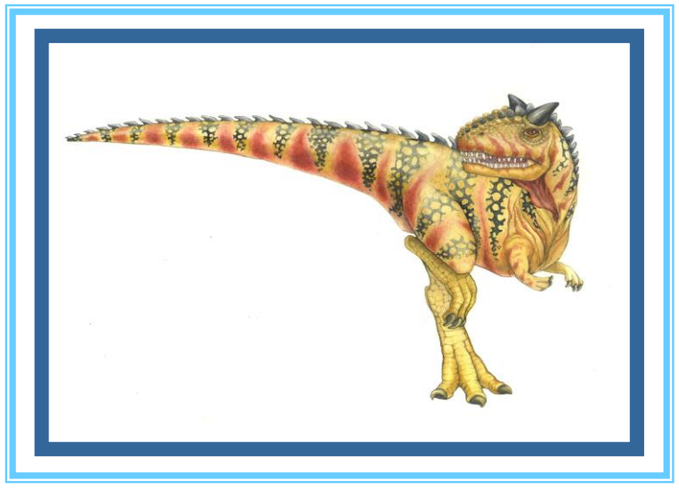
Bölüm 8: Ana Bellek (Main Memory)



Bölüm 8: Bellek Yönetimi

- Arka plan
- Takas (Swapping)
- Ardışık Bellek Tahsisi (Contiguous Memory Allocation)
- Sayfalama (Paging)
- Sayfa Tablosunun Yapısı
- Segmentasyon
- Örnek: Intel 32 and 64-bit Mimarisi
- Örnek: ARMv8 Mimarisi

Hedefler

- Bellek donanımını organize etme yollarını detaylı bir şekilde açıklamak
- Sayfalama ve segmentasyon da dahil olmak üzere çeşitli bellek yönetim teknolojilerini tartışmak
- Sadece segmentasyon ve sayfalamalı segmentasyon tekniklerinden her ikisini de destekleyen Intel Pentium'u detaylı bir şekilde tanımlamak

Arkaplan

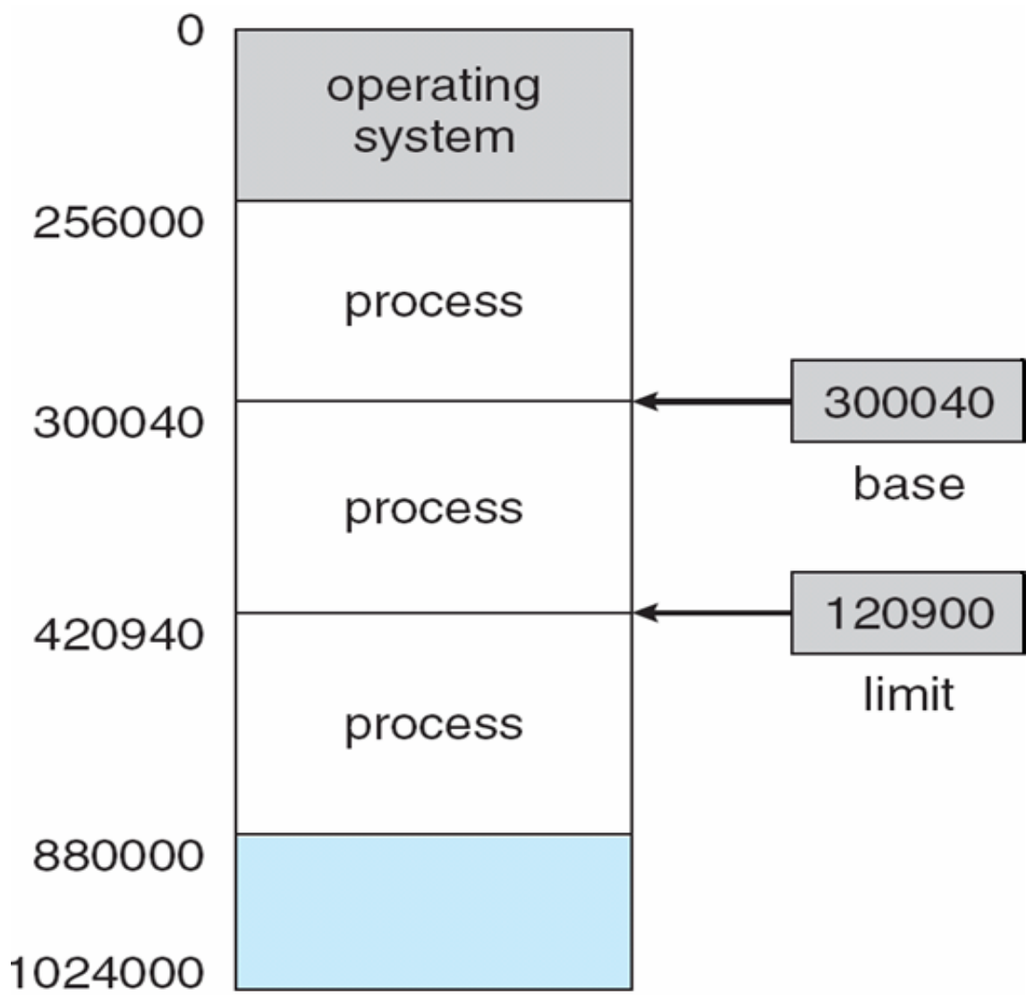
- Bir bilgisayar sisteminin ana amacı, programları çalıştırmaktır.
- Bu programlar, eriştikleri verilerle birlikte, yürütme sırasında en azından kısmen ana bellekte olmalıdırlar.
- Hem CPU'nun kullanımını hem de kullanıcılara verdiği yanıtın hızını iyileştirmek için, genel amaçlı bir bilgisayar, birkaç işlemin bellekte tutulması gerekir.
- Çeşitli yaklaşımları yansıtan bir çok bellek yönetim şeması mevcuttur ve her bir algoritmanın etkinliği duruma bağlıdır.
- Bir sistem için bir bellek yönetimi şemasının seçilmesi birçok faktöre bağlıdır, özellikle sistemin donanım tasarımı üzerinde.
- Çoğu algoritma, donanım desteği gerektirir.

Arkaplan

- Program, diskten belleğe getirilip çalışması için bir process'e yerleştirilmelidir.
- Ana bellek ve kaydediciler CPU'nun doğrudan erişebildiği kayıt ortamlarıdır
- Bellek ünitesi yalnızca adresler + okuma istekleri veya adres + veri ve yazma istekleri ile ilgilenir
- Kaydedici erişimi bir CPU çevriminde (veya daha az) yerine getirilir
- Ana bellek bir den fazla çevrimde erişilebilir
- **Ön bellek (Cache),** ana bellek ve CPU kaydedicileri arasında yer alır
- Belleğin korunması belleğin doğru çalışmasını sağlamak için gereklidir.

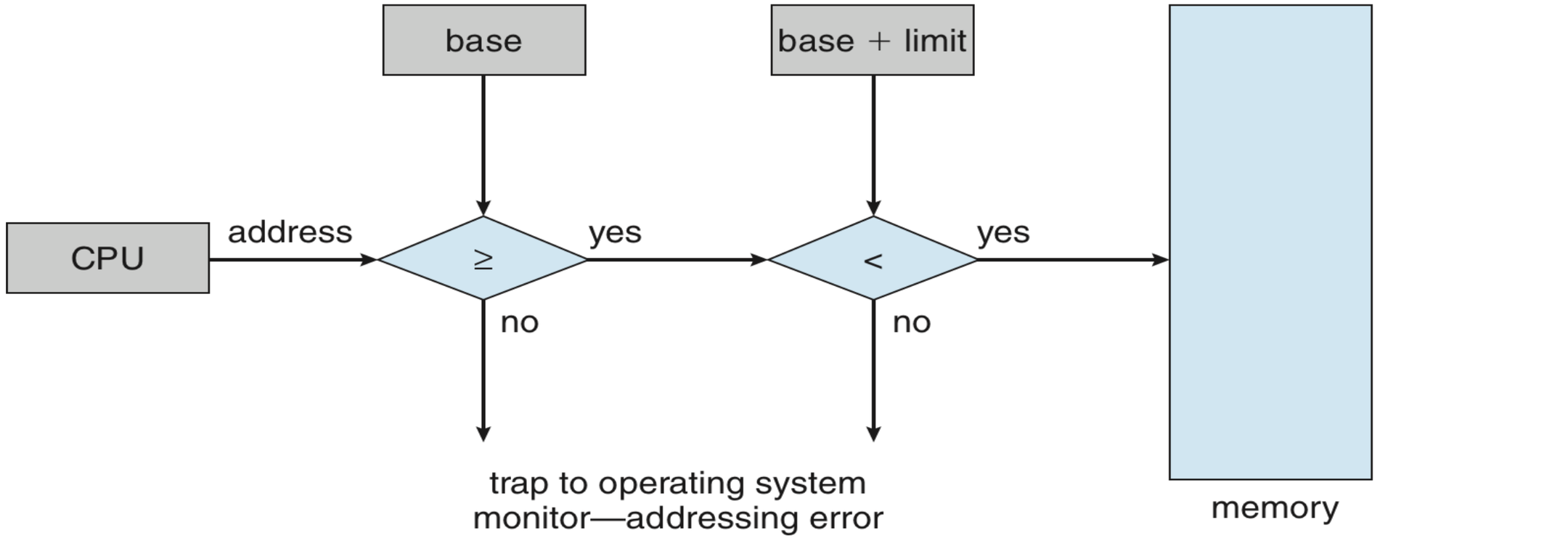
Taban ve Limit Kaydedici

- **Base** ve **limit** kaydedici çifti mantıksal adres alanı olarak tanımlanırlar.
- CPU'nun, kullanıcı modunda üretilen her bir bellek erişimini denetlemesi gerekir (bu kullanıcı için tanımlı olan base ve limit sınırları içinde olduğunu garantilemek için)



Base ve Limit Kaydedicileri ile Donanım Adresi Koruma

- CPU, kullanıcı modunda oluşturulan her bellek erişimini kontrol etmeli ve bu kullanıcı için taban ve limit arasında olduğundan emin olmalıdır.



Adres Bağlama

- Genellikle, bir program, bir disk üzerinde ikili (binary) bir yürütülebilir dosya olarak bulunur.
- Uygulanacak program, belleğe alınmalı ve bir prosese yerleştirilmelidir.
- Kullanılan bellek yönetimine bağlı olarak, işlem yürütülürken disk ve bellek arasında taşınabilir.
- Diskte duran ve yürütülmek için belleğe alınmayı bekleyen prosesler, girdi kuyruğunu (input queue) oluşturur.
- Normal prosedüre göre, girdi kuyruğundaki işlemlerden birini seçilir ve bu işlem belleğe yüklenir.
- İşlem yürütülürken, bellekteki komutlara ve verilere erişir.
- Sonunda proses sona erer ve bellek alanı kullanılabilir olarak bildirilir. Yani adres alanı geri verilir.

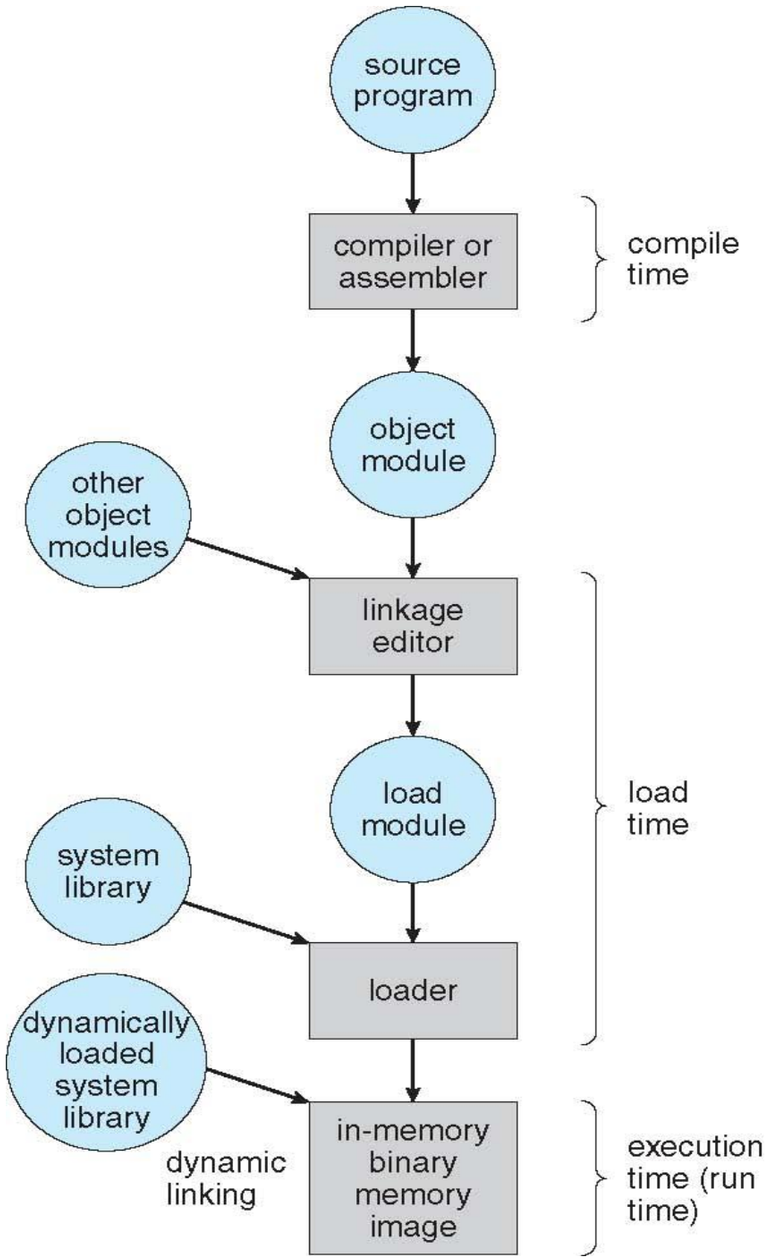
Adres Bağlama

- Çoğu sistem bir kullanıcı prosesinin fiziksel belleğin herhangi bir bölümünde bulunmasına izin verir.
- Böylece, bilgisayarın adres alanı 00000'de başlayabilmesine rağmen, kullanıcı işleminin ilk adresi 00000 olmasına gerek yoktur. Daha sonra, bir kullanıcı programının bir işlemi gerçekte fiziksel bellekte nasıl bulunduğunu göreceksiniz.
- Ayrıca, adresler bir programın yaşamının farklı aşamalarında farklı yollarla gösterilir.
  - Kaynak kod adresleri genellikle semboliktir. («Sayaç» değişkeni gibi)
  - Bir derleyici, genellikle bu sembolik adresleri yeniden taşınabilir adreslere **bağlar**.
    - örneğin. “bu modülün başından itibaren 14 bytes”
  - Bağlayıcı (linkage) veya Yükleyici (loader) yeniden konumlandırılabilir bu adresleri değişmez adreslere bağlar
    - örneğin. 74014
  - Her bir bağlama bir adres uzayını diğerine dönüştürür

Komutları ve Veriyi Belleğe Bağlama

- Komutların ve verinin bellek adreslerine bağlanması üç durumda olabilir:
  - **Derleme zamanı**: Eğer bellek konumu önceden bilinirse, **mutlak kod** üretilebilir. Eğer başlangıç konumu değişirse yeniden derlenmelidir.
  - **Yükleme zamanı:** Bellek konumu derleme zamanında bilinmiyorsa **yeniden konumlandırılabilir kod** üretilmelidir.
  - **Çalışma zamanı**: Eğer proses çalışma esnasında bir bellek kesiminden diğerine hareket ederse, bağlama çalışma anına kadar geciktirilir.
    - Adres haritalama için donanım desteği gerekir (örneğin, taban (base) ve tavan (limit) kaydedicileri)

