Bölüm 3: Bellek Yönetimi

İşletim Sistemleri

Bellek Yönetimi

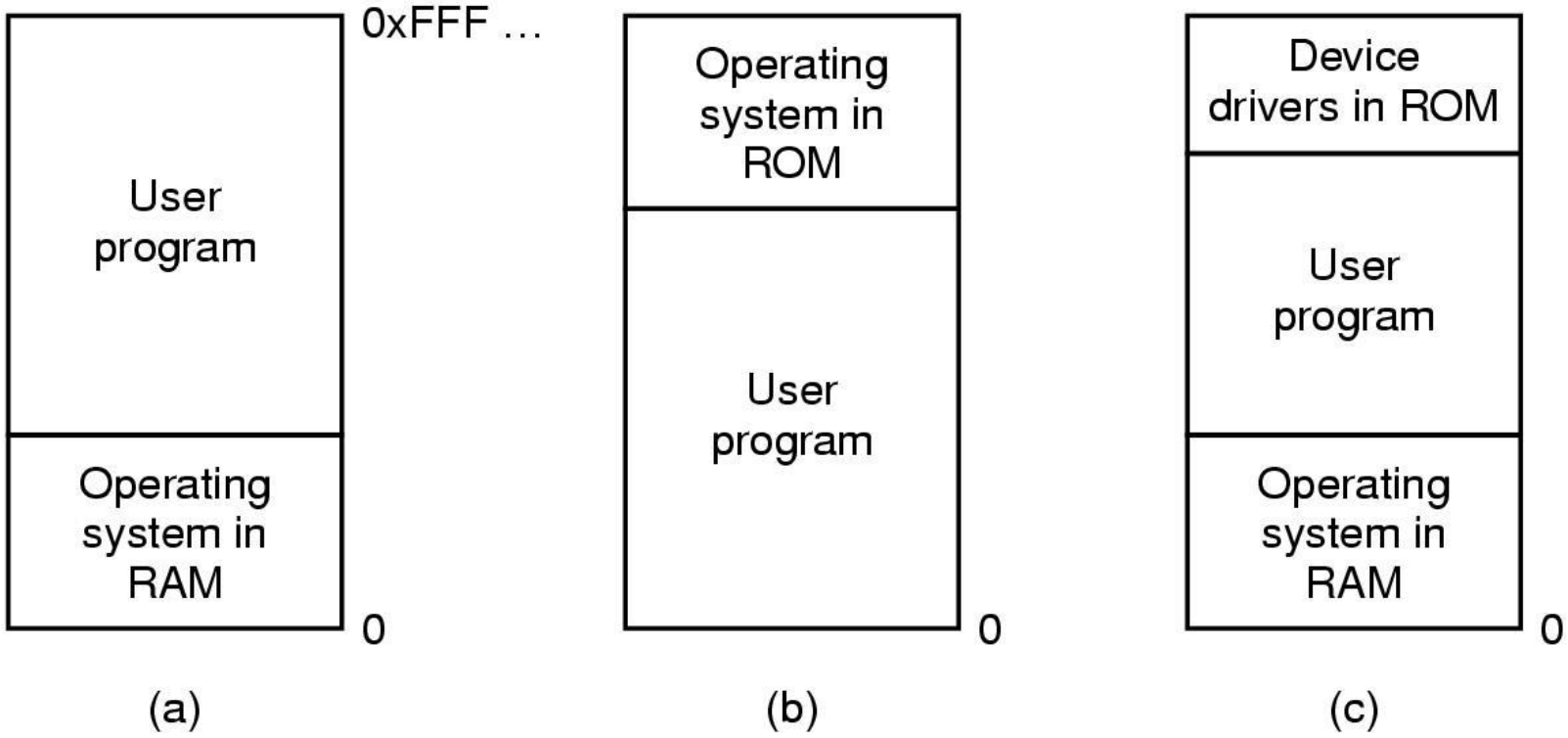
- Bellek (RAM) önemli ve nadir bulunan bir kaynaktır
  - Programlar, genişleyerek kendilerine sunulan belleği doldururlar
- Programcının istediği
  - Bellek gizli, sonsuz büyük, sonsuz hızlı, ve kalıcı olmalıdır...
- Gerçekte olan
  - İnsanların aklına gelen en iyi şey: bellek hiyerarşisi
  - Yazmaç, önbellek, bellek, disk, teyp
- Bellek yöneticisi
  - Belleği verimli bir şekilde yönetir
  - Boşalan bellek alanlarını takip eder, programlara bellek tahsis eder

Soyutlama Yapılmadığında

- Eski anabilgisayar, mini bilgisayarlar, kişisel bilgisayarlarda bellek soyutlaması yoktu…
  - MOV REGISTER1, 1000
  - fiziksel bellek adresi 1000'in içeriğini yazmaca taşır
- Bellekte aynı anda iki program yer alamaz

Soyutlama Yapılmadığında

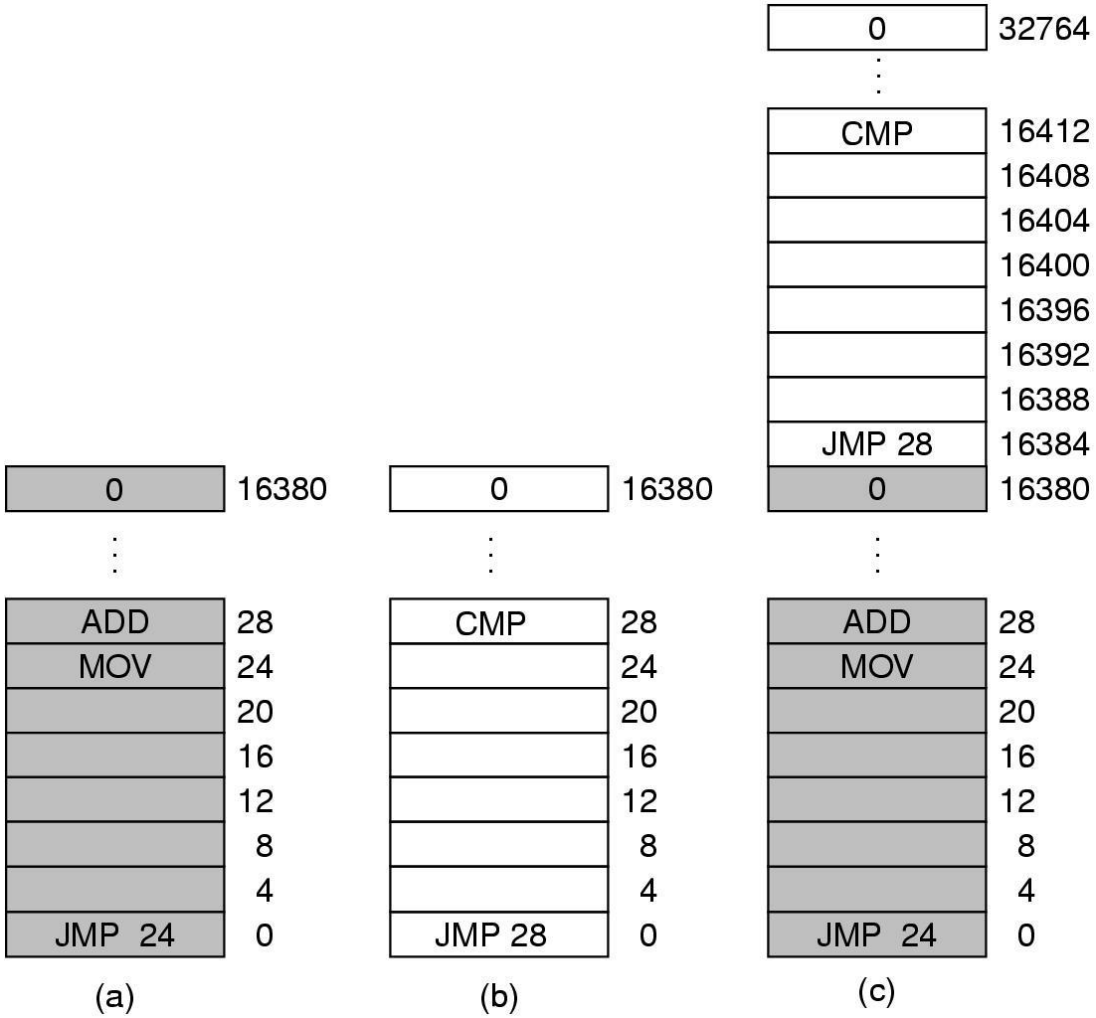
- İşletim sistemi ve bir süreç ile belleğin düzenlenmesi



Soyutlama Yapılmadığında Problemler

- IBM 360
- 2 KB bloklara bölünmüş bellek ve her biri 4 bit koruma anahtarına sahip
- PSW 4 bitlik bir koruma anahtarına sahip
- Donanım, PSW anahtarından farklı bir koruma koduyla belleğe erişme girişimlerini yakalar.

