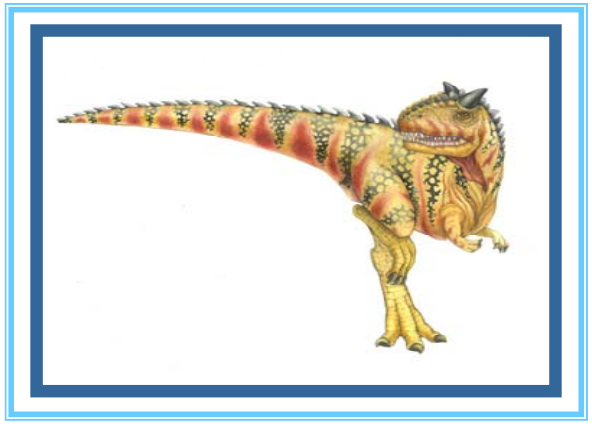
Bölüm 9: Ana Bellek (Main Memory)



Bölüm 8: Bellek Yönetimi

- Arkaplan
- Takas (Swapping)
- Ardışık Bellek Tahsisi (Contiguous Memory Allocation)
- Sayfalama
- Sayfa Tablosunun Yapısı
- Segmentasyon
- Örnek: Intel Pentium

Hedefler

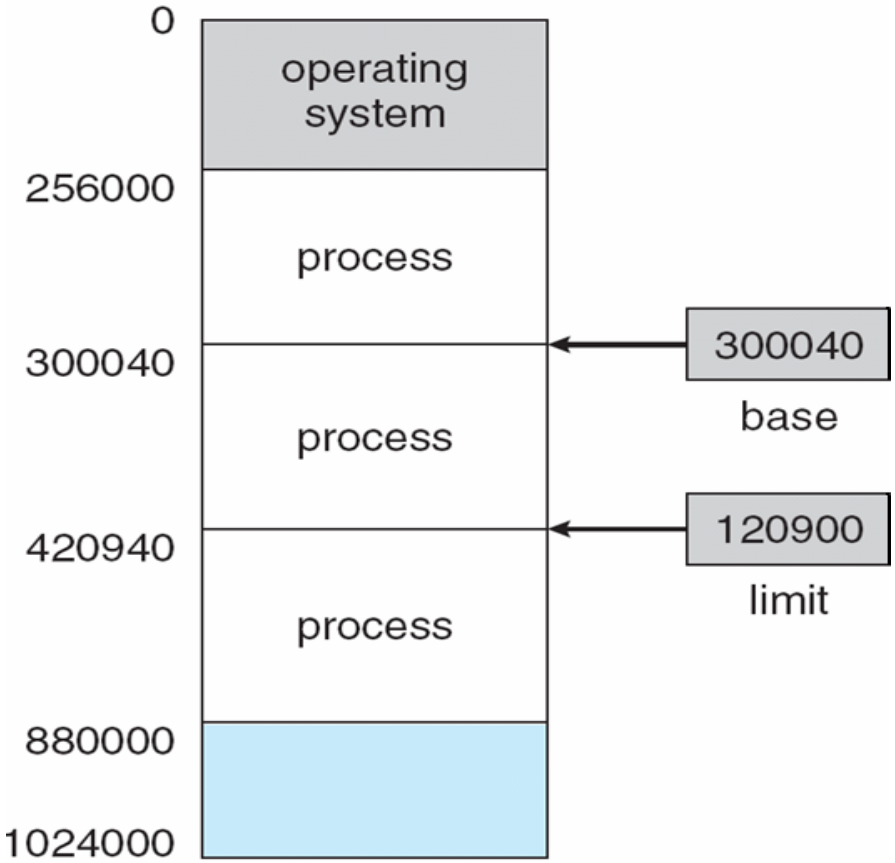
- Bellek donanımını organize etme yollarını detaylı bir şekilde açıklamak
- Sayfalama ve segmentasyon da dahil olmak üzere çeşitli bellek yönetim teknolojilerini tartışmak
- Sadece segmentasyon ve sayfalamalı segmentasyon tekniklerinden her ikisini de destekleyen Intel Pentium'u detaylı bir şekilde tanımlamak

Arkaplan

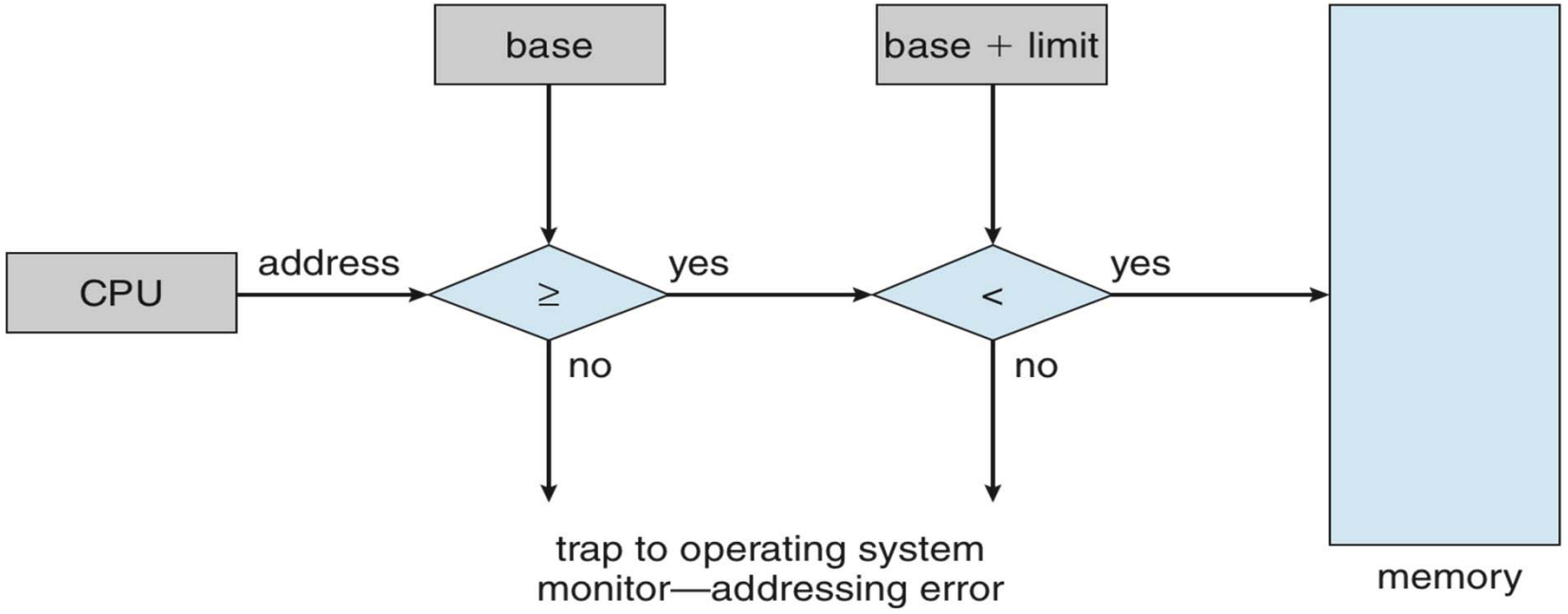
- Program, diskten belleğe getirilip çalışması için bir process'e yerleştirilmelidir.
- Ana bellek ve kaydediciler CPU'nun doğrudan erişebildiği kayıt ortamlarıdır
- Bellek ünitesi yalnızca adresler + okuma istekleri veya adres + veri ve yazma istekleri ile ilgilenir
- Kaydedici erişimi bir CPU çevriminde (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** register çifti mantıksal adres alanı olarak tanımlanırlar.



Base ve Limit Kaydedicileri ile Donanım Adresi Koruma



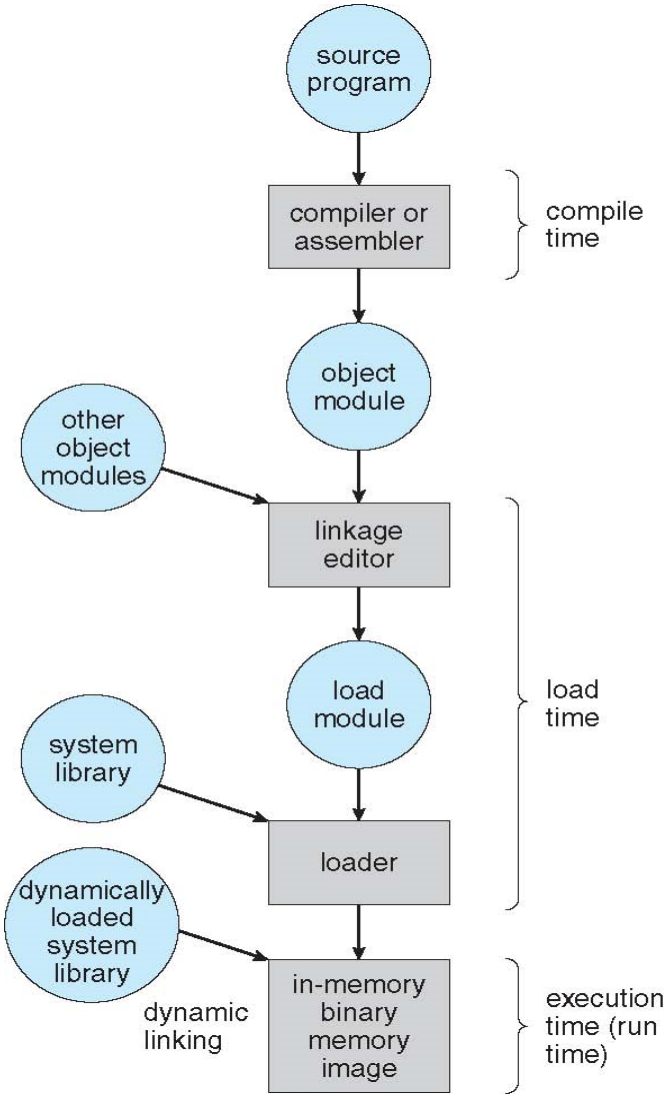
Adres Bağlama

- İlk kullanıcı prosesinin daima 0000 adresinde olması uygun değildir
  - Nasıl?
- Ayrıca, adresler bir programın yaşamının farklı aşamalarında farklı yollarla gösterilir.
  - Kaynak kod adresleri genellikle semboliktir.
  - Derlenmiş kod adresleri yeniden konumlandırılabilir adreslere **bağlanır**.
    - örneğin. “bu modülün başından itibaren 14 bytes”
  - Bağlayıcı veya Yükleyici 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 gecikir.
    - Adres haritalama için donanım desteği gerekir (örneğin, taban ve tavan kaydedicileri)

Bir Kullanıcı Programının Çok Adımlı Çalışması



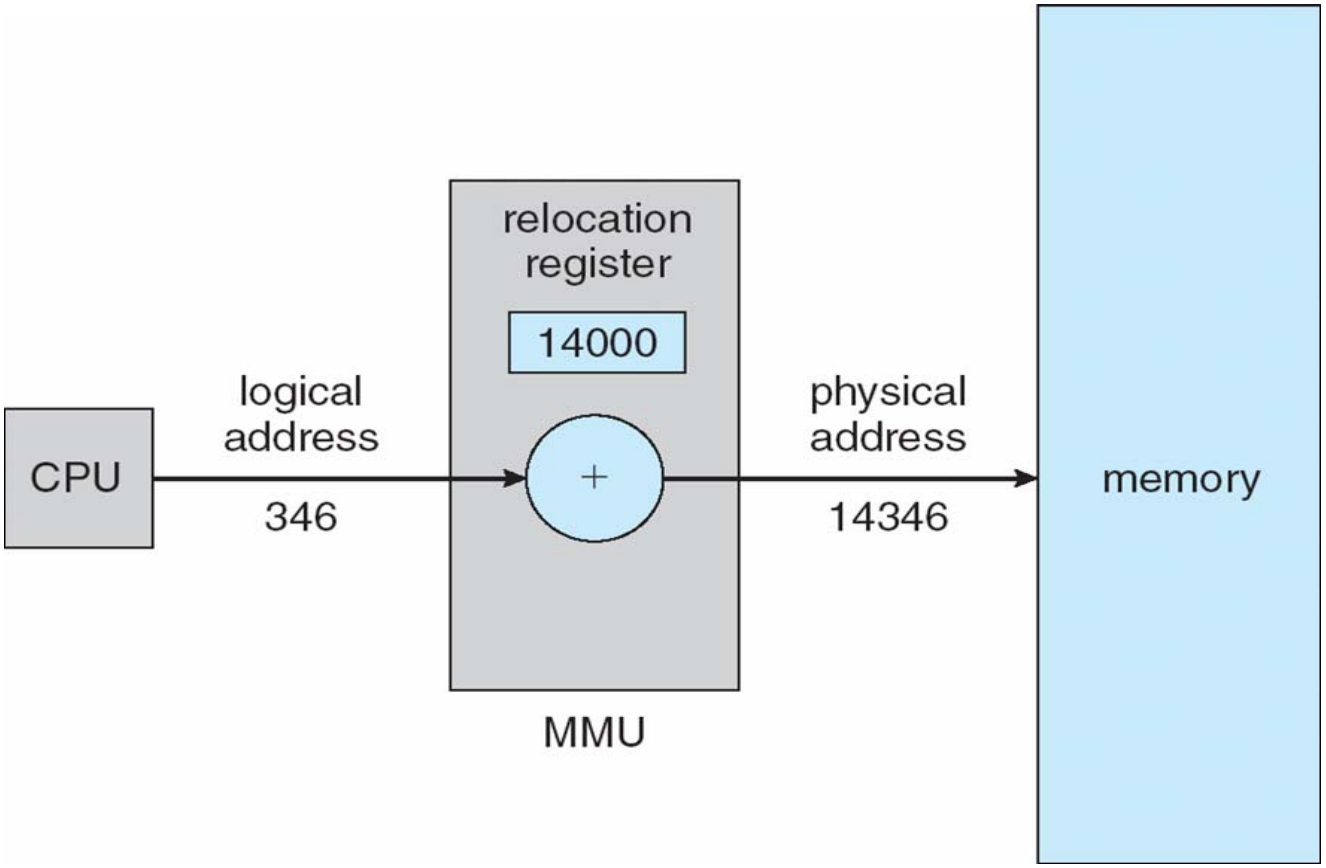
Mantıksal vs. Fiziksel Adres Uzayı

- Ayrı bir fiziksel adres uzayına bağlı olan mantıksal adres uzayı kavramı, 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.
- Mantıksal ve fiziksel adresler derleme ve yükleme adres bağlama düzenlerinde aynıdır, çalışma anında farklıdır
- **Mantıksal adres uzayı** bir program tarafından oluşturulan tüm mantıksal adreslerin kümesidir.
- **Fiziksel adres uzayı** bir program tarafından oluşturulan tüm fiziksel adreslerin 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 herbir adrese eklendiği basit bir düzeni ele alalım.
  - Taban 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.

