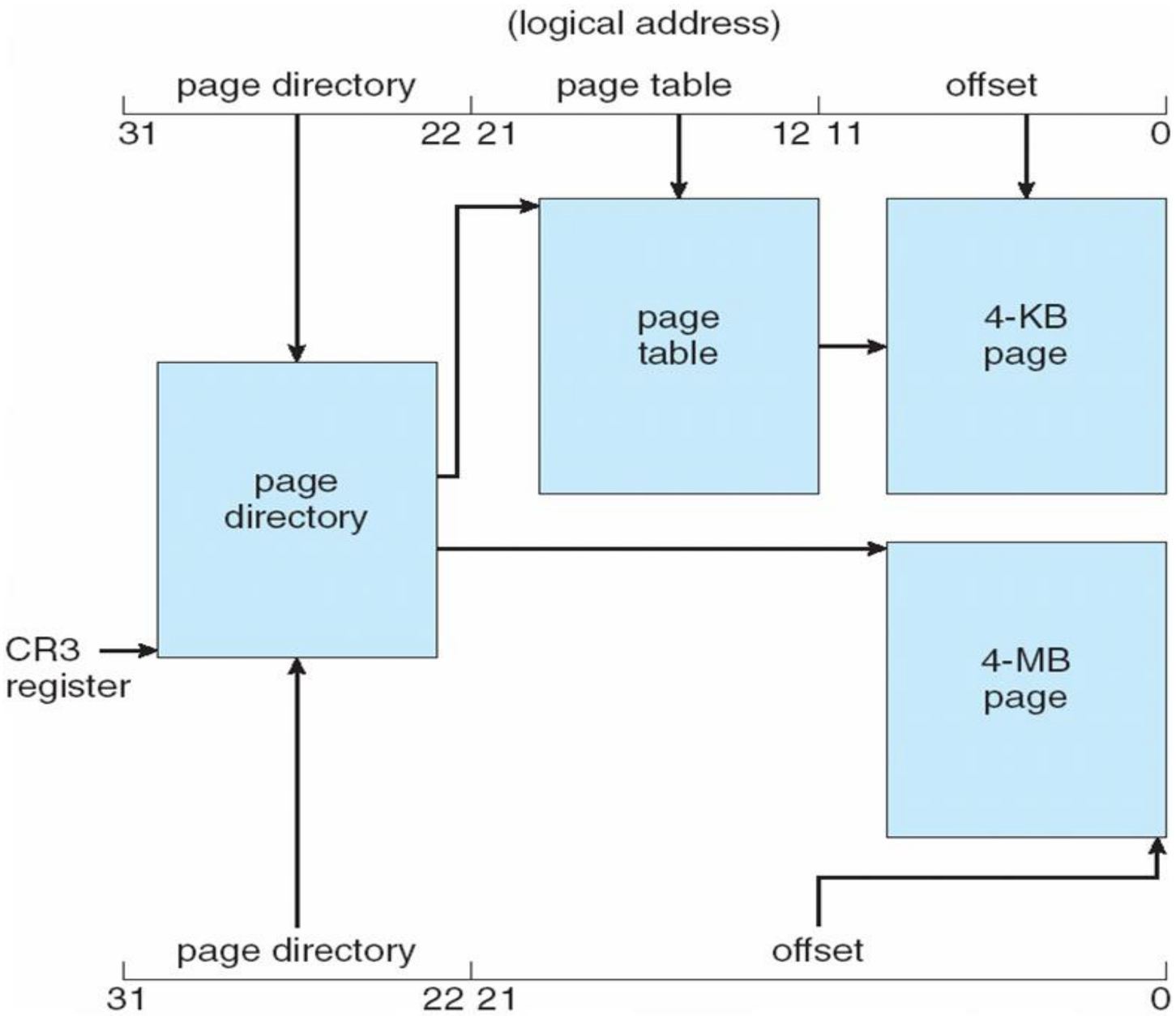




Intel Pentium Segmentasyonu

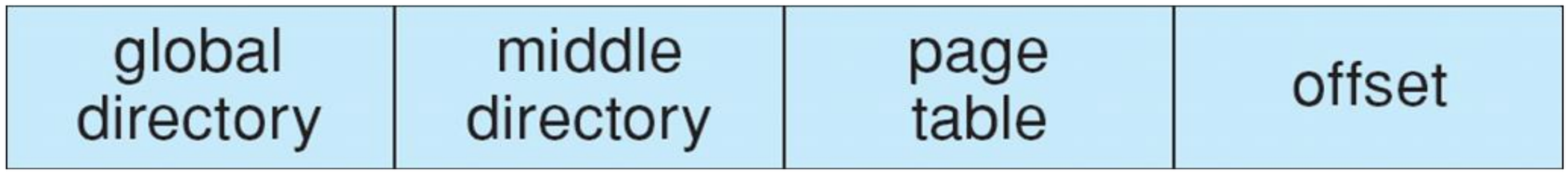


Pentium Sayfalama Mimarisi



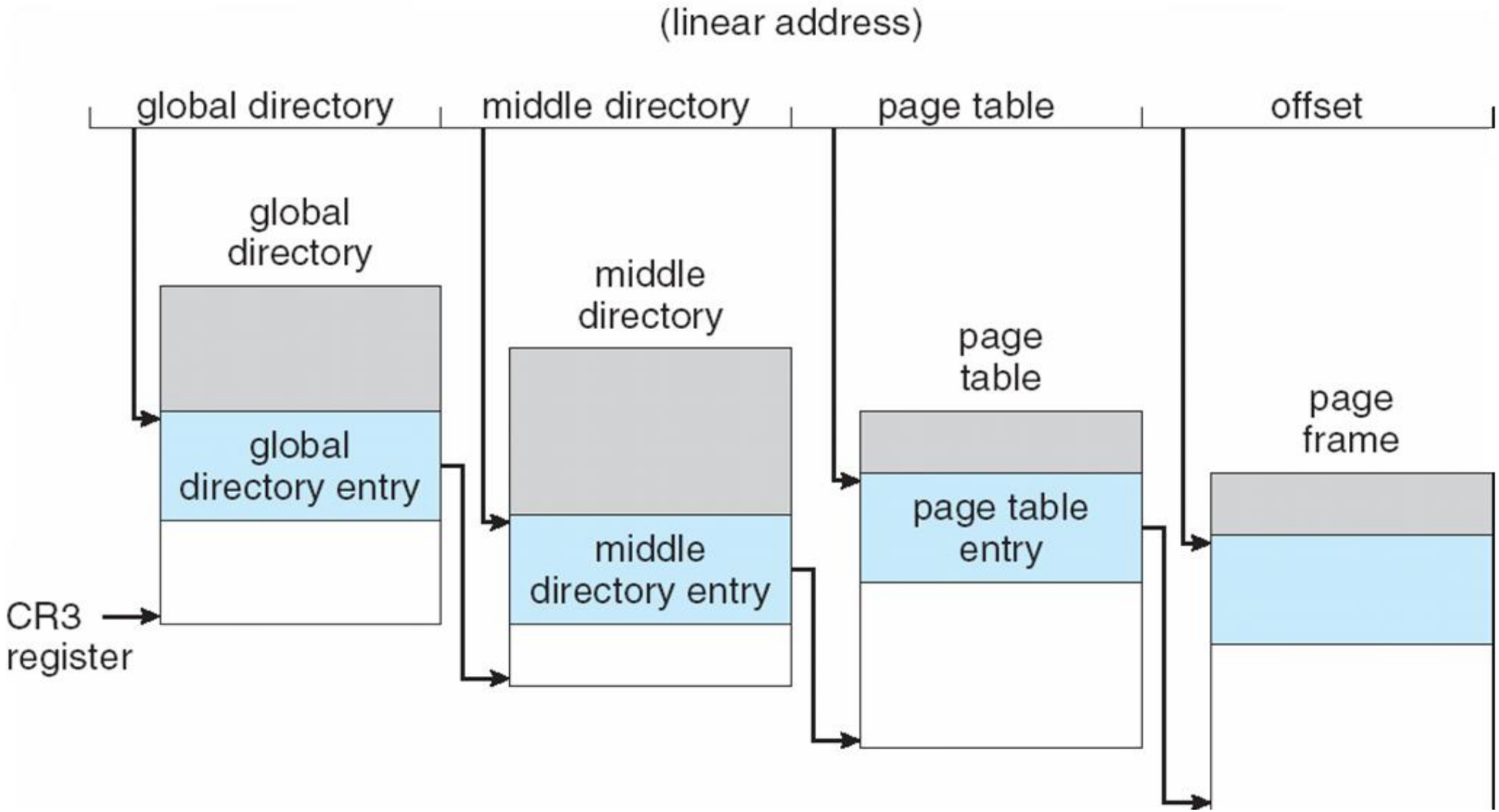
Linux'ta Doğrusal Adres

- Linux yalnızca 6 segment kullanır.(kernel kodu, kernel verisi, kullanıcı kodu, kullanıcı verisi, görev-durum segmenti (TSS), default LDT segmenti)
- Linux 4 olası modun yalnızca ikisini kullanır.– kernel ve kullanıcı
- 32-bit ve 64-bit sistemlerde sağlıklı çalışan üç-aşamalı sayfalama stratejisi kullanır.
- Doğrusal adres dört bölüme ayrılır:



- Fakat Pentium sadece 2-aşamalı sayfalamayı destekler?!

Linux’ta Üç Aşamalı Sayfalama



7. Bölüm Sonu