Yer Değiştirme Problemi



Soyutlama Olmamasının Dezavantajı

- Sorun, her iki programın da mutlak fiziksel belleğe referans vermesidir.
- İnsanlar mahrem bir alana, yani yerel adreslere sahip olabilmek isterler.
- IBM 360
  - İkinci program belleğe yüklenirken adresler değiştirilir
  - Statik yer değiştirme
    - 16384'e bir program yüklenirken, her adrese bir sabit değer eklenir.
    - Yüklemeyi yavaşlatır, ek bilgi gerektirir
- Gömülü ve akıllı sistemlerde soyutlama olmadan bellek yönetimi var

Soyutlama: Adres Uzayı

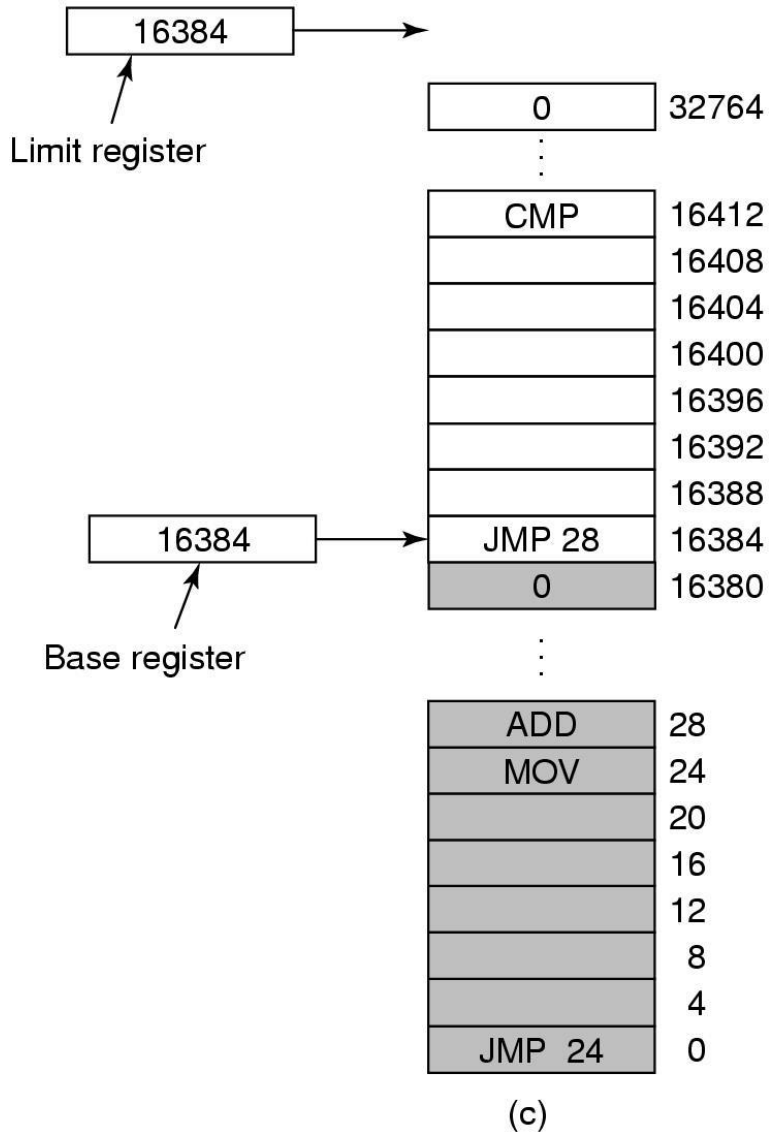
- Fiziksel adresi programcılara gösterme (not expose)
  - işletim sistemi çökmesi
  - Paralelleştirmek zor
- Çözülmesi gereken iki problem:
  - Koruma
  - yer değiştirme
- Adres alanı:
  - Belleği adreslemek için bir dizi bellek süreci kullanılabilir
  - Her sürecin birbirinden bağımsız kendi adres alanı vardır.
  - Nasıl?

Dinamik Yer Değiştirme

- İşlemciye iki özel yazmaç:
  - taban ve limit
- Program ardışık bir boşluğa yüklenecek
- Yükleme sırasında yer değiştirme yok
- Süreç çalıştırıldığında ve bir adrese referans verildiğinde, CPU otomatik olarak limit değerini aşıp aşmadığını kontrol ederek taban değerini o adrese ekler

Taban ve Limit Yazmaçları

- Her bellek erişiminde bir ekleme ve karşılaştırma yapılması gerekiyor

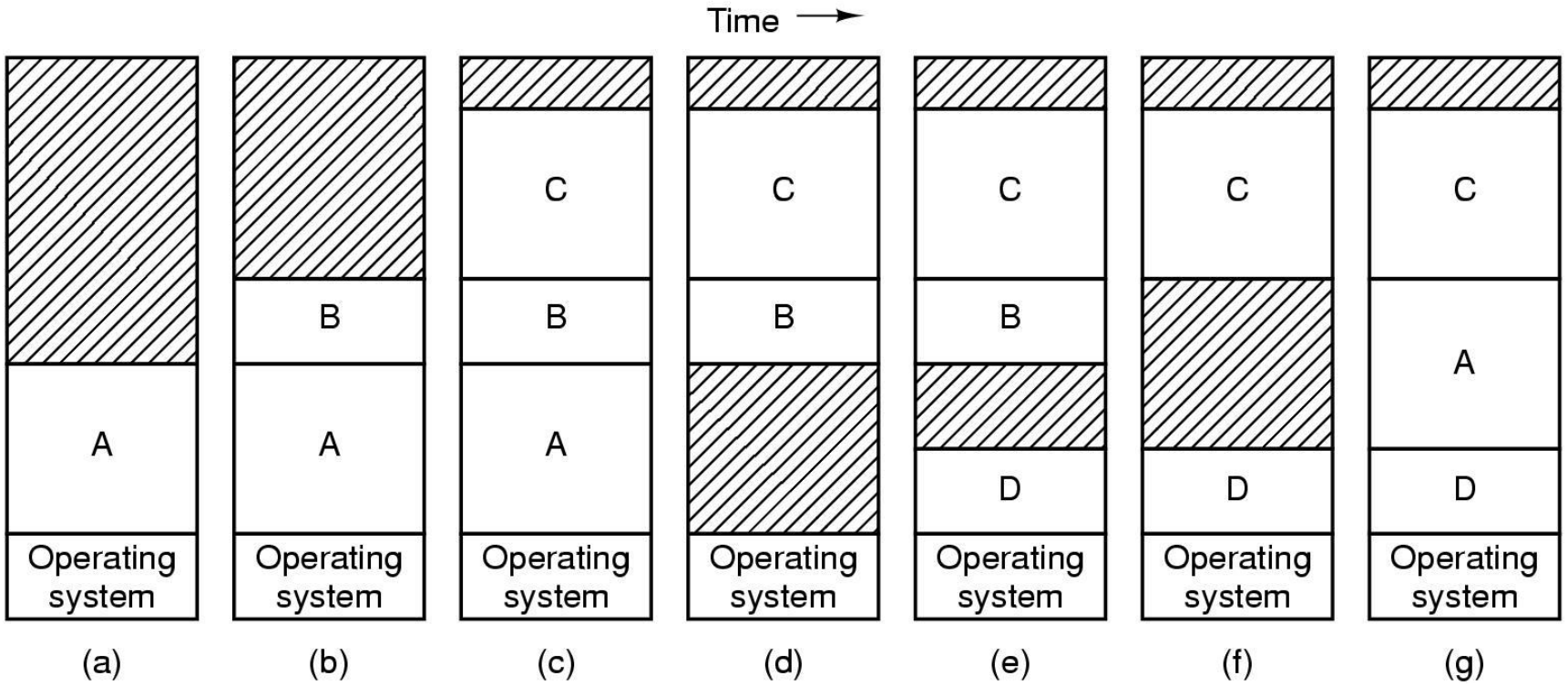


Değiş Tokuş Yapmak (swap)

- Sistemde arka planda çalışan bir çok sunucu süreci vardır
- Fiziksel bellek tüm programları tutacak kadar büyük değildir
  - Değiş tokuş
    - Programları belleğe getir, değiş tokuş yapıp götür
  - Sanal bellek
    - Programları kısmen bellekte olsalar bile çalıştır

Bellek Düzeni Değişimi

- Süreçler belleğe girip çıktıkça bellek tahsisi değişir. Gölgeli bölgeler kullanılmayan bellektir.

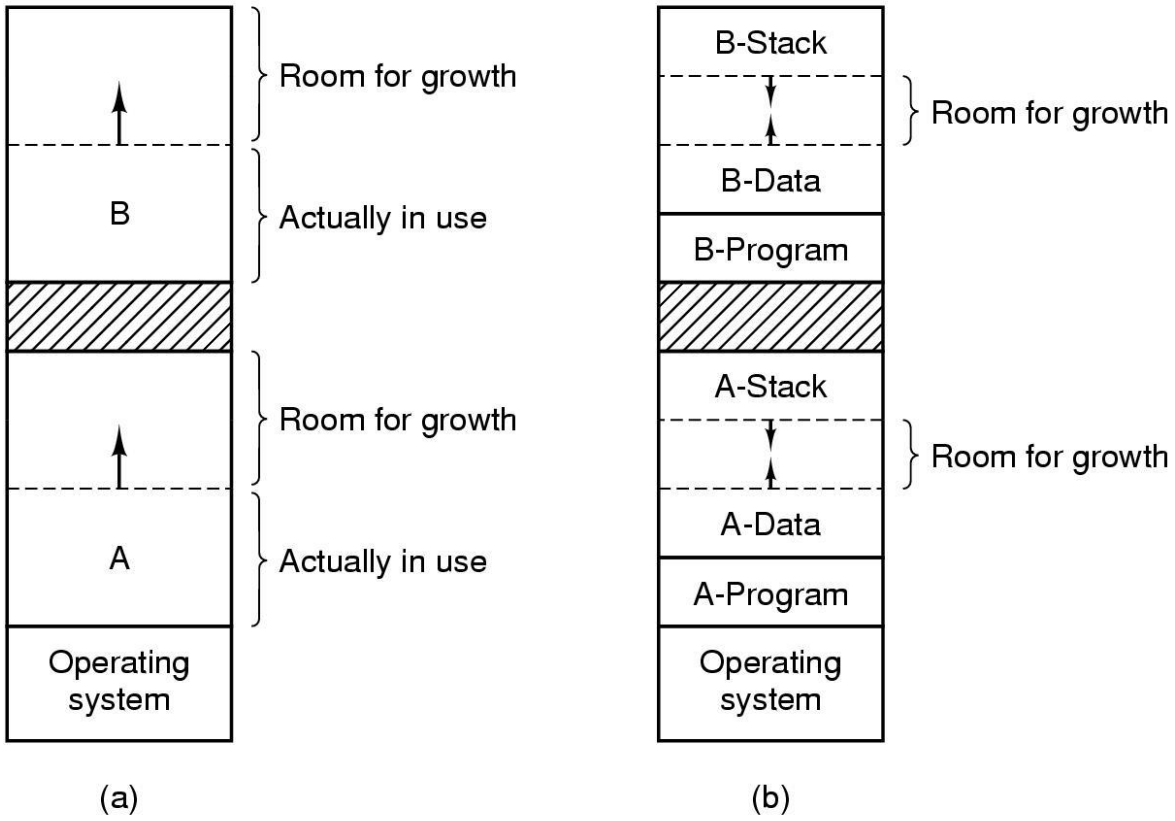


Problemler

- Giriş ve çıkış takası sonrası farklı adresler
  - Statik yer değiştirme/dinamik yer değiştirme
- Bellek delikleri (hole)
  - Bellek sıkıştırma
    - İşlemci zamanı gerekir
    - 20 ns'de 4 byte taşır, ardından 1 GB'ı sıkıştırmak için 5 saniye
- Bir program için ayrılan bellek miktarı ne kadar?
  - Programlar büyüme eğilimindedir
  - Hem veri kesimi (segment) hem de yığın