Yeniden Konumlandırma Kaydedicisi Kullanılarak Dinamik Konumlandırma



Dinamik Yükleme

- 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.
- İşletim sistemi tarafından özel bir desteğe ihtiyaç duyulmaz.
  - Programın tasarımına bağlı olarak uygulanır.
  - İşletim sistemi dinamik yüklemeyi uygulaman için kütüphaneler sağlayarak yardım edebilir.

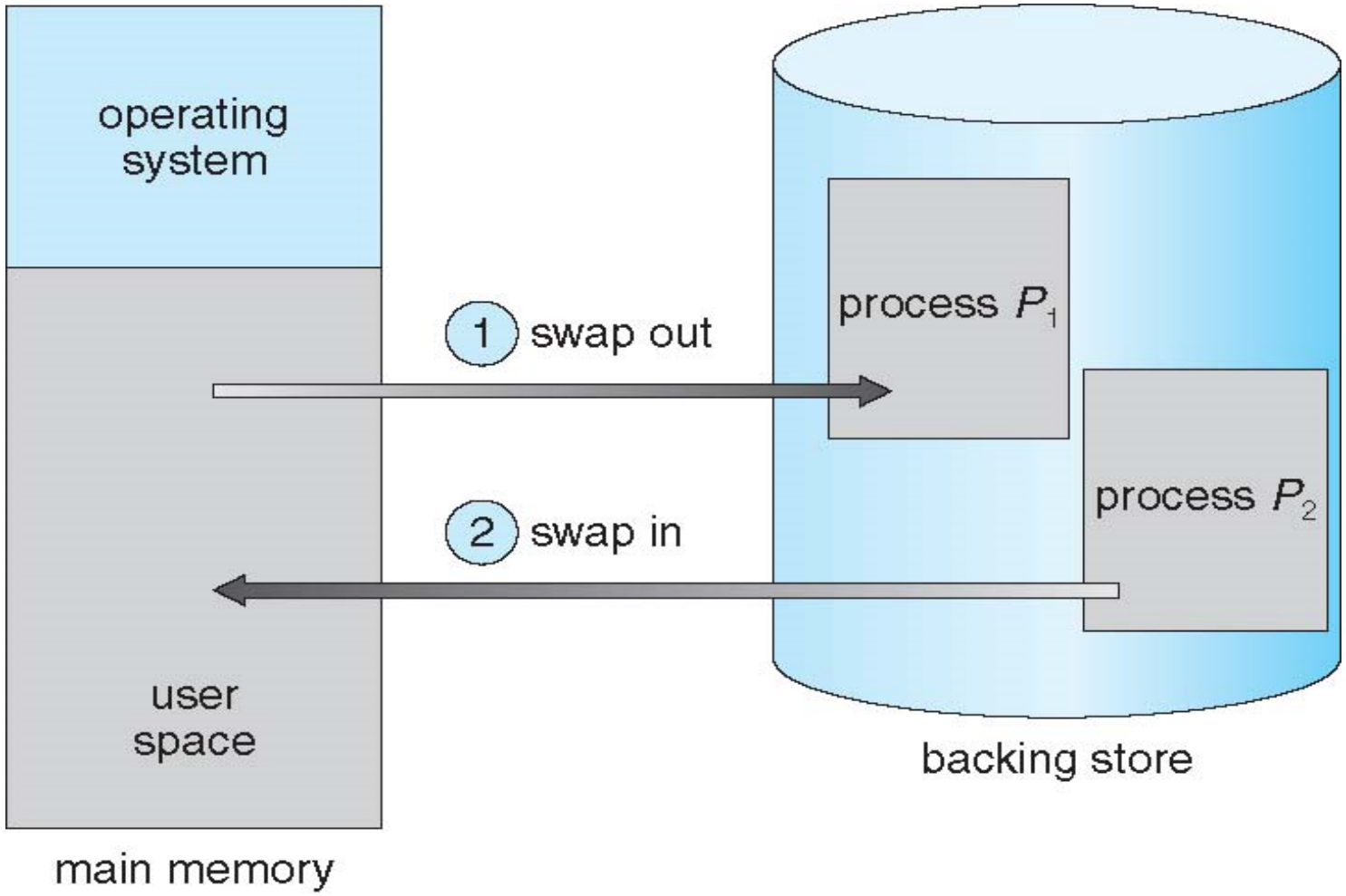
Dinamik Bağlama

- 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 kitaplığı altprogramının yerini tespit etmek için kullanılır.
- Stub altprogramın adresi ile kendisinin yerini değiştirerek altprogramı yürütür.
- İşletim sistemi altprogramın 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.

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 resimlerinin kopyalarını barındıracak kadar büyük ve hızlı disk ; bu hafıza görsellerine doğrudan erişim sağlanmalıdır.
- **Roll out, roll in (Dışa taşıma, içe taşıma)** – Öncelik tabanlı planlama algoritmaları için kullanılan değişkenin farklılaştırılmasıdır. Düşük öncelikli bir process değiştirilerek 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 çalıştırılmaya diskte bellek görüntüleri var olan hazır processlerin ready queue (hazır kuyruğunda) tutar.
- Does the swapped out process need to swap back in to same physical addresses?
- Depends on address binding method
  - Plus consider pending I/O to / from process memory space
- Swapping işleminin değiştirilmiş versiyonları pek çok sistemde bulunur.(ör., UNIX, Linux, ve Windows)
  - Takas işlemi normalde devre dışıdır.
  - Ayrılmış bellek alanın 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

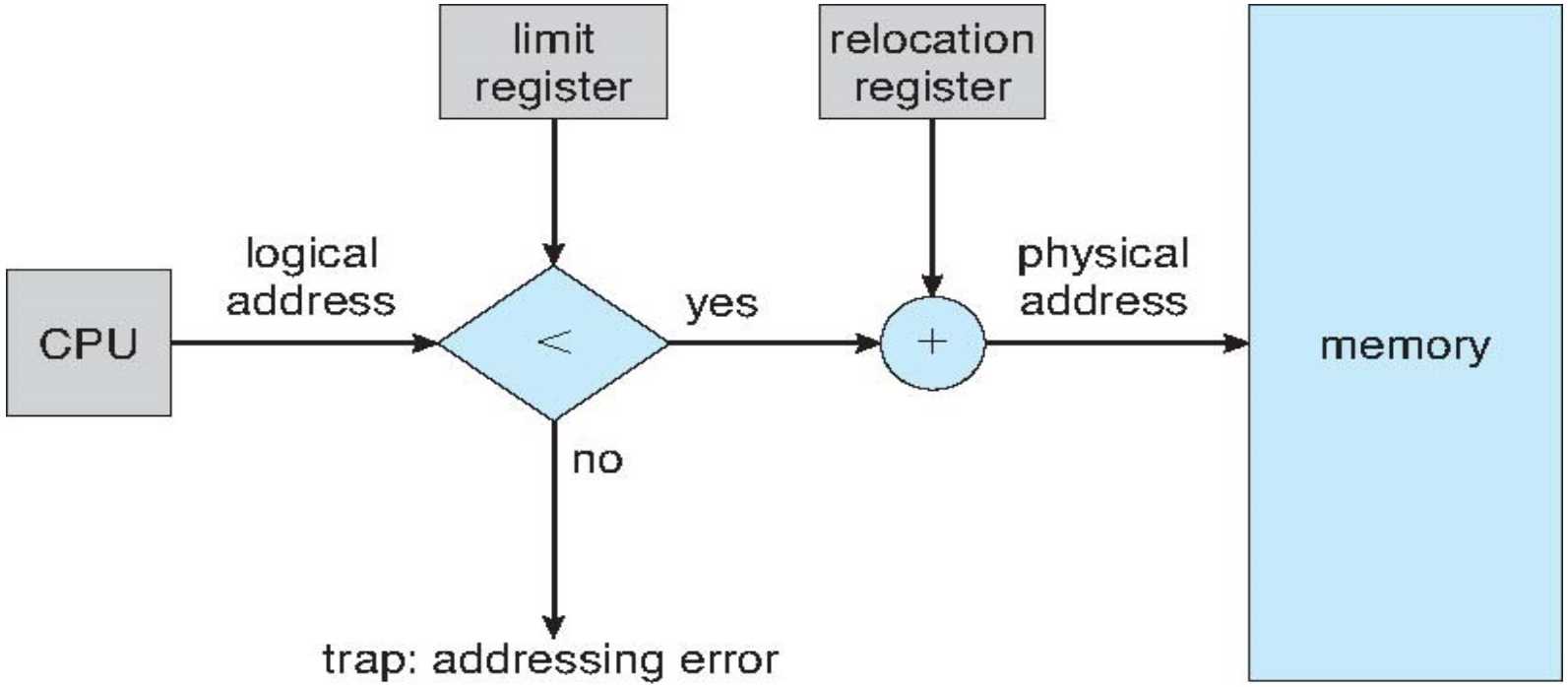
- If next processes to be put on CPU is not in memory, need to swap out a process and swap in target process
- Context switch time can then be very high
- 100MB process swapping to hard disk with transfer rate of 50MB/sec
  - Plus disk latency of 8 ms
  - Swap out time of 2008 ms
  - Plus swap in of same sized process
  - Total context switch swapping component time of 4016ms (> 4 seconds)
- Can reduce if reduce size of memory swapped – by knowing how much memory really being used
  - System calls to inform OS of memory use via `request memory` and `release memory`

Sürekli Tahsis (Contiguous Allocation)

- 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.

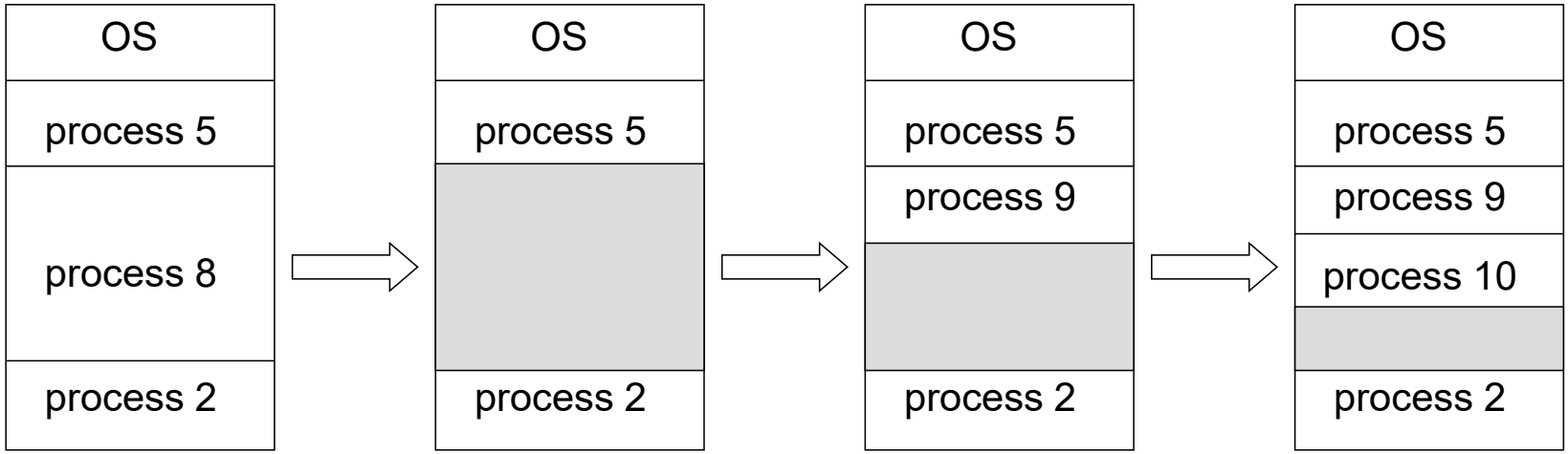
- Yerdeğiştirme register'ı (relocation register) kullanıcı process'lerini bir diğerinden korumak için kullanılır.
  - Base register küçük fiziksel adres değerini içerir.
  - Limit register mantıksal adresler dizisini içerir - her mantıksal adres limit register'dan daha kısa olmalıdır.
  - MMU mantıksal adresi *dinamik olarak* haritalar.
  - Daha sonra kernel kodunun geçici olarak (**transient**) kernel boyutunu değiştirmesi gibi eylemlere izin verebilir.

