Bölüm 9: Sanal Bellek

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

Değiş Tokuş – Sanal Bellek

- Sistemde arka planda çalışan bir çok süreç vardır
- Fiziksel bellek tüm programları tutacak kadar büyük değildir
  - Değiş tokuş (swap)
    - Programları belleğe getir, değiş tokuş yapıp götür
  - Sanal bellek
    - Programları kısmen bellekte olsalar bile çalıştır
- **Değiş tokuş** bir süreci birincil bellekten ikincil belleğe geçici olarak aktarır.
- **Talebe bağlı sayfalama** pager, tüm süreci yer değiştirmek yerine yalnızca gerekli sayfaları belleğe getirir.

Sanal Bellek

- Kodun yürütülebilmesi için bellekte olması gerekir, ancak tüm program nadiren kullanılır
  - Hataları ele alan kod parçası, sıra dışı prosedürler, büyük veri yapıları
- Tüm program kodu aynı anda gerekli değildir
- Kısmen yüklenmiş program yürütülebilir!
  - Program artık fiziksel bellek limitiyle sınırlanamaz
  - Her program çalışırken daha az bellek kullanır, aynı anda daha fazla program çalışabilir
  - Tepki veya geri dönüş süresinde artış olmaksızın CPU kullanımı ve verimi artar
  - Programları belleğe yüklemek veya takas için daha az G/Ç gerekir

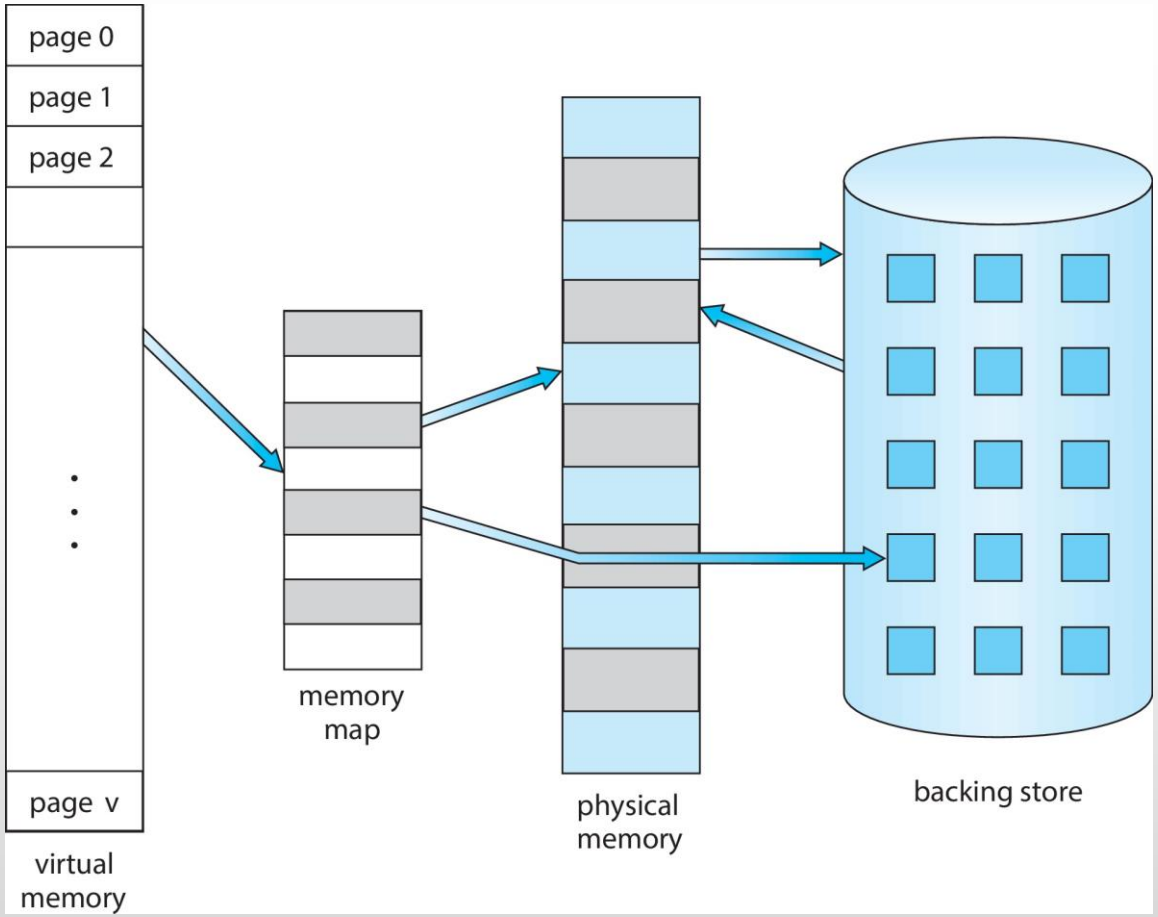
Sanal Bellek

- Kullanıcı mantıksal belleğinin fiziksel bellekten ayrılması
- Programın yürütülmesi için bir kısmının bellekte olması yeterli
- Mantıksal adres alanı, fiziksel adres alanından çok daha büyük olabilir.
- Adres alanlarının birkaç süreç tarafından paylaşılmasına izin verir
- Daha verimli süreç oluşturmaya izin verir
- Aynı anda daha fazla program çalışabilir

Sanal Bellek Adres Uzayı

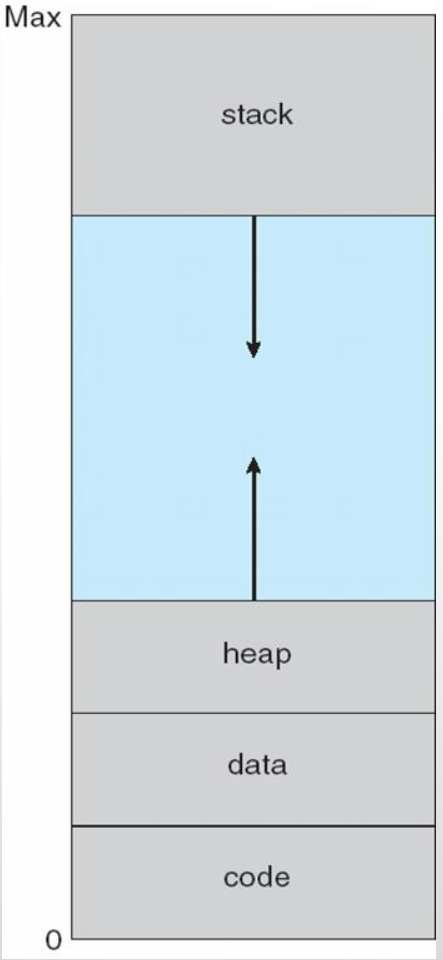
- Sürecin bellekte nasıl saklandığına ilişkin mantıksal görünüm
- Genellikle 0 adresinde başlar, alanın sonuna kadar bitişik adresler
- Fiziksel bellek sayfa çerçeveleri olarak düzenlenir
- MMU mantıksal ve fiziksel adresleri eşler
- Sanal bellek şu yollarla uygulanabilir:
  - Talep olduğunda sayfalama (demand paging)
  - Talep olduğunda kesimleme (demand segmentation)

Sanal Bellek

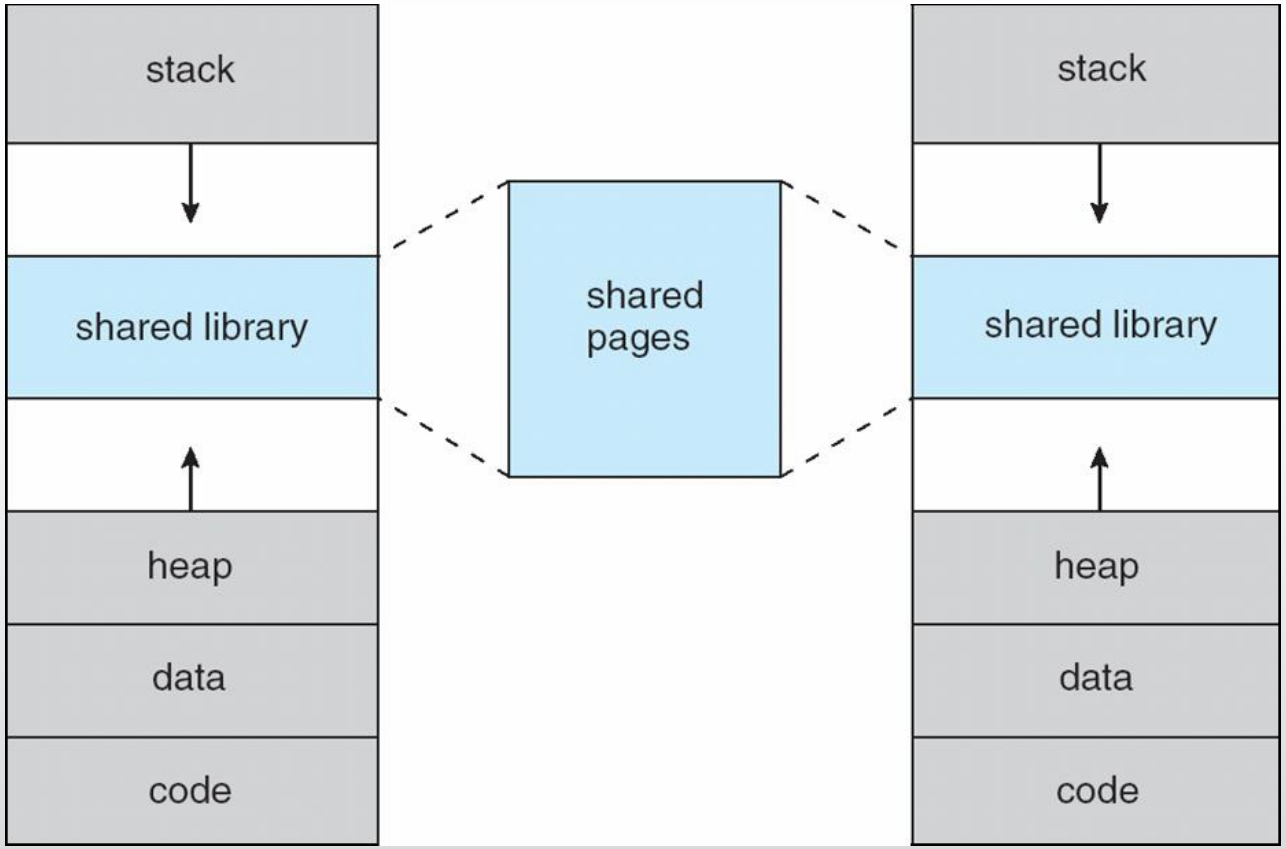


Sanal Bellek Adres Uzayı

- Yığıt Maks mantıksal adreste başlar ve büyüdükçe mantıksal adres alanı aşağı doğru iner.
- Kullanılmayan adres alanı deliktir (hole)
- Yığıt veya yığın yeni sayfaya ulaşana kadar fiziksel belleğe ihtiyaç yoktur
- Sanal adres alanına eşleme yoluyla sistem kütüphaneleri paylaşılabilir