Bellekte Alan Tahsisi

- (a) Büyüyen veri segmenti için. (b) Büyüyen yığın, ve veri kesmi için.

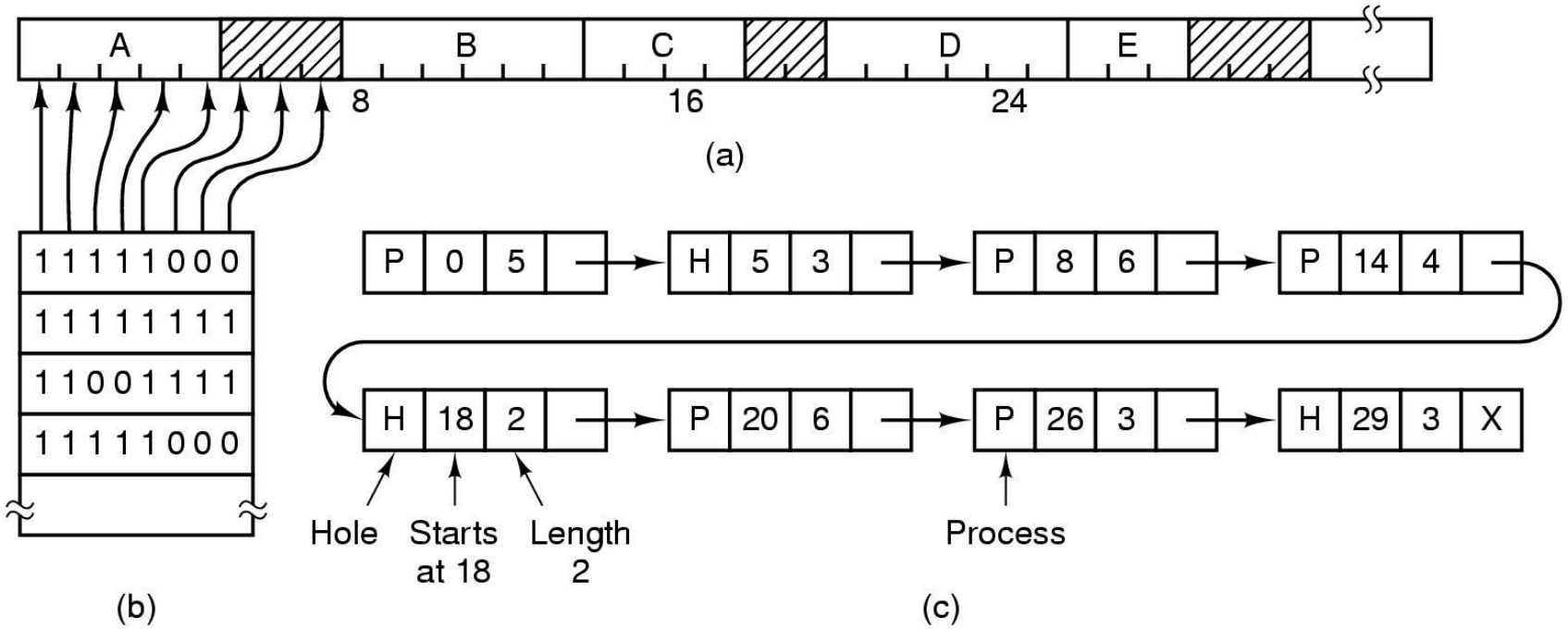


Bellekte Boş Alan Yönetimi

- Biteşlem (bitmap) ve bağlı listeler
- Biteşlem
  - Bellek, ayırma birimlerine bölünmüştür (birkaç sözcükten KB boyutuna kadar)
  - Her birime karşılık gelen, biteşlem'de bir bit var
  - İstenen uzunlukta boş alan bulmak zor

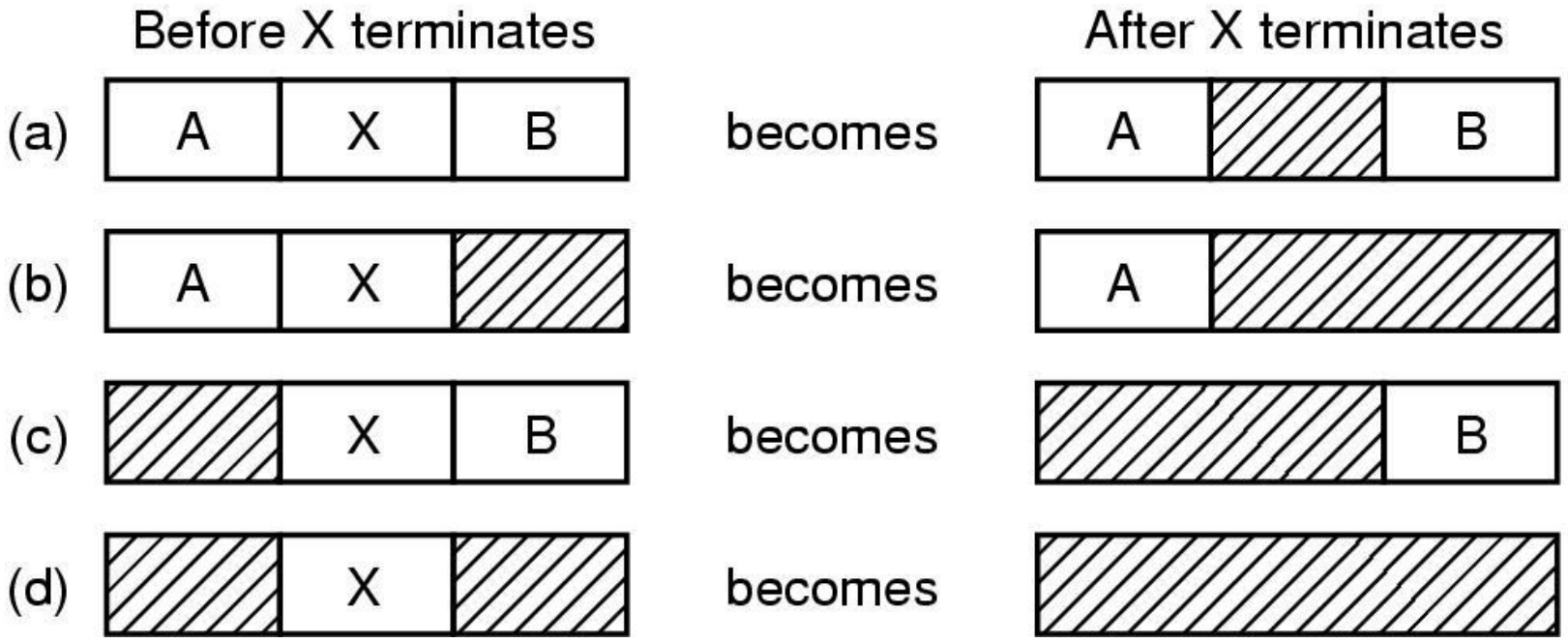
Biteşlem ile Bellek Yönetimi

(a) Belleğin bir bölümü, beş işlem ve üç delik var. İm işaretleri, bellek ayırma birimlerini gösterir. Gölgeli bölgeler (bit eşlemde 0) boştur. (b) ilgili bit eşlem. (c) bağlı liste gösterimi.



Bağlı Liste ile Bellek Yönetimi

- X Sürecini sonlandırdıktan sonra oluşan bellek düzeni.