Yer Değiştirme Register'ları ve Limit Register'lar İçin Donanım Desteği



Sürekli Tahsis (Devam)

- Çoklu bölüm tahsisi
  - Degree of multiprogramming limited by number of partitions
  - 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)

- **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 durum 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.
  - Shuffle memory contents to place all free memory together in one large block
  - Sıkıştırma işlemi mümkündür ancak, sadece takas (relocation) işlemi dinamik ise yapılır ve yürütme zamanında gerçekleştirilir.
  - I/O (Giriş / Çıkış Sorunu)
    - Latch job in memory while it is involved in I/O
    - Sadece işletim sistemi buffer'larıyla I/O işlemleri yapın.
- Şimdi yedekleme deposunda da (backing store) aynı sorunun olduğunu düşünün.

Sayfalama

- Bir process'in fiziksel bellek alanı sürekli olmayabilir, sonraki bellek alanı kullanılabilir olduğu sürece fiziksel bellek alanı ayrılır.
- Fiziksel belleğin sabit boyutlu bloklar halinde bölünmüş haline **frames (çerçeveler)** denir.
  - Size is power of 2, between 512 bytes and 16 Mbytes
- Mantıksal adres eş boyutlara bölünür ve bu bölümlere **pages (sayfalar)** deriz.
- Tüm boş frame'leri takip eder.
- *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) kurun
- Yedekleme deposu aynı şekilde sayfalara bölünür.
- Hala iç parçalanma (Internal fragmentation) mevcuttur.

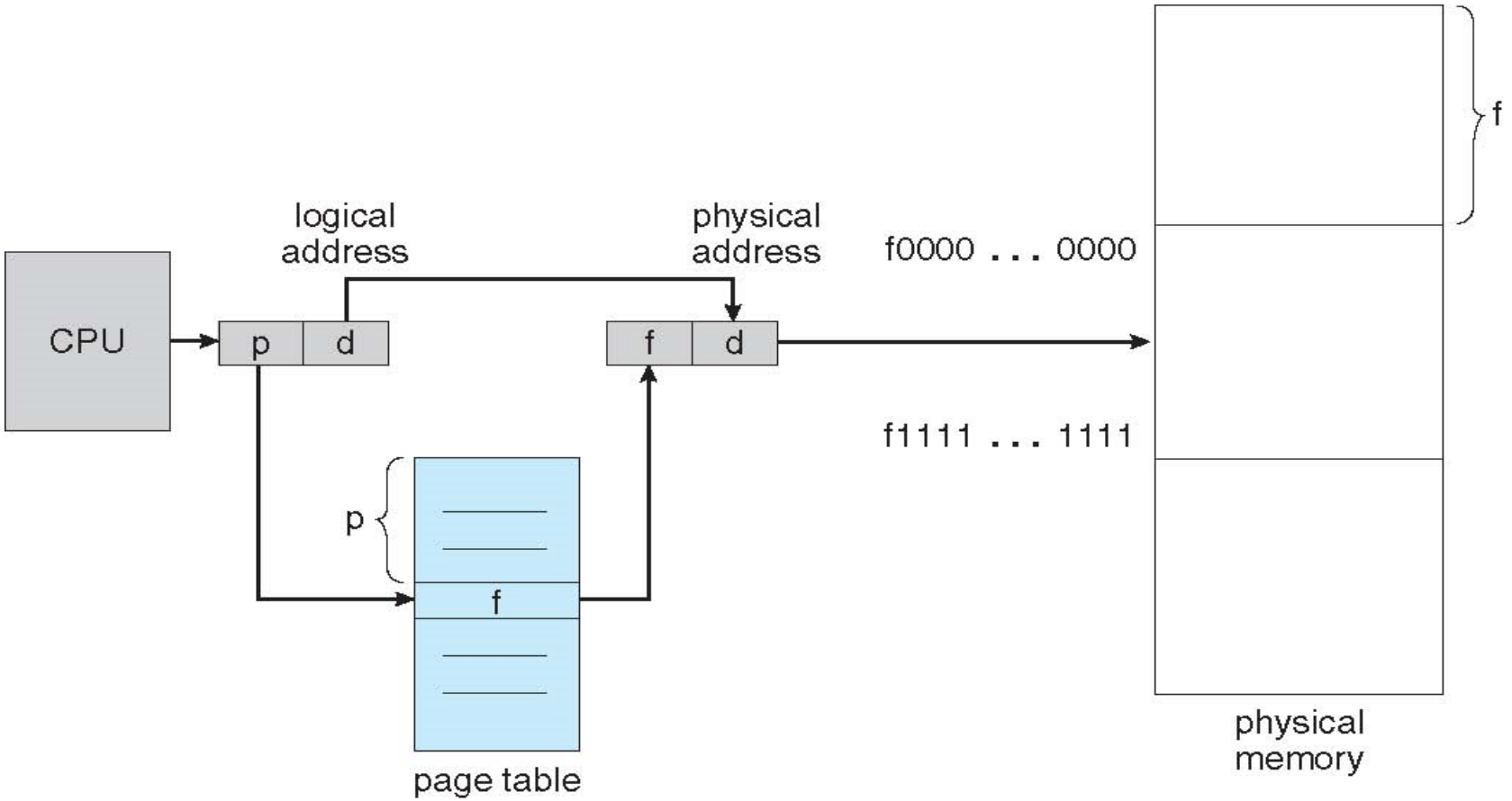
Adres Çeviri Şeması

- 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*)** – combined with base address to define the physical memory address that is sent to the memory unit