Paylaşımlı Kütüphane



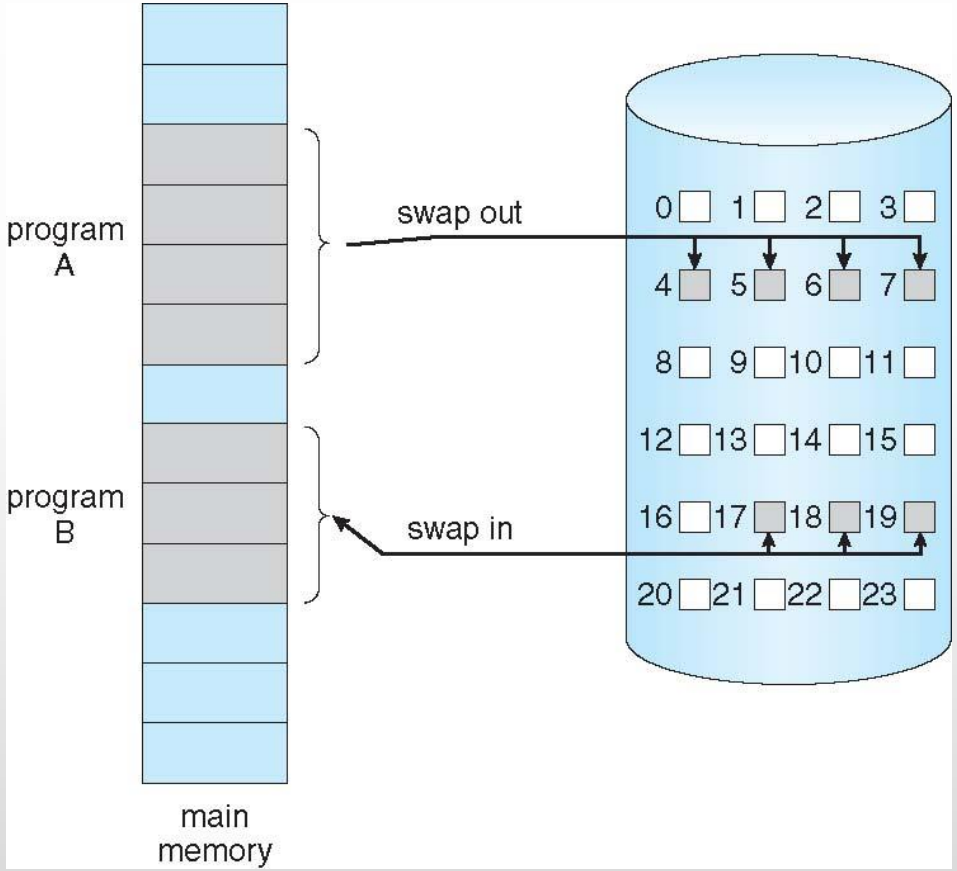
Sanal Bellek – Şişkin (bloat) Yazılım Yönetimi

- Programların belleğe sığmayacak kadar büyük olabilir
- Programları bölmek 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

Talep Olduğunda Sayfalama

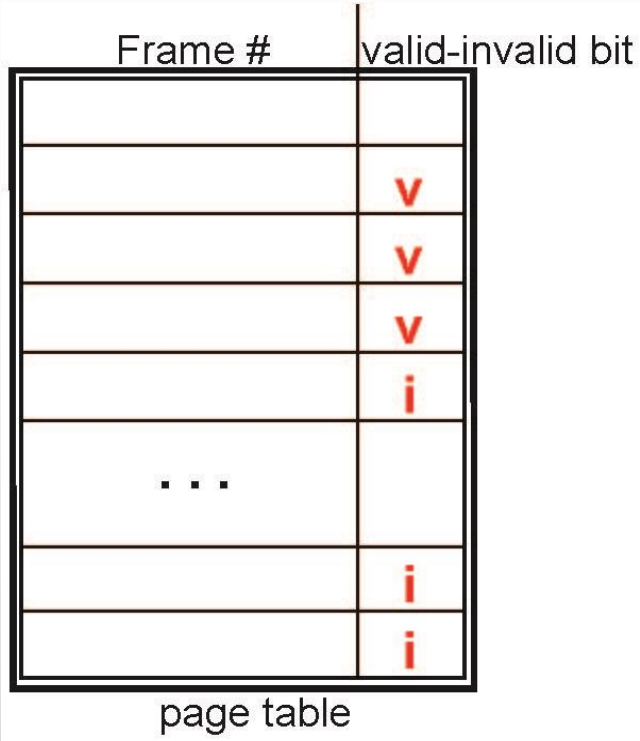
- Tüm süreci yükleme zamanında belleğe getirebilir
- Veya bir sayfayı yalnızca gerektiğinde belleğe getirir
  - Daha az G/Ç gerekli, Gereksiz G/Ç yok, Daha az bellek gerekli, Daha hızlı yanıt, Daha fazla kullanıcı
- Değiş tokuş (swap) sayfalamaya benzer
  - Sayfa gerekli olduğunda referans verilir
    - Referans geçersiz ise işlemi iptal et
    - Bellekte değil ise belleğe getir
- Lazy swapper - sayfa gerekmedikçe asla bir sayfayı belleğe getirmez
- Pager - sayfalarla ilgilenen takasçı birim

Talep Olduğunda Sayfalama

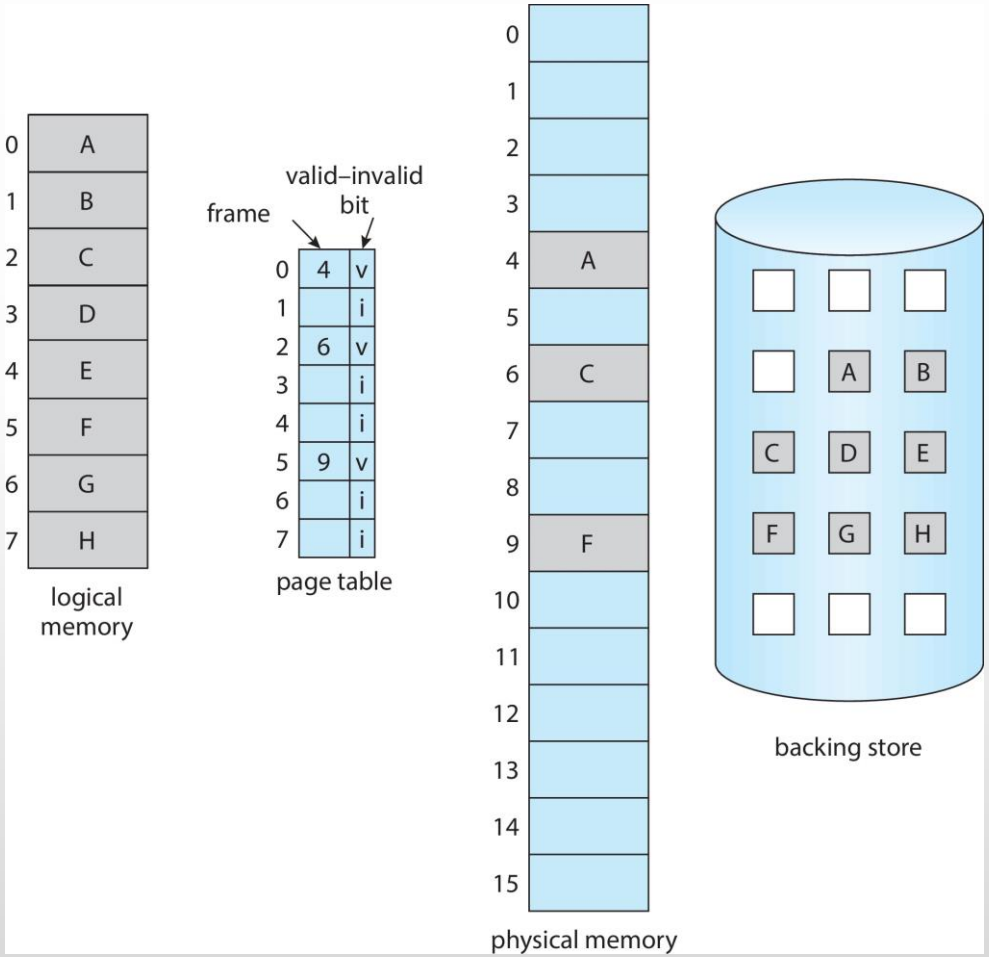


Geçerli Geçersiz Biti

- Her sayfa tablosu girdisi ile geçerli-geçersiz bir bit ilişkilendirilir
  - v - bellekte yerleşik,
  - i - bellekte değil
- Başlangıçta geçerli-geçersiz bit, tüm girdilerde i olarak ayarlanır
- MMU adres çevirisi sırasında, geçerli-geçersiz bit i ise → sayfa hatası