Bir Kullanıcı Programının Çok Adımlı Çalışması



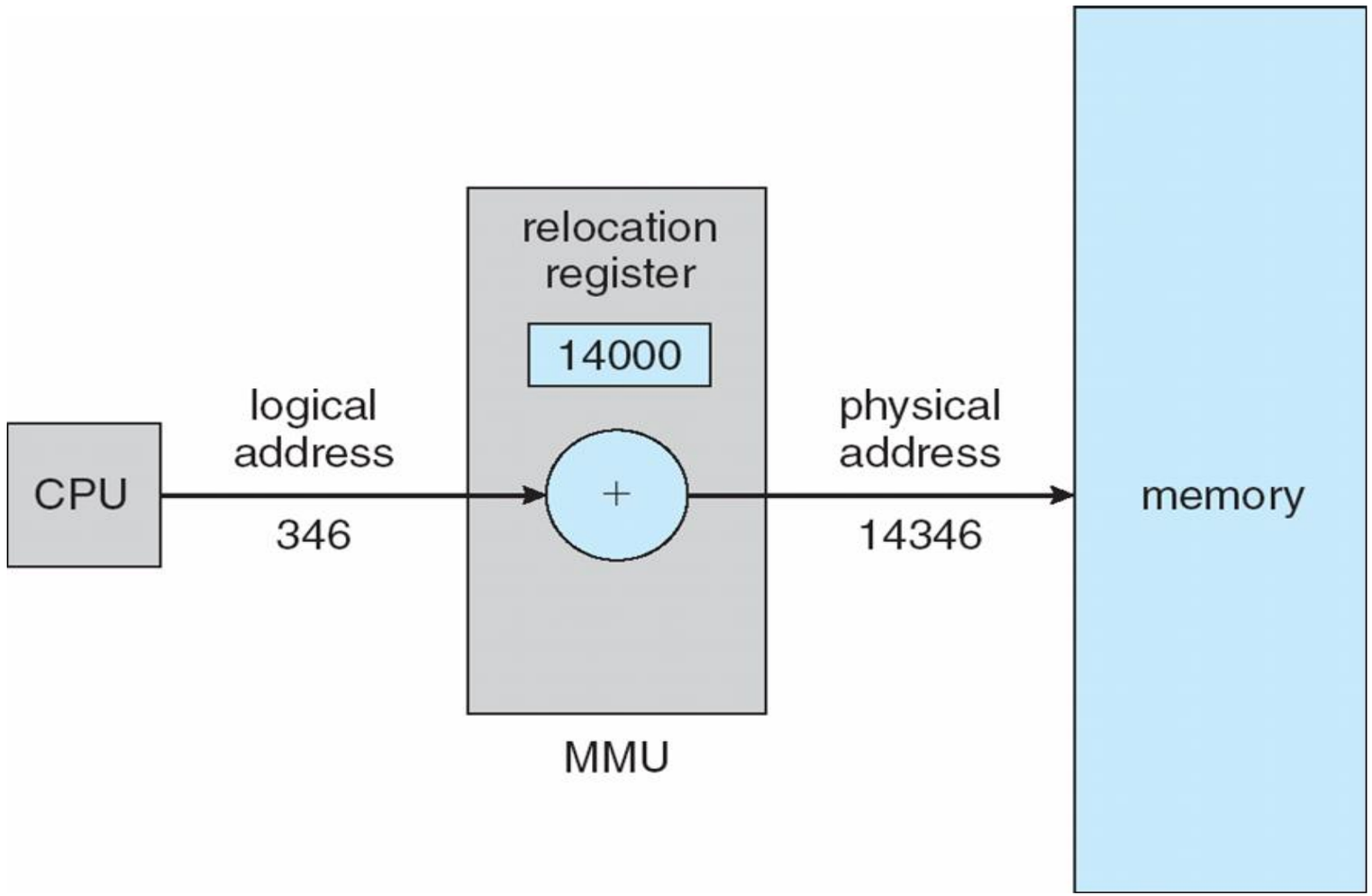
Mantıksal vs. Fiziksel Adres Uzayı

- Fiziksel adres uzayının mantıksal adres uzayı kavramından ayrılması, sağlam bir bellek yönetiminin merkezinde yer alır.
  - **Mantıksal adres (Logical address)** – CPU tarafından oluşturulur; ayrıca sanal adres (**virtual address**) olarak ta adlandırılır.
  - **Fiziksel adres (Physical address)** – adres, bellek birimi tarafından görülür.
- Derleme zamanı ve yükleme zamanı adresiyle bağlama yöntemleri aynı mantıksal ve fiziksel adresleri üretir. Bununla birlikte, yürütme zamanı adres-bağlama şeması farklı mantıksal ve fiziksel adreslere neden olur.
- **Mantıksal adres uzayı** bir program tarafından oluşturulan tüm mantıksal adreslerin kümesidir.
- **Fiziksel adres uzayı** bir program tarafından oluşturulan bu mantıksal adreslere karşılık gelen tüm fiziksel adres kümesidir.

Bellek Yönetim Birimi
(Memory-Management Unit - MMU)

- Çalışma anında sanal adresi fiziksel adrese dönüştüren donanım
- Bu bölümün ilerleyen kısımlarında göreceğiniz üzere pek çok yöntem mevcuttur.
- Başlangıç için, yeniden konumlandırma kaydedicisindeki bir değerin bir kullanıcı prosesi tarafından belleğe gönderildiği anda üretilen her bir adrese eklendiği basit bir düzeni ele alalım.
  - Taban(base) kaydedicisi **yeniden konumlandırma kaydedicisi** olarak adlandırılır
  - Intel 80x86 üzerinde MS-DOS, 4 adet yeniden konumlandırma kaydedicisi kullanmıştır.
- Kullanıcı programları *mantıksal* adreslerle çalışırlar; asla *gerçek* fiziksel adresleri göremezler.
  - Çalışma anında bağlama bellekteki bir konuma referans yapıldığında meydana gelir
  - Mantıksal adres, fiziksel adrese bağlıdır.

Konumlandırma Kaydedicisi Kullanılarak Dinamik Konumlandırma



Dinamik Yükleme

- Tüm programın yürütülmesi için çalışan kısmın bellekte olması gerekir
- Rutin çağrılana kadar yüklenmez.
- Bellek alanının daha iyi kullanımını sağlar. Kullanılmamış rutin asla yüklenmez.
- Tüm rutinler yeniden konumlandırılabilir yük biçiminde hazır durumda diskte tutulur.
- Nadir meydana gelen olayları yönetmek için büyük miktarda koda ihtiyaç duyulduğunda kullanışlıdır.

Dinamik Bağlama

- **Statik bağlama**– sistem kütüphaneleri ve program kodunun yükleyici (loader) tarafından ikilik (binary) program görüntüsü altında birleştirilmesidir.
- **Dinamik bağlama**– bağlama işleminin çalışma zamanına kadar ertelenmesidir.
- Küçük bir kod parçası olan *stub (kalıntı)*, uygun olan hafıza-yerleşim yerini tespit etmek için kullanılır.
- Stub rutinin adresi ile kendisinin yerini değiştirerek rutini yürütür.
- İşletim sistemi rutinin bellek adresinde bulunup bulunmadığını kontrol eder.
  - Adres alanında değilse, adres alanına ekler.
- Dinamik linking özellikle kütüphaneler için kullanışlıdır.
- Sistem aynı zamanda **shared libraries (paylaşımlı kütüphaneler)** olarak da bilir.
- Sistem kütüphanelerini güncellemeleri için uygulanabilirliğini düşünün.
  - Sürümleme (versiyonlar oluşturma) gerekebilir.

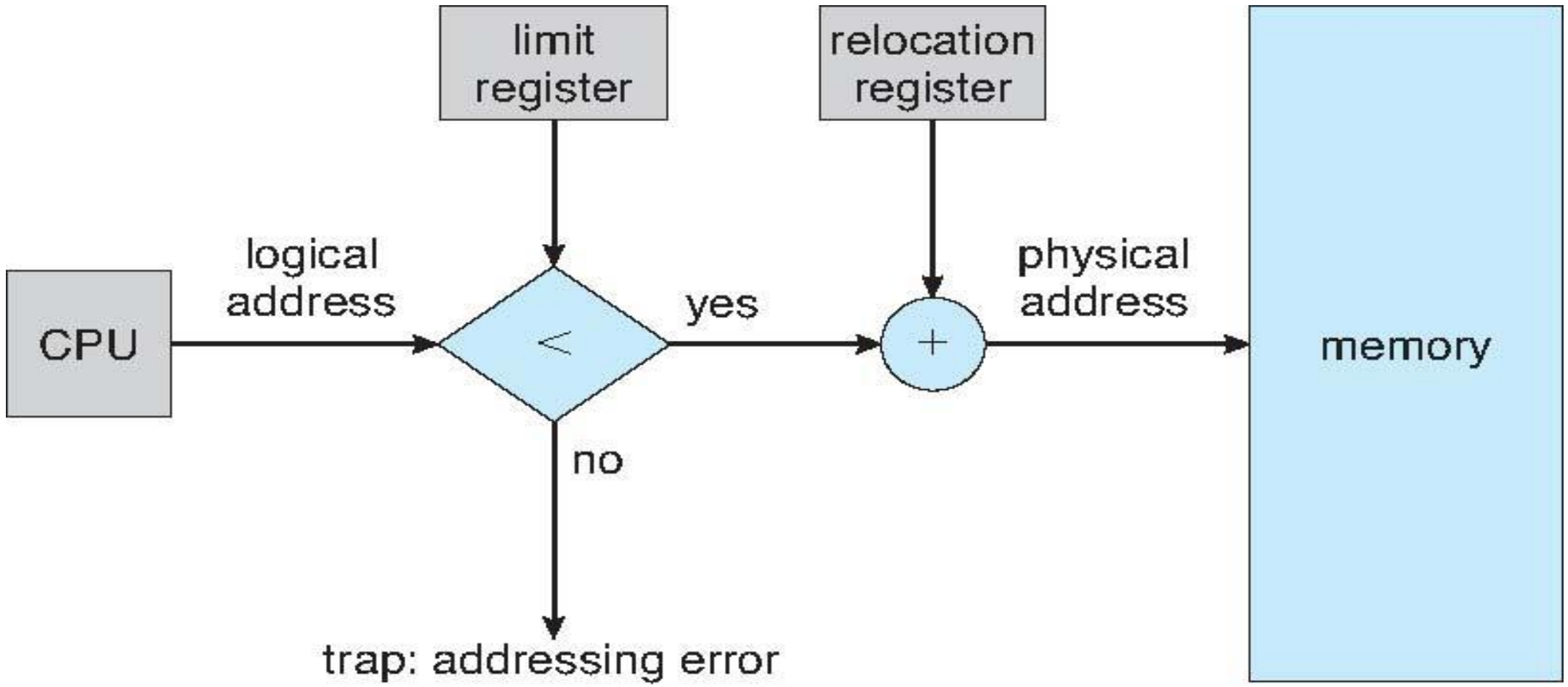
Bitişik Tahsis (Contiguous Allocation)

- Ana bellek, hem OS hem de kullanıcı süreçlerini desteklemelidir
- Sınırlı kaynak verimli bir şekilde tahsis edilmelidir
- Bitişik tahsis, eski bir yöntemdir
- Ana bellek genellikle iki bölümden oluşur:
  - Yerleşik işletim sistemi genellikle kesme vektörü ile düşük bellekte (low memory) tutulur.
  - Kullanıcı işlemleri ise yüksek hafızada (high memory) tutulur
  - Her process belleğin bitişik tek bir bölümünde yer alır.

Bitişik Tahsis (Contiguous Allocation)

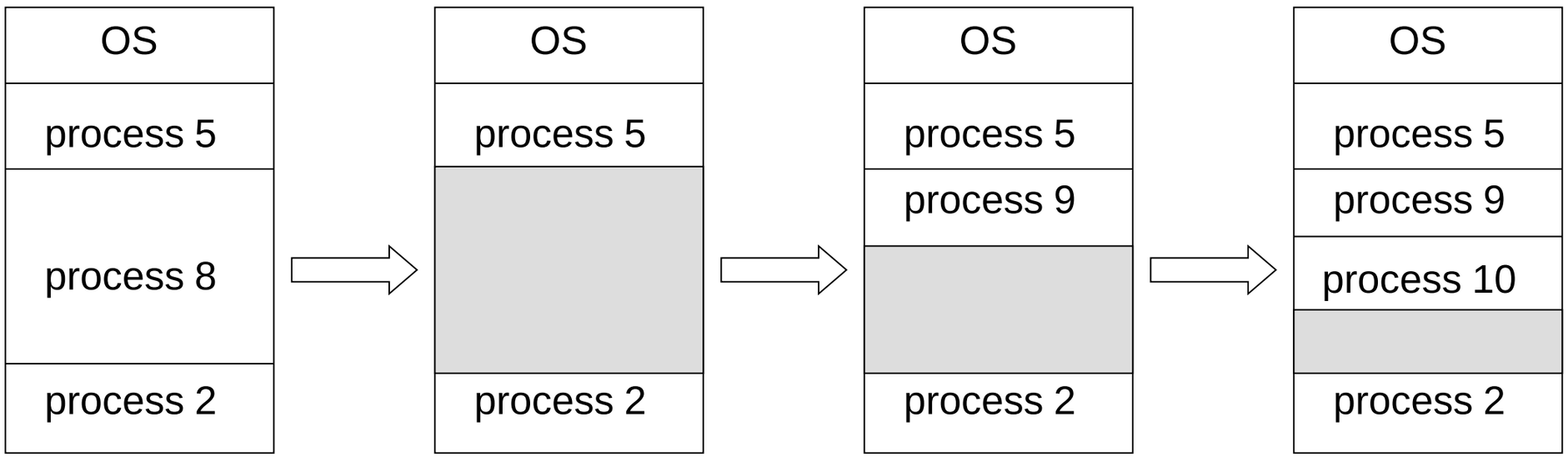
- Konumlandırma kaydedicisi (relocation register) kullanıcı process'lerini bir diğerinden korumak için kullanılır.
  - Taban kaydedicisi (Base register) en küçük fiziksel adres değerini içerir.
  - Sınır Kaydedicisi (Limit register) mantıksal adres aralığını içerir - her mantıksal adres Sınır Kaydedicisi ’den daha kısa olmalıdır.
  - MMU mantıksal adresi *dinamik olarak* haritalar.

Konumlandırma Register'ları ve Limit Register'lar İçin Donanım Desteği



Bitişik Tahsis (Devam)

- Çoklu-bölüm tahsisi
  - Verimlilik için değişken bölüm boyutları (belirli bir prosesin ihtiyaçlarına göre boyutlandırılır)
  - Hole (Delik) – kullanılabilir bellek bloğu;çeşitli büyüklükteki holes(boşluklar) bellek boyunca dağılmıştır.
  - Bir process geldiğinde, onun sığabileceği büyüklükte bir hole(delik) bellek tahsis edilir.
  - Process serbest bölüme geçerken komşu serbest bölümle birleştirilir.
  - İşletim sistemi şu bilgileri tutar:
    a) ayrılan bölümleri b) serbest bölümleri (hole)



Dinamik Depolama-Tahsis Problemi

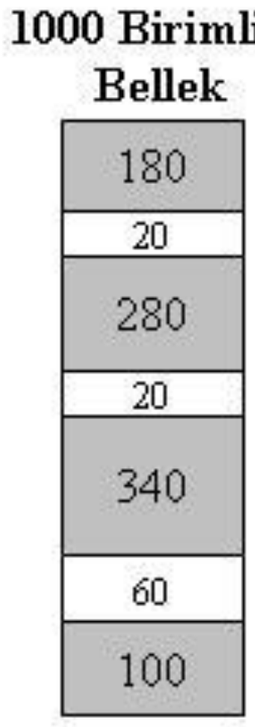
Serbest alanlara gelen N boyutlu bir istek nasıl yerine getirilir ?

- **First-fit (İlk durum)**: İlk bulduğu yeterli alana yerleştirir.
- **Best-fit (En uygun durum)**: Tüm liste aranır boyutlarına bakarak en az boşluk bırakacak şekilde yerleştirilir.
  - En az artık alan üretir.
- **Worst-fit (En kötü durum)**: En büyük alana yerleştirir. Aynı zamanda tüm liste aranır.
  - En fazla artık alan üretir.

Hız ve depolama alanı açısından first-fit ve best-fit, worst-fit'ten daha iyidir.

Fragmentation (Parçalanma)

- Veriler pek çok sistemde belleğe ardışıl bir biçimde yerleştirilmez. Çünkü ardışıl yerleştirme işlemi bir süre sonra parçalanma (*fragmentation*) denilen soruna yol açmaktadır.
- **External Fragmentation (Dış Parçalanma)**– Toplam bellek alanı çalıştırılacak programa yettiği halde boşluklar farklı bölgelerde olduğundan yerleştirilemez.



- **Internal Fragmentation (İç Parçalanma)**– Ayrılan bellek alanı istenen bellek alanından biraz daha büyük olabilir; bu boyut farkı bellekte bir bölüm olarak mevcuttur ancak kullanılmamıştır.