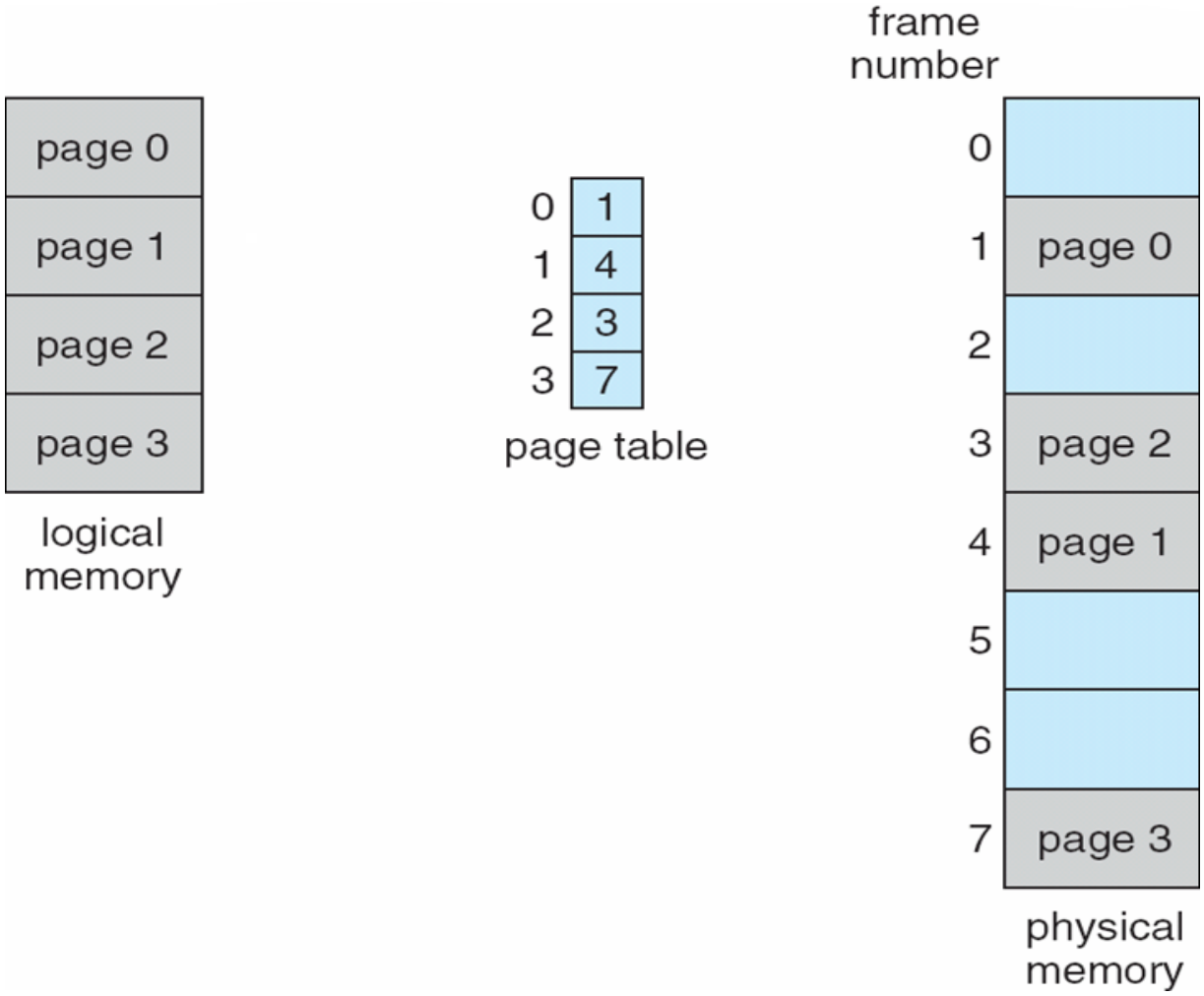
page number	page offset
p	*d*
m - n	*n*

- For given logical address space 2^m and page size 2^n

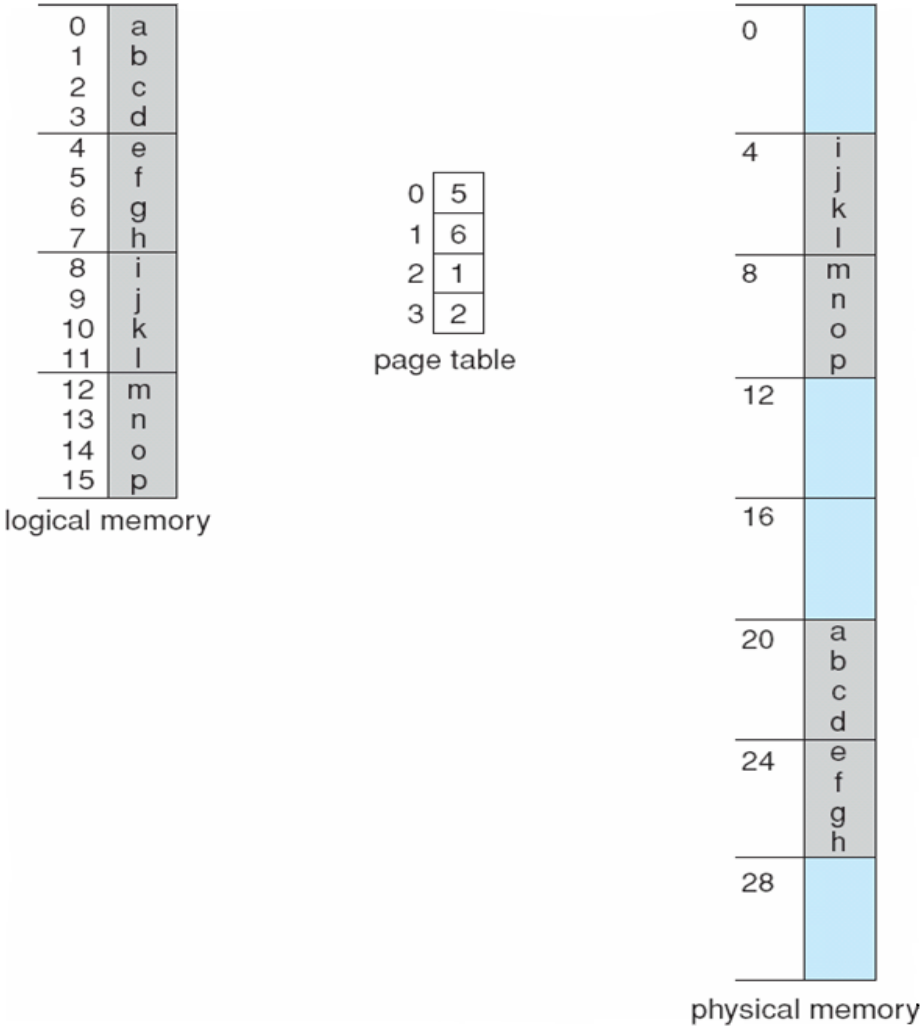
Donanım Sayfalama



Mantıksal ve Fiziksel Bellek Sayfalama Modeli



Sayfalama Örneps

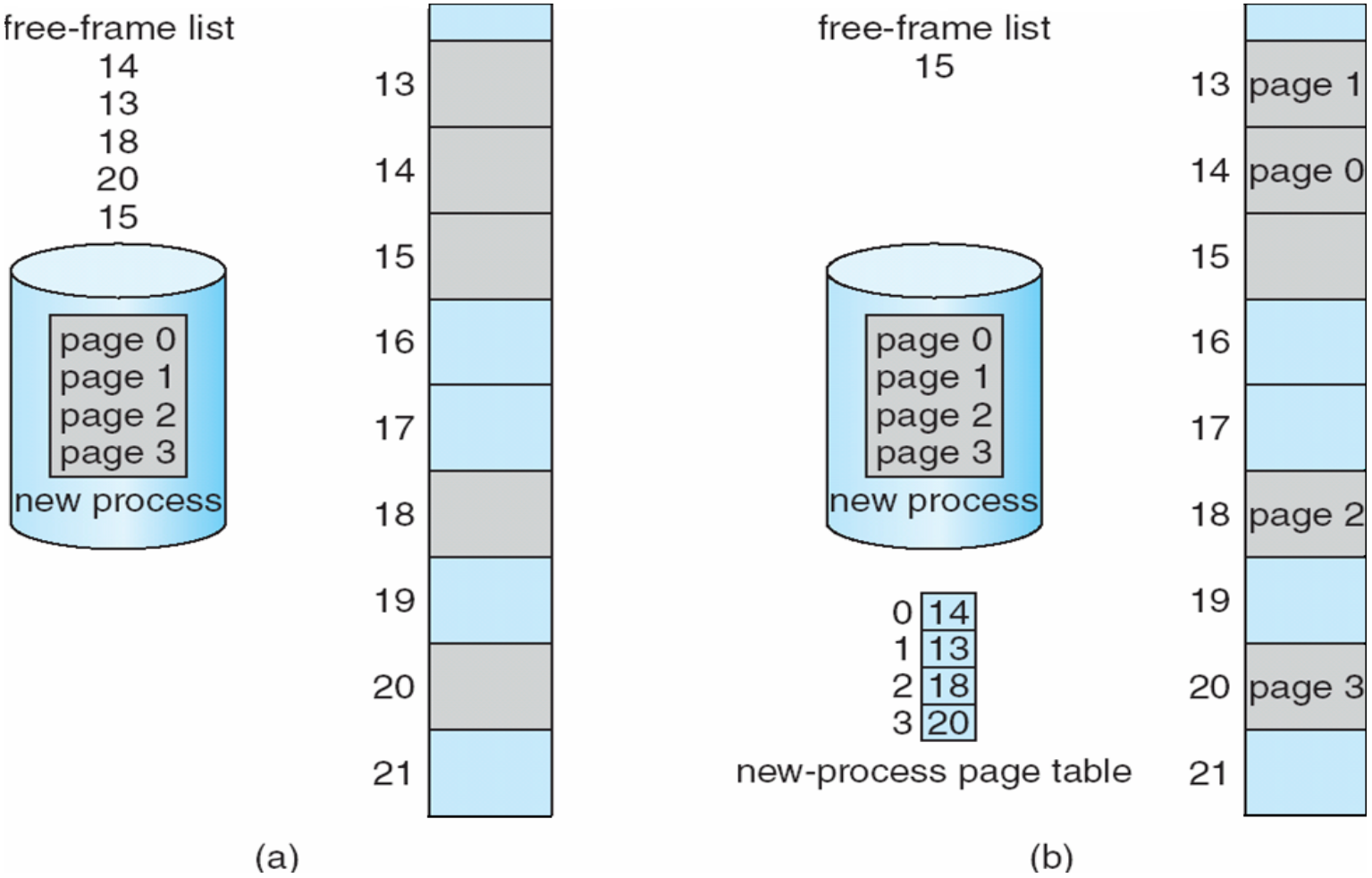


n=2 ve *m*=4 32-byte bellek ve 4-byte'lık sayfalar

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
  - En kötü durumdaki parçalanma = 1 frame – 1 byte
  - Ortalama parçalanma = 1 / 2 frame size
  - So small frame sizes desirable?
  - But each page table entry takes memory to track
  - Sayfa boyutları zaman içinde büyür.
    - Solaris iki sayfa boyutu destekler. – 8 KB ve 4 MB
- Process view and physical memory now very different
- Uygulama process'i sadece kendi belleğine erişebilir.

Serbest Frame'ler



Before allocation

After allocation

Sayfa Tablosu Uygulaması

- 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 donanım taraflı **associative memory** ya da **translation look-aside buffers (TLBs)** olarak isimlendirilen özel hızlı-arama önbelleği ile çözülebilir.
- Some TLBs store **address-space identifiers (ASIDs)** in each TLB entry – uniquely identifies each process to provide address-space protection for that process
  - Otherwise need to flush at every context switch
- TLBs typically small (64 to 1,024 entries)
- On a TLB miss, value is loaded into the TLB for faster access next time
  - Replacement policies göz önünde bulundurulmalıdır.
  - Some entries can be **wired down** for permanent fast access

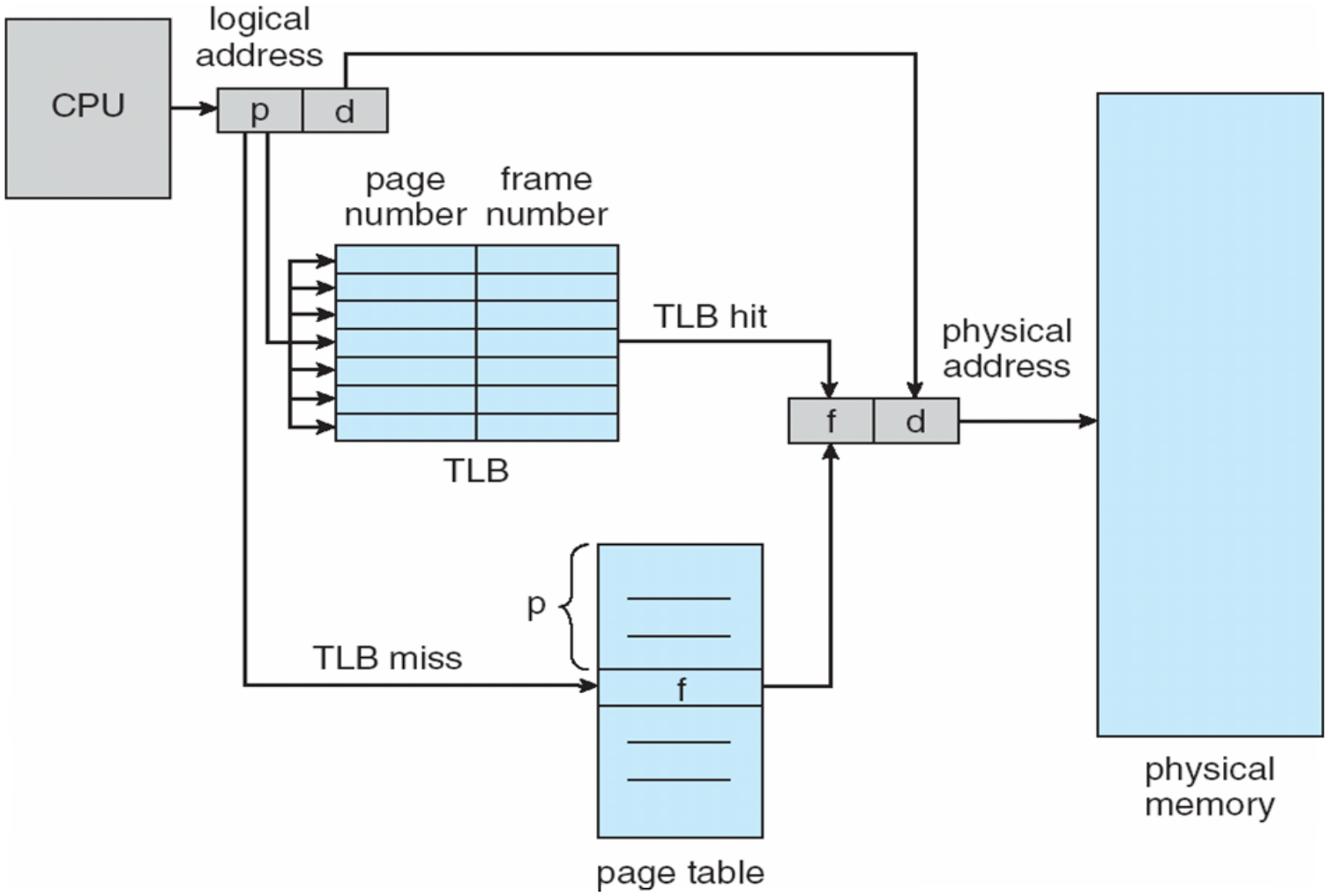
İlişkisel Bellek

- İlişkisel bellek – paralel arama

Page #	Frame #

- Adres dönüştürme (p, d)
  - If p is in associative register, get frame # out
  - Otherwise get frame # from page table in memory

TLB İle Donanım Sayfalama



Etkin Erişim Süresi

- İlişkisel arama = ε zaman birimi
  - Bellek erişim süresinin %10'undan az olabilir.
- Hit oranı = α
  - Hit oranı – ilişkisel kayıtlar içerisinde bir sayfa bulunma süresinin yüzdesi; ratio related to number of associative registers
- Consider α = 80%, ε = 20ns for TLB search, 100ns for memory access
- **Etkin Erişim Süresi (Effective Access Time - EAT)**

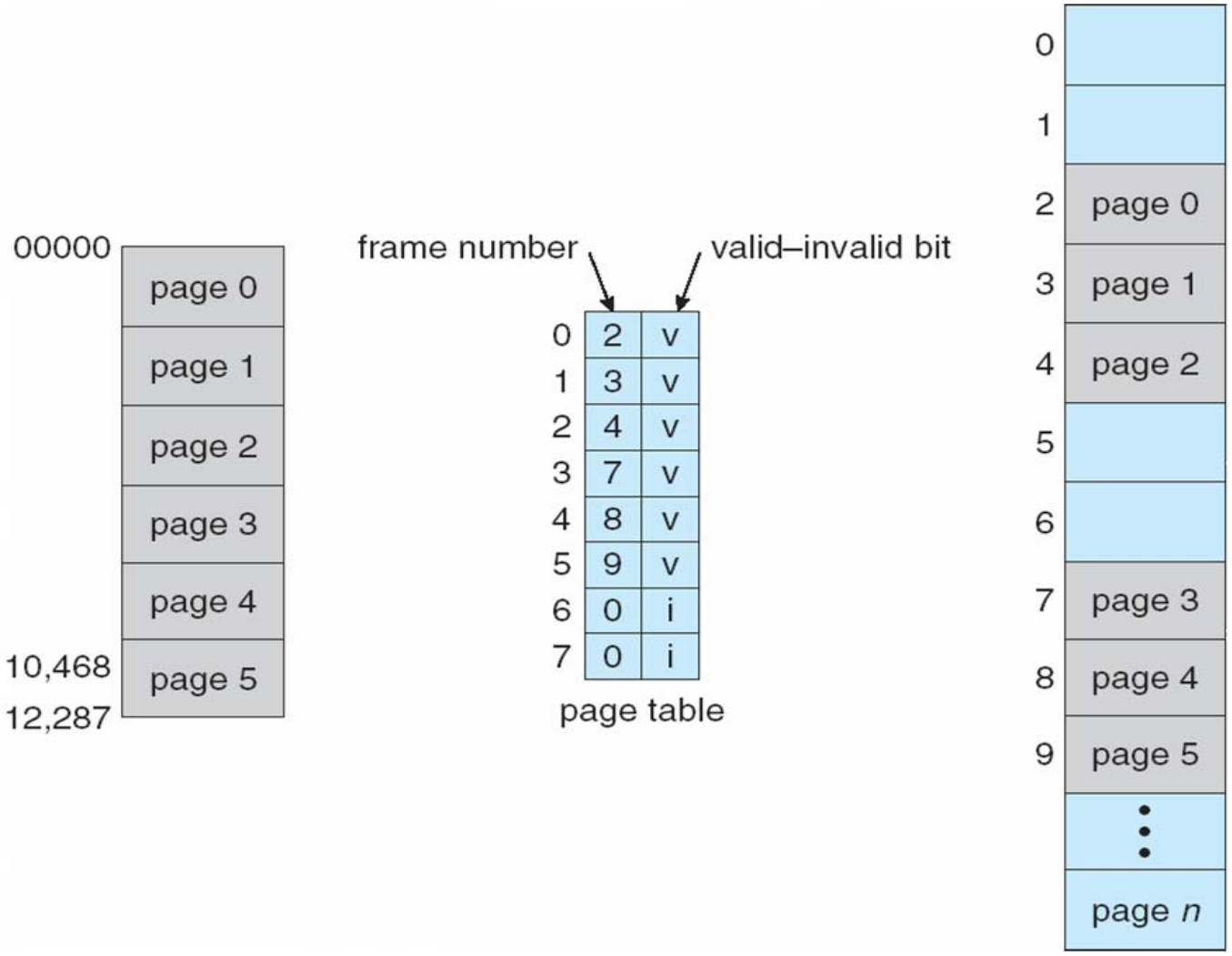
$$EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$
$$= 2 + \varepsilon - \alpha$$

- Consider α = 80%, ε = 20ns for TLB search, 100ns for memory access
  - EAT = 0.80 x 120 + 0.20 x 220 = 140ns
- Consider slower memory but better hit ratio -> α = 98%, ε = 20ns for TLB search, 140ns for memory access
  - 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 sayfa göstermek için daha fazla bit eklenebilir, vb.
- Sayfa tablosunda her girdi için geçerli-geçersiz (**Valid-invalid**) biti eklenir:
  - “valid” indicates that the associated page is in the process’ logical address space, and is thus a legal page
  - “invalid” indicates that the page is not in the process’ logical address space
  - Ya da PTLR kullanılır.
- Any violations result in a trap to the kernel