Geçerli Geçersiz Biti



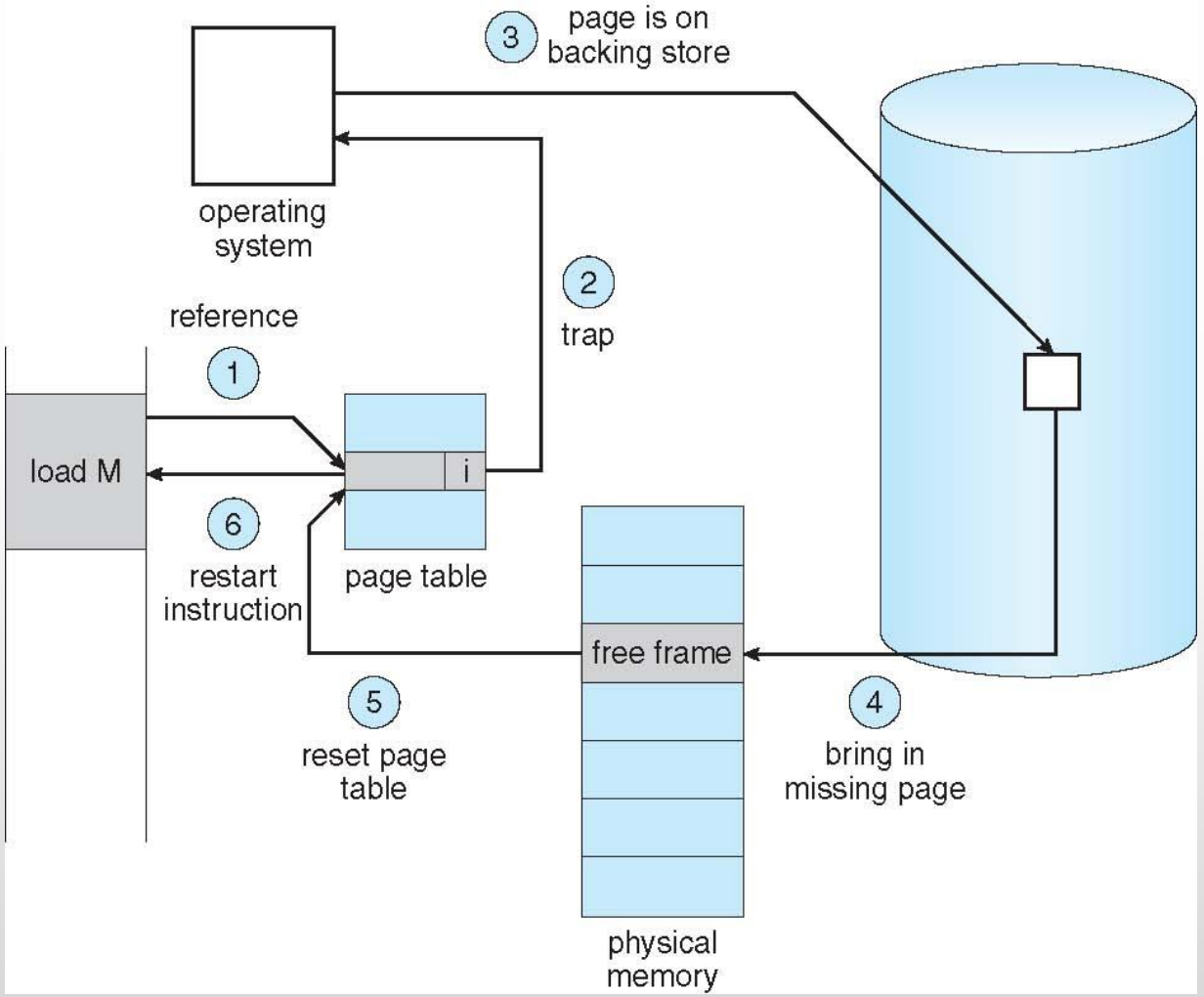
Hatalı Sayfa İşlemi

- Mevcut/yok biti, sayfanın bellekte olup olmadığını söyler
- Adres bellekte yoksa ne olur?
- İşletim sistemine tuzak (trap)
  - İşletim sistemi diske yazmak için bir sayfa seçer
  - (Gerekli) olan sayfayı belleğe getirir
  - Komutu yeniden başlatır

Hatalı Sayfa İşlemi

- Hatalı bir sayfaya erişim olduğunda, yürütme işletim sistemine geçer.
- İşletim sistemi karar vermek için sayfa tablosuna bakar
  - Geçersiz referans → süreç iptal edilir
  - Sadece sayfa bellekte değil
- Boş bir çerçeve bulur
- Sayfayı bir çerçeve içine yerleştirmek için disk operasyonu çizelgeler
- Sayfa tablosunu günceller
  - Geçerli (valid) bitini ayarlar
- Sayfa hatasına neden olan komutu yeniden başlatır

Hatalı Sayfa İşlemi



Talep Olduğunda Sayfalama Adımlar

- Yürütmeyi işletim sistemine bırak
- Kullanıcı yazmaçları ve süreç durumunu kaydet
- Kesilmenin bir sayfa hatası olduğunu belirle
- Sayfa referansının geçerli olup olmadığını kontrol et
  - Sayfanın diskteki konumunu belirle
- Sayfayı diskten bellekte boş bir çerçeveye taşı
  - Taşıma işlemi sonlanana kadar bu cihaz için sırada bekle
  - Aygıt arama ve/veya gecikme süresi boyunca bekle
  - Sayfayı boş bir çerçeveye aktarmaya başla

Talep Olduğunda Sayfalama Adımlar

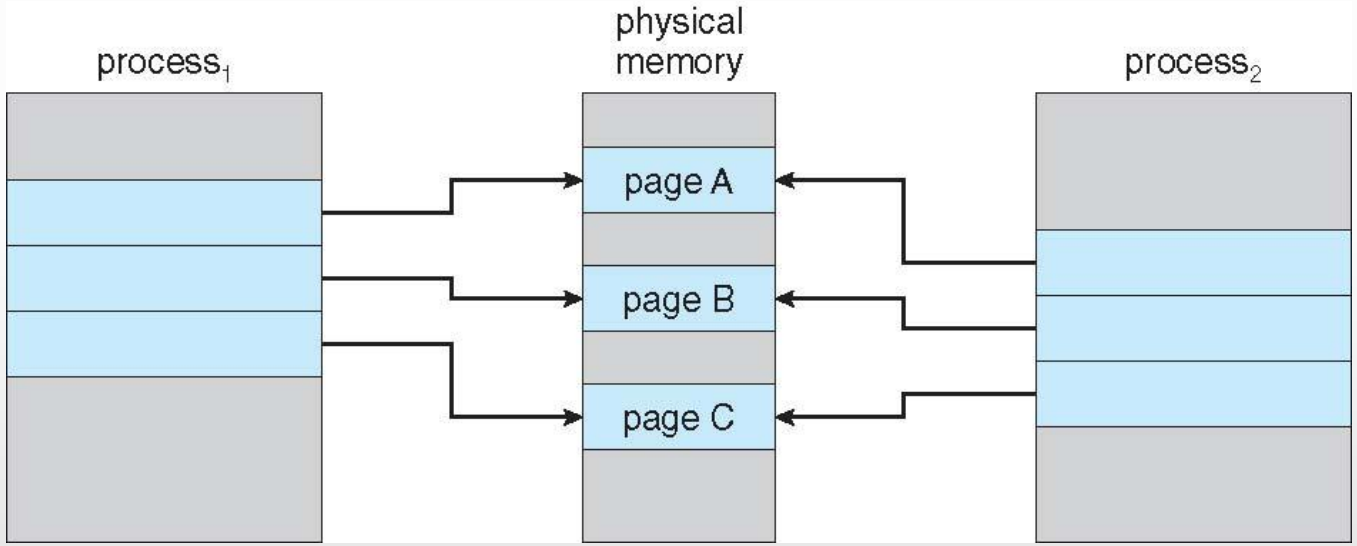
- Beklerken CPU'yu başka bir sürece tahsis et
- Disk G/Ç alt sisteminden bir kesilme al (G/Ç tamamlandı)
- Diğer kullanıcı için yazmaç ve süreç durumunu kaydet
- Kesilmenin diskten olduğunu belirle
- Sayfanın artık bellekte olduğunu göstermek için sayfa tablosunu güncelle
- CPU'nun bu sürece tekrar tahsis edilmesini bekle
- Kullanıcı yazmaçlarını, süreç durumunu ve yeni sayfa tablosunu geri yükle
- Yarıda kesilen komutu devam ettir

Copy-on-Write

- Copy-on-Write (COW), hem üst hem de alt süreçlerin başlangıçta aynı sayfaları bellekte paylaşmasına izin verir
  - Süreçlerden biri paylaşılan sayfayı değiştirirse, sayfa kopyalanır
- COW, yalnızca değiştirilen sayfalar kopyalandığından daha verimli süreç oluşturmaya olanak tanır
- Genel olarak, boş sayfalar, talep üzerine sıfır ile doldurulan sayfalardan oluşan bir havuzdan tahsis edilir.
  - Hızlı talep yürütme için havuzda her zaman boş çerçeveler olmalıdır
  - Sayfa hatasında bir çerçeveyi serbest bırakmak zorunda kalınmaz

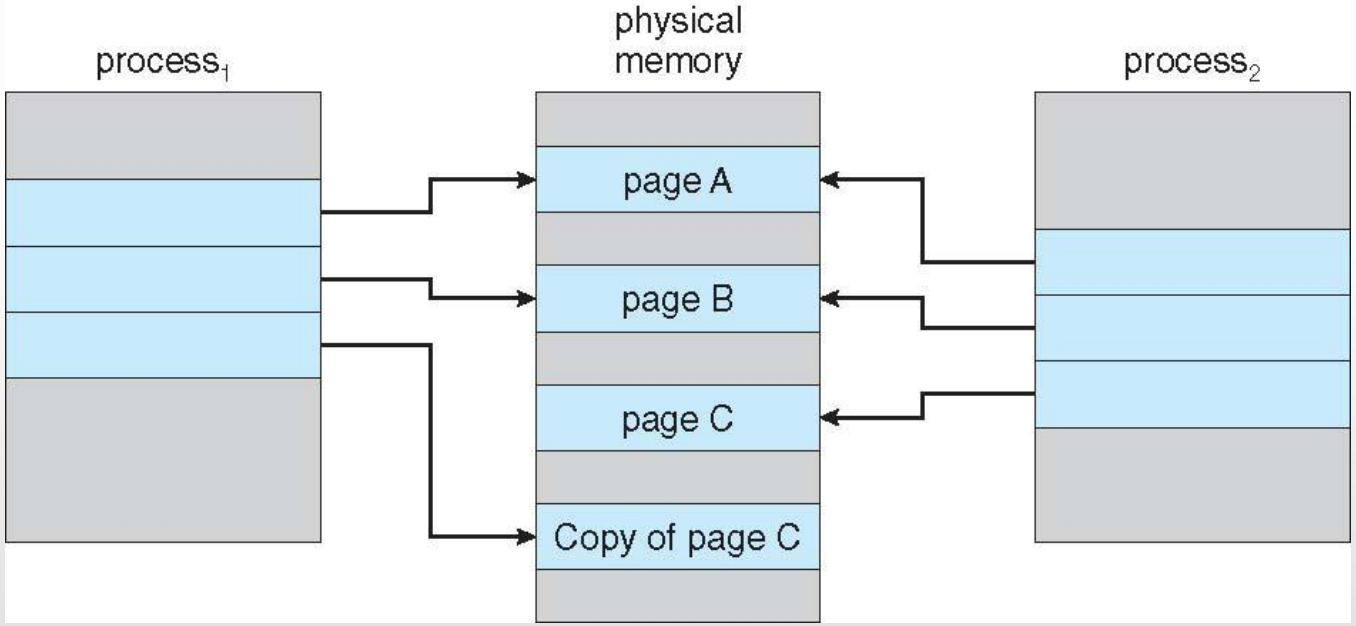
Copy-on-Write C Sayfası Değişmeden Önce

- .