Boş Alan Yönetimi – Bağlı Liste

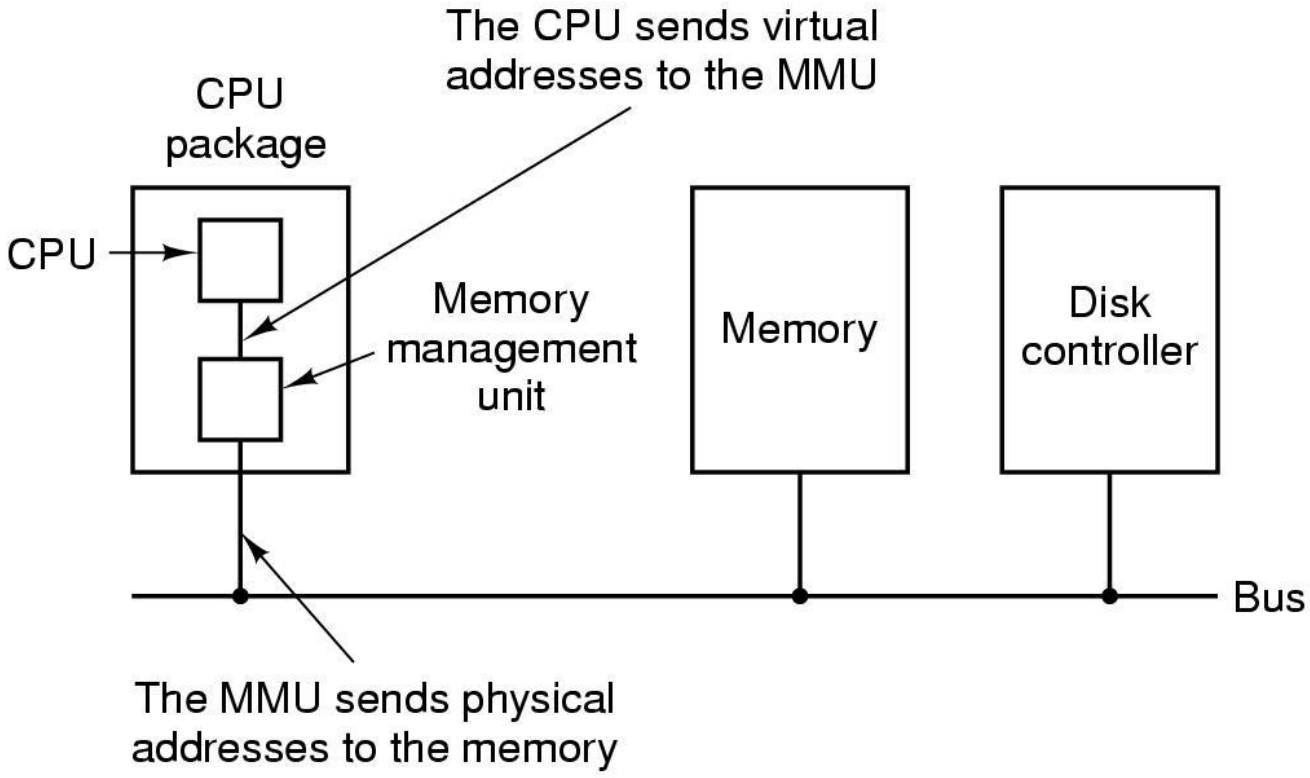
- Çift bağlı liste
- Programlara boş hafıza nasıl tahsis edilir?
  - İlk uyan (first fit)
    - hızlı; başlangıç daha sık kullanılır; büyük bir boş alanı kırmak
  - Sonraki uyan (next fit)
    - Her seferinde en son kullanılan yerden
  - En iyi uyan (best fit)
    - Tüm listeyi arayıp, gerekli boyuta yakın deliği bulan
  - En kötü uyan (worst fit)
    - En büyük deliği bulur
  - Hızlı uyan (quick fit)
    - İşlemler ve delikler için ayrı kuyruklarda tutulur

Sanal Bellek – Şişkin (bloat) Yazılım Yönetimi

- Programların belleğe sığmayacak kadar büyük olduğu yerler
- Programlar tarafından bölünmek kötü bir fikir
- Sanal bellek
  - Her programın kendi adres alanı vardır
  - Adres alanı, sayfa adı verilen parçalara bölünmüştür.
  - Her sayfa bitişik bir alandır ve fiziksel adresle eşlenir
  - Tüm sayfaların fiziksel bellekte olması gerekmez.
  - İşletim sistemi, sayfa adreslerini ve fiziksel adresleri ihtiyaç anında eşler
  - Gerekli bir sayfa bellekte olmadığında, işletim sistemi halleder
  - Her sayfa değiş tokuşa ihtiyaç duyar

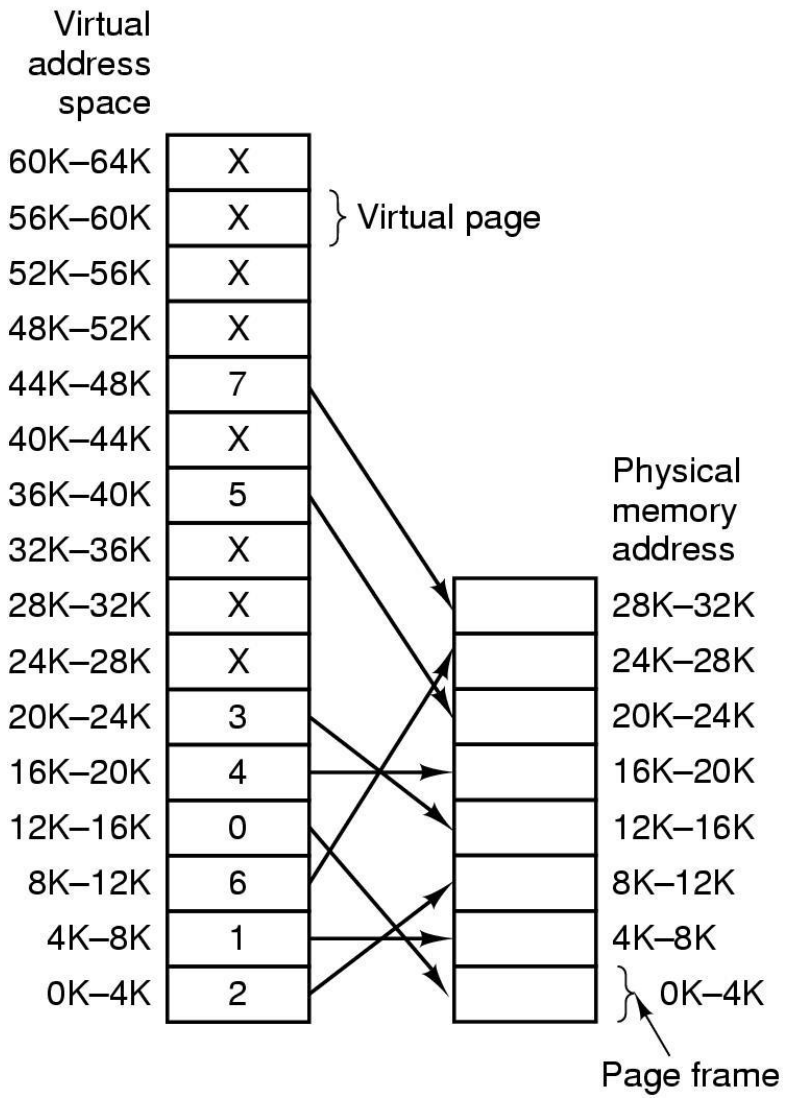
Sanal Bellek - Sayfalama

- MMU'nun konumu ve işlevi – işlemcinin bir parçası olarak gösterilir



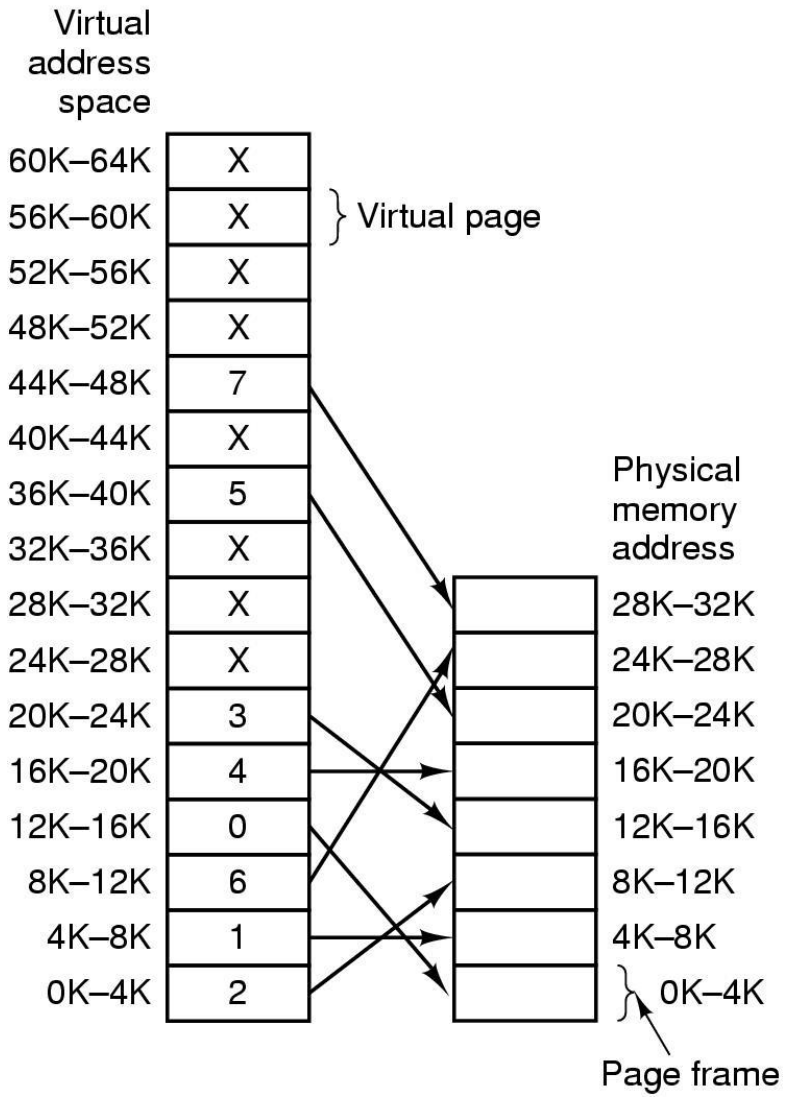
Sanal Bellek – Sayfa Tablosu

Sayfa tablosu (page table) sanal ve fiziksel bellek adresleri arasındaki ilişkiyi tutar.