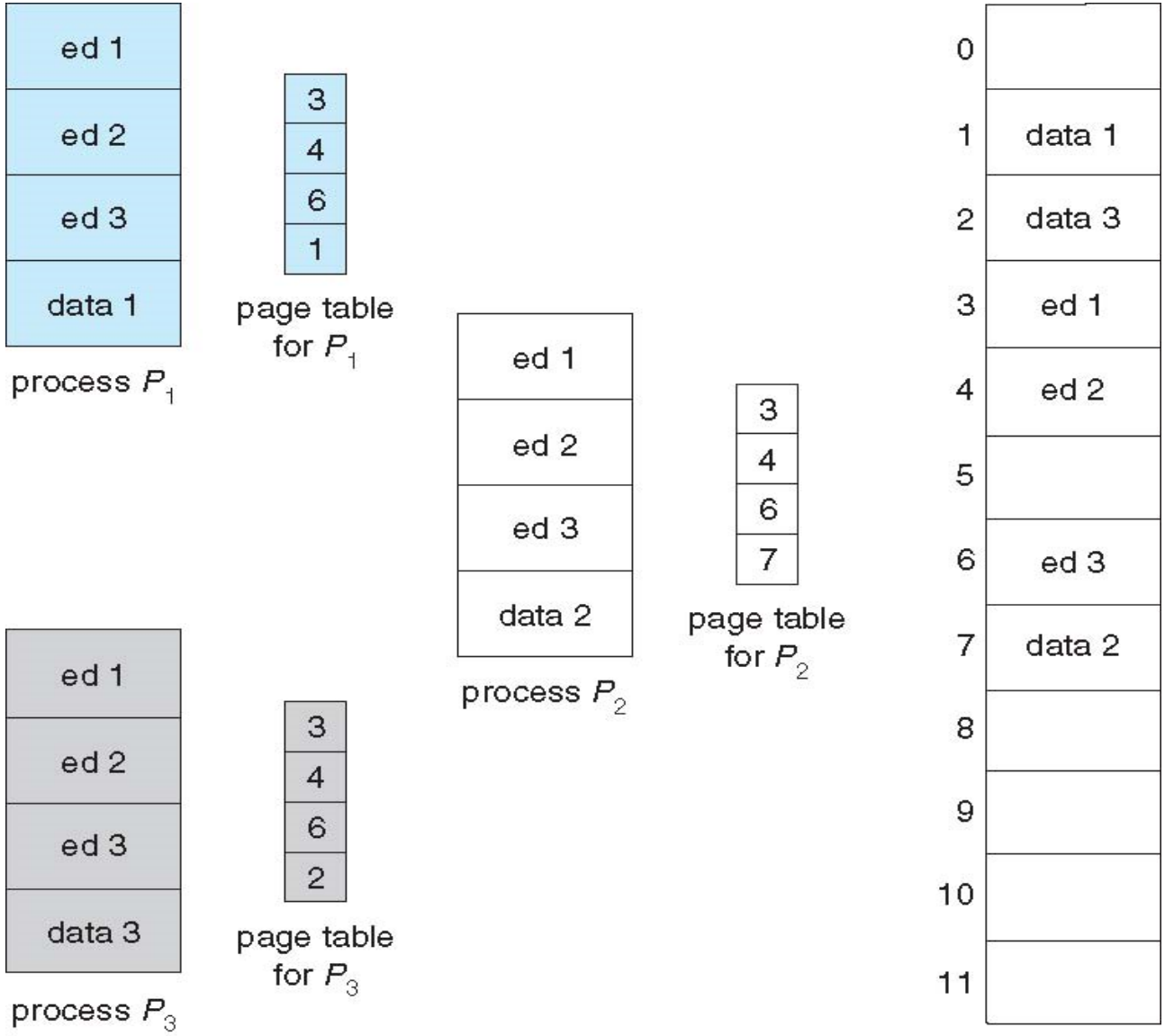
Sayfa Tablosundaki Geçerli (v) ya da Geçersiz (i) Bit



Paylaşımlı Sayfalar

- **Paylaşımlı kod**
  - Salt okunur (**reentrant – evrensel**) kodun tek kopyası processler arasında paylaşılır. (i.e., metin editörleri, derleyiciler, windows sistemleri)
  - Birden çok iş parçacığının aynı process alanını paylaşması gibi
  - Also useful for interprocess communication if sharing of read-write pages is allowed
- **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



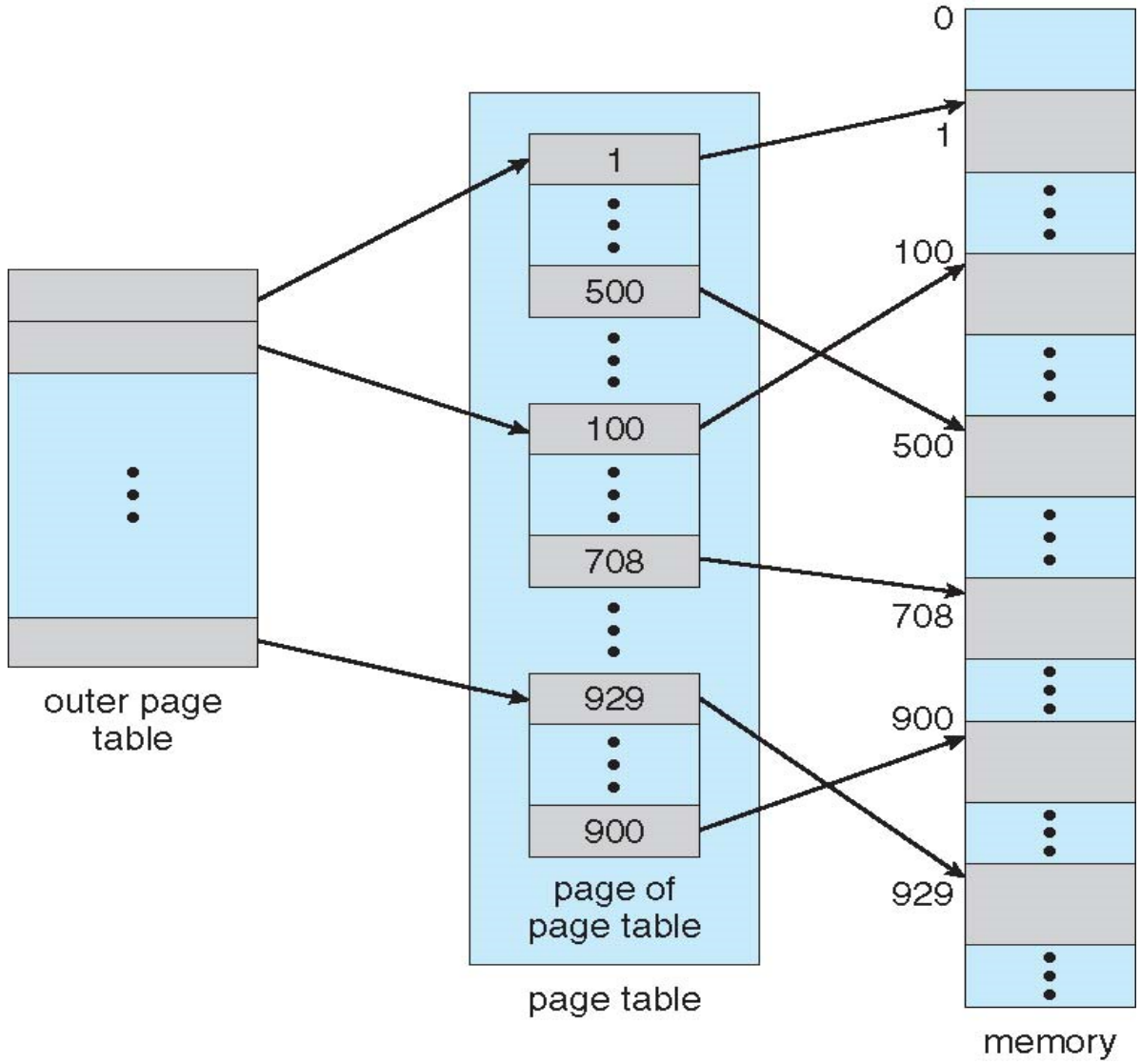
Sayfa Tablosunun Yapısı

- Memory structures for paging can get huge using straight-forward methods
  - Modern bilgisayarlarda 32-bit'lik mantıksal adresler olduğunu gözönüne alın.
  - Sayfanın boyutu 4 KB (2^{12})
  - Page table would have 1 million entries (2^{32} / 2^{12})
  - If each entry is 4 bytes -> 4 MB of physical address space / memory for page table alone
    - That amount of memory used to cost a lot
    - Don’t want to allocate that contiguously in main memory
- Hiyerarşik Sayfalama
- Hashed Sayfa Tabloları
- Inverted Page Tables

Hiyerarşik Sayfa Tablolama

- Break up the logical address space into multiple page tables
- İki aşamalı sayfa tablosu basit bir tekniktir.
- We then page the page table

İki Aşamalı Sayfa-Tablo Şeması



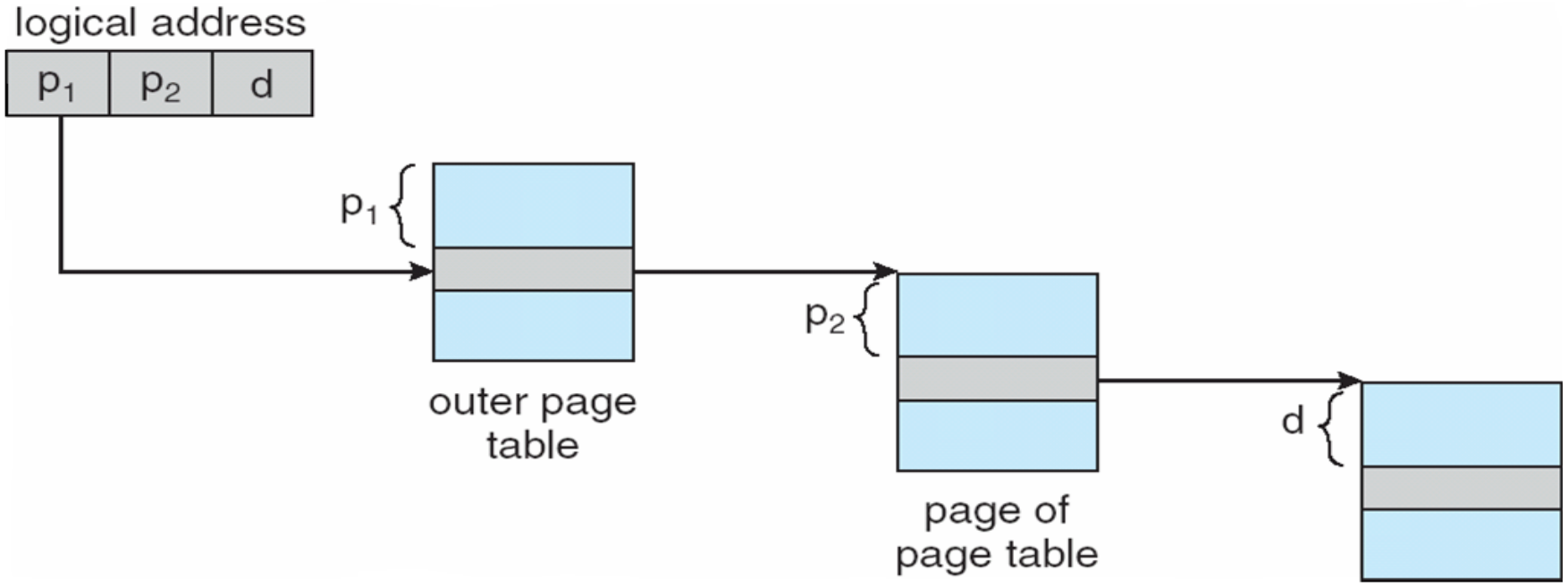
İki Aşamalı Sayfalama Örneği

- 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.
- Since the page table is paged, the page number is further divided into:
  - 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

- where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the inner page table
- **forward-mapped page table** olarak bilinir.