- «İlk uyan» incelendiğinde *N* blokluk alan tahsis edilmiş, 0.5 *N* blokluk alan fragmentation nedeniyle kaybedilmiştir.
  - 1/3'ü kullanılamaz olabilir -> **yüzde 50 kuralı**

Fragmentation (Devam)

- **Compaction (sıkıştırma)** ile dış parçalanmayı azaltın.
  - Boş belleği tek bir büyük blokta bir araya getirmek için bellek içeriğini karıştır
  - Sıkıştırma işlemi mümkündür ancak, sadece takas (relocation) işlemi dinamik ise yapılır ve yürütme zamanında gerçekleştirilir.
  - I/O (Giriş / Çıkış Sorunu)
    - I/O işlem isteği olduğunda görevi belleğe sabitle

- Şimdi yedekleme deposunun(örn: hard disk) aynı parçalanma sorunlarına sahip olduğunu düşünün

- Sıkıştırma işlemi yükü fazla, bütün proseslerin yeri değişiyor ve zaman gerektirir. Bu yüzden çözüm için; sayfalama, segmentasyon

Sayfalama

- Bir process'in fiziksel bellek alanı bitişik olmayabilir, sonraki bellek alanı kullanılabilir olduğu sürece fiziksel bellek alanı tahsis edilir.

- Fiziksel belleğin sabit boyutlu bloklar halinde bölünmüş haline **frames (çerçeveler)** denir.
  - Boyut 2'nin kuvvetleri şeklinde, 512 bytes ve 16 Mbytes arasında

- Mantıksal adres eş boyutlara bölünür ve bu bölümlere **pages (sayfalar)** deriz.

Sayfalama

- *N* sayfa boyutundaki bir programı çalıştırmak için, *N* tane serbest frame'e ve programın yüklenmesine ihtiyaç vardır
- Mantıksal adresi fiziksel adrese çevirmek için **page table** (sayfa tablosu) gerekli
- Yedekleme deposu aynı şekilde sayfalara bölünür.
- Hala iç parçalanma (Internal fragmentation) mevcuttur.

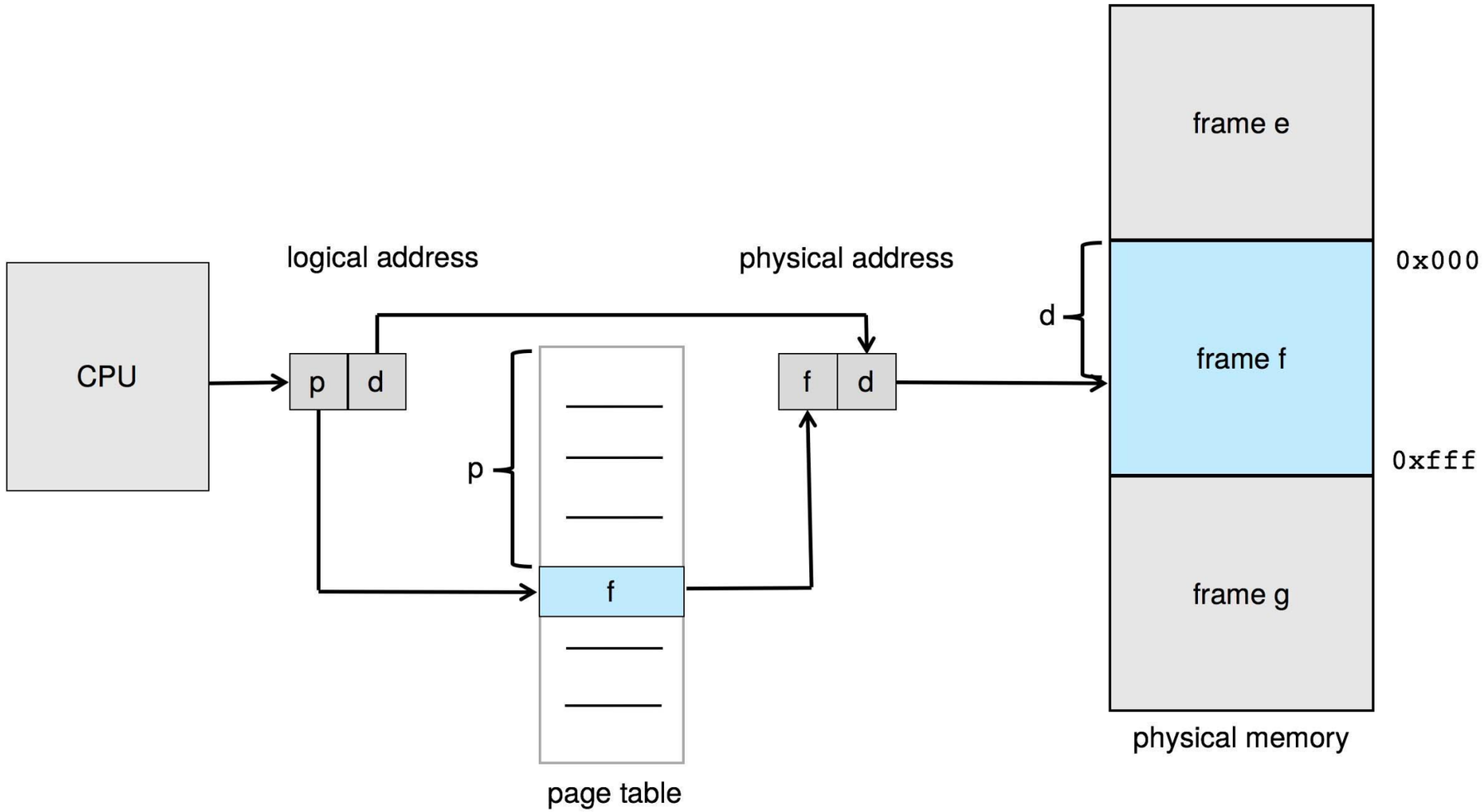
Adres Çeviri Şeması

- İşlemci tarafından üretilen adres aşağıdaki gibi bölünmüştür:
  - **Page number (Sayfa numarası -*p*)**– Fiziksel bellekteki her sayfanın base addressini içeren bir sayfa tablosunda index olarak kullanılır.
  - **Page offset (Sayfa ofset -*d*)** – Bellek birimine gönderilen fiziksel bellek adresini tanımlamak için taban adresi ile birleştirilir.

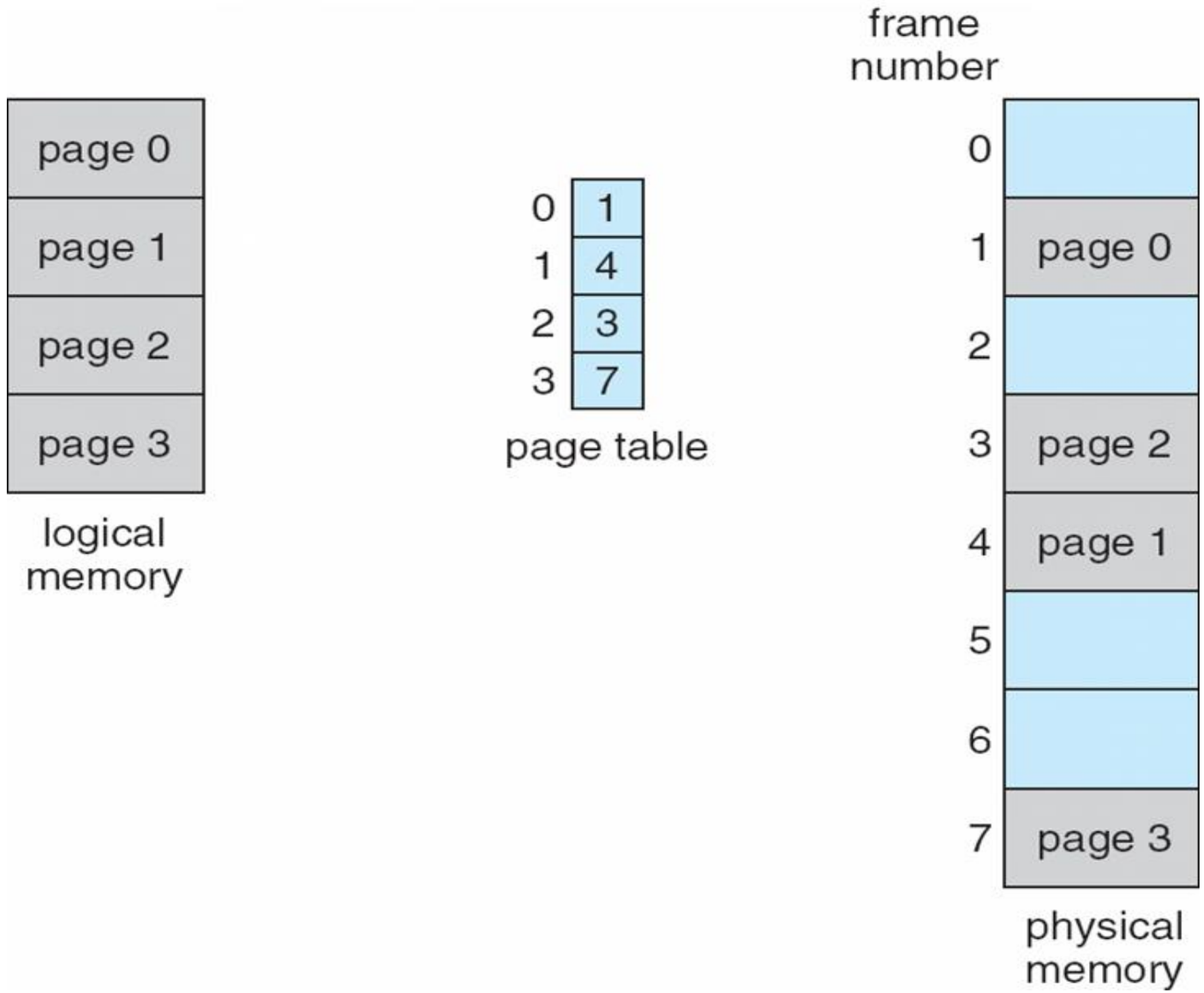
page number	page offset
p	*d*
m - n	*n*

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

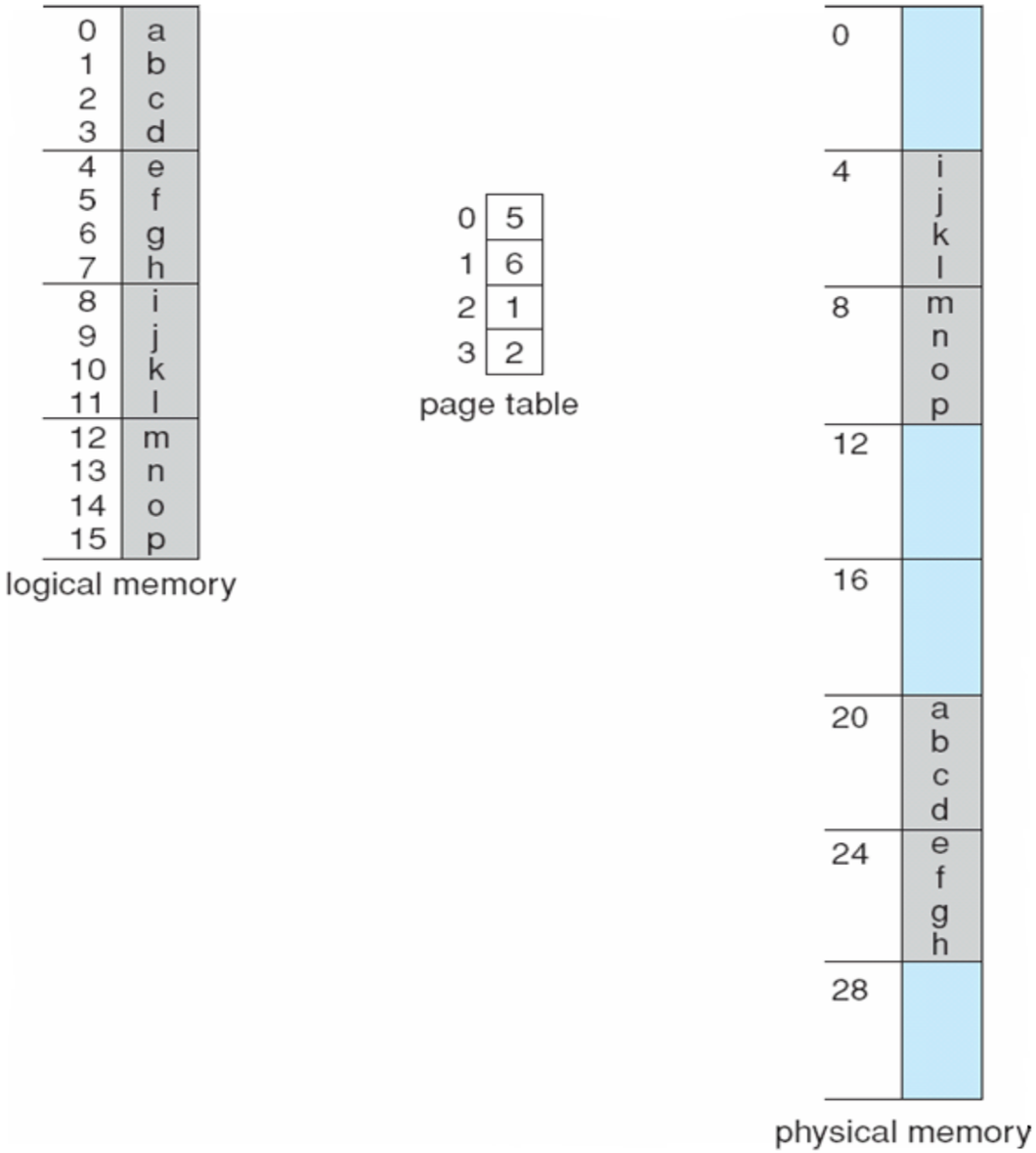
Donanım Sayfalama



Mantıksal ve Fiziksel Bellek Sayfalama Modeli



Sayfalama Örneği

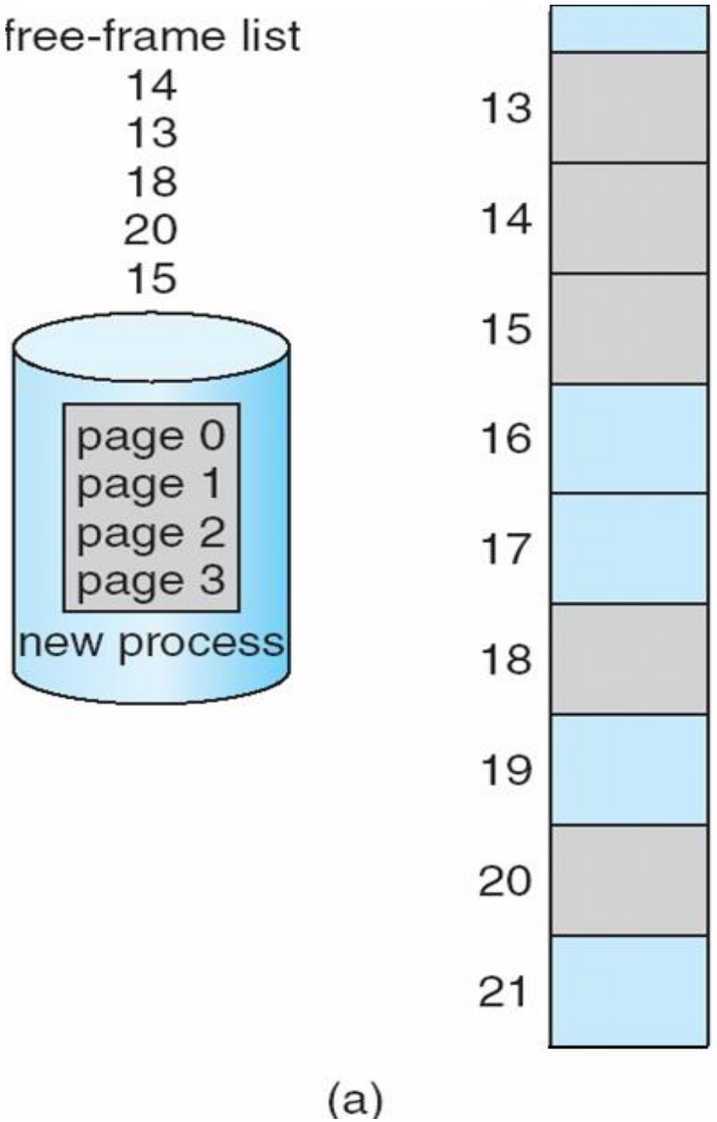


n=2 ve *m*=4 32-byte bellek ve 4-byte'lık sayfalar (8 sayfa)