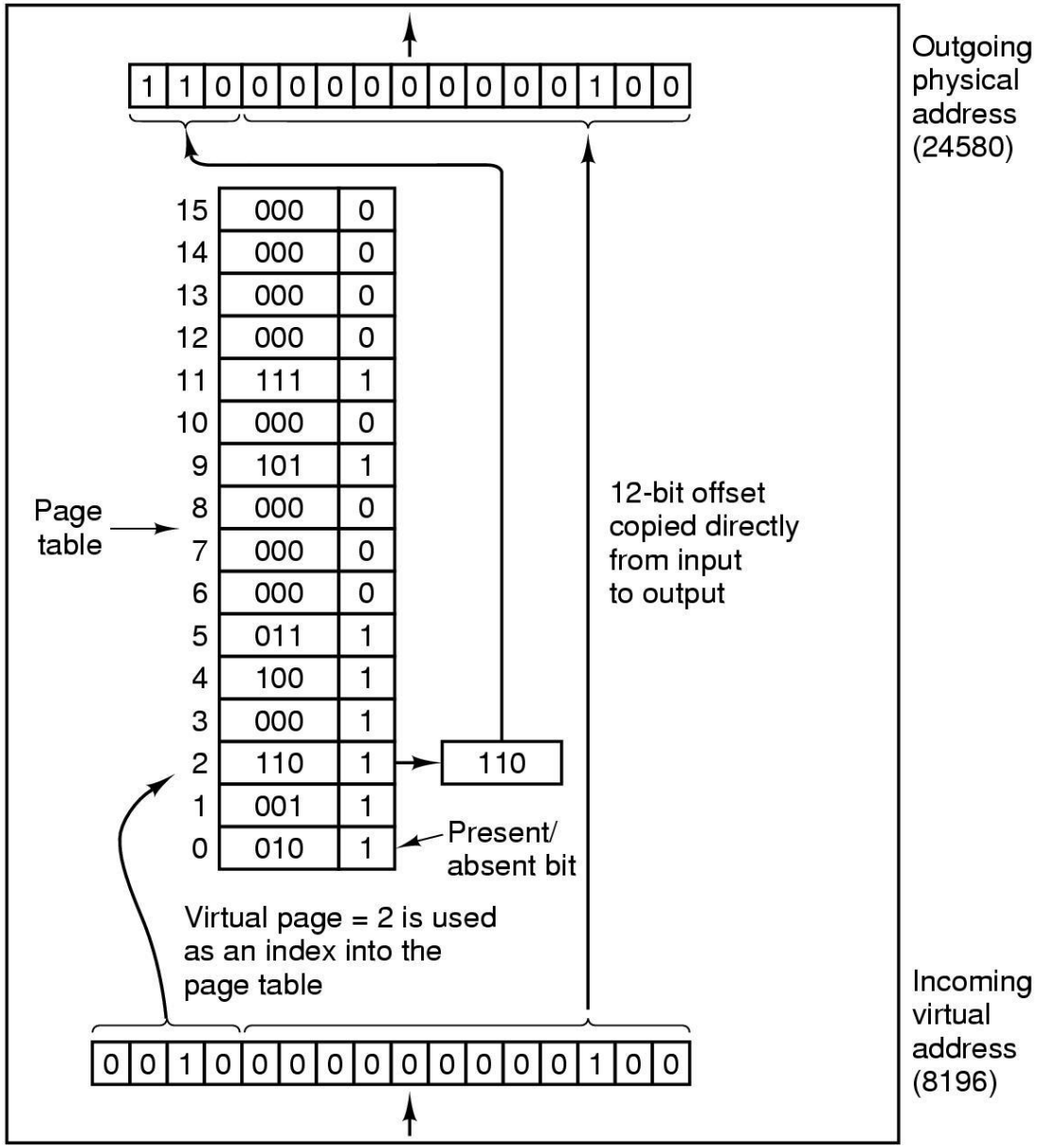
Bellek Yönetim Birimi

- MMU(memory management unit)
  - CPU: MOV REG, 0
  - MMU: MOV REG, 8192
  - CPU: MOV REG 8192
  - MMU: MOV REG 24567
  - CPU:MOV REG 20500
  - MMU:MOV REG 12308
  - CPU: MOV REG 32780
  - MMU: page fault



Bellek Yönetim Birimi

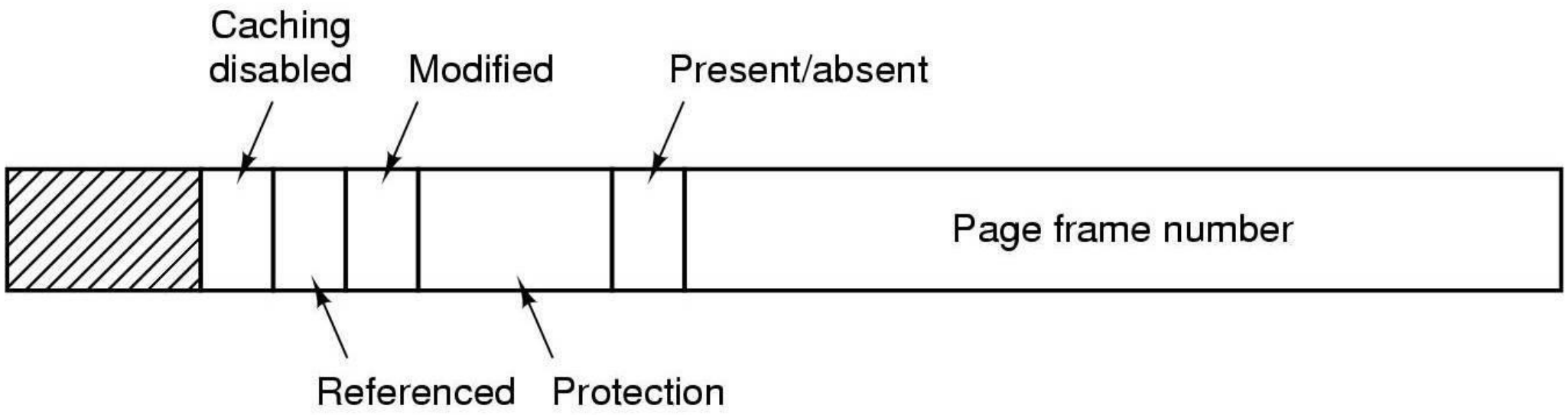
- 16 adet 4 KB sayfalı MMU'nun dahili çalışması



Sanal Adres Eşleme

- Sanal adres, sanal sayfa numarası ve ofset
- 16 bit adres: 4 KB sayfa boyutu; 16 sayfa
- Sanal sayfa numarası: sayfa tablosundaki indis (index)
- Sayfa tablosunun amacı
  - Sanal sayfaları sayfa çerçevelerine eşleme

Sayfa Tablosu Elemanı Yapısı



Sayfa Tablosu Yapısı

- Koruma
  - Ne tür erişimlere izin verilir?
- Değiştirilmiş:
  - Bir sayfa yazıldığında (kirli)
- Erişilen:
  - Bir sayfa referans alındığında
- Önbellek devre dışı bırakma
  - Veri tutarsızlığı

Sayfalama Uygulama Sorunları

- Sanal adresten fiziksel adrese eşleme hızlı olmalıdır.
- Sanal adres alanı büyükse, sayfa tablosu da büyük olacaktır. (32bit/64bit)
- Her sürecin bellekte kendi sayfa tablosu olmalıdır.

Sayfalamayı Hızlandırma

- Sayfa tablosunu yazmaçta tutmak?
  - Süreç koşarken bellek erişimi gerekmez
  - Karşılanmayacak derecede pahalı
- Sayfa tablosunu tamamen bellekte tutmak?
  - Her sürecin kendi sayfa tablosu vardır
  - Sayfa tablosu bellekte tutulur
  - Mantıksal bir bellek erişimi gerçekleştirmek için kaç bellek erişimi gerekir?

Sayfalamayı Hızlandırma

- Etkili bellek erişim süresi, her veri/komut erişimi için gereken süre
  - İki kez bellek erişim süresi; performansı yarı yarıya azaltır
  - Sayfa tablosuna erişin & verilere/komutlara erişin

- Çözüm:
  - İlişkili yazmaçlar (associative registers) veya çeviriye bakma arabellekleri (translation look-aside buffers) (TLB'ler) adı verilen özel hızlı arama yapan donanım önbelleği

Translation Lookaside Buffers

- .

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

TLB

- TLB genellikle MMU'nun içindedir ve az sayıda elemandan oluşur
- Sanal bir adres alındığında
  - MMU öncelikle sanal sayfa numarasının TLB'de olup olmadığını kontrol eder
  - TLB’de ise, sayfa tablosunu ziyaret etmeye gerek yok
  - değilse, TLB'den bir elemanı çıkarır ve yeni elemanı sayfa tablosunda bir eleman ile değiştirir.

Etkili Erişim Süresi

- İlişkili Arama = ε zaman birimi;
- bellek çevrim (cycle) süresi = *t* zaman birimi;
- İsabet oranı = α
- Etkili Erişim Süresi
  - EES = $(t + \varepsilon)\,\alpha + (2t + \varepsilon)(1 - \alpha) = 2t + \varepsilon - t\,\alpha$
- ε(20 ns), *t*(100 ns), α 1(%80), α 2(%98) ise:
  - TLB hit: 20+100=120 ns
  - TLB miss: 20+100+100=220 ns
  - EES1 = 120*0,8 + 220 * 0,2 = 140 ns
  - EES2 = 120*0,98 + 220 * 0,02 = 122 ns

Büyük Bellek için Sayfa Tablosu

- Adres alanı: 32 bit
- Sayfa boyutu: 4 KB
- Sayfa Numaraları: 20 bit, 1 milyon sayfa
- Sayfa eleman başına 32 bit, sayfa tablosunu tutmak için 4 MB
- 64 bit sistem için?