Adres Dönüşüm Şeması



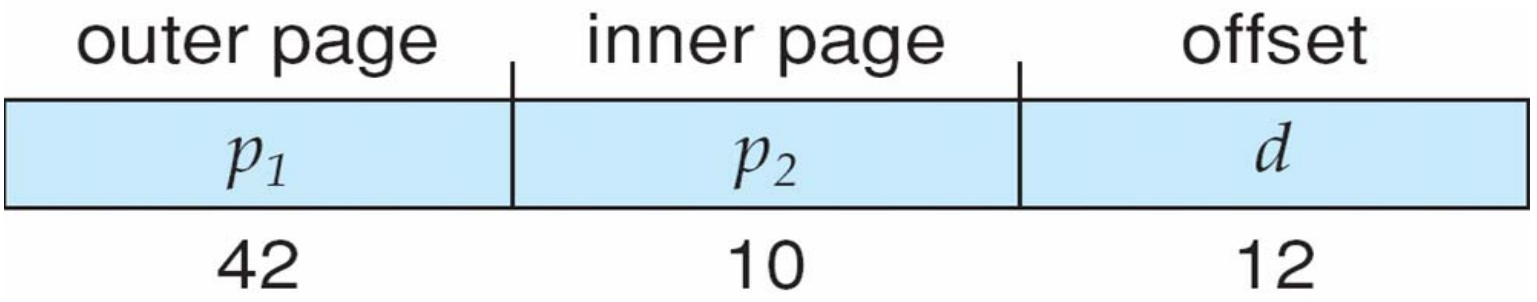
64-bit Mantıksal Adres Alanı

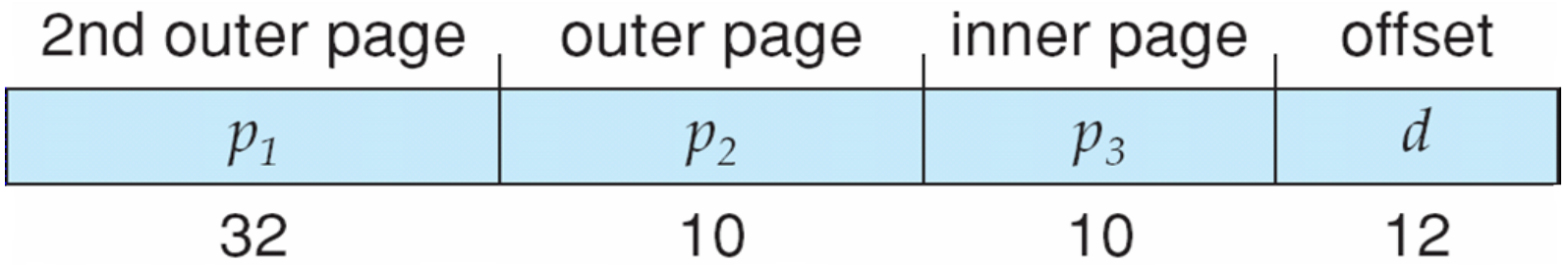
- İki aşamalı sayfalama şeması yeterli değildir.
- Eğer sayfa boyutu 4 KB (2^{12}) ise
  - Sayfa tablosunda 252 kayıt vardır.
  - If two level scheme, inner page tables could be 2^{10} 4-byte entries
  - Adres şunun gibi görünecektir:

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

  - 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

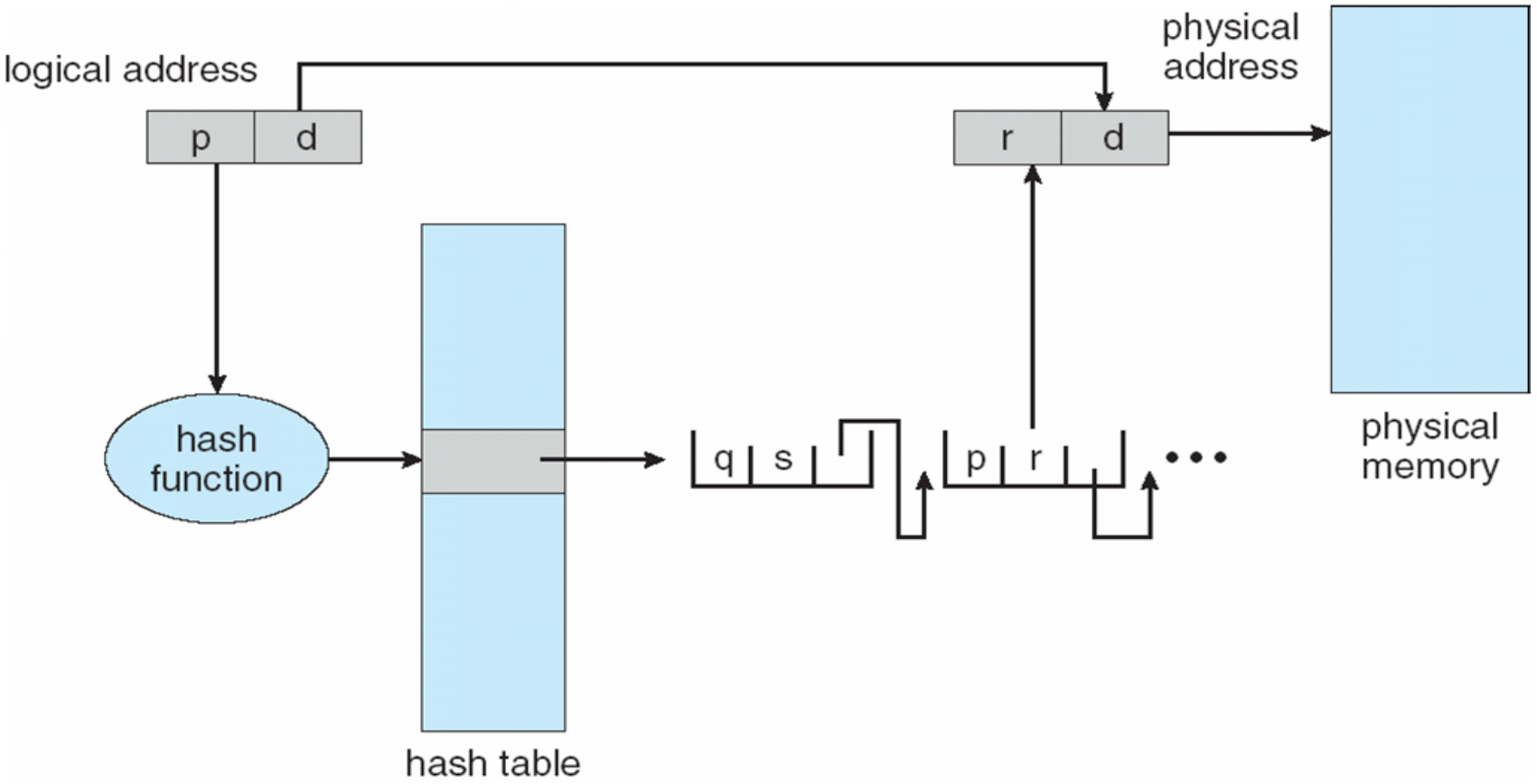




Hashed Sayfa Tabloları

- Ortak adres alanları > 32 bit
- Sanal sayfa numarası, bir sayfa tablosu içinde hashed durumdadır.
  - Bu sayfa tablosu contains a chain of elements hashing to the same location
- Her eleman (1) sanal sayfa numarası (2) eşlenmiş sayfa frame 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.

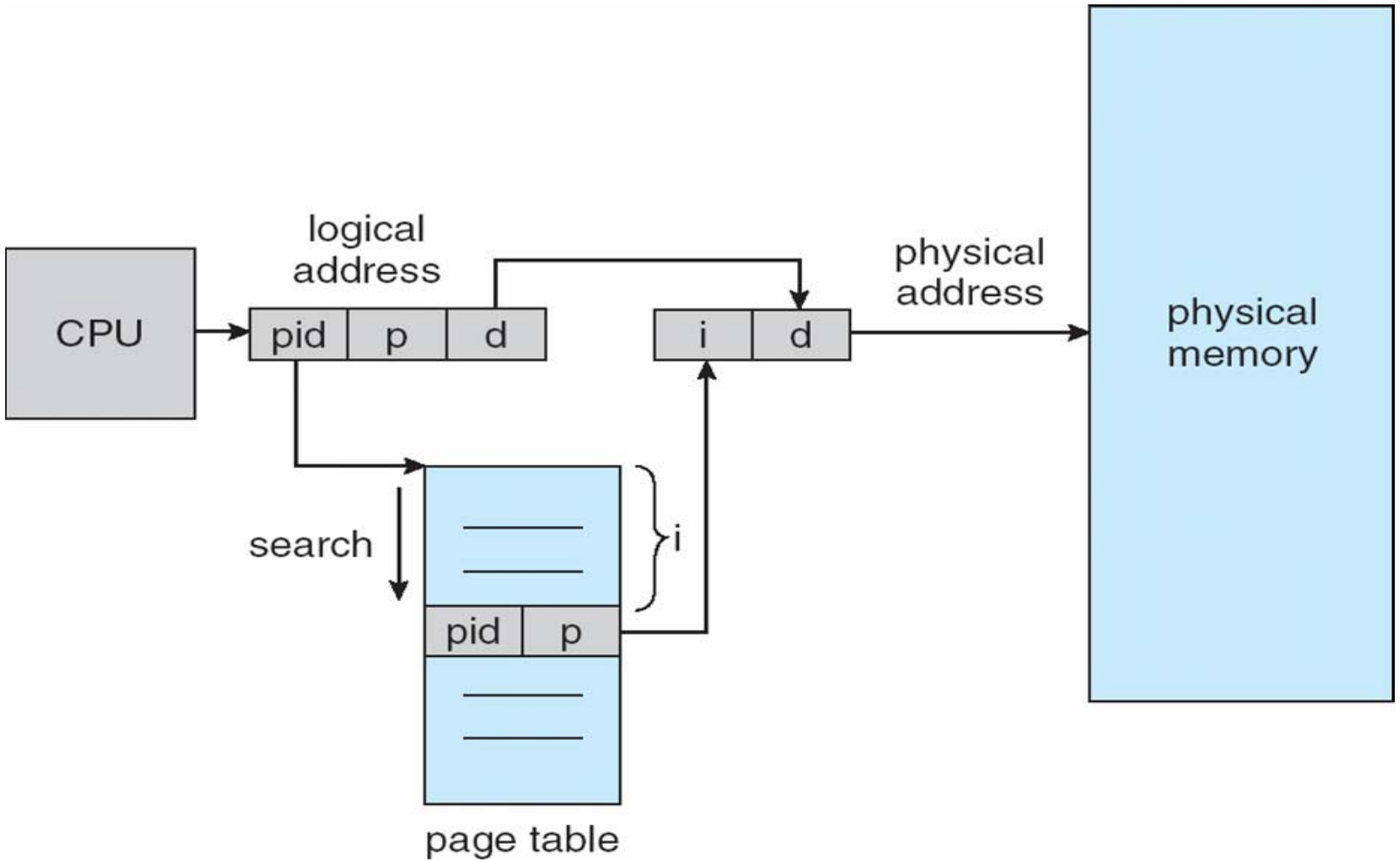
Hashed Sayfa Tablosu



Inverted Sayfa Tablosu

- Rather than each process having a page table and keeping track of all possible logical pages, track all physical pages
- Belleğin her gerçek sayfası için bir girdi
- Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page
- Gerekli bellek miktarını azaltmak için her sayfa tablosu depolanmalıdır, fakat bir sayfa talebi olduğunda tablo aramak için gereken zaman artar.
- Use hash table to limit the search to one — or at most a few — page-table entries
  - TLB erişimi hızlandırabilir.
- Fakat paylaşımlı bellek nasıl uygulanabilir?
  - Paylaşılan fiziksel adres için sanal bir adres bilgisi tutularak.