Copy-on-Write C Sayfası Değiştikten Sonra

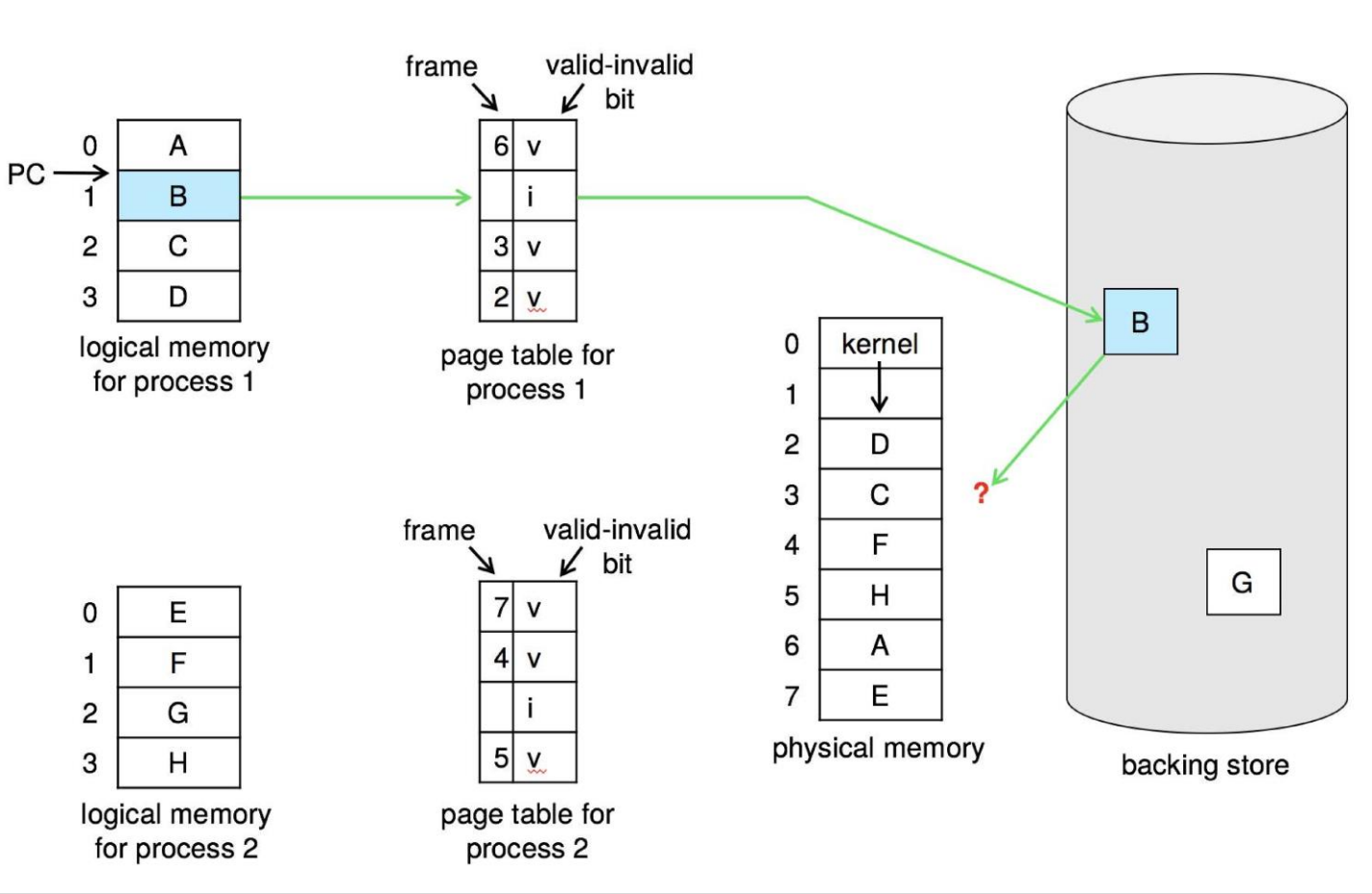
- .



Boş Çerçeve Yoksa

- Sayfalar süreçler tarafından kullanıldı
- Ayrıca çekirdekten, G/Ç arabelleklerinden vb. istekler gelebilir
- Her birine ne kadar tahsis edilmeli?
- Sayfa değiştirme: bellekte kullanımda olmayan sayfaları bul ve çıkar
  - Algoritma: sonlandırılsın mı? değiş tokuş mu? sayfayı değiştir?
  - Performans: minimum sayfa hatasıyla sonuçlanacak bir algoritma
- Aynı sayfa birkaç kez belleğe alınabilir

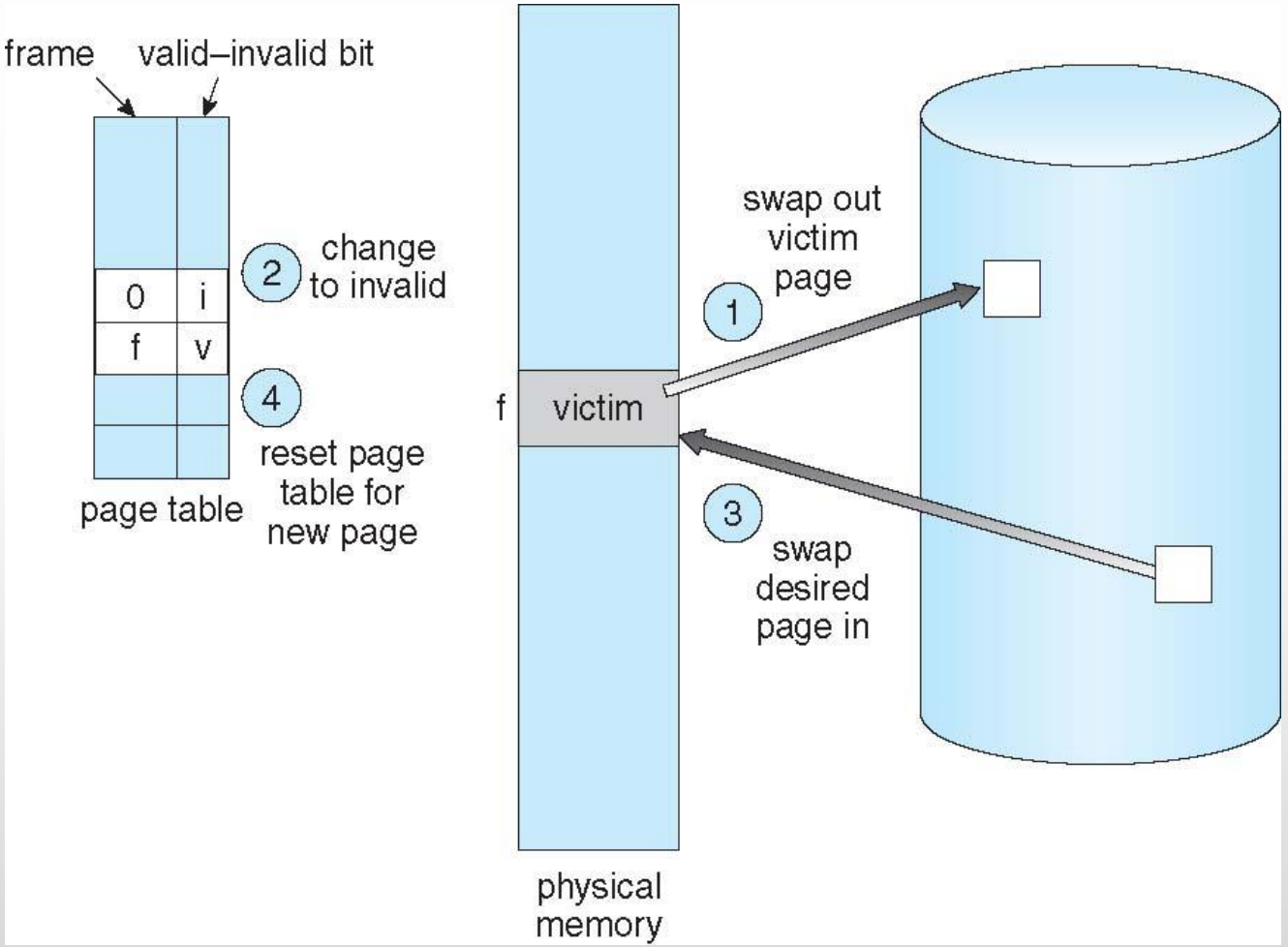
Sayfa Değiştirme



Sayfa Değiştirme

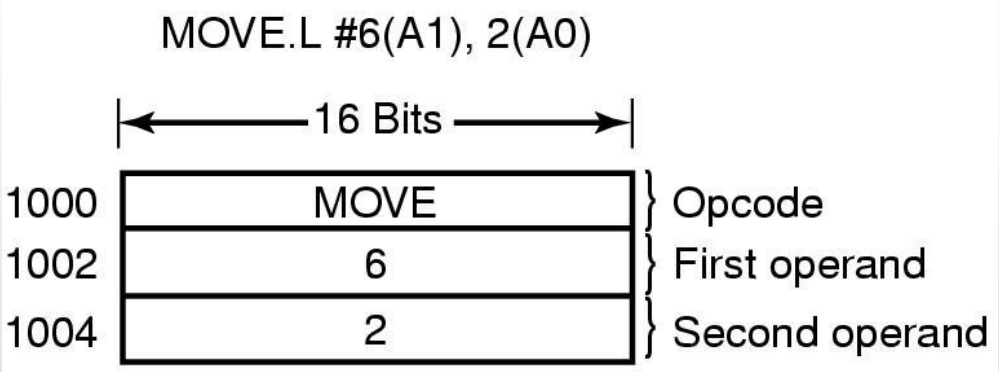
- Diskte istenen sayfanın konumunu bul
- Boş bir çerçeve bul:
  - Boş bir çerçeve varsa onu kullan
  - Boş çerçeve yoksa, bir kurban çerçeve seçmek için bir sayfa değiştirme algoritması kullan
  - Kirliyse kurban çerçeveyi diske yaz
- İstenen sayfayı (yeni) boş çerçeveye getir; sayfa ve çerçeve tablolarını güncelle
- Tuzağa neden olan komutu yeniden başlatarak işleme devam et

Sayfa Değiştirme



Sayfa Hatasına Neden Olan Bir Komut

- Talimat yeniden nereden başlatılır? PC, talimatın hangi bölümünün gerçekten hatalı olduğuna bağlıdır. 1002'de hata verirse OS, komutun 1000'de başladığını nereden biliyor?