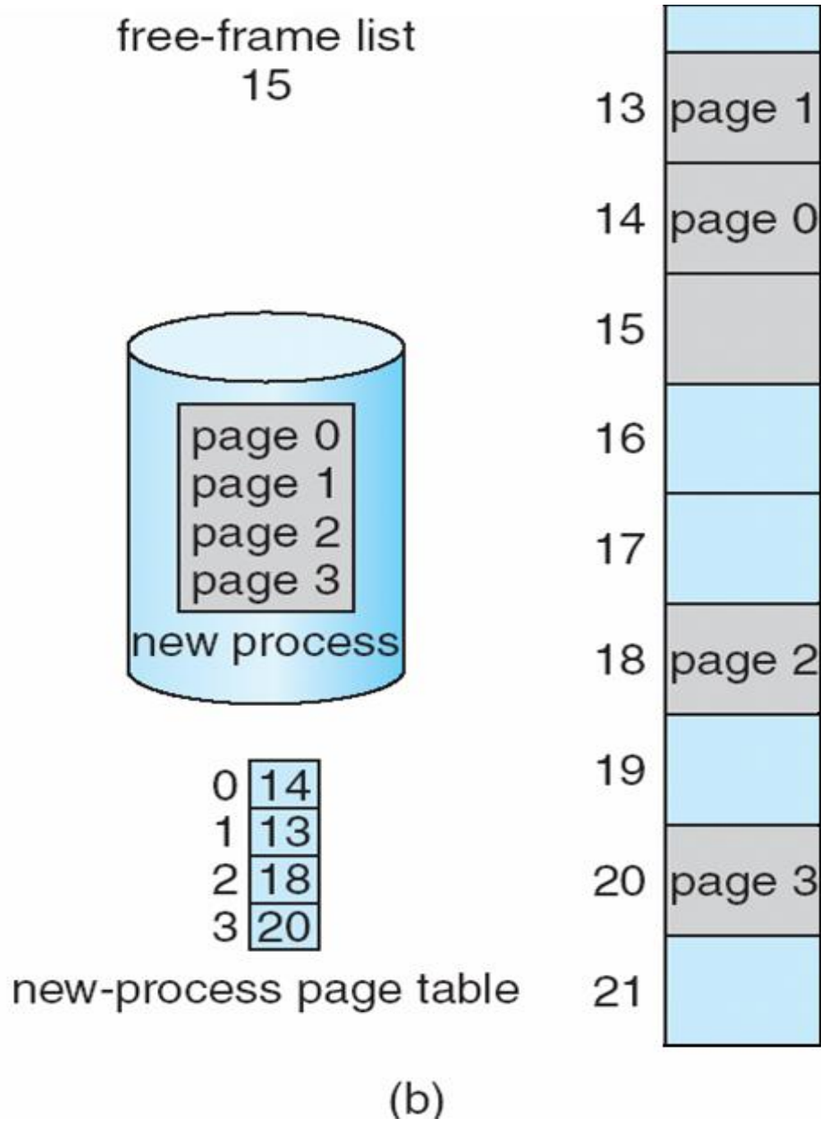
Sayfalama (Devam)

- İç parçalanma hesaplanıyor
  - Sayfa boyutu = 2,048 bytes
  - Process boyutu = 72,766 bytes
  - 35 sayfa + 1,086 byte
  - İç parçalanma 2,048 - 1,086 = 962 bytes
  - Çok küçük çerçeve boyutları arzulanır mı?
  - Ancak her sayfa tablosu girişi, izlemek için bellekte yer tutar
  - Solaris iki sayfa boyutu destekler. – 8 KB ve 4 MB
- İşlem görünümü ve fiziksel hafıza artık çok farklı
- Uygulama process'i sadece kendi belleğine erişebilir.

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**

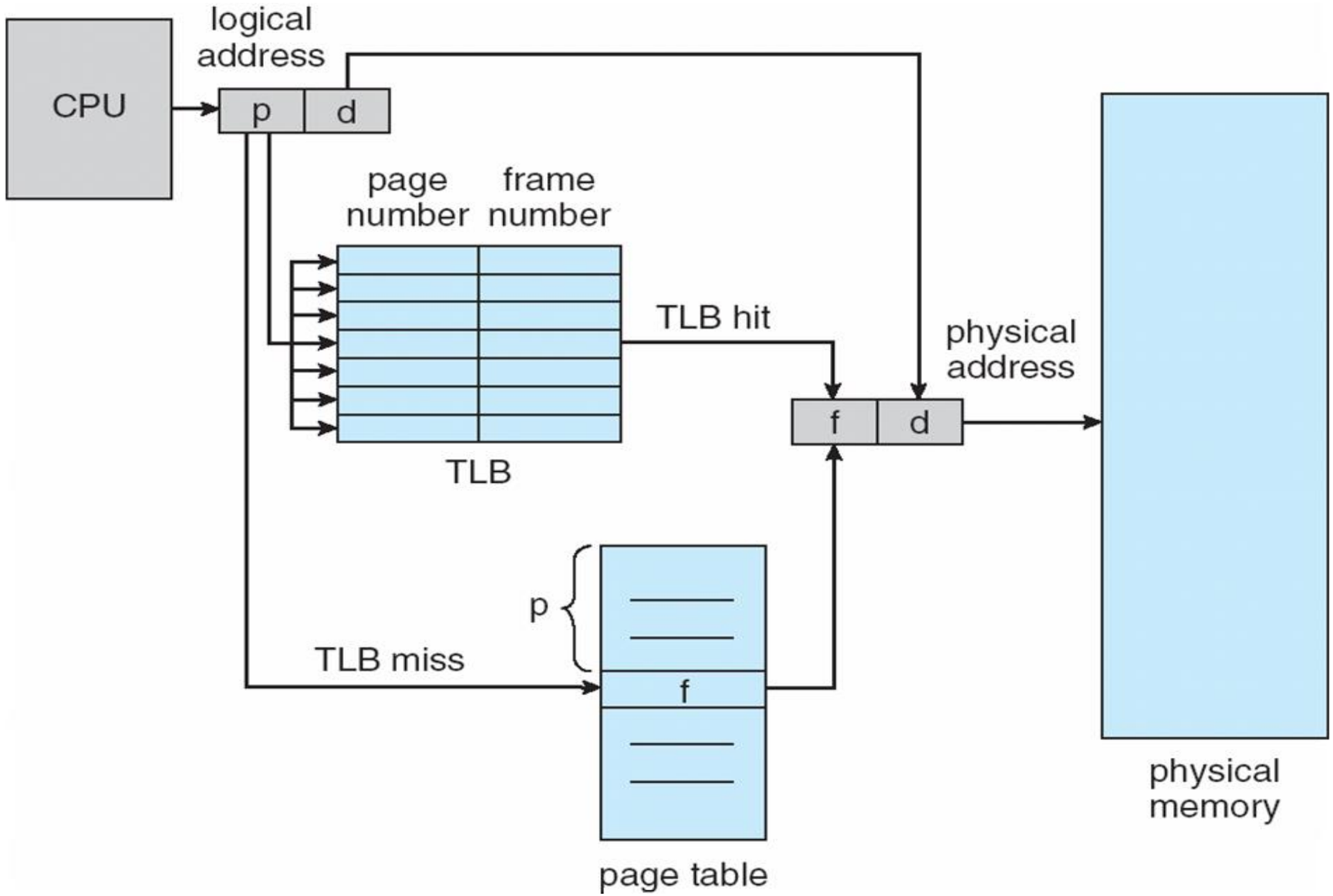
İlişkisel Bellek

- 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)**

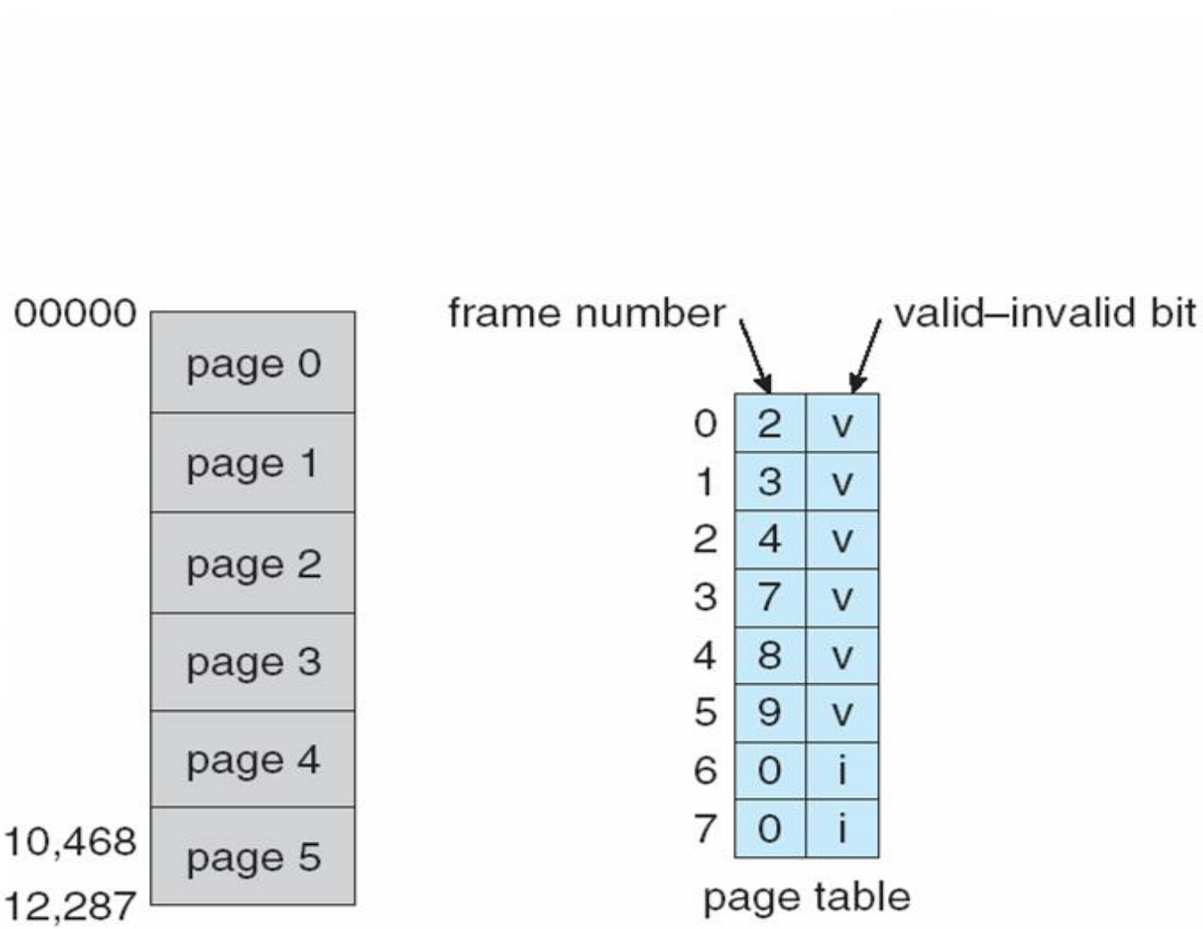
$$EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$
$$= 2 + \varepsilon - \alpha$$

- 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

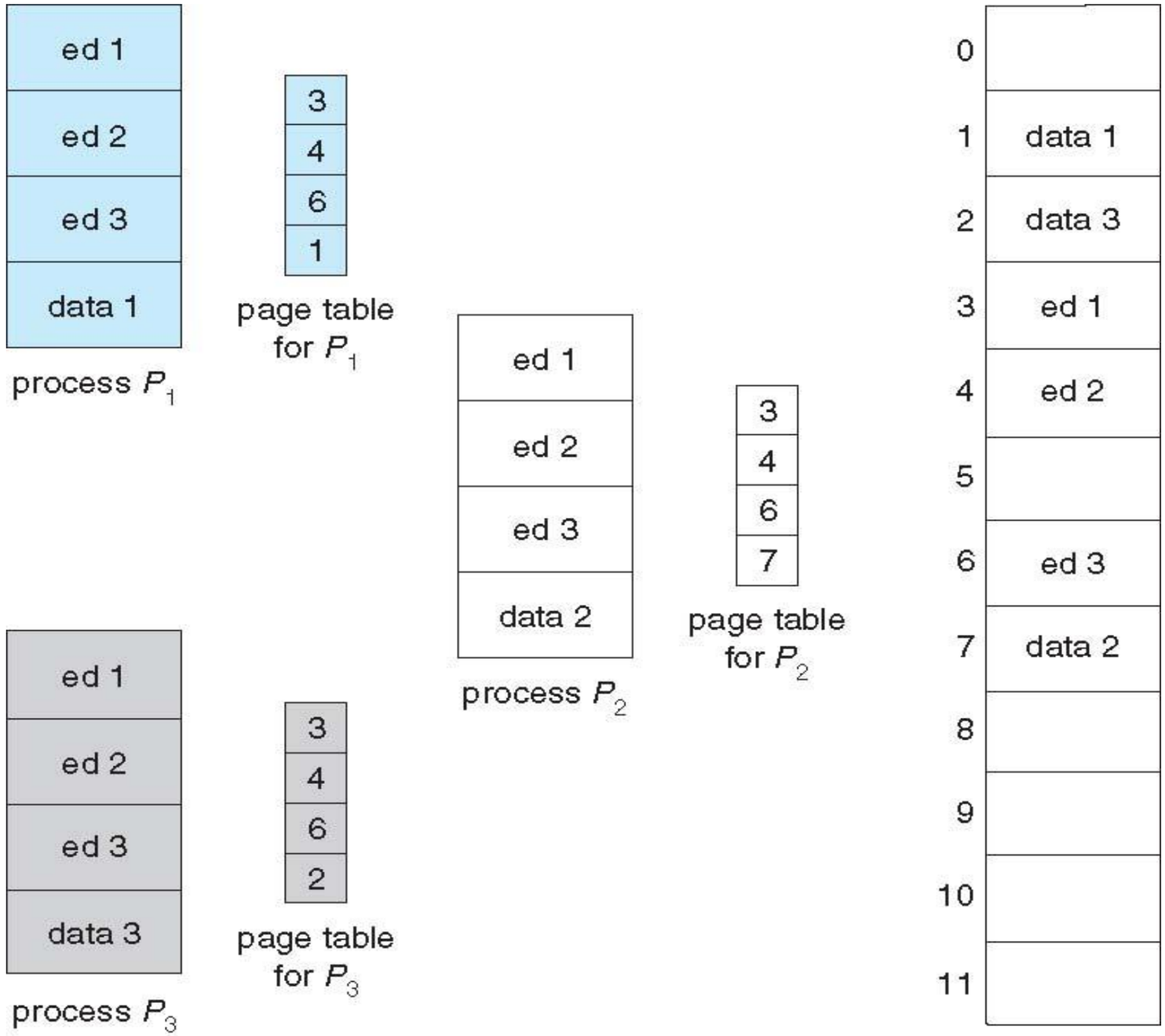


Paylaşımlı Sayfalar

- **Paylaşımlı kod**
  - Salt okunur kodun bir kopyası processler arasında paylaşılabilir. (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