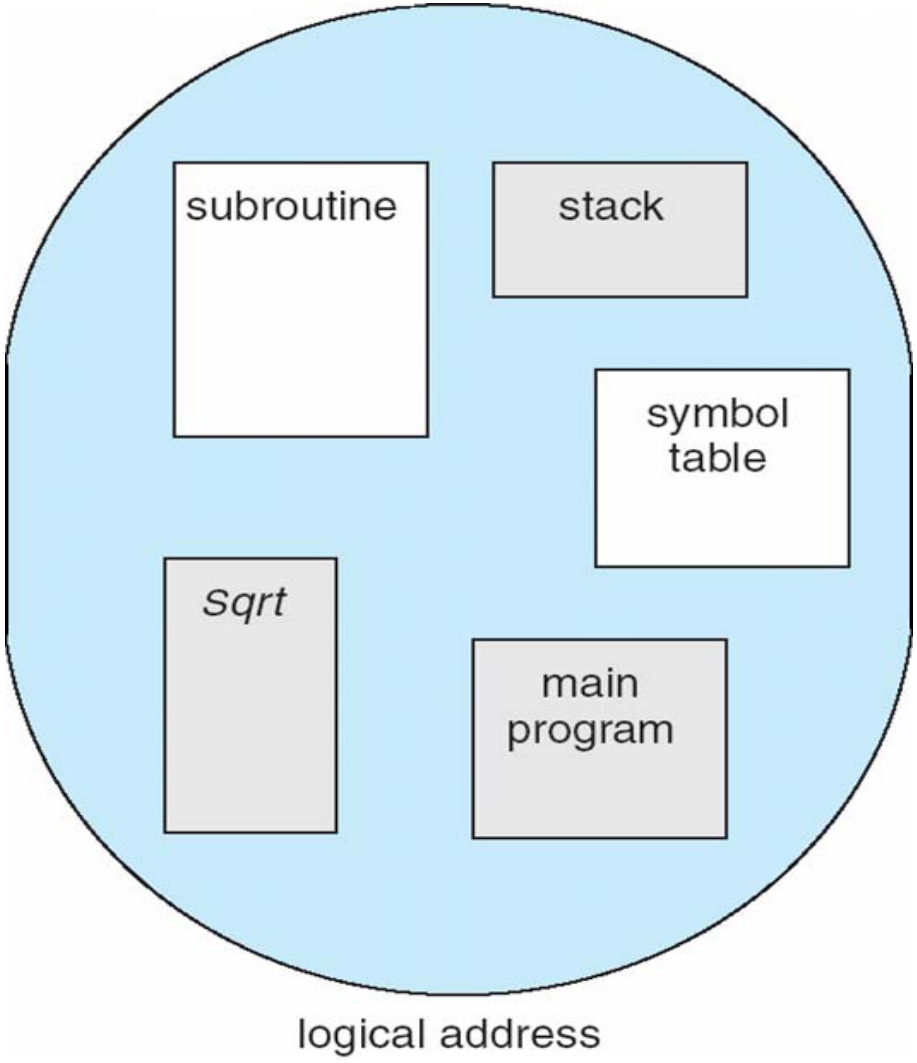
Inverted Page Table Architecture



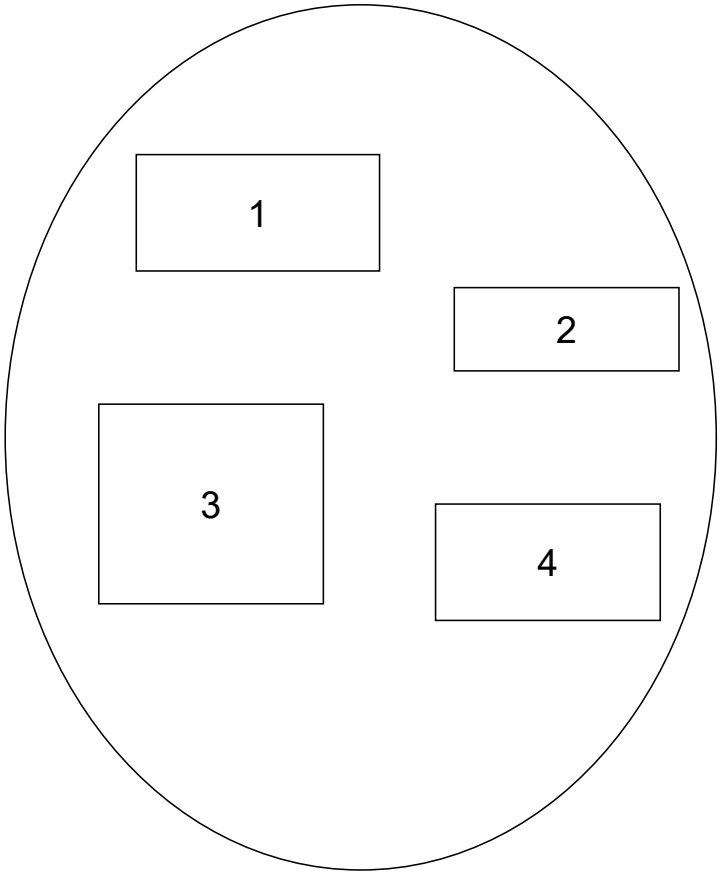
Segmentasyon

- Bellek yönetim şeması, kullanıcının belleği görebilmesini destekler.
- 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

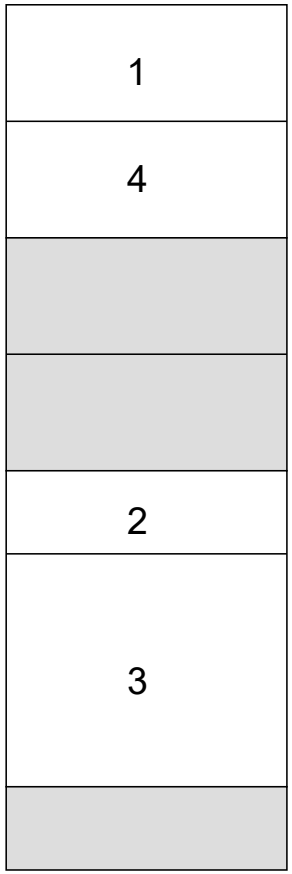
Bir Programın Kullanıcı Görünümü



Mantıksal Segmentasyon Görünümü



user space



physical memory space

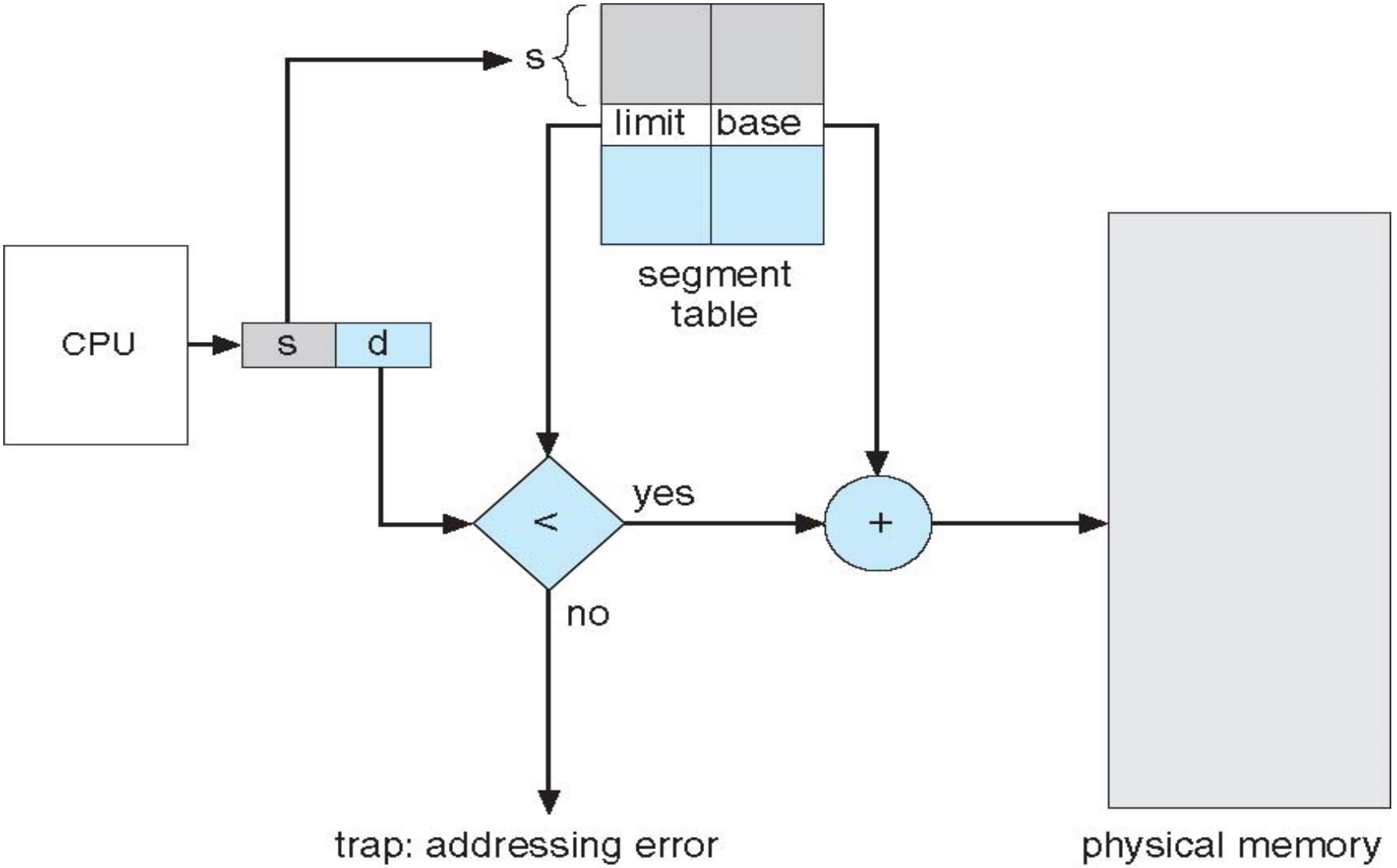
Segmentasyon Mimarisi

- Mantıksal adres iki bölümden oluşur:
  <segment-numarası, ofset>,
- **Segment table (Segment Tablosu)**– iki boyutlu fiziksel adresleri haritalar; her tablonun şu girdileri 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.

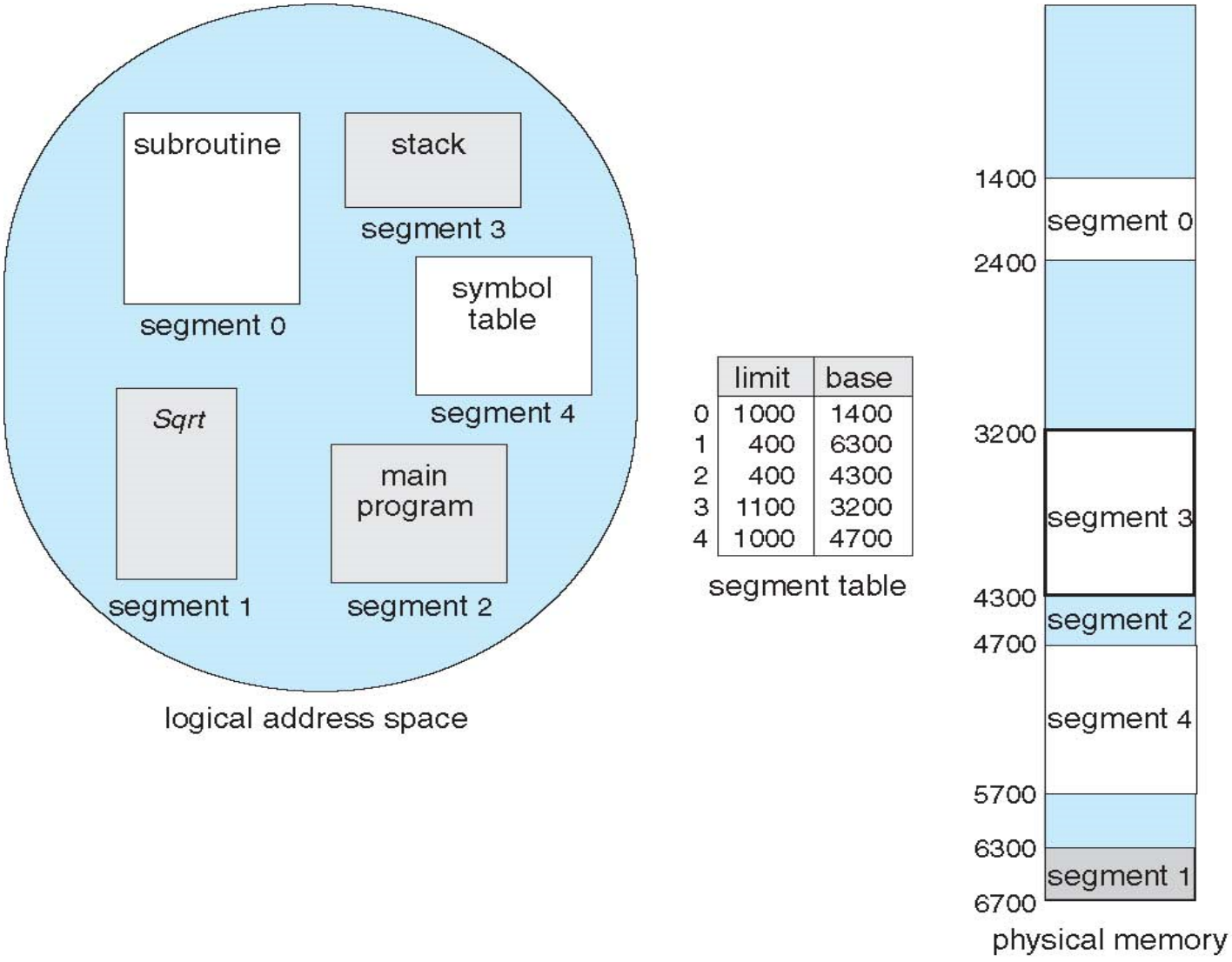
Segmentasyon Mimarisi (Devam)

- Koruma
  - With each entry in segment table associate:
    - Doğrulama bit = 0 $\Rightarrow$ illegal segment
    - okuma/yazma/çalıştırma ayrıcalıkları
- Koruma biti segmentler ile ilgilidir; kod paylaşımı segment düzeyinde gerçekleşir.
- Segment uzunlukları değişebildiğinden bu yana, bellek tahsisi bir dinamik depolama-tahsis problemidir.
- Bir segmentasyon örneği aşağıdaki diyagramda gösterilmiştir.

Donanım Segmentasyonu



Segmentasyon Örneği

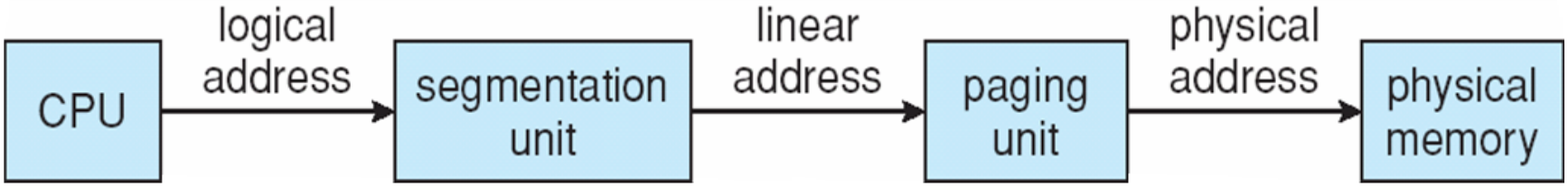


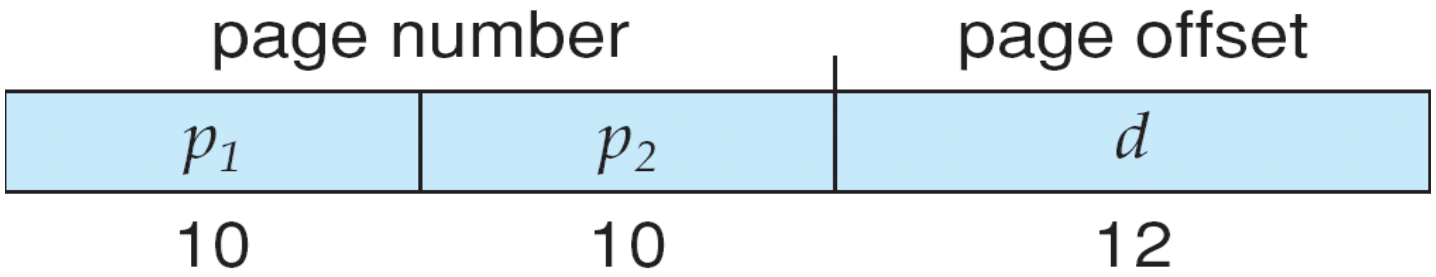
Örnek: The Intel Pentium

- Hem segmentasyonu hem de sayfalama ile segmentasyonu destekler.
  - Her segment 4 GB olabilir.
  - Process başına en fazla 16K kadardır.
  - İki bölüme ayrılmıştır.
    - 8 K kadar olan ilk bölüm segmentleri işlemeye özeldir. (**local descriptor table LDT** 'de tutulur.)
    - 8 K kadar olan ikinci bölüm tüm processler arasında paylaşılır.(**global descriptor table GDT** 'de tutulur.)