Sayfa Hatasının Etkisi

- Sayfa hatası süresi = 25 ms (page fault)
- Bellek erişim süresi = 100 ns (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 < 110 ns olması gerekiyorsa, o zaman 100 + 24.999.900*p < 110 yani p < 10/24.999.900 < 10/25.000.000 = 1/2.500.000 = 4×10^{-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?

Sayfa Değiştirme Algoritmaları

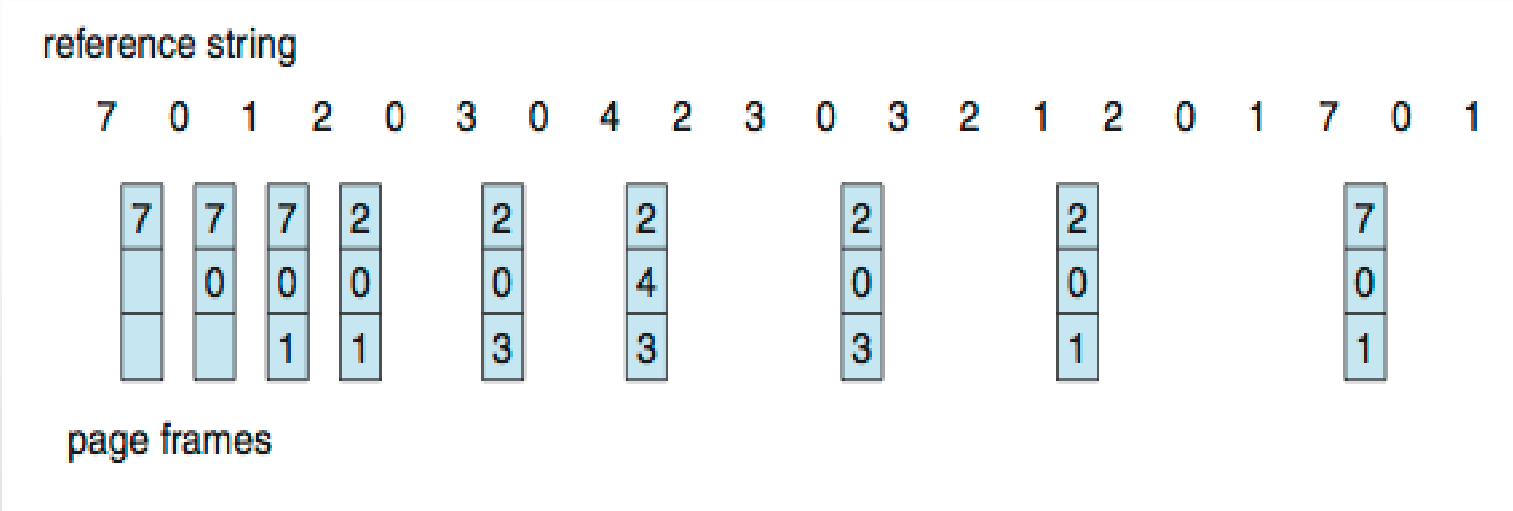
- Optimum (optimal)
- Yakın zamanda kullanılmayan (not recently used)
- İlk Giren İlk Çıkar (first-in first-out)
- İkinci şans (second chance)
- Saat (clock)
- En son kullanılan (least recently used)
- Çalışma kümesi (working set)
- Çalışma kümesi saat (working set clock)

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
- En uzun süre kullanılmayacak olanı seçilir
- İşletim sistemi hangi sayfaya ne zaman erişileceğini bilemez (crystal ball)
- Ancak; gerçekleştirilebilir algoritmaların performansını karşılaştırmak için kullanılabilir

Optimum Sayfa Değişimi

- Sayfa hatası 9 kez, değiştirme 6 kez



Yakın Zamanda Kullanılmayan Sayfa Değişimi

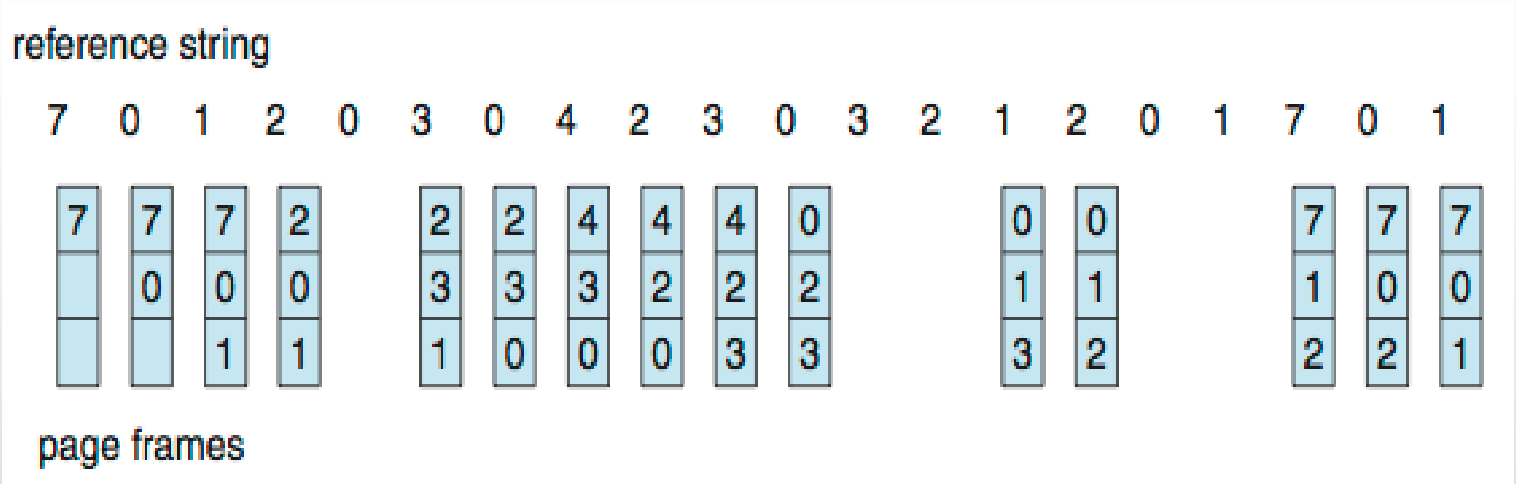
- Sayfaları kullanımlarına göre değiştirme,
- Çıkarmak için en düşük öncelikli sayfayı seç
- Sayfalarda R ve M biti bulunur (referenced, modified)
- Bir işlem başlatıldığında, her iki sayfa bitine de 0 atanır, periyodik olarak, R biti temizlenir
- Dört farklı durum oluşabilir
  - Sınıf 0: erişilmedi,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

İlk Giren İlk Çıkar Sayfa Değişimi

- Listeyi zamana göre sıralı tut (en sonuncusu listenin sonuna gelir)
- En yaşlıyı, yani sıranın başını tahliye et
- Uygulaması kolay
- En eskisi en yoğun şekilde kullanılmış olabilir!
- FIFO'ya hiçbir kullanım bilgisi dahil değildir