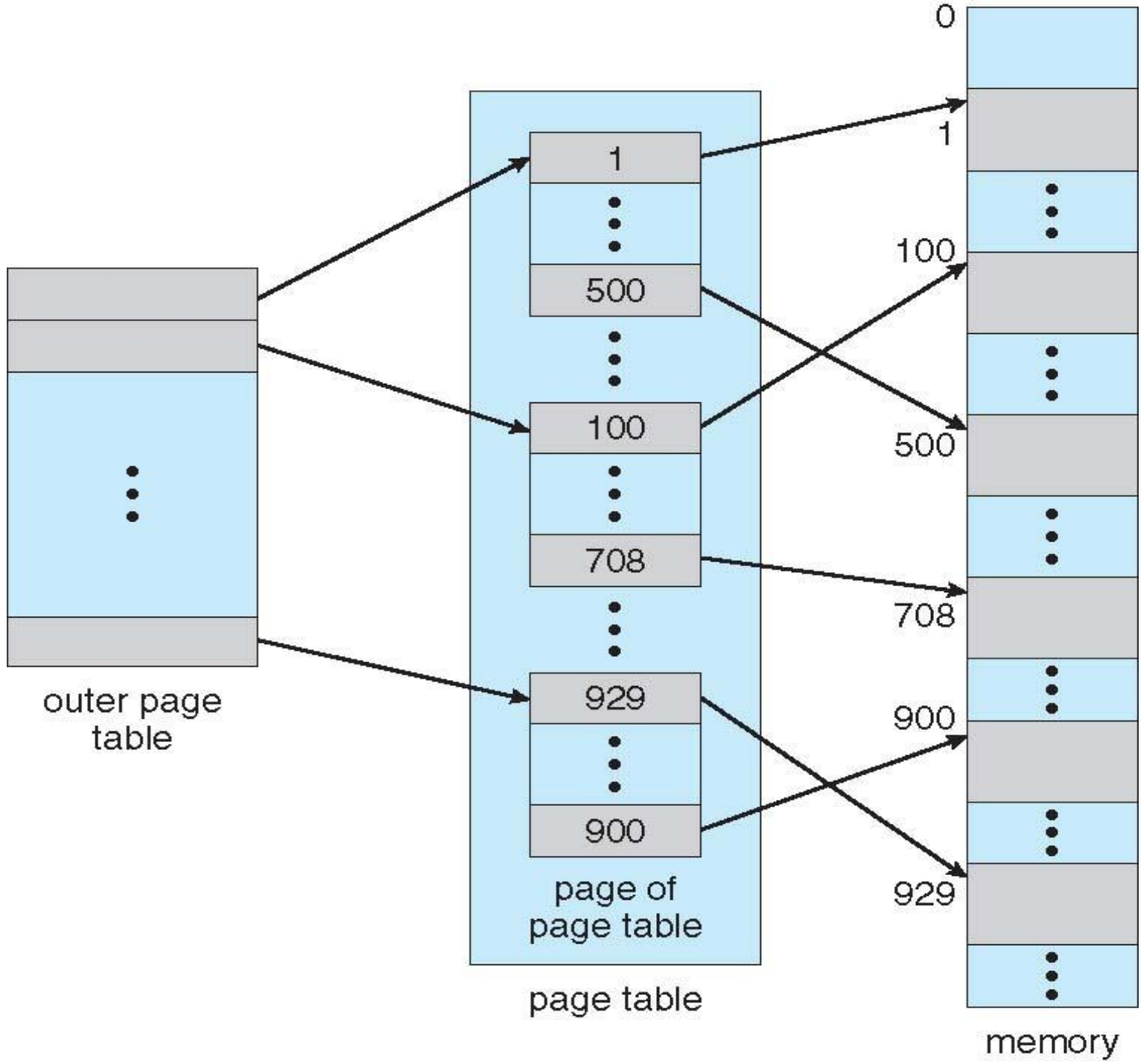
Sayfa Tablosunun Yapısı

- 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 bunu bitişik tahsis etmek istemeyiz.
  - Basit bir çözüm, sayfa tablosunu daha küçük birimlere bölmektir.
    - 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.

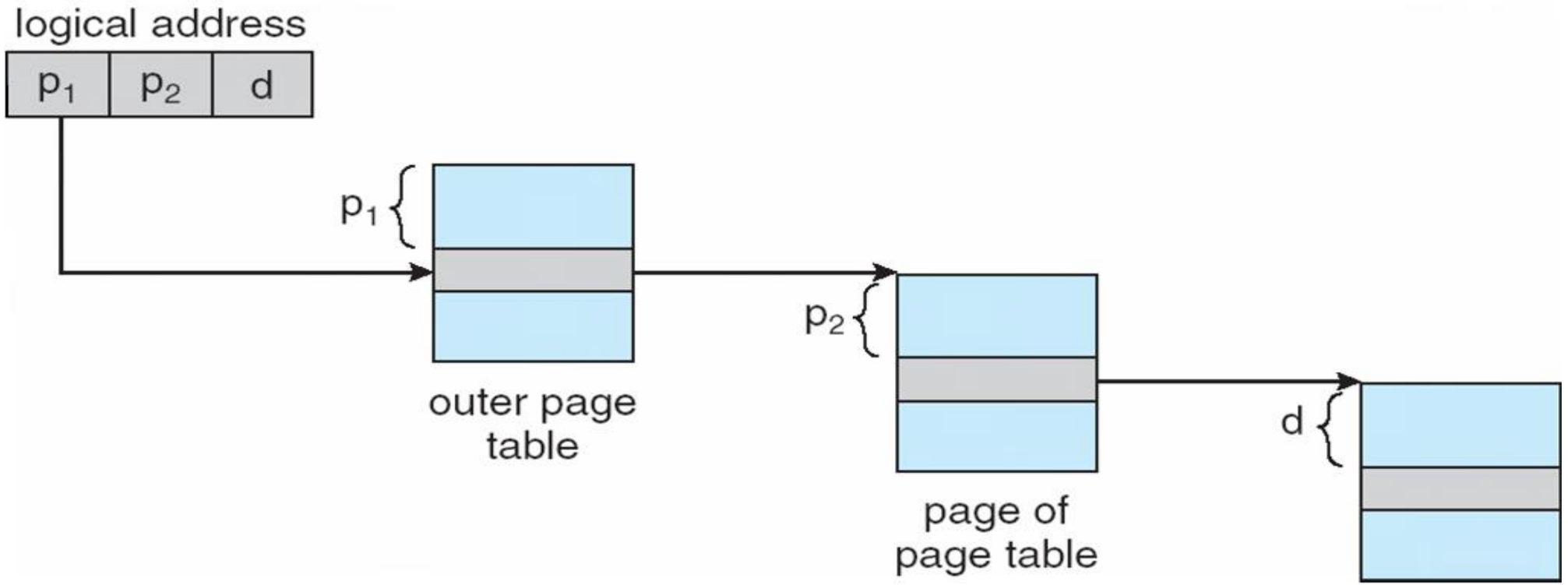
- Sayfa tablosu sayfalandırıldığından, sayfa numarası daha da bölünmüştür :
  - 12-bit'lik bir sayfa numarası
  - 10-bit'lik bir sayfa ofseti

- Böylece,bir mantıksal adres aşağıdaki gibi olur:

page number		page offset
p_1	p_2	d
12	10	10

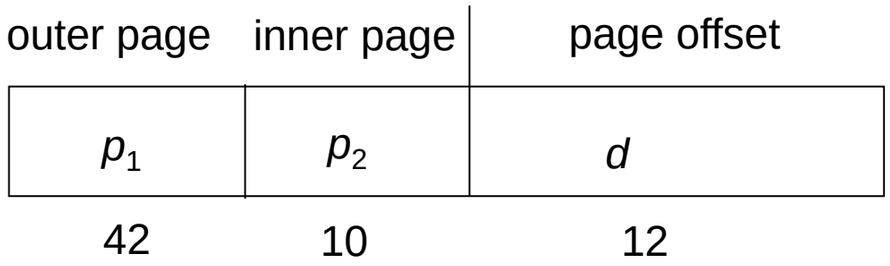
- burada p_1 dış sayfa tablosundaki bir dizindir ve p_2 iç sayfa tablosunun sayfasındaki yer değiştirmedir
- **forward-mapped page table** olarak bilinir.

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



- 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

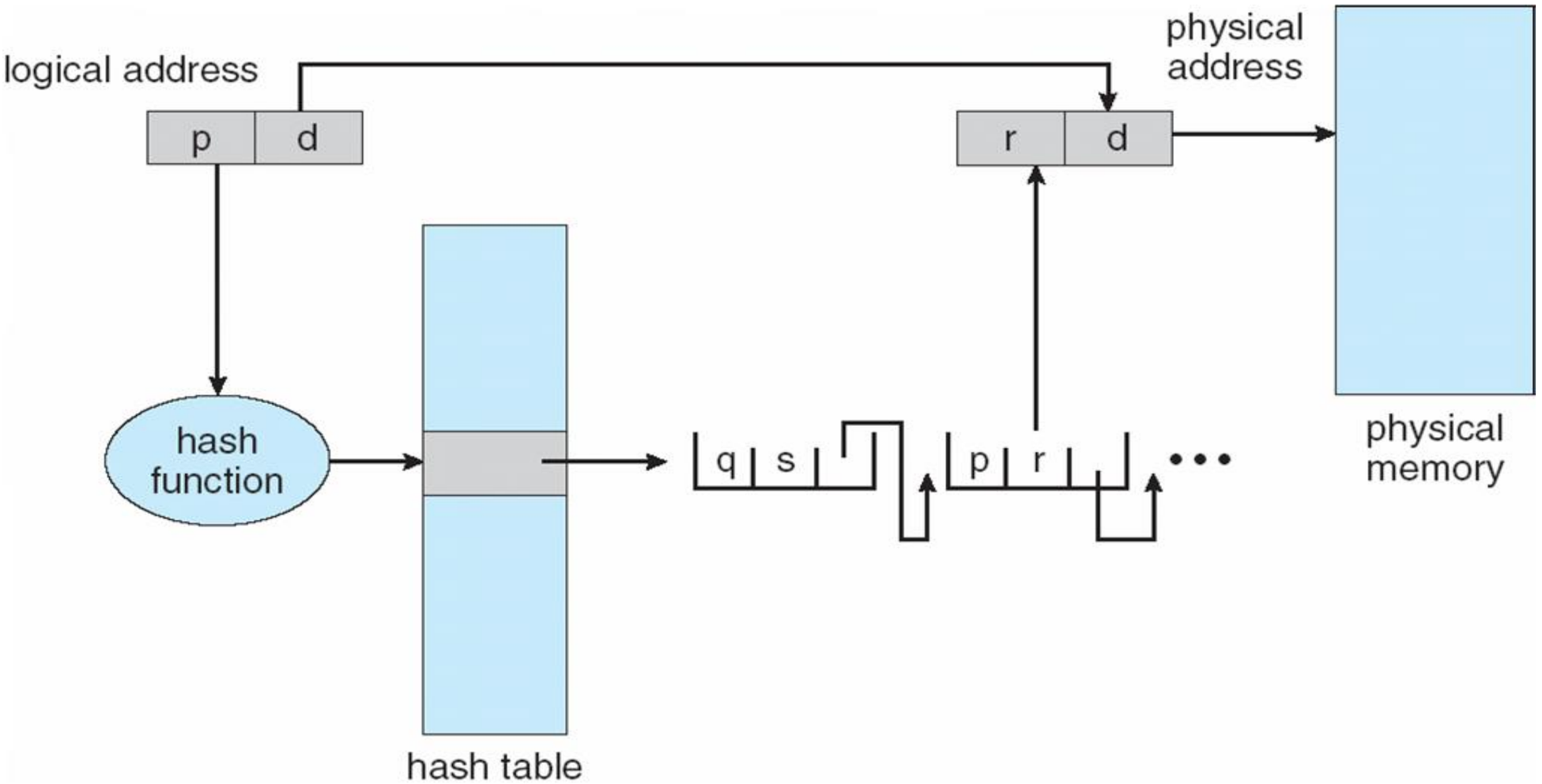
outer page	inner page	offset
p_1	p_2	d
42	10	12

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.

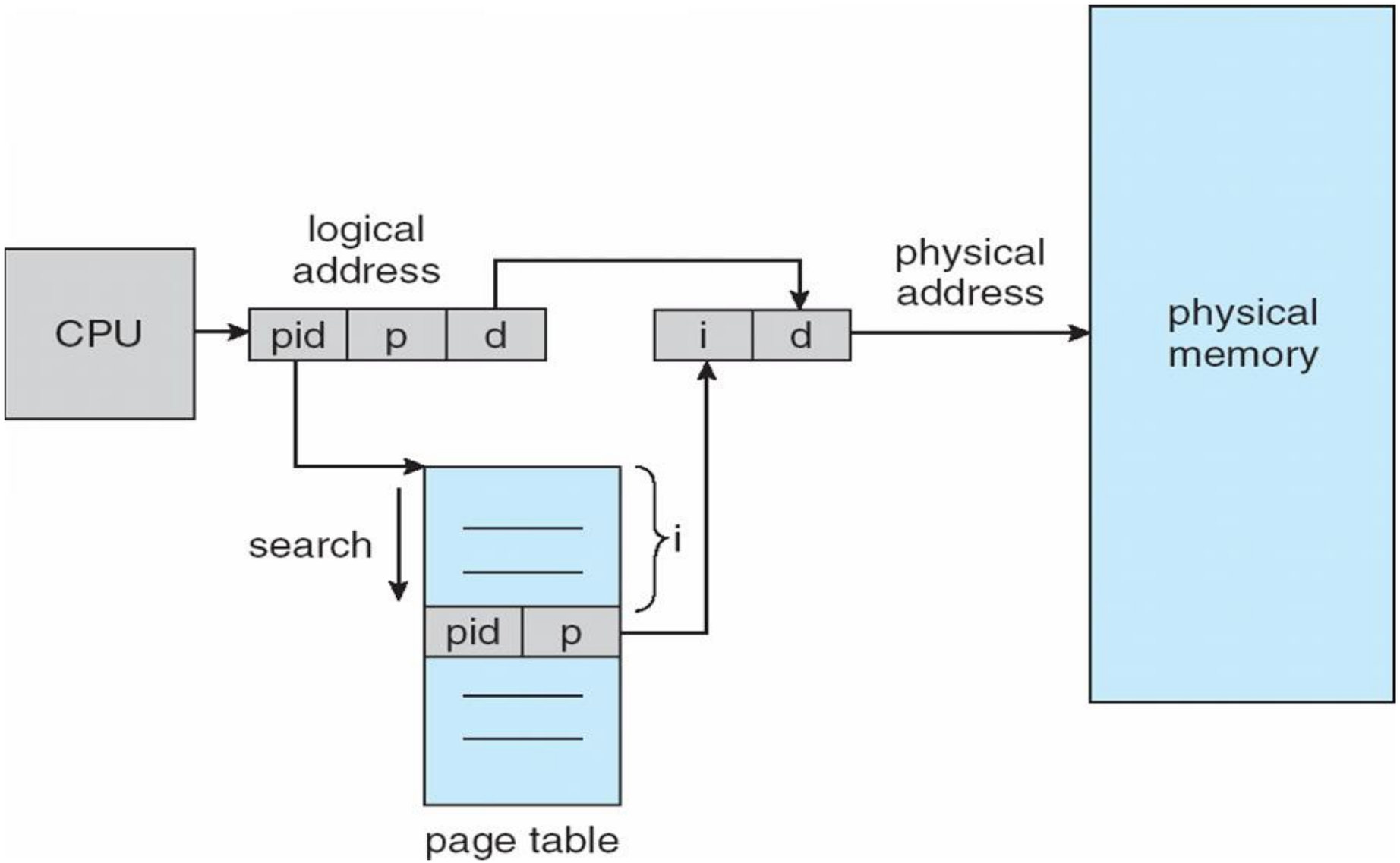
Hashed Sayfa Tablosu



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



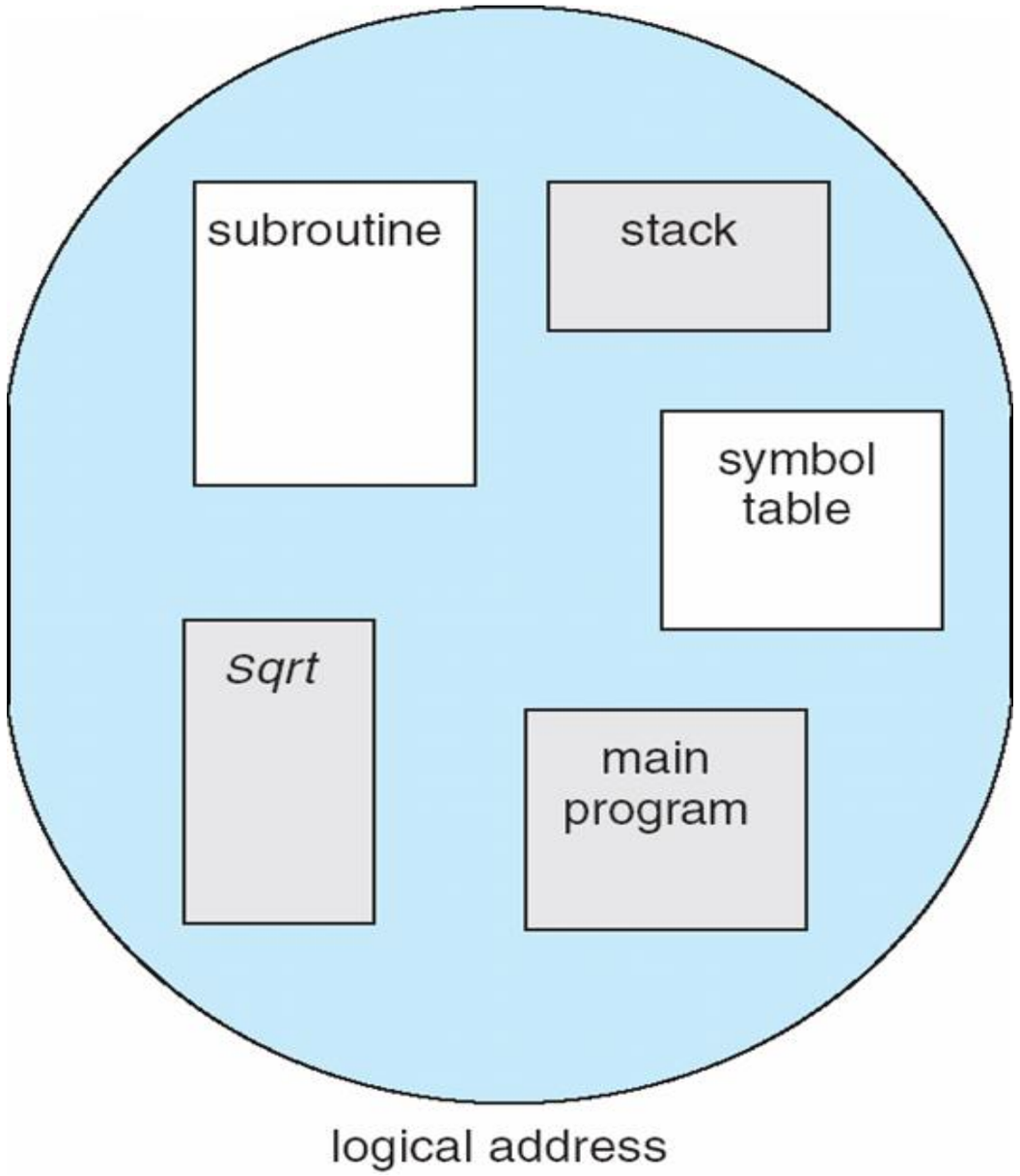
Segmentasyon

- Belleğin kullanıcı görünümünü destekleyen bellek yönetimi şeması.