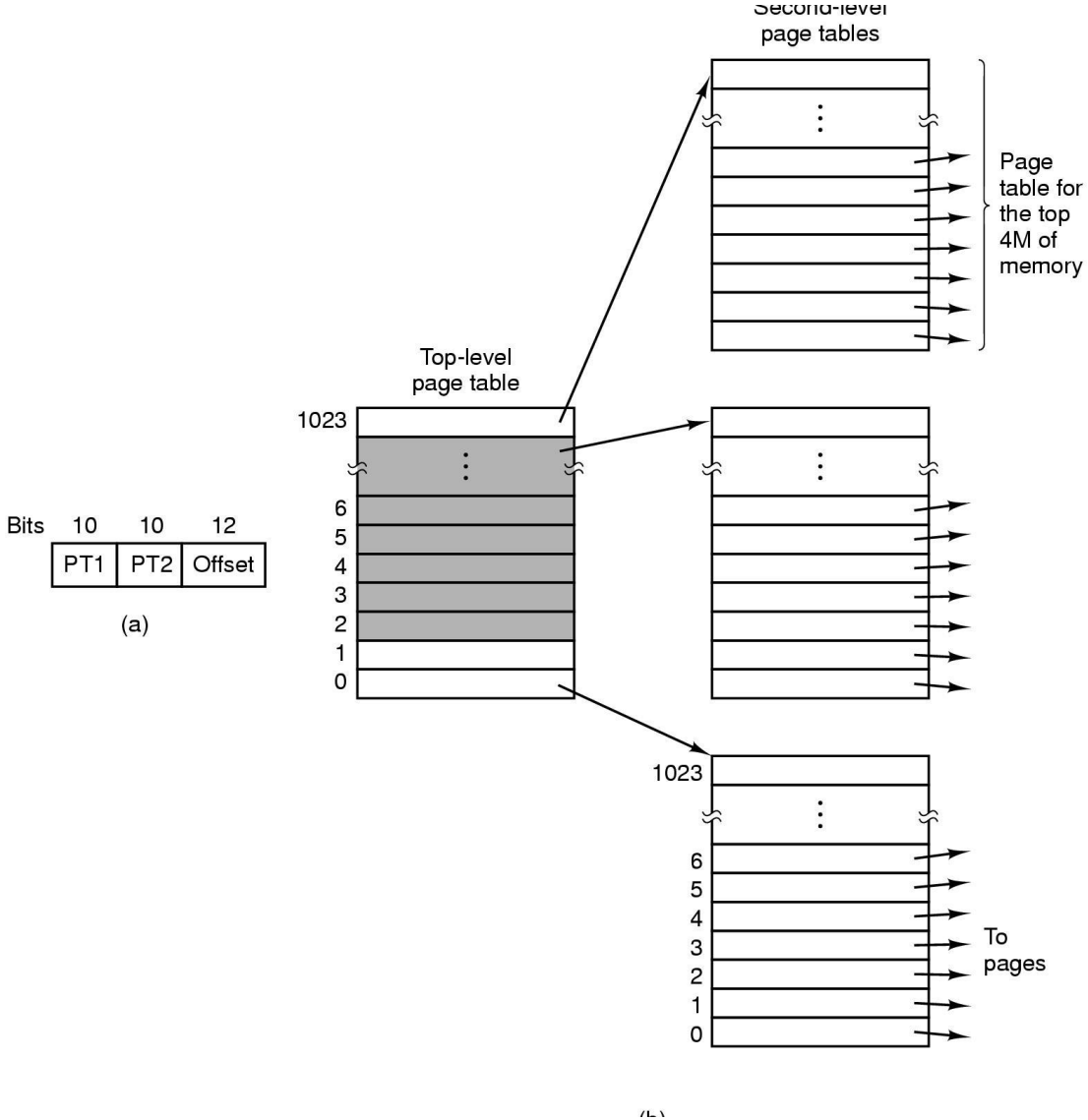
Çoklu Seviye Sayfa Tablosu

- 32 bit sanal bellek üç bölüme ayrılmıştır
  - 10 bit PT1, 10 bit PT2, 12 bit ofset
- Çoklu seviye sayfa tablosu
  - Tüm sayfa tablolarını daima bellekte tutmak gerekmez
  - Sayfa tabloları da sayfalarda saklanır
  - Örnek: bir program 4G adres alanına sahiptir, koşmak için 12M'ye ihtiyaç duyar: 4M kod, 4M veri, 4M yığın için

Çoklu Seviye Sayfa Tablosu

(a) İki sayfa tablo alanına sahip 32 bitlik bir adres.

(b) İki seviyeli sayfa tabloları.

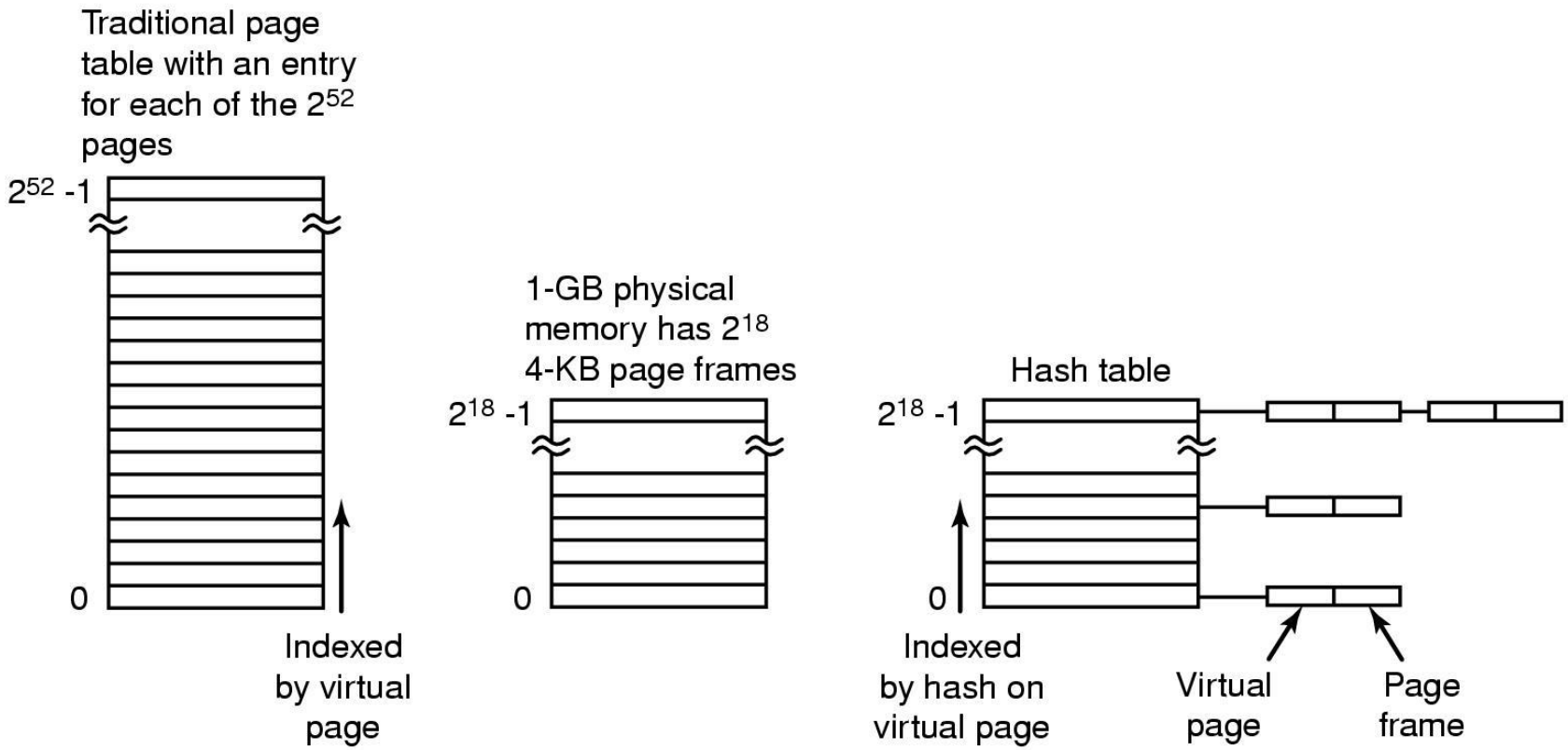


Ters Sayfa Tablosu

- Sanal adres alanları fiziksel bellekten çok daha büyük olduğunda
- Ters sayfa tablosu: sayfa yerine sayfa çerçevesi başına eleman
- Arama çok daha zor
  - TLB
  - Hash
- Ters sayfa tablosu 64 bit sistemlerde yaygındır

Geleneksel ve Test Sayfa Tablosu

- .