İlk Giren İlk Çıkar Sayfa Değişimi

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

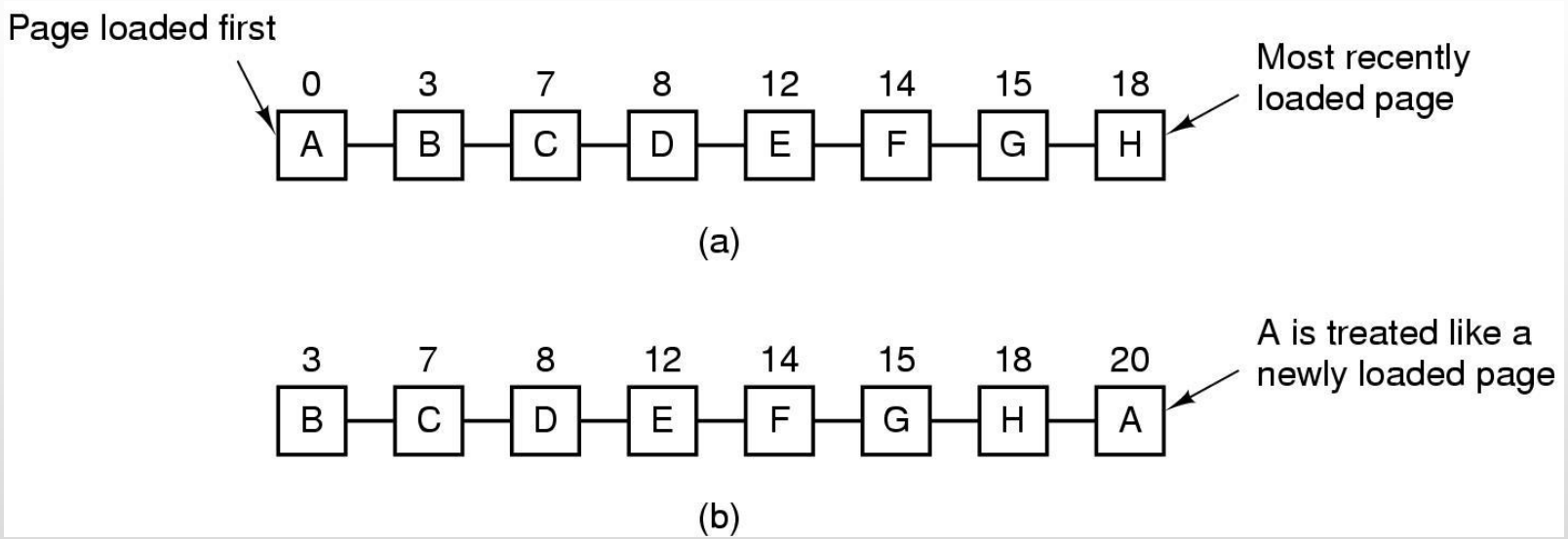


İkinci Şans Sayfa Değişimi

- «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

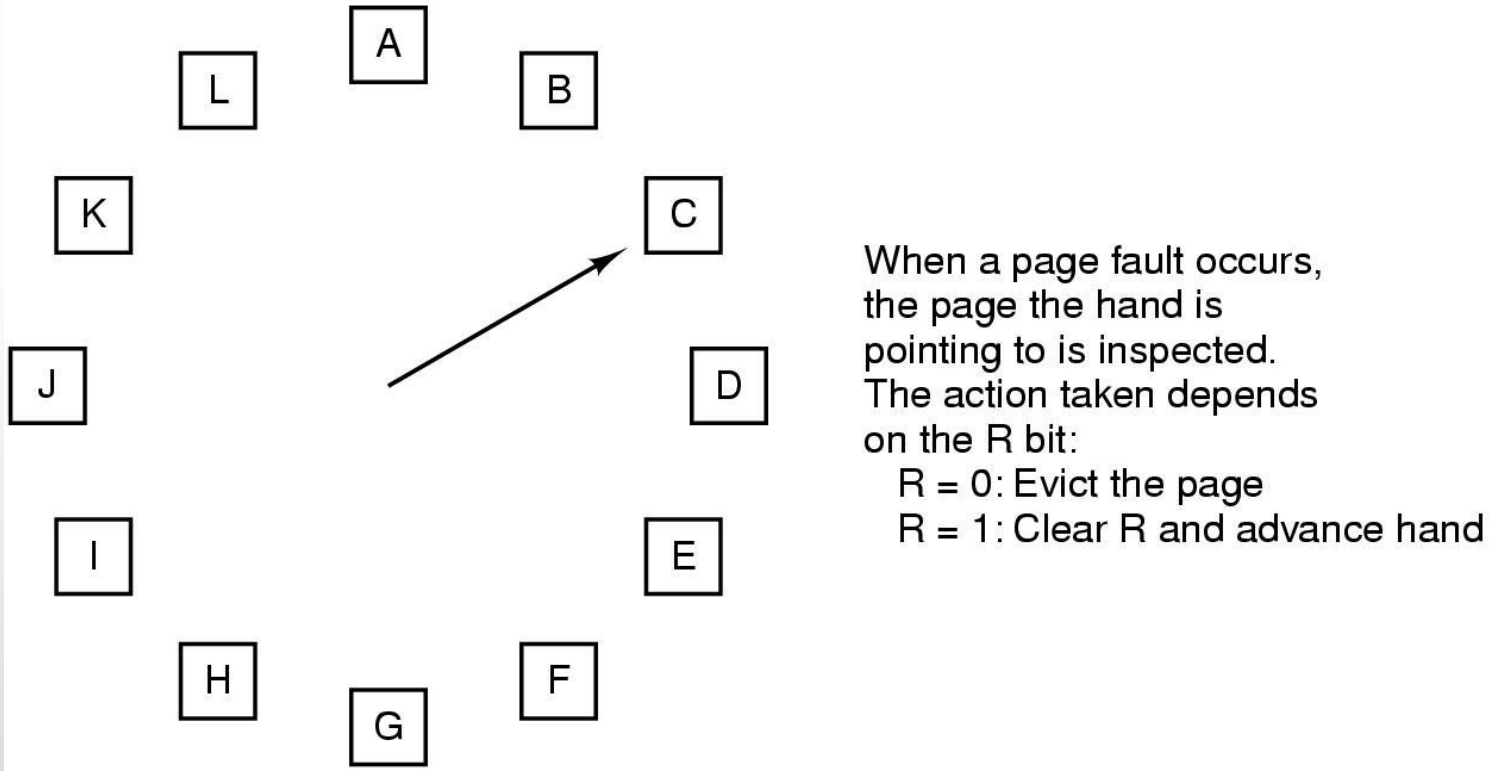
(a) FIFO sırasına göre sıralanmış sayfalar. (b) Zaman 20'de bir sayfa hatası oluşursa ve A'nın R biti 1 ise sayfa listesi. Sayfaların üzerindeki sayılar yükleme süreleridir.



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

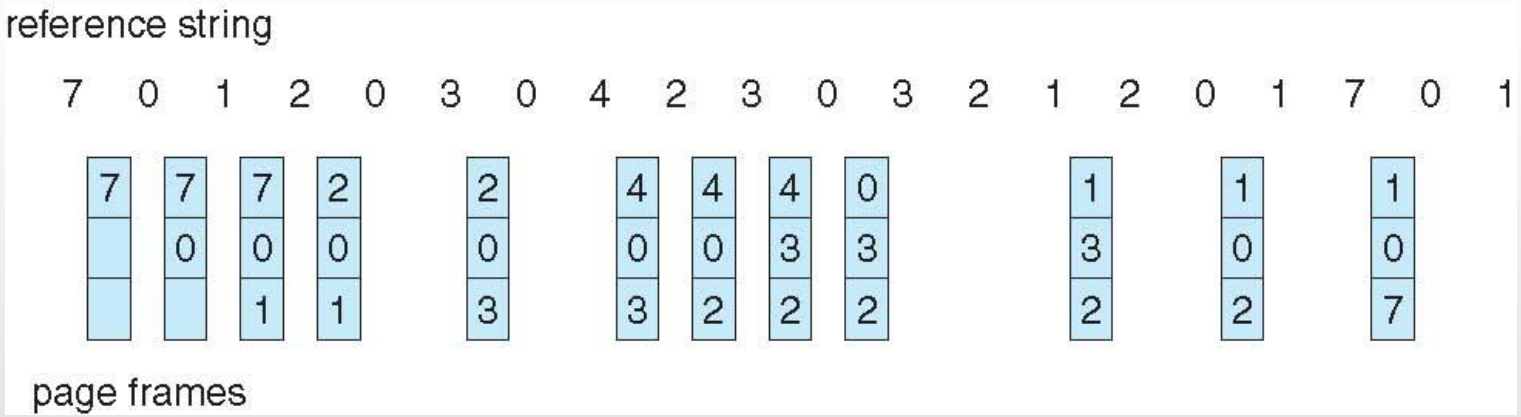


En Son Kullanılanı Değiştir Algoritması

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

En Son Kullanılanı Değiştir Algoritması

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

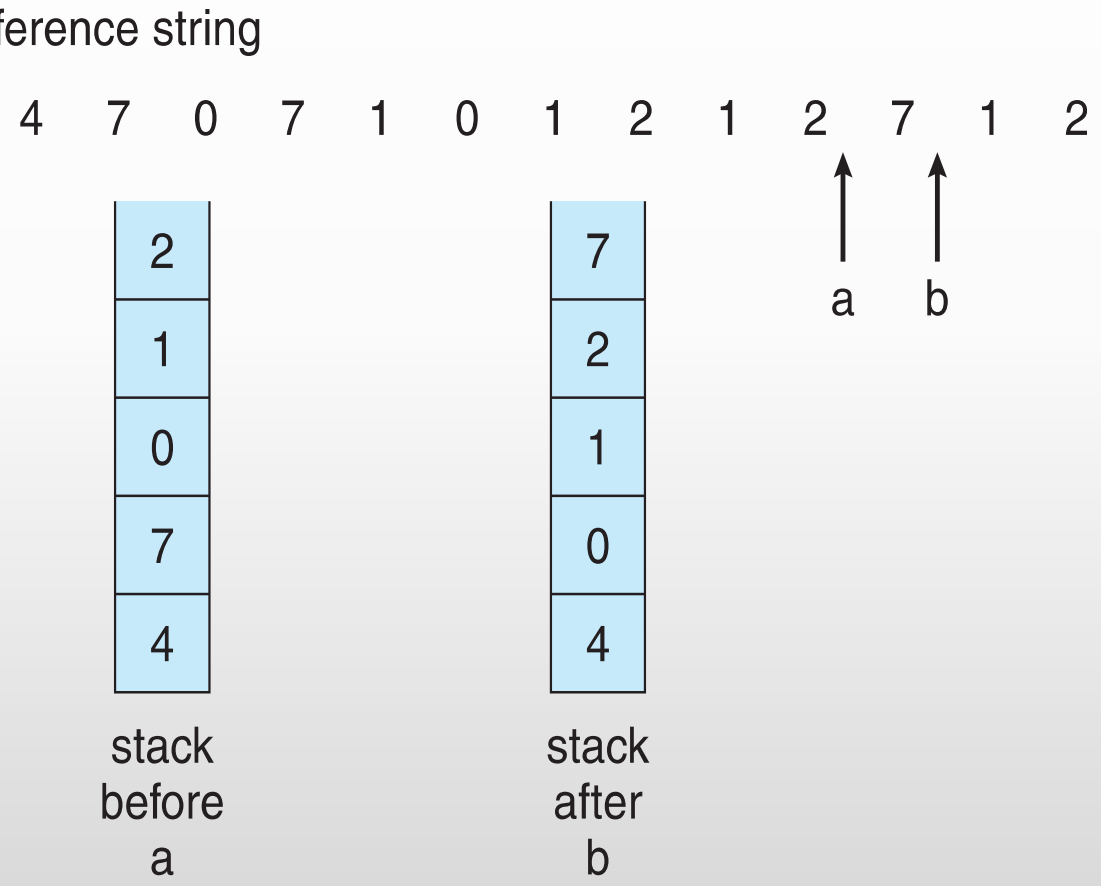


En Son Kullanılanı Değiştir Algoritması

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

En Son Kullanılanı Değiştir (yığın)

- .



En Son Kullanılanı Değiştir (donanım)

- Sayfalara 0, 1, 2, 3, 2, 1, 0, 3, 2, 3 sırasıyla erişildiğinde kullanılan matrisin içeriğinin değişimi.

(a)

	Page 0	1	2	3
0	0	1	1	1
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

(b)

Page 0	1	2	3
0	0	1	1
1	0	1	1
0	0	0	0
0	0	0	0

(c)

Page 0	1	2	3
0	0	0	1
1	0	0	1
1	1	0	1
0	0	0	0

(d)

Page 0	1	2	3
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0

(e)

Page 0	1	2	3
0	0	0	0
1	0	0	0
1	1	0	1
1	1	0	0

(f)

0	0	0	0
1	0	1	1
1	0	0	1
1	0	0	0

(g)

0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

(h)

0	1	1	0
0	0	1	0
0	0	0	0
1	1	1	0

(i)

0	1	0	0
0	0	0	0
1	1	0	1
1	1	0	0

(j)

0	1	0	0
0	0	0	0
1	1	0	0
1	1	1	0

Sık Kullanılmayanı Değiştir

- «En son kullanılanı değiştir» yöntemini yazılımda simüle etmek için, (Not frequently used)
- Sayfaların erişim zamanları kaydedilir
- Bir sayfa hatası oluştuğunda, değeri en düşük olan çıkarılır
- Sorun: uzun bir log tutmak gerekir