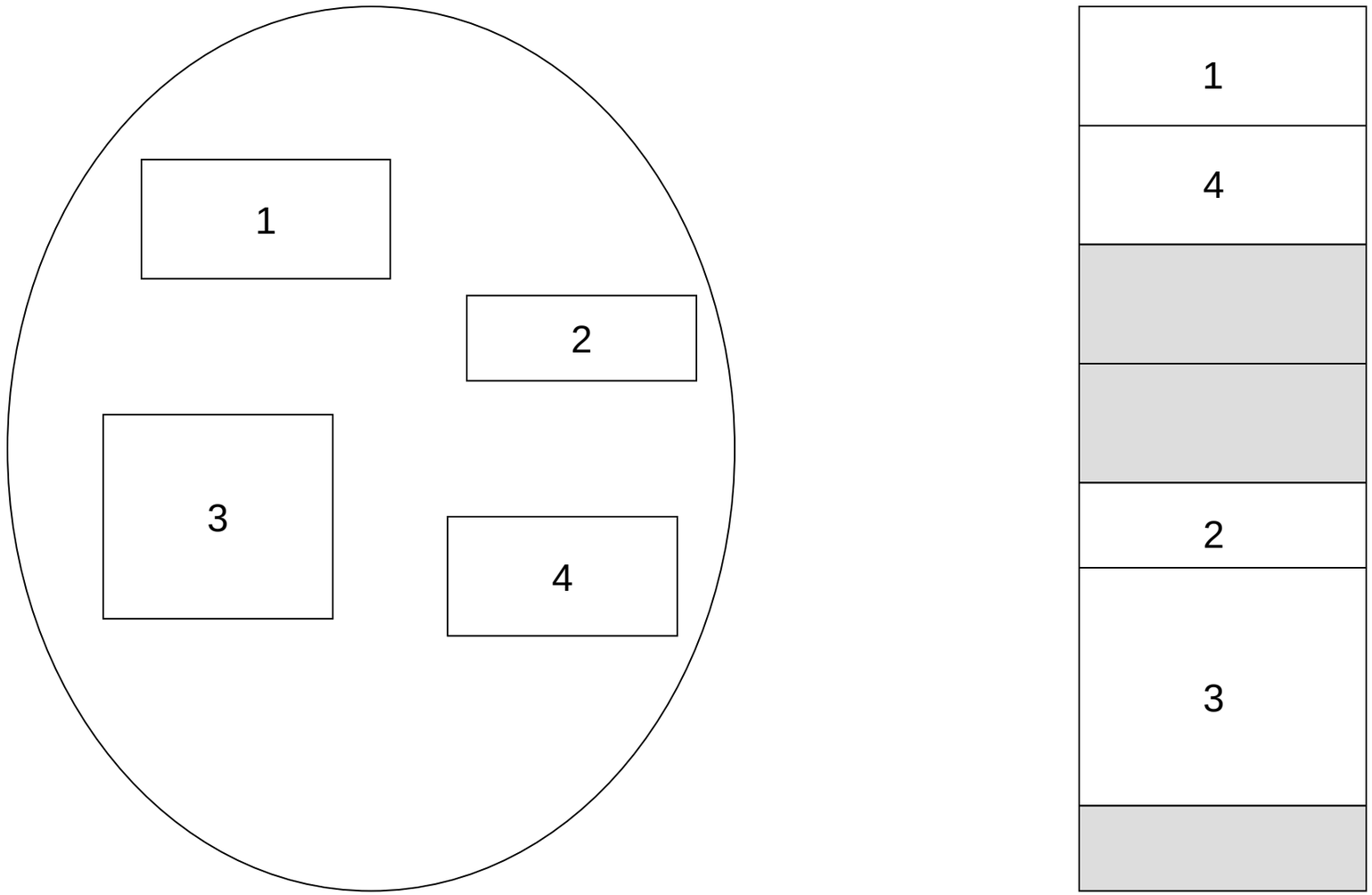
- Program bir segmentler topluluğudur.
  - Bir segment, şunlar gibi mantıksal bir birimdir:

    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)

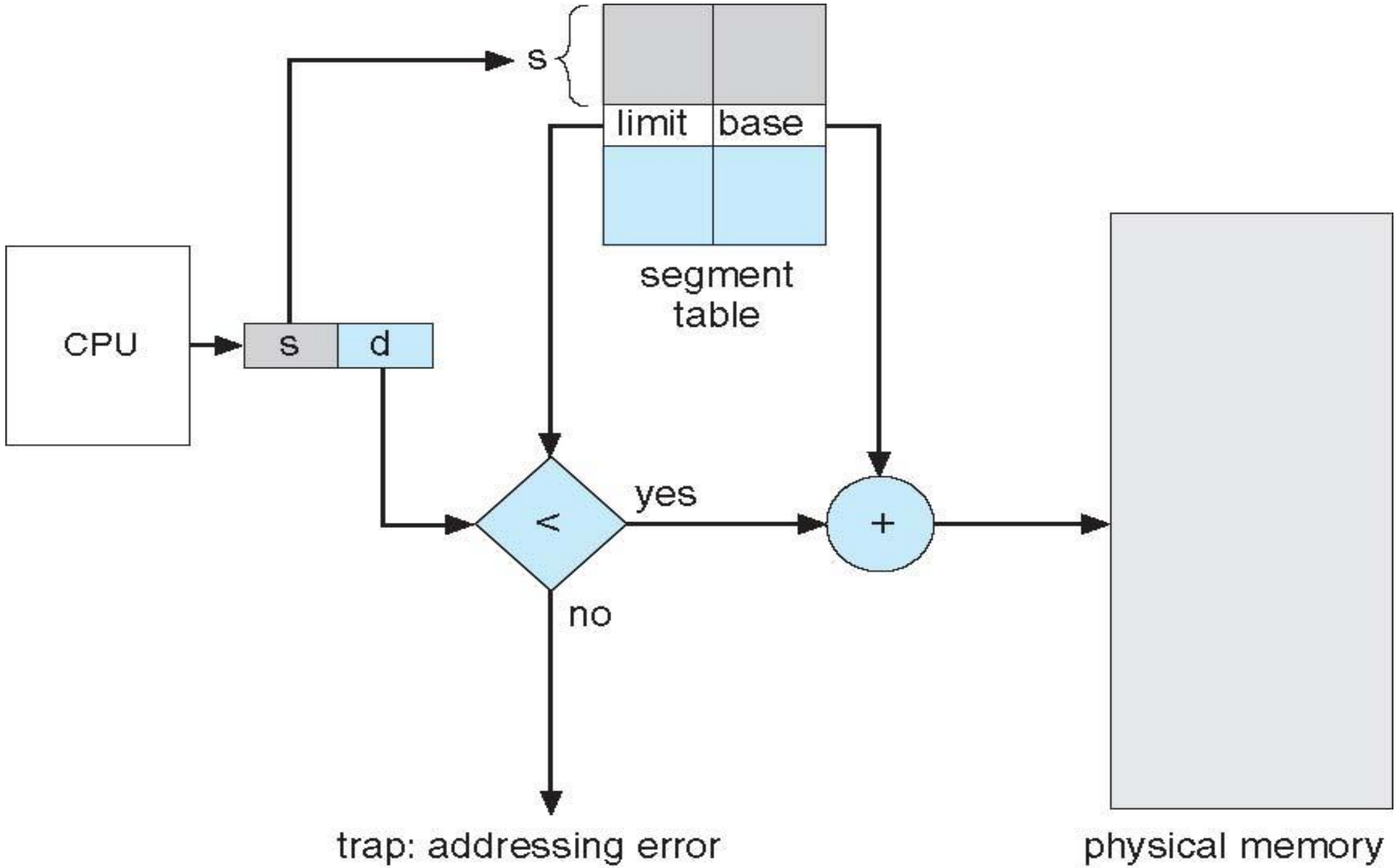
- Koruma
  - Segment tablosundaki her kayıt şunlar ile ilişkilendirilir:
    - Doğrulama bit = 0 ⇒ illegal segment
    - okuma/yazma/çalıştırma ayrıcalıkları

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

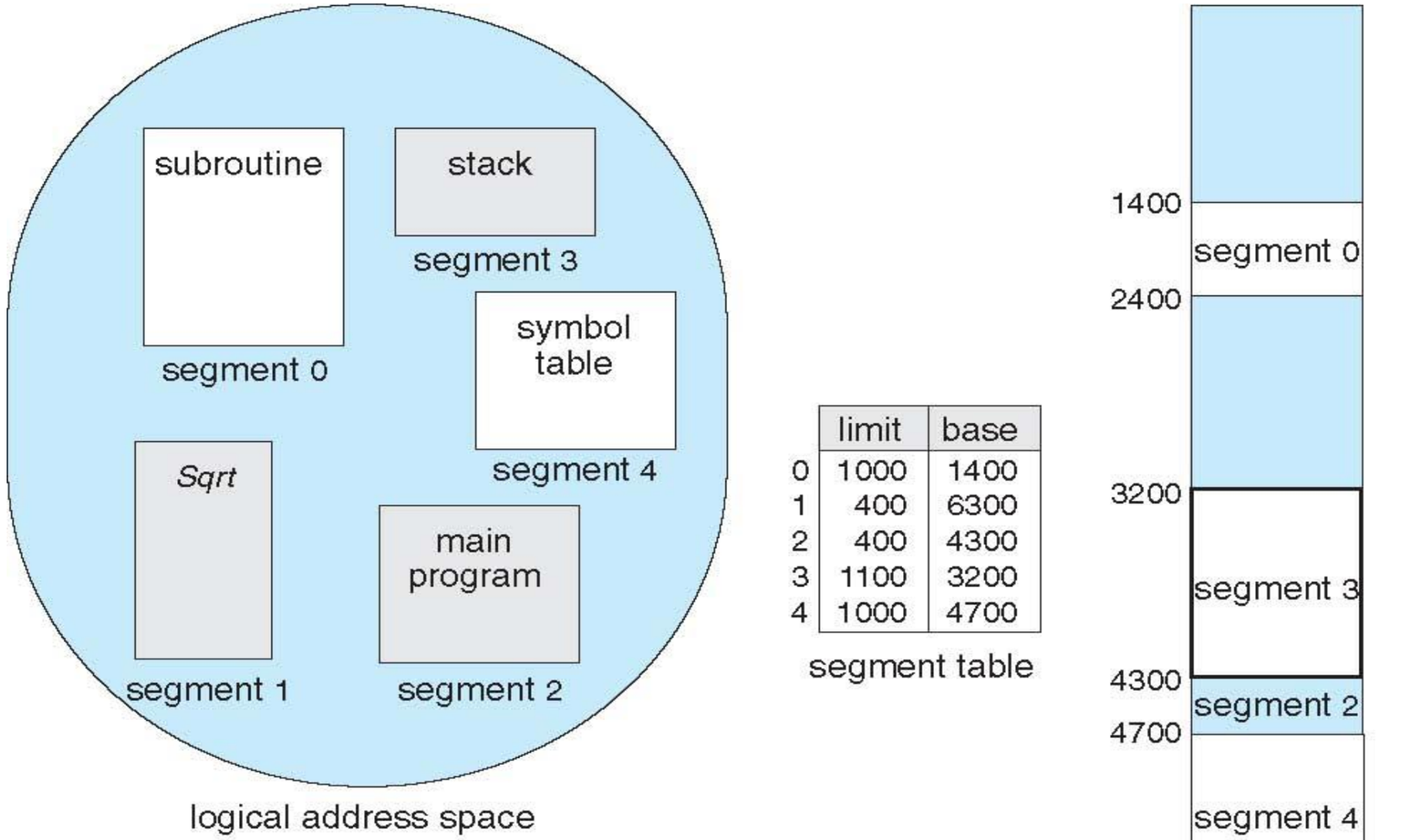
- Segment uzunlukları değişebildiğinden dolayı, bellek tahsisi bir dinamik depolama-tahsis problemidir.

- 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

Swapping (Takas)

- Bir process geçici olarak bellekten bir yedekleme deposuna alınabilir ve sonra yürütmeye devam etmek için belleğe geri gönderilebilir
  - Process'lerin toplam fiziksel bellek alanı fiziksel bellek miktarını aşabilir.

- **Backing store (yedekleme deposu)** – Tüm kullanıcılar için tüm bellek görüntülerini (image) kopyalarını barındıracak kadar büyük ve hızlı disk ; bu hafıza görüntülerini (image) doğrudan erişim sağlanmalıdır.

- **Roll out, roll in (Dışa taşıma, içe taşıma)** – Öncelik tabanlı planlama algoritmaları için kullanılır. Düşük öncelikli bir process dışa taşınır (swap out) çok daha yüksek öncelikli işlem takas edilip yürütülür.