Sayfa Değiştirme Algoritmaları

- Optimum (optimal)
- Yakın zamanda kullanılmayan (not recently used)
- İlk Giren İlk Çıkar (first-in first-out)
- İkinci şans (second chance)
- Saat (clock)
- En son kullanılan (least recently used)
- Çalışma kümesi (working set)
- WSClock

Sayfa Hatasının Etkisi

- Sayfa hatası süresi = 25ms (page fault)
- Bellek erişim süresi = 100ns (memory access)
- Sayfa kayıp oranı *p* olsun: (miss rate)
- EAT = $100(1-p) + 25*10^6*p = 100 + 24.999.900*p$
- p = 1/1000 ise, EAT =25.099,9 ns
- EAT < 110ns olması gerekiyorsa, o zaman $100 + 24.999.900*p < 110$ yani $p < 10/24.999.900 < 10/25.000.000 = 1/2.500.000 = 4\times10^{-7}$
- Sayfa hatası oranı p, 4×10^{-7}'den küçük olmalıdır

Sayfa Değişimi

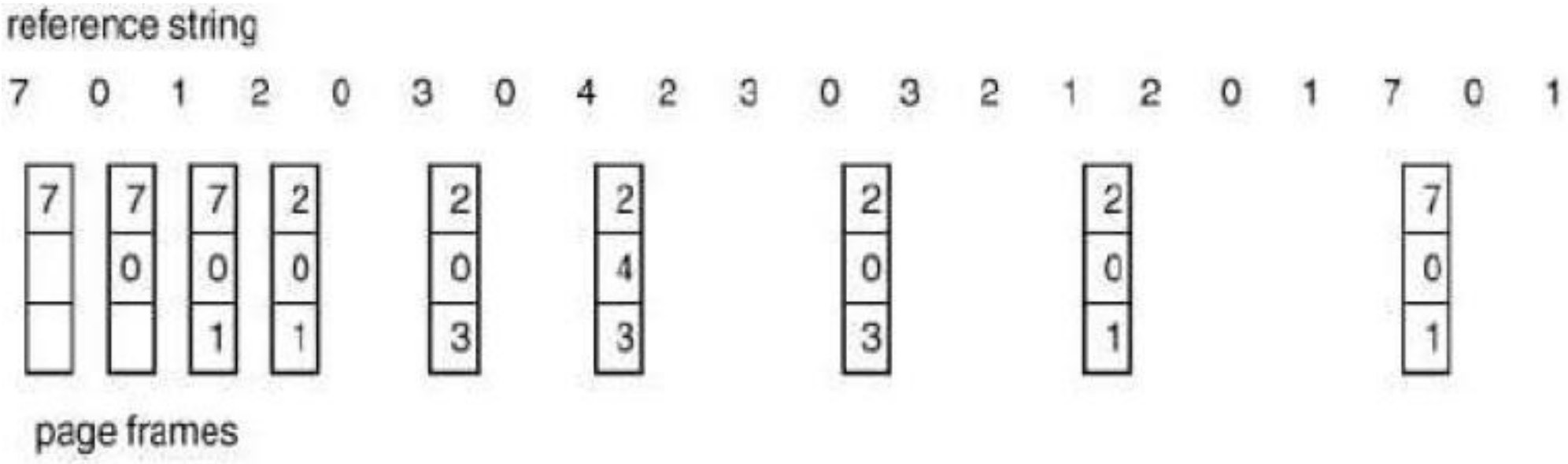
- Bir sayfa hatası oluştuğunda, bazı sayfaların bellekten çıkarılması gerekir.
- Çıkarılacak sayfa bellekteyken değiştirilmişse, diske geri yazılması gerekir.
- Çok kullanılan bir sayfa taşınırsa büyük ihtimalle kısa süre sonra geri getirilecektir.
- Değiştirilecek sayfa nasıl seçilmeli?

Optimum Sayfa Değişimi

- Tarif etmesi kolay ama uygulaması imkansız
- Her sayfa, o sayfaya ilk erişimden önce çalıştırılacak komutların sayısı ile etiketlenir.
- İhtiyaç duyulduğunda en yüksek etikete sahip sayfa kaldırılır
- İşletim sistemi hangi sayfaya ne zaman erişileceğini bilemez
- Ancak; gerçekleştirilebilir algoritmaların performansını karşılaştırmak için kullanılabilir

Optimum Sayfa Değişimi

Sayfa hatası 9 kez, değiştirme 6 kez

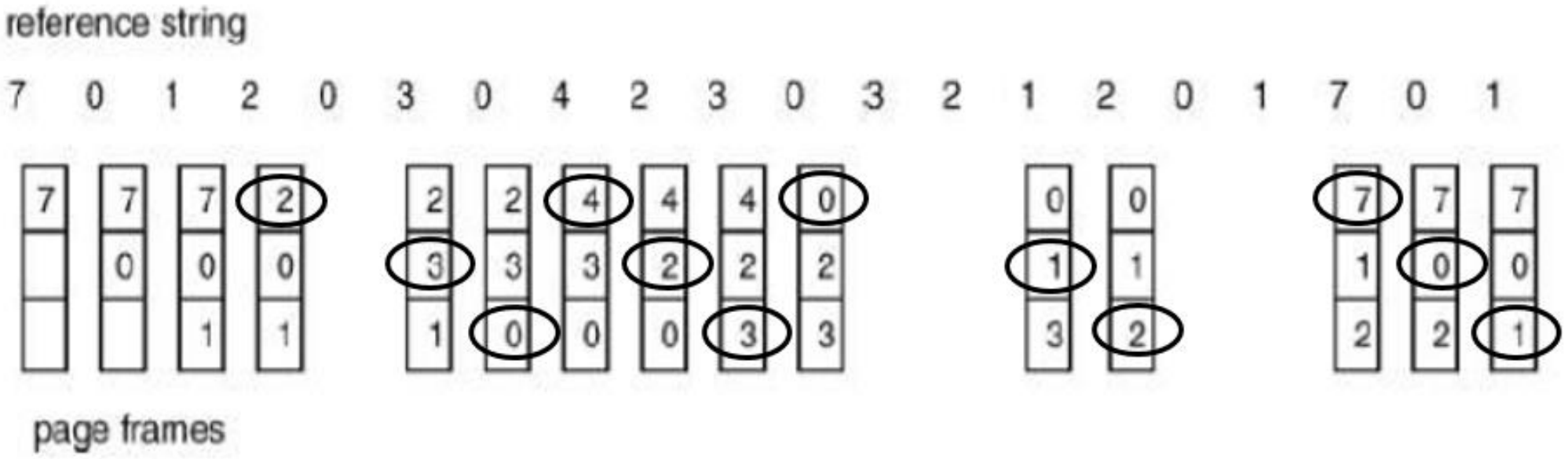


Yakın Zamanda Kullanılmayan Sayfa Değişimi

- Sayfaları kullanımlarına göre değiştirme
- Sayfalarda R ve M biti bulunur (referenced, modified)
- Bir işlem başlatıldığında, her iki sayfa bitine de 0 atanır, periyodik olarak, R biti temizlenir
- Dört farklı durum oluşabilir
  - Sınıf 0: erişilmedi, değiştirilmedi
  - Sınıf 1: erişilmedi, değiştirildi
  - Sınıf 2: erişildi, değiştirilmedi
  - Sınıf 3: erişildi, değiştirildi

İlk Giren İlk Çıkar Sayfa Değişimi

- "En eski olanı" değiştir, Tatmin edici olmayan performans ama basit, Sayfa hatası 15, değiştirme sayısı 12

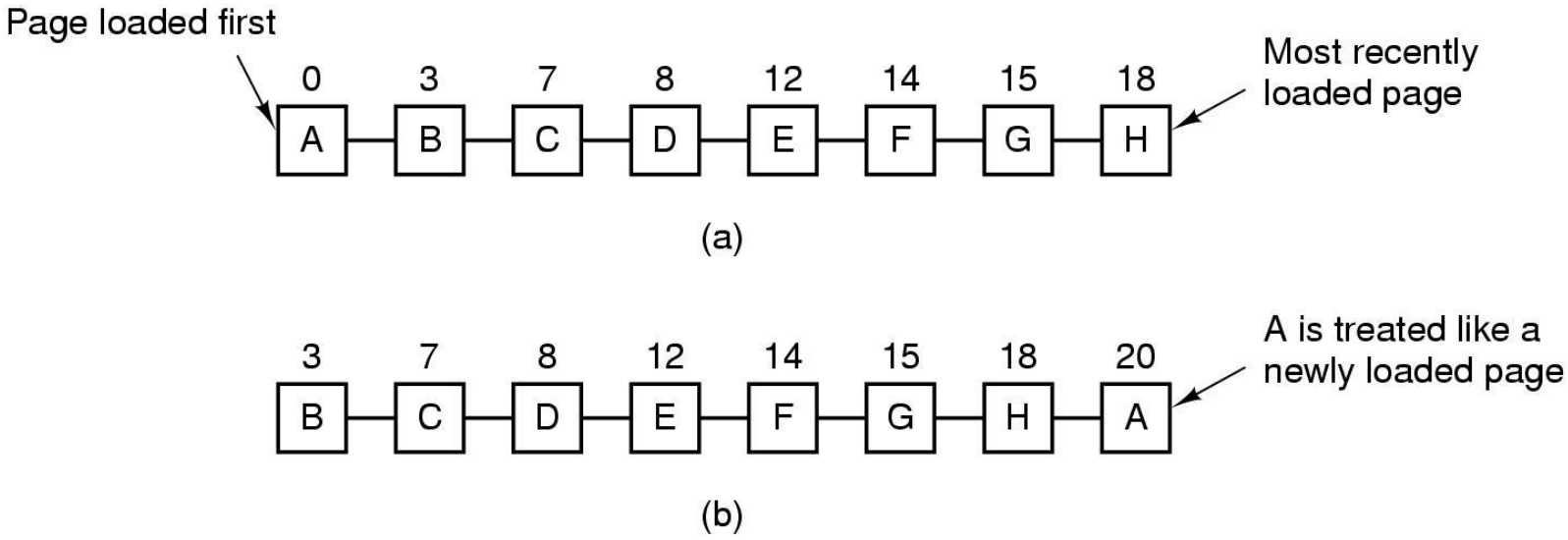


İkinci Şans Sayfa Değişimi

- «İlk giren ilk çıkar» yöntemine basit bir değişiklik yapılarak çok kullanılan bir sayfanın değiştirilmesi önlenmiştir
  - R biti incelenir
  - Eğer R değeri 0 ise, sayfa eskidir ve kullanılmamıştır
  - Eğer R değeri 1 ise sayfaya ikinci şansı ver
  - Eğer tüm sayfalara erişim olduysa, «İlk giren ilk çıkar» gibi çalışır

İkinci Şans Algoritması

(a) FIFO sırasına göre sıralanmış sayfalar. (b) Zaman 20'de bir sayfa hatası oluşursa ve A'nın R biti 1 ise sayfa listesi. Sayfaların üzerindeki sayılar yükleme süreleridir.