Sık Kullanılmayanı Değiştir

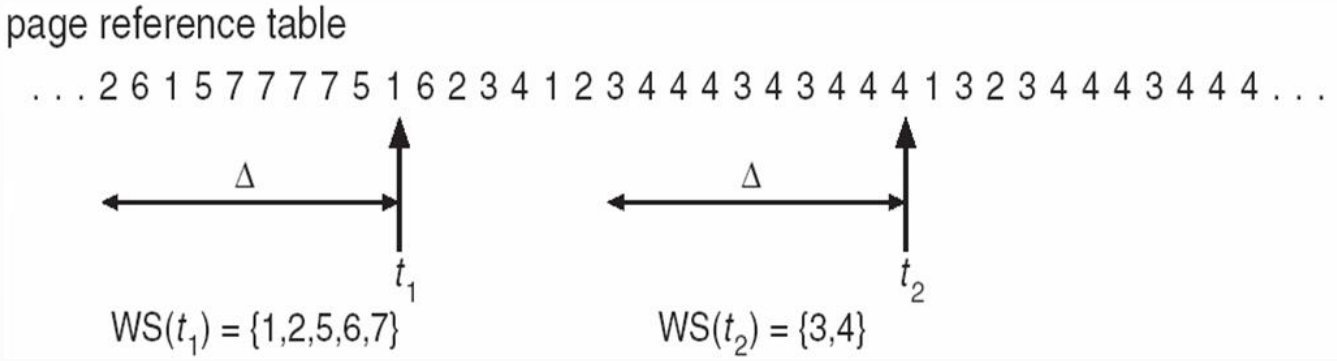
Altı sayfa ve beş zaman dilimi için algoritma çıktısı

Page	R bits for pages 0-5, clock tick 0	R bits for pages 0-5, clock tick 1	R bits for pages 0-5, clock tick 2	R bits for pages 0-5, clock tick 3	R bits for pages 0-5, clock tick 4
	1 0 1 0 1 1	1 1 0 0 1 0	1 1 0 1 0 1	1 0 0 0 1 0	0 1 1 0 0 0
0	10000000	11000000	11100000	11110000	01111000
1	00000000	10000000	11000000	01100000	10110000
2	10000000	01000000	00100000	00100000	10010000
3	00000000	00000000	10000000	01000000	00100000
4	10000000	11000000	01100000	10110000	01011000
5	10000000	01000000	10100000	01010000	00101000
	(a)	(b)	(c)	(d)	(e)

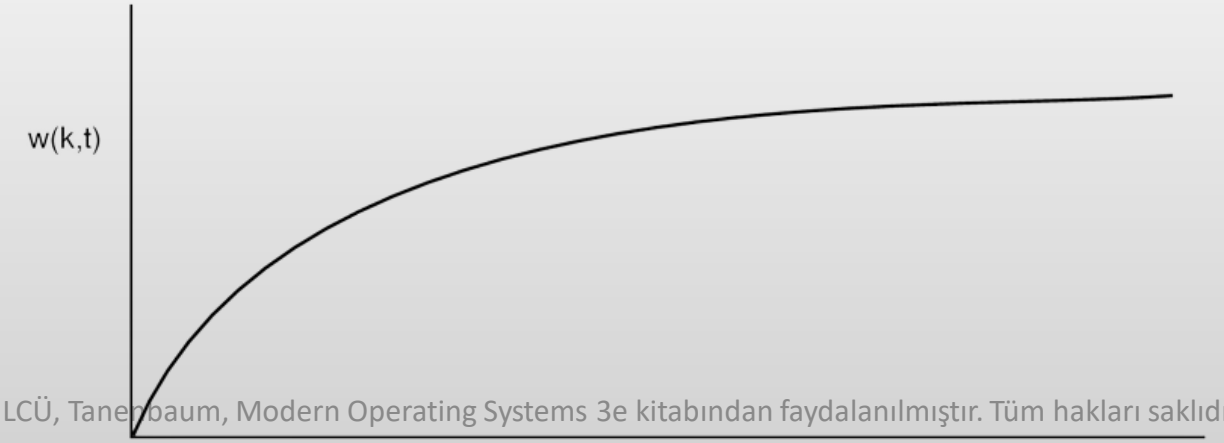
Çalışma Kümesi Sayfa Değiştirme Algoritması

- Sayfaları önceden değil talep üzerine yükler
- Süreç koşarken, sayfalarının yalnızca nispeten küçük bir kısmına erişir
- Çalışma seti:
  - Bir sürecin herhangi bir anda kullanmakta olduğu sayfalar kümesi
  - Çalışma kümesi varsa, bir sonraki aşamaya kadar çok fazla sayfa hatasına neden olmaz
- Bir süreç yer değiştirilip bellekten çıkarıldıktan ve daha sonra yer değiştirilip tekrar belleğe alındığında, önce çalışma kümesi yüklenir, böylece sayfa hatası sayısı büyük ölçüde azaltılır

Çalışma Kümesi Sayfa Değiştirme Algoritması



- Çalışma kümesi, k adet en son bellek erişimi tarafından kullanılan sayfa setidir. w(k,t) fonksiyonu, t zamanında çalışan kümenin boyutudur.



Çalışma Kümesi Yer Değiştirme

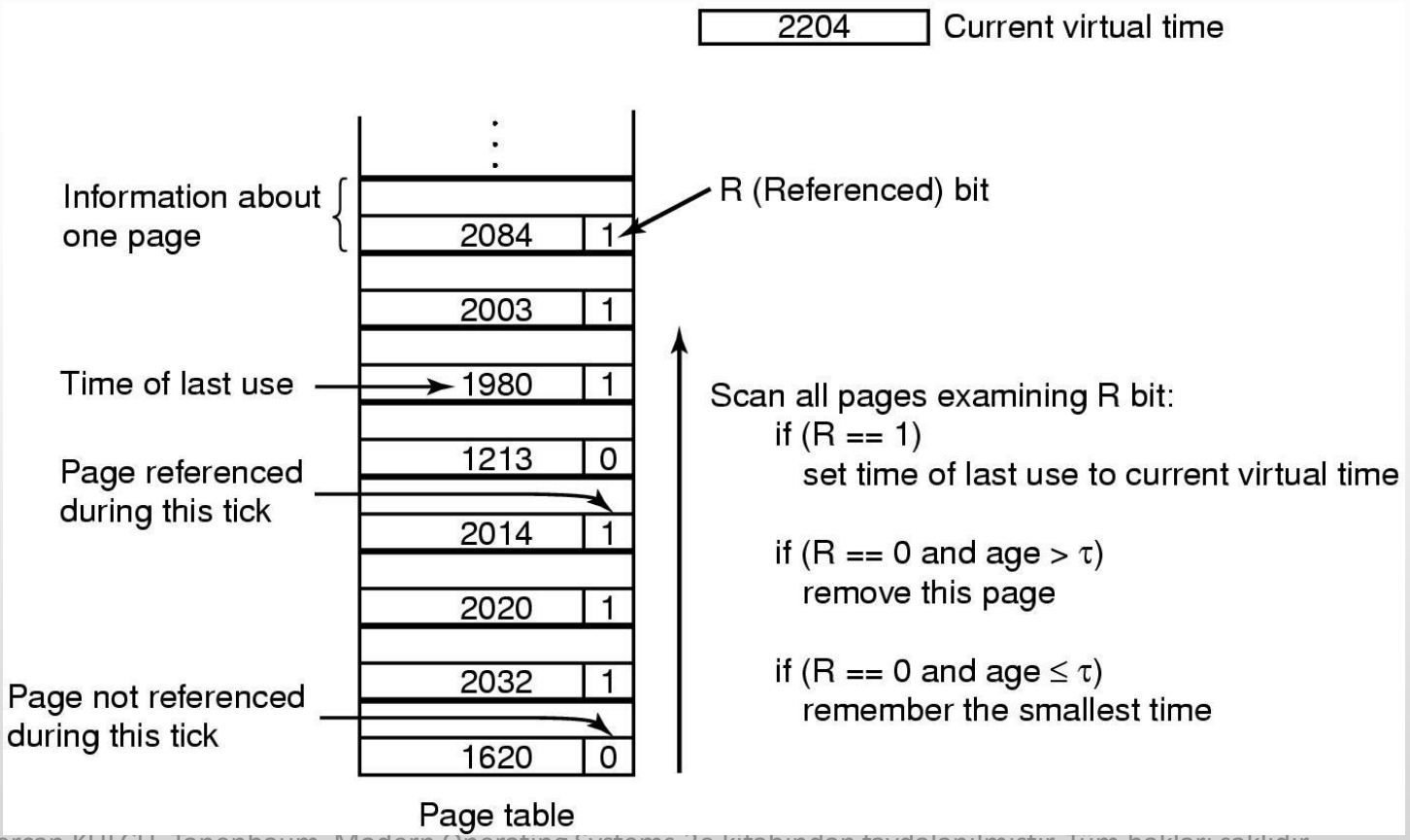
- Sayfa hatası oluştuğunda, çalışma kümesinde olmayan sayfa çıkarılır
- Çalışan bir küme elde etmek için: bir yazmaç tutulur ve maliyetli olacak her erişimde değeri bir kaydırılır (shift)
- Yakınsamalar
  - K adet bellek erişimi yerine koşma süresini (tau) kullan
- Her bellek erişiminde bellekteki sayfalar takip edilebilir.
- Her k erişim, çalışan bir kümeyle sonuçlanır.
- Masraflı

Erişim Sayısı Yerine Sanal Zaman Kullan

- t koşma (CPU) süresi boyunca erişilen son k sayfanın kaydı tutulur
- Sanal zaman, bir süreç için başladığından beri kullandığı işlemci zamanı miktarıdır.
- Bir sürecin ne kadar iş yaptığının ölçüsü