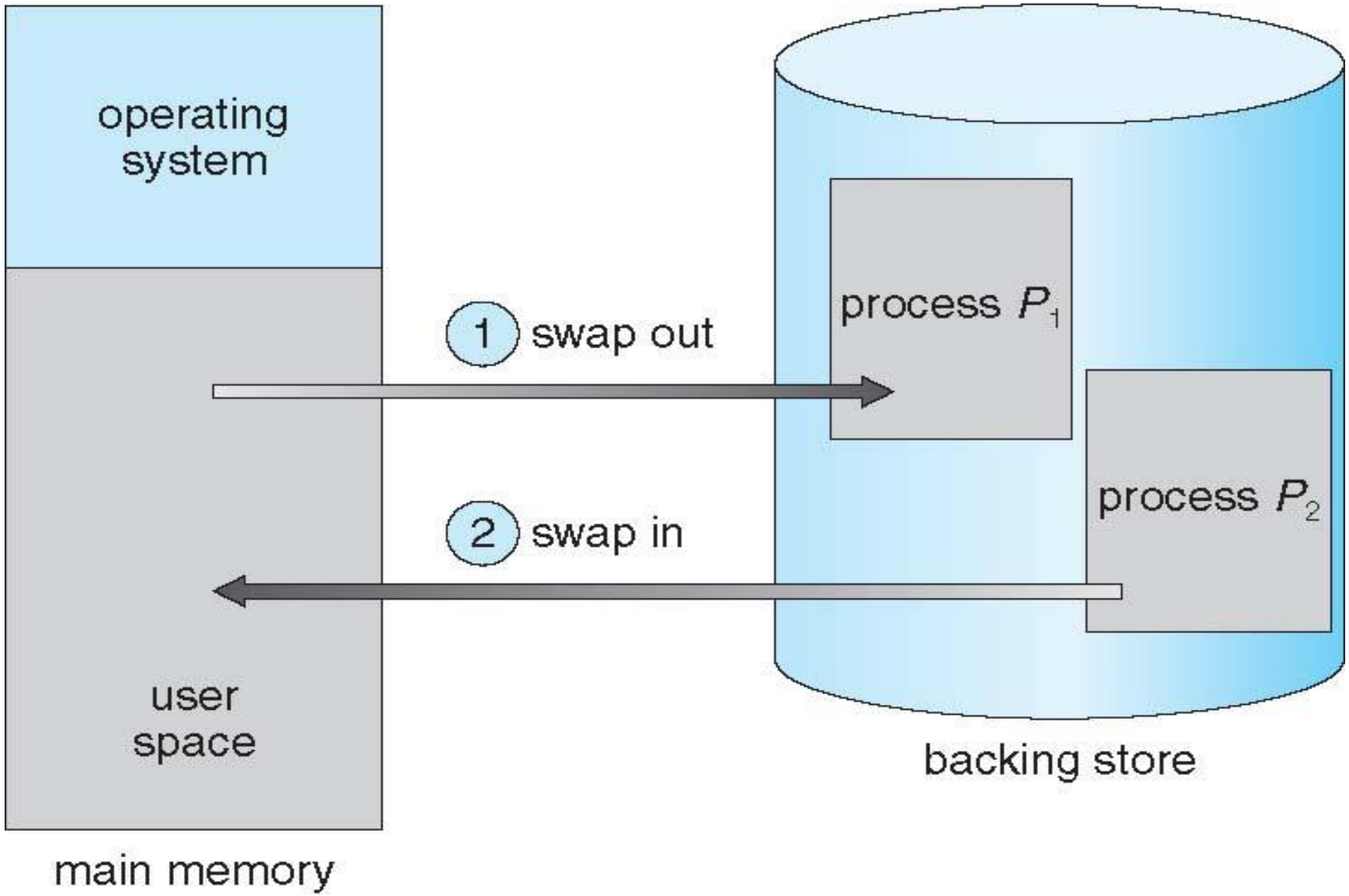
- Takas süresinin büyük bir kısmı transfer süresidir; toplam transfer süresi takas edilen bellek miktarı ile doğru orantılıdır.

- Sistem, disk üzerinde bellek görüntüleri olan çalışmaya hazır (ready-to-run) prosesleri hazır kuyruğunda tutar.

Swapping (Takas)

- Değiştirilen prosesin aynı fiziksel adreslere geri dönmesi gerekiyor mu?
- Adres bağlama yöntemine bağlı
- Swapping işleminin değiştirilmiş versiyonları pek çok sistemde bulunur.(ör., UNIX, Linux, ve Windows)
  - Takas işlemi normalde devre dışıdır.
  - Ayrılmış bellek alanı eşik değerinden fazla ile başlatılır.
  - Talep edilen bellek miktarı eşik değerinin altına inerse tekrar devre dışı bırakılır.

Swapping Şematik Görünümü

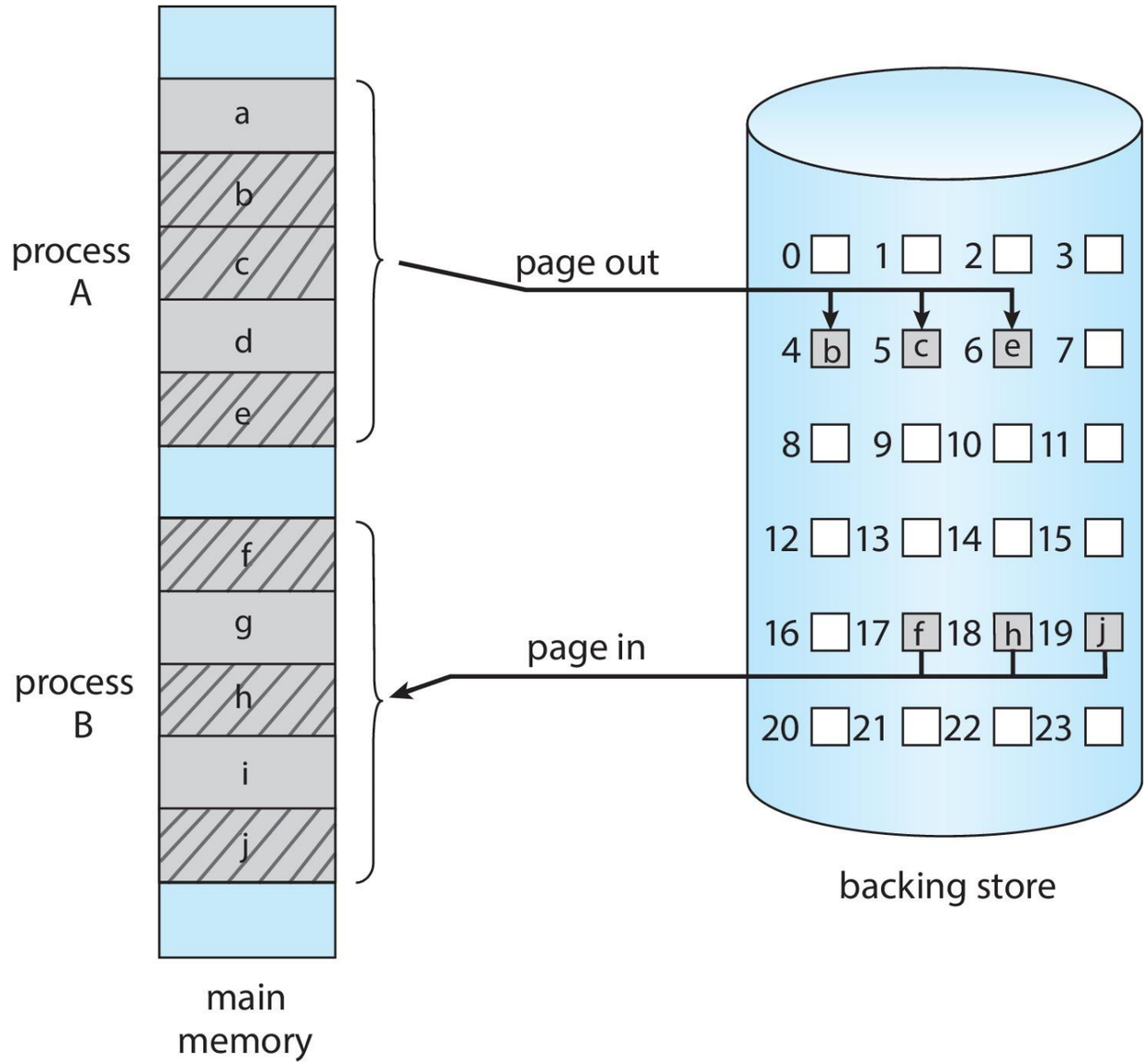


Context Switch Time including Swapping

- CPU'ya konacak sonraki işlemler bellekte değilse, bir prosesi dışa takas etmeye (dışa taşıma) ve hedef prosesinde içe takas etmeye ihtiyaç duyarız
- İçerik geçiş zamanı daha sonra çok yüksek olabilir
- 50MB / sn aktarım hızıyla sabit diske 100MB prosesin takası;
  - Artı 8 ms'lik disk gecikmesi ile
  - 2008 ms dışa takas (swap out) zamanı
  - + Aynı boyutlu prosesi de içe takas yapın
  - Toplam bağlam değiştirme bileşeni süresi 4016ms (> 4 saniye)

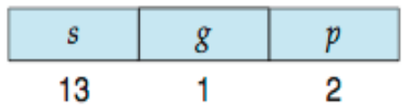
- Gerçekten ne kadar bellek kullanıldığını bilerek, bellek takas boyutu ve tabiki zamanı azaltabilir.
- Sistem çağrıları, (`request memory` ve `release memory`) OS'yi bellek kullanımı hakkında bilgilendirilmesi için kullanır

Swapping with Paging



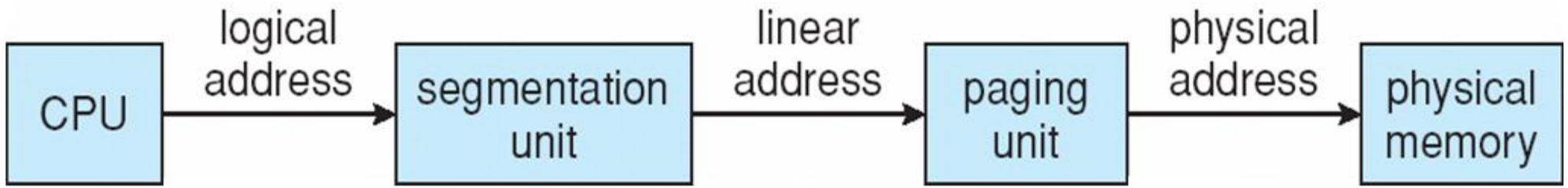
Örnek: The Intel Pentium (32 Bit)

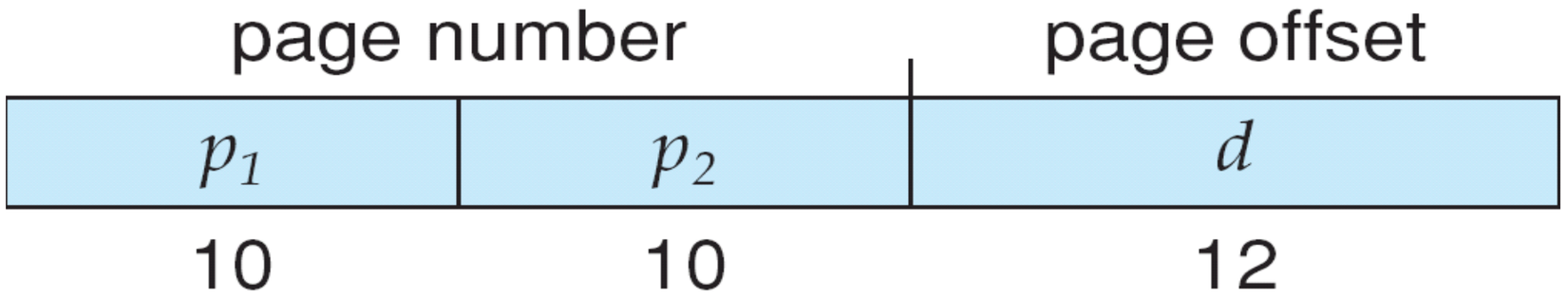
- Hem segmentasyonu hem de sayfalama ile segmentasyonu destekler.
  - Her segment 4 GB olabilir.
  - Process başına en fazla 16K segment olabilir
  - İki bölüme ayrılmıştır.
    - 8 K kadar olan ilk bölüm segmentleri prosese özeldir. (**local descriptor table LDT** ‘de tutulur.)
    - 8 K kadar olan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



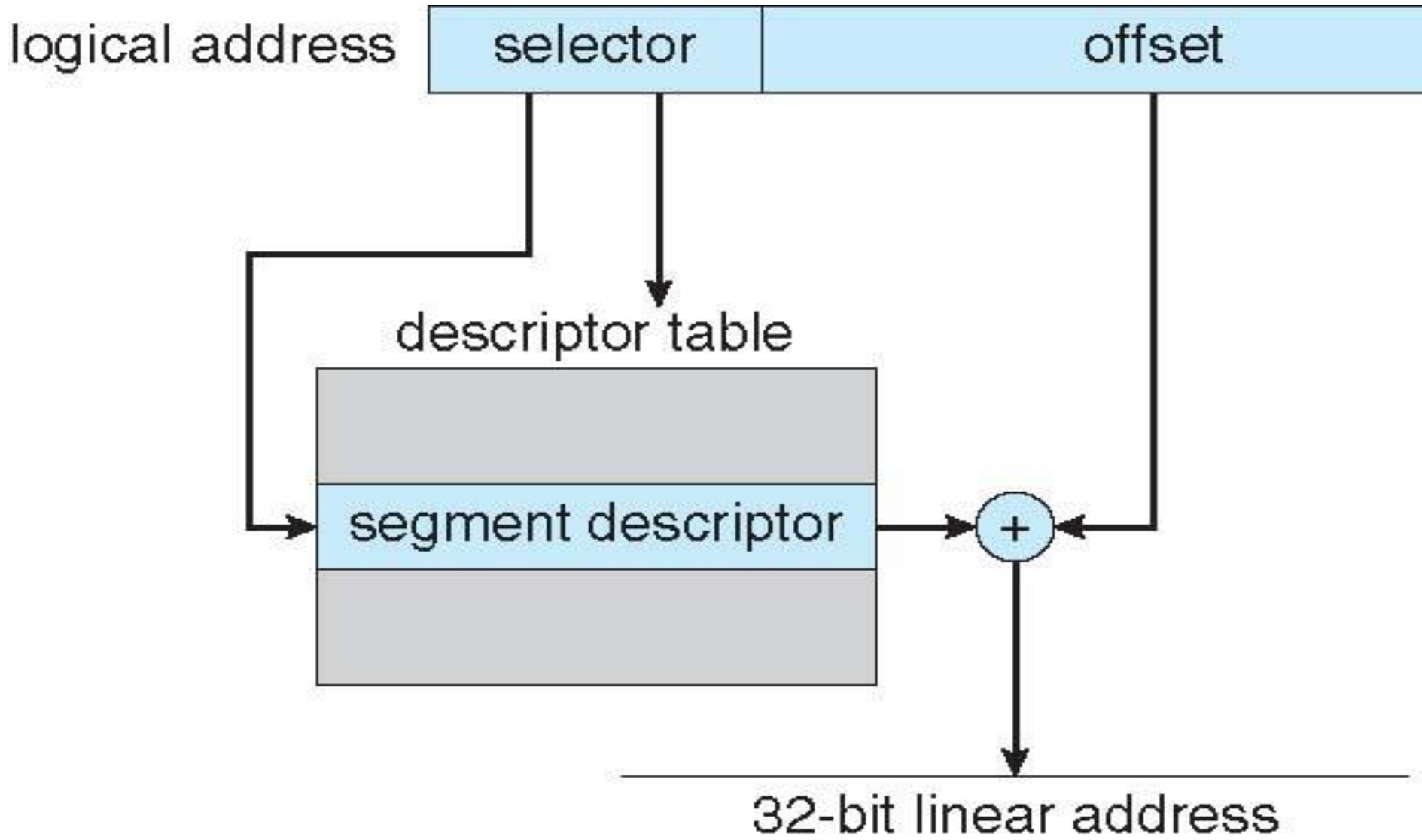
  - Doğrusal adres sayfalama birimine verilir.
    - Ana bellekte fiziksel adres üretir.
    - Sayfalama birimleri eşdeğer MMU oluşturur.
    - Sayfaların boyutları 4 KB ya da 4 MB olabilir.