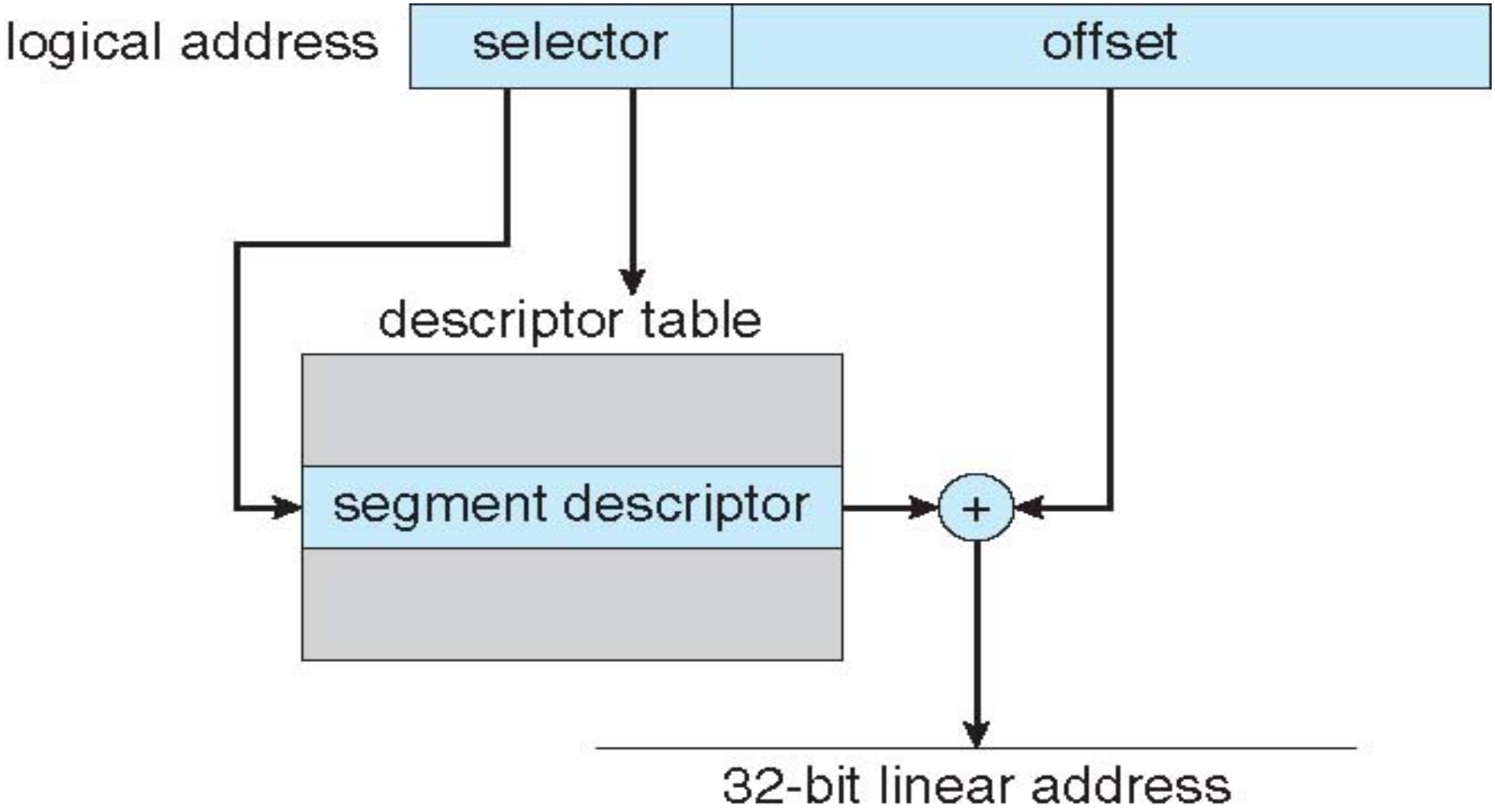
- CPU mantıksal adresi oluşturur.
  - Segmentasyon birimine verir.
    - Doğrusal adresler üretilir.
  - Doğrusal adres sayfalama birimine verilir.
    - Ana bellekte fiziksel adres üretir.
    - Sayfalama birimleri eşdeğer MMU oluşturur.
    - Sayfaların boyutları 4 KB ya da 4 MB olabilir.

Pentium'da Mantıksal Adresin Fiziksel Adrese Dönüştürülmesi

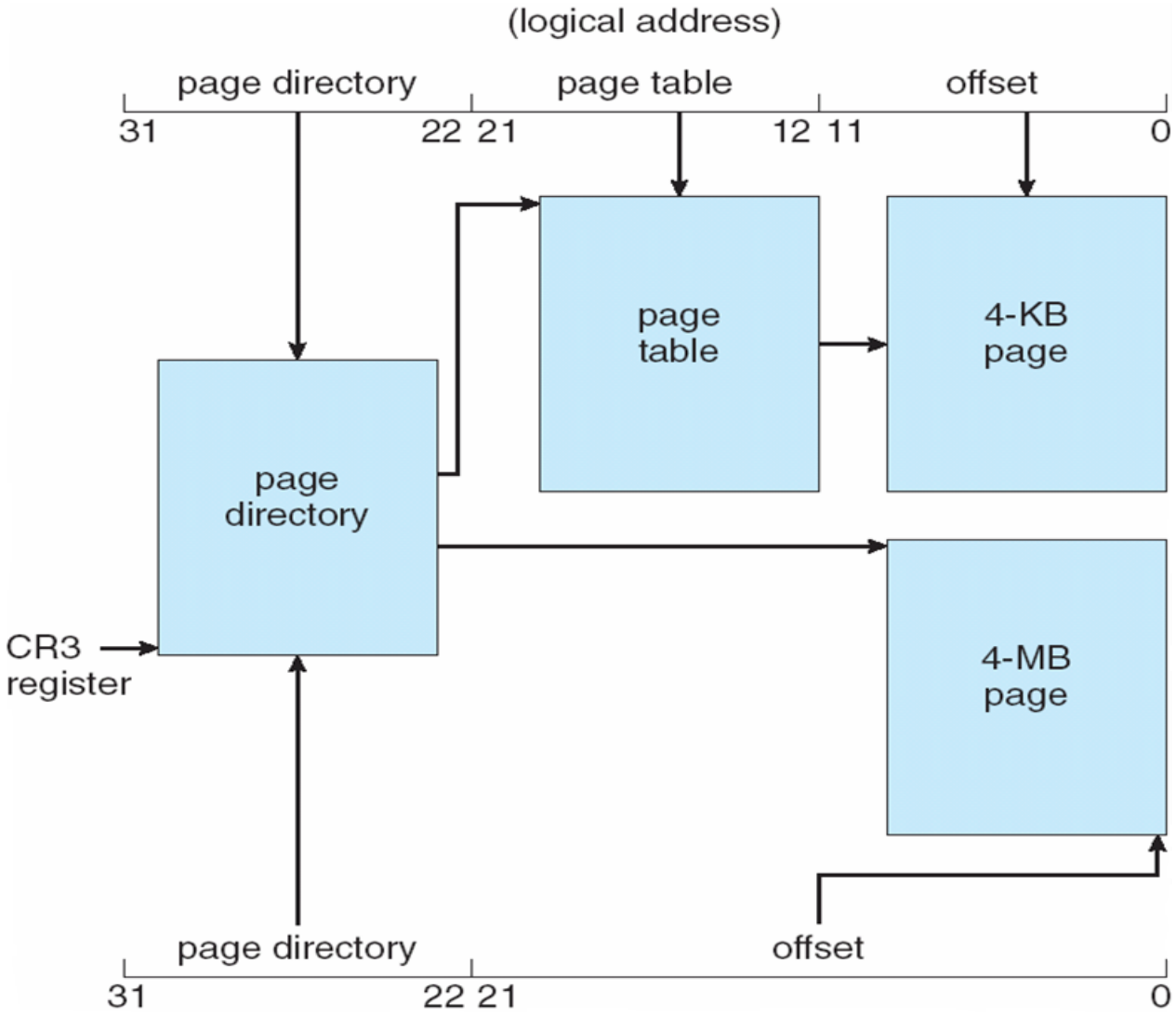




Intel Pentium Segmentasyonu



Pentium Sayfalama Mimarisi

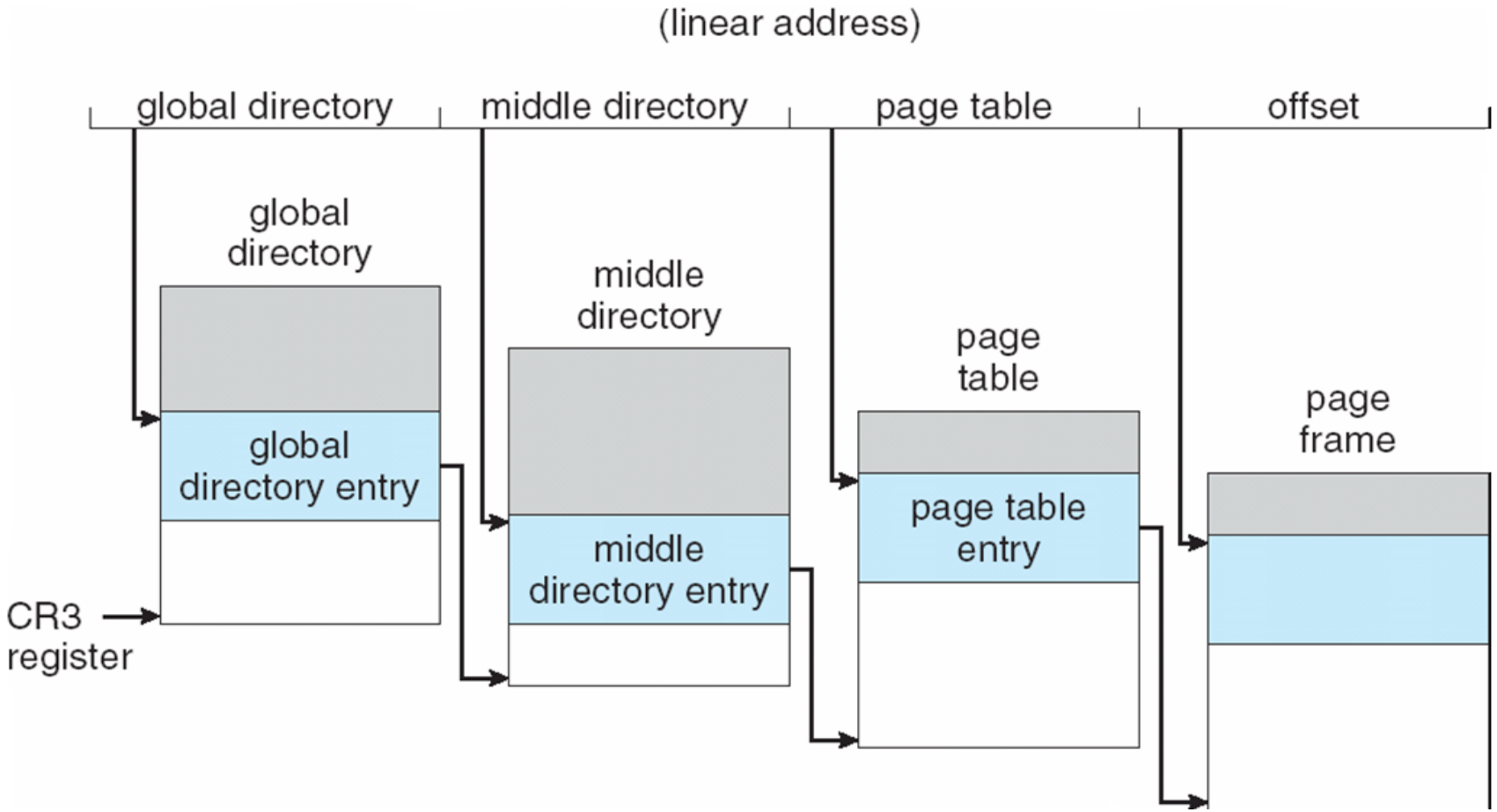


Linux'ta Doğrusal Adres

- 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-bir 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