Çalışma Kümesi Yer Değiştirme

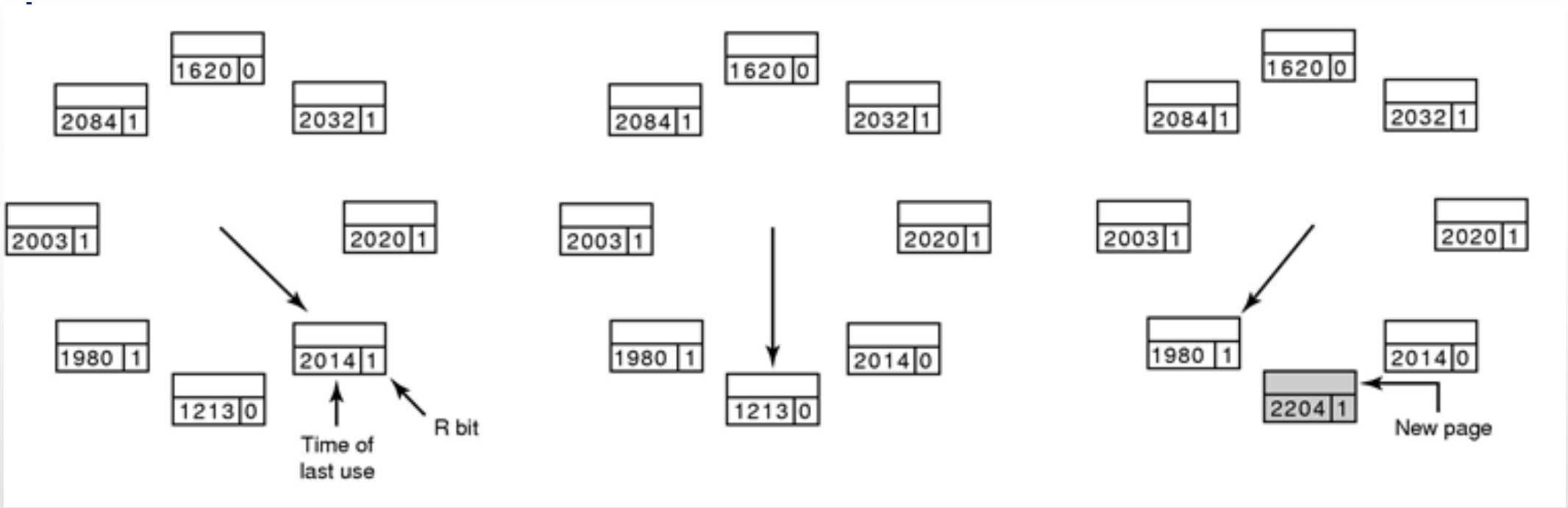
- R biti periyodik olarak temizlenir



Çalışma Kümesi Saat

- Temel çalışma kümesi algoritmasının her sayfa hatasında tüm sayfa tablosunu taraması gerekir
- WSClock: basit ve etkili
  - Sayfa çerçevelerinin dairesel listesi
  - R=1 ise, sayfaya geçerli tıklama sırasında erişilmiştir; R'ye 0 ata ve eli ilerlet
  - R=0 ise, ve M=0 ise ve yaş > tau ise yer değiştir; M=1 ise, eli ilerlet ve geri yazmayı çizelgele

Çalışma Kümesi Saat



Çalışma Kümesi Saat

- İki durumda el başlangıç noktasına kadar gelir:
  - En az bir yazma çizelgelendi:
    - ibre temiz bir sayfa bulana kadar hareket etmeye devam eder
  - Hiçbir yazma çizelgelenmedi:
    - tüm sayfalar çalışma kümesinde,
    - temiz bir sayfa seç
    - veya temiz sayfa yoksa, geçerli sayfa kurbandır ve diske geri yaz

Özet

Algoritma	Yorum
Optimum	Uygulanabilir değil, ancak kıyaslama olarak kullanışlı
NRU (Yakın Zamanda Kullanılmayan)	Çok kaba
FIFO (İlk Giren İlk Çıkar)	Önemli sayfaları atabilir
İkinci şans	FIFO'ya göre büyük gelişme
Saat	Gerçekçi
LRU (En Son Kullanılanlar)	Mükemmel, ancak tam olarak uygulanması zor
NFU (Sık Kullanılmayan)	LRU'ya yakınsayan adilce kaba
Yaşlanma	LRU'ya iyi yakınsayan verimli algoritma
Çalışma seti	Uygulanması biraz pahalı
WSClock	İyi verimli algoritma

Yerel ve Genel Tahsis Politikaları

- Genel tahsis: süreç, tüm çerçeveler kümesinden bir yedek çerçeve seçer; bir süreç diğerinden çerçeve alabilir
  - Süreç yürütme süresi büyük ölçüde değişebilir
  - Daha yüksek verim, çok daha yaygın
- Yerel tahsis: her süreç yalnızca kendi tahsis edilen çerçeve kümesinden seçim yapar
  - Süreç başına daha tutarlı performans
  - Muhtemelen yeterince kullanılmayan bellek

Yerel ve Genel Tahsis Politikaları

(a) Orijinal konfigürasyon. (b) Yerel sayfa değişimi. (c) Küresel.

	Age
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

(a)

A0
A1
A2
A3
A4
A6
B0
B1
B2
B3
B4
B5
B6
C1
C2
C3

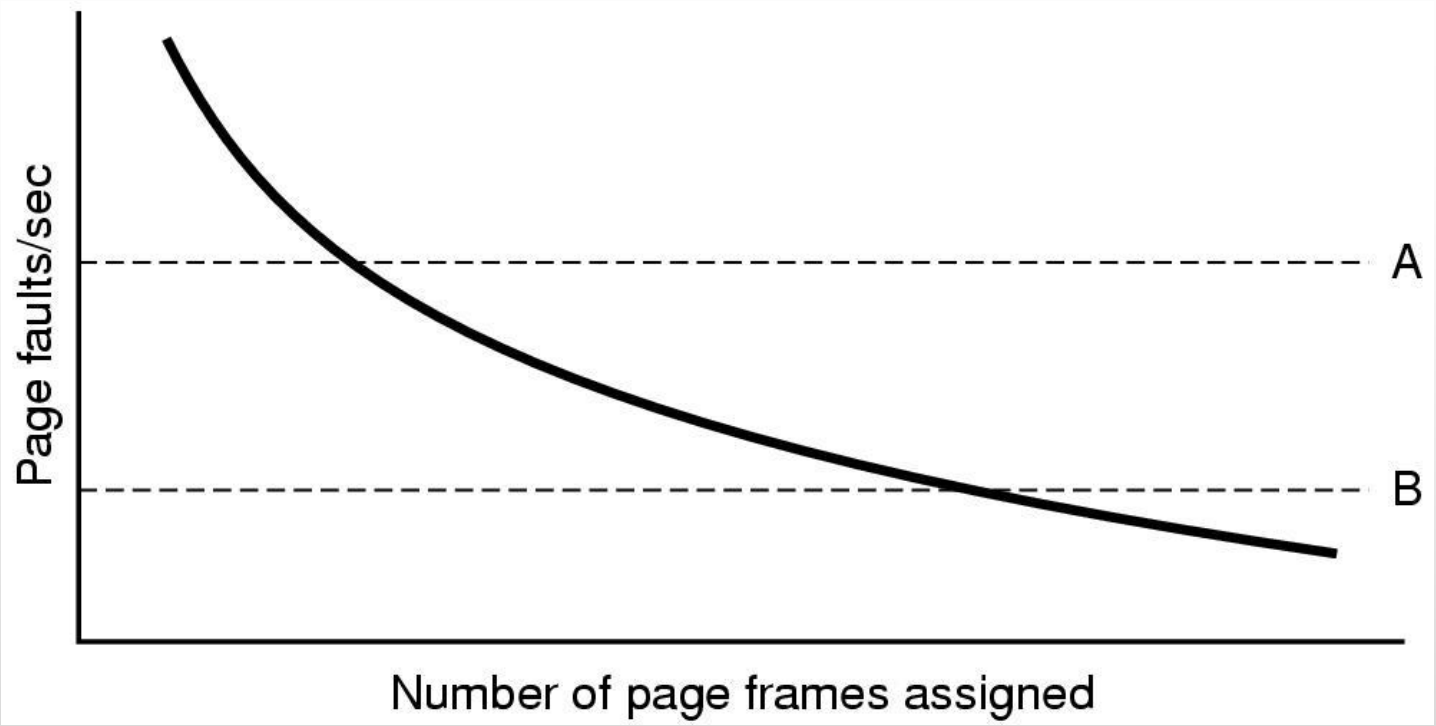
(b)

A0
A1
A2
A3
A4
A5
B0
B1
B2
A6
B4
B5
B6
C1
C2
C3

(c)

Yerel ve Genel Tahsis Politikaları

- .



Belady Anomalisi

- Çerçeve büyüklükleri,
- Erişim sıralaması: 1 2 3 4 1 2 5 1 2 3 4 5
- 4 çerçeve: sayfa hatası: 10, yer değiştirme 6

1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3

- 3 çerçeve: sayfa hatası: 9, yer değiştirme 6

1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	4
		3	3	3	2	2	2	2	2	4	4

Sayfa Boyutu

- Ek yük = s*e/p + p/2 [sayfa girişlerinin boyutu + parça]
  - p sayfa boyutudur,
  - s süreç boyutudur,
  - e, sayfa tablosu girdisinin boyutudur

- p = √(2s*e)
  - s = 1 MB, e = 8 bayt, p = 4 KB en uygun
  - 1 KB tipik
  - 4-8 KB yaygın
  - bu kaba bir yaklaşım

Bellek Eşlemeli Dosyalar

- Süreç, bir dosyayı sanal adres alanının bir parçasına eşlemek için sistem çağrısı yapar. (memory mapped file)
- Paylaşımlı bellek (shared memory) yoluyla iletişim kurmak için kullanılabilir.
- Süreçler aynı dosyayı paylaşır.
- Okumak ve yazmak için kullanılır.

Temizleme İlkesi

- İhtiyaç duyulduğunda kurban aramak yerine, ihtiyaç duymadan önce tahliye edilecek sayfaları bulmak için bir arka plan (daemon) programı olmalı
- Daemon çoğu zaman uyur, periyodik olarak uyanır
- Eğer "çok az" çerçeve varsa, çerçeveleri atar
- Tahliye etmeden önce temiz olduklarından emin olunmalı

Sanal Bellek Problemler

- İşletim sistemi, süreç oluşturulduğunda, yürütüldüğünde, sayfa hatası olduğunda, sonlandırıldığında sayfalamaya çok fazla dahil olur
- Özel sorun ve problemler
  - Sayfa hatası işleme
  - Komut yedekleme
  - Bellekteki sayfaları kilitleme
  - Yedekleme deposu - sayfaların diskte yerleştirileceği yer

Komut Yedekleme

- Otomatik arttırma yazmaçları, komut yürütülmeden önce veya sonra yükler.
  - Önce yüklerse, işlemin geri alınması gerekir.
  - Sonra yüklenirse, hiç yapılmaması gerekir.
- Komutun yedeklenmesi için donanım çözümü
  - Komut yürütülmeden önce mevcut komutu bir yazmaca kopyala
- Aksi takdirde işletim sistemi bataklığın derinliklerindedir

Bellekte Sayfa Kilitleme

- İşlem, G/Ç çağrısı yapar, verileri bekler
- Beklerken askıya alınır, yeni süreç okunur, ve yeni süreç sayfa hataları alır
- Global sayfalama algoritması => gelen veriler yeni sayfanın üzerine yazılır
- Çözüm: G/Ç'de devreye giren sayfaları kilitleyin

Yedekleme Deposu - Statik Bölme

- Sayfa takas edildiğinde diskte nereye konur?

- Süreç başladığında sabit bir bölüm tahsis edilir
- Boş parçalar, liste olarak yönetilir
- Süreç için yeterince büyük parça atanır
- Süreç tablosunda tutulan bölümün başlangıç adresi tutulur.
- Sanal adres uzayındaki sayfa ofseti, diskteki adrese karşılık gelir.
- Veri, metin, ve yığın için farklı alanlar atanabilir, yığın zamanla genişleyebilir