Pentium'da Mantıksal Adresin Fiziksel Adrese Dönüştürülmesi

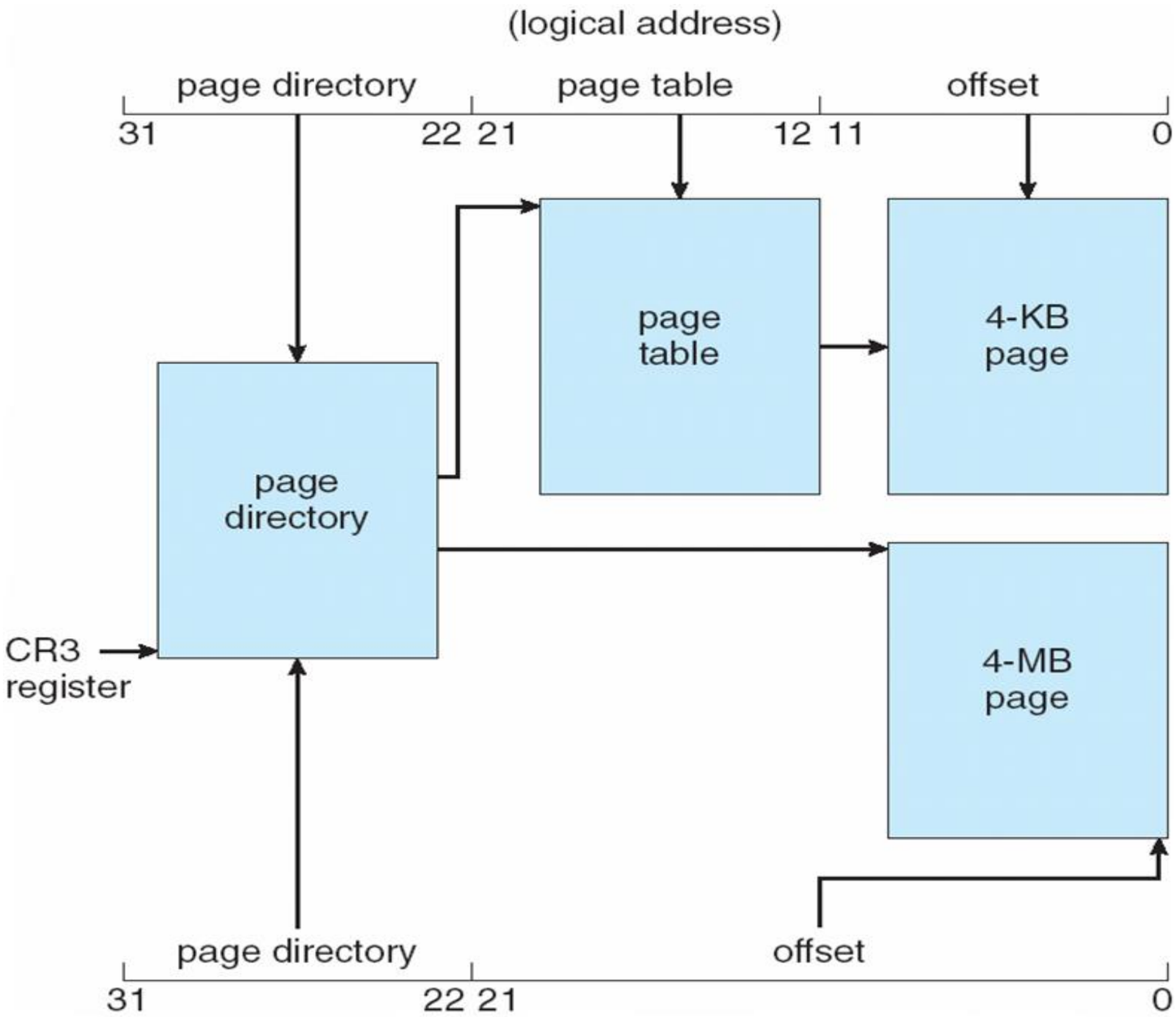




Intel Pentium Segmentasyonu

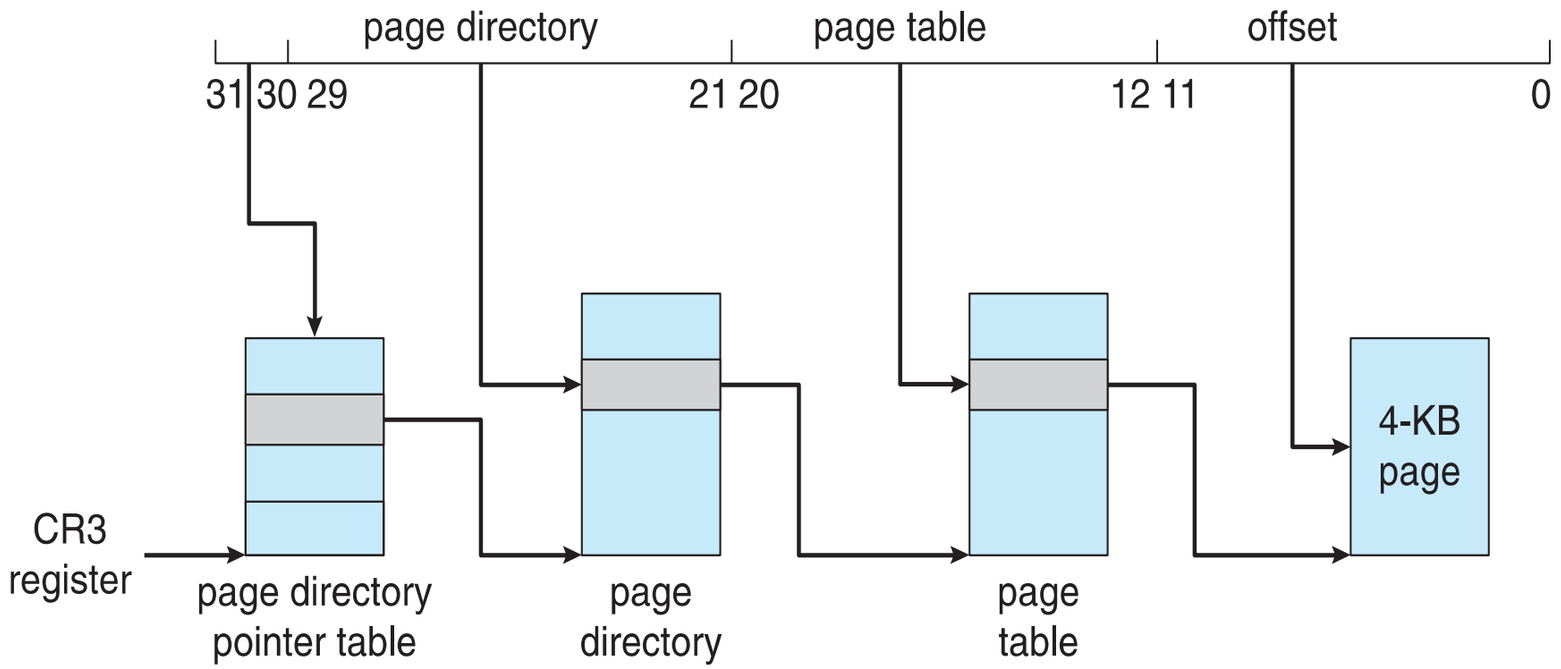


Pentium Sayfalama Mimarisi



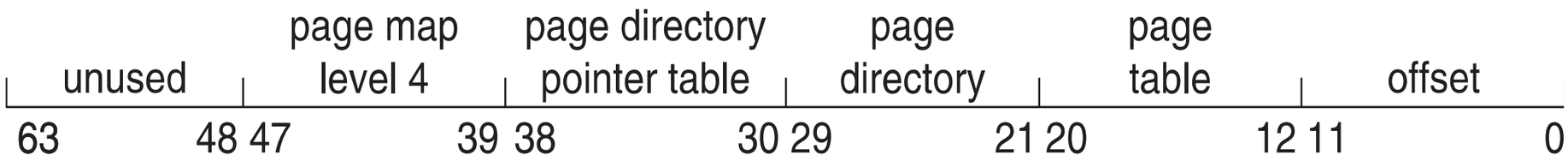
Intel IA-32 Page Address Extensions

- 32 bit adres limitleri Intel'in sayfa adresi uzantısını (**page address extension** (**PAE**)) oluşturmasına ve 32 bit uygulamaların 4 GB'den daha fazla bellek alanına erişmesine izin verdi
  - Sayfalama 3 seviyeli şema
  - İlk iki bit, bir sayfa dizini işaretçi tablosuna başvurur **page directory pointer table**
  - Sayfa dizin ve sayfa tablosu girdileri, boyut olarak 64 bit oldu
  - Net etki adres alanını 36 bit'e yükseltiyor - 64 GB fiziksel bellek



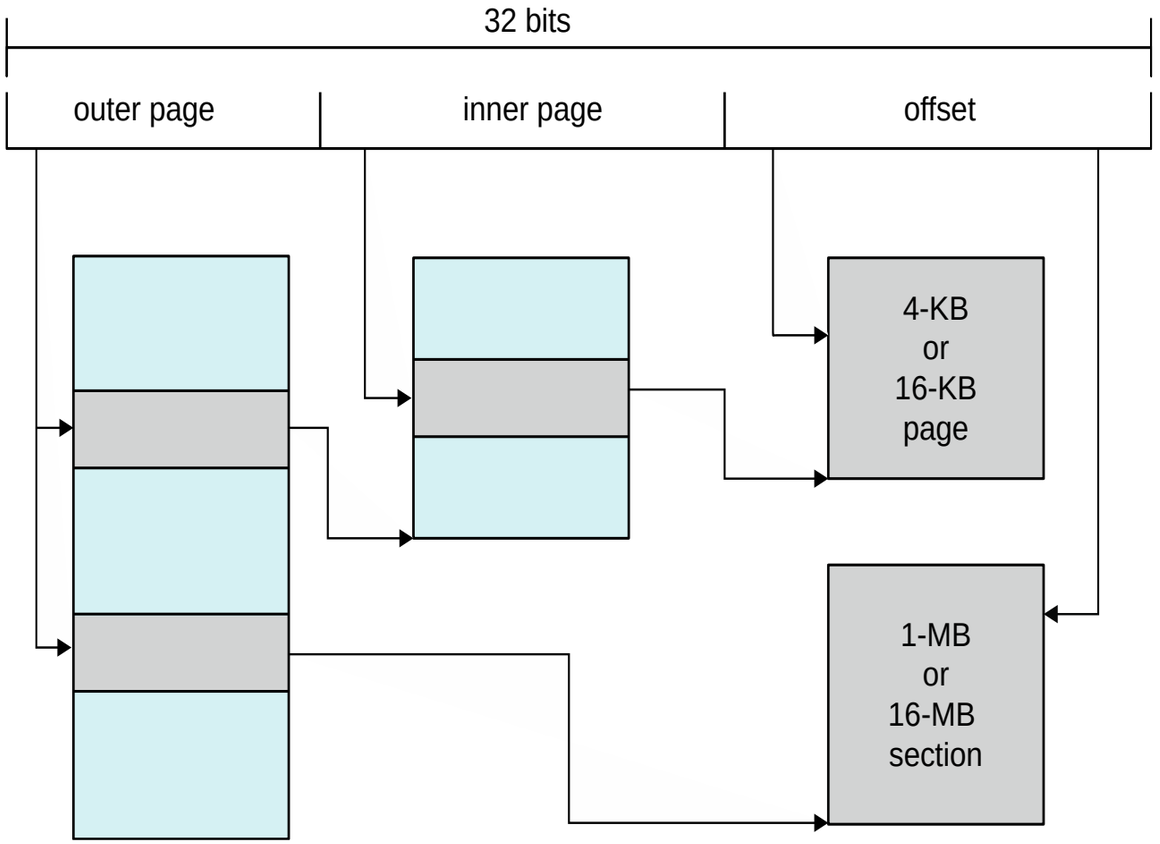
Intel x86-64

- Mevcut nesil Intel x86 mimarisi
- 64 bit devasadır (> 16 exabyte)
- Pratikte sadece 48 bit adresleme gerçekleştirilir
  - Sayfa boyutları 4 KB, 2 MB, 1 GB
  - Dört seviye sayfalama hiyerarşisi
- PAE'yi de kullanabilirsiniz, böylece sanal adresler 48 bit ve fiziksel adresler 52 bittir



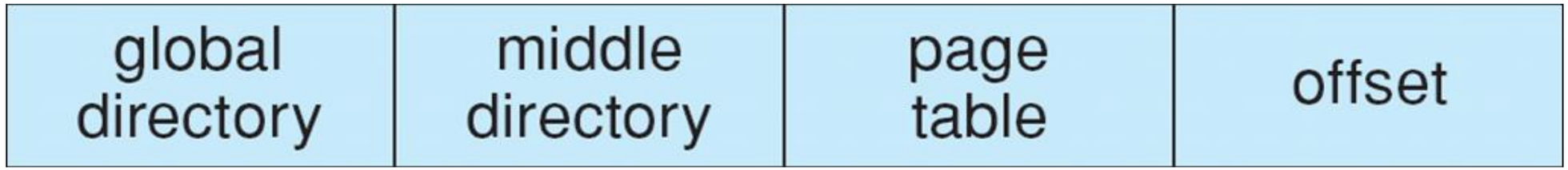
Example: ARM Architecture

- Dominant mobil platform çipi (örneğin Apple iOS ve Google Android cihazları)
- Modern, enerji verimli, 32 bit CPU
- 4 KB ve 16 KB sayfalar
- 1 MB ve 16 MB sayfalar (bölümler - **sections -** olarak adlandırılır)
- Bölümler için tek düzey sayfalama, daha küçük sayfalar için iki seviyeli
- İki TLB seviyesi
  - Dış (Outer) seviyenin iki mikro TLB'si vardır (bir veri, bir komut)
  - İç (Inner) tek ana TLB'ye sahip
  - İlk iç denetlenir, yoksa dış denetlenir ve CPU tarafından gerçekleştirilir.



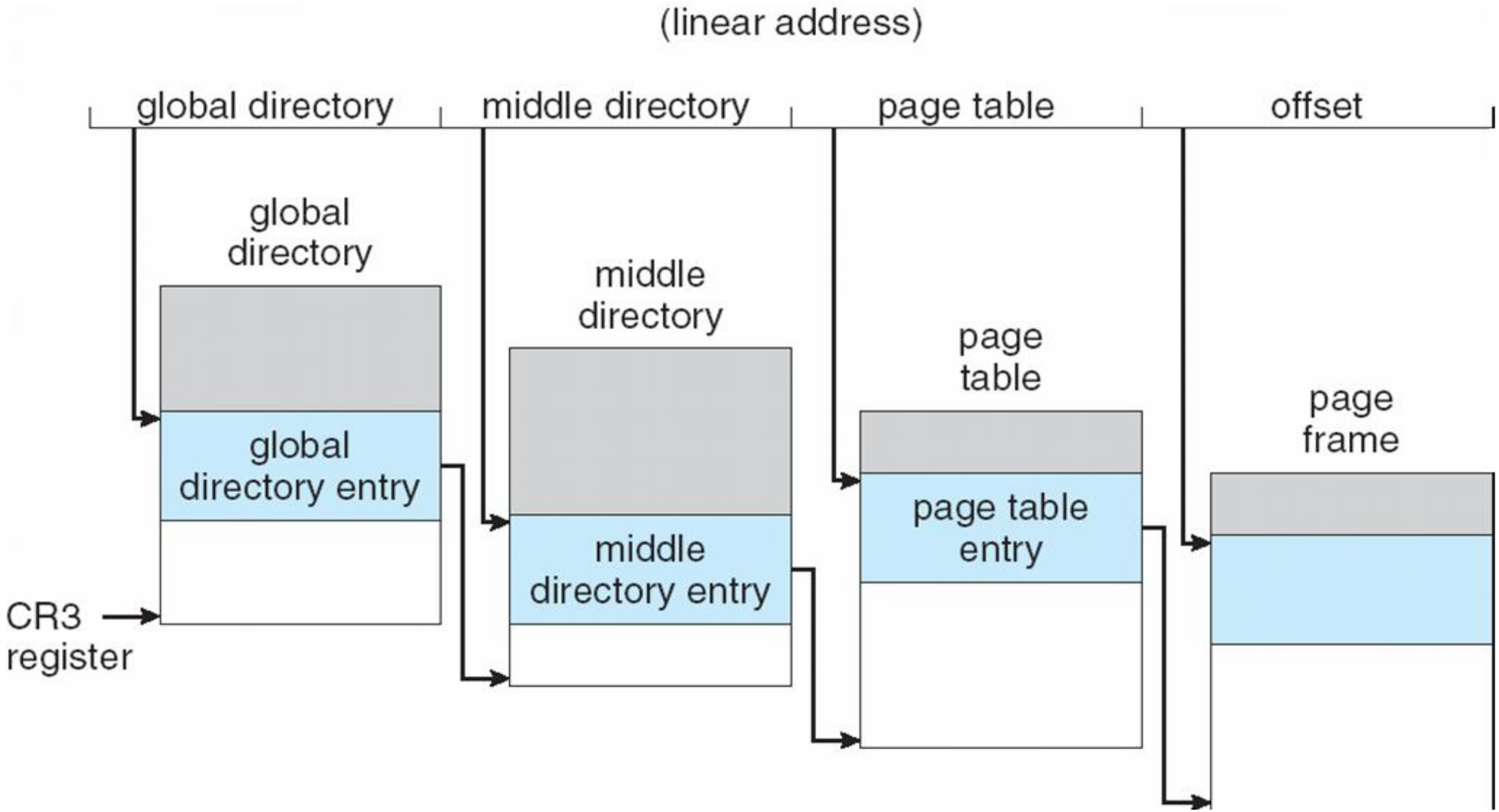
Linux'ta Doğrusal Adres

- 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



8. Bölüm Sonu