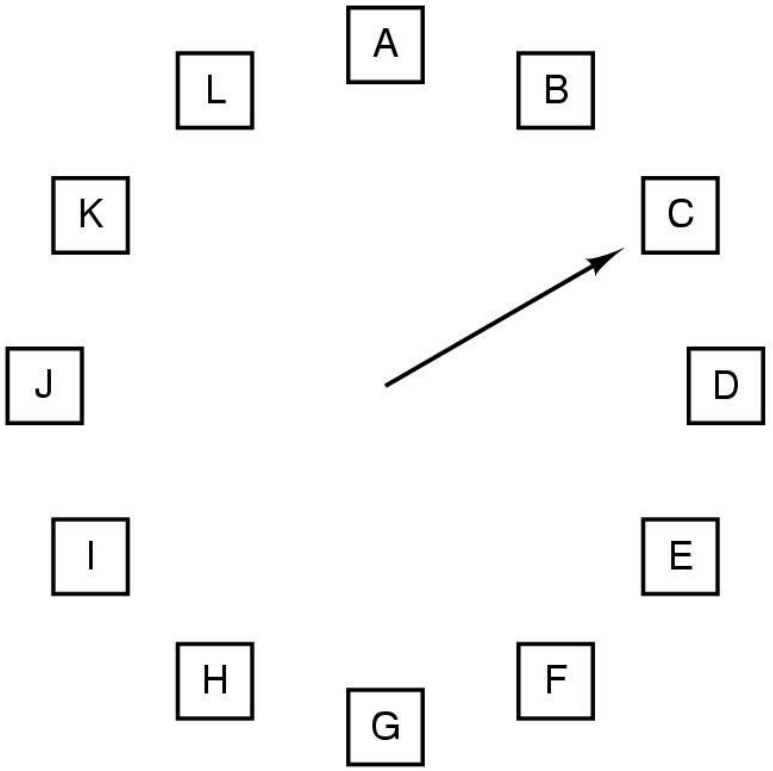


Saat Sayfa Değiştirme

- İkinci şans algoritması verimsizdir
  - Sayfaları listesinde sürekli olarak hareket ettirir
- Alternatif
  - Tüm sayfa çerçevelerini saat şeklinde dairesel bir listede tut

Saat Sayfa Değiştirme Algoritması

- .



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:

R = 0: Evict the page

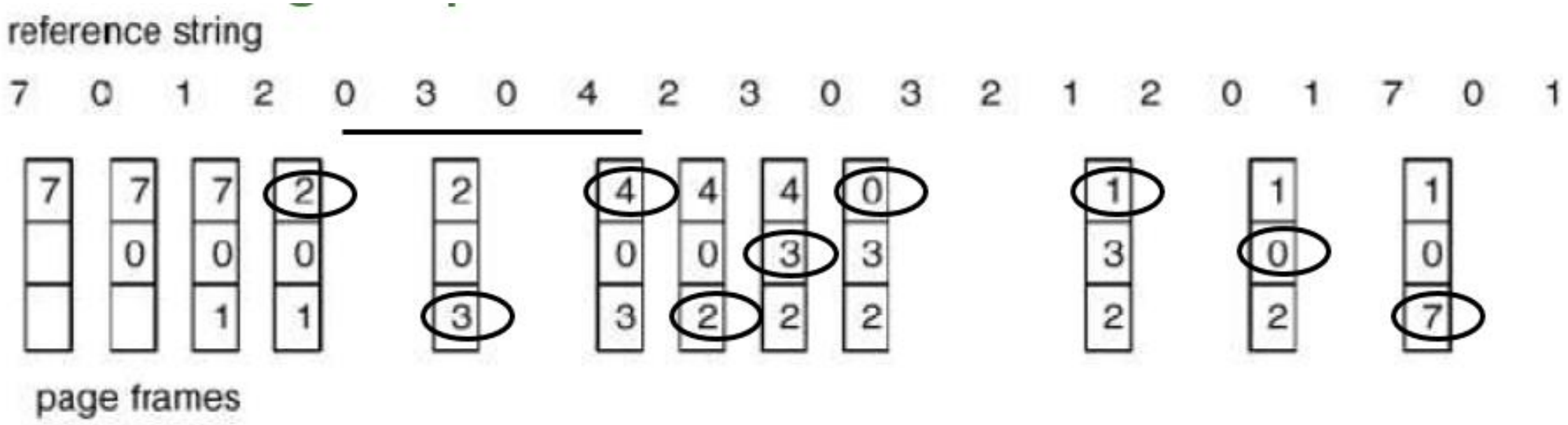
R = 1: Clear R and advance hand

En Son Kullanılan Sayfa Değiştirme

- Optimum algoritmaya yakın bir yaklaşım
  - Uzun zamandır kullanılmayan sayfalar muhtemelen uzun süre daha kullanılmayacaktır
  - Bir sayfa hatası oluştuğunda, en uzun süre kullanılmayan sayfayı at
- LRU'yu gerçeklemek için
  - Zamanı kaydetmek için bellekteki tüm sayfaları tutan bağlı liste kullan
  - Yada donanım sayacı veya bir matris kullan

Son Kullanılan Sayfa Değiştirme Algoritması

- Sayfa hata sayısı: 12, Sayfa değiştirme sayısı: 9

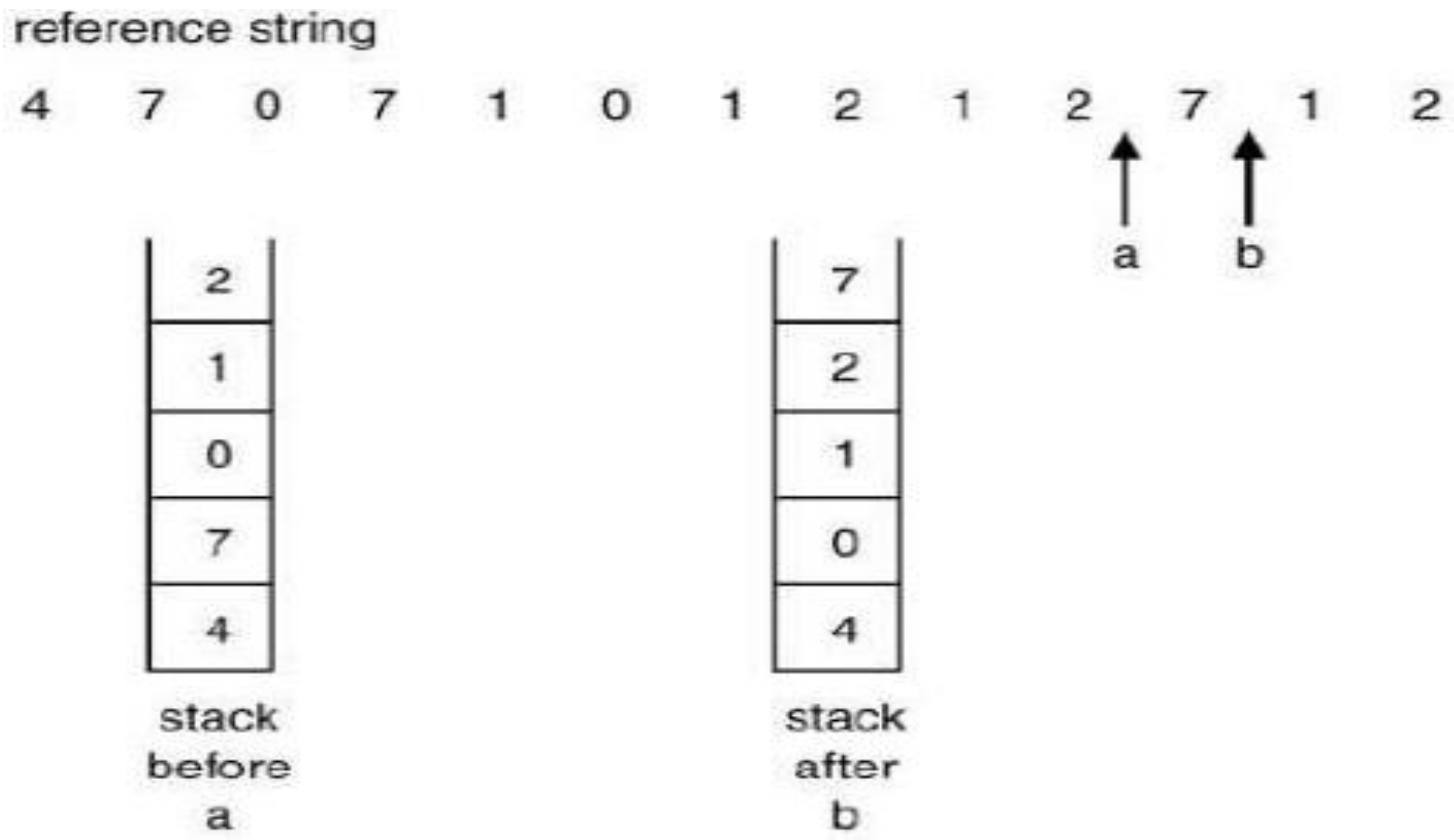


En Son Kullanılan Algoritma Gerçekleme

- Verimli bir şekilde gerçeklenmesi zor
- Sayfada fazladan bir sayaç (kullanım süresi) alanı kullan
  - En eski sayfayı değiştir
  - Sayaçlı donanım gerektirir
  - Sayfa hatasında, en düşük olanı bulmak için tüm sayaçları kontrol eder
- Bir yığında sayfa numaralarını sakla (yazılımsal)
  - En alttaki sayfayı değiştir
  - Yığının boyutu, fiziksel çerçevelerin boyutu kadar
  - Eğer sayfa yığının içindeyse çıkar ve geri koy,
  - Değilse, yığına koy (push)

Yığın Kullanan En Son Kullanılan Algoritması

- .



Sanal Bellek - Sayfalama

Sanal Bellek - Sayfalama

Sanal Bellek - Sayfalama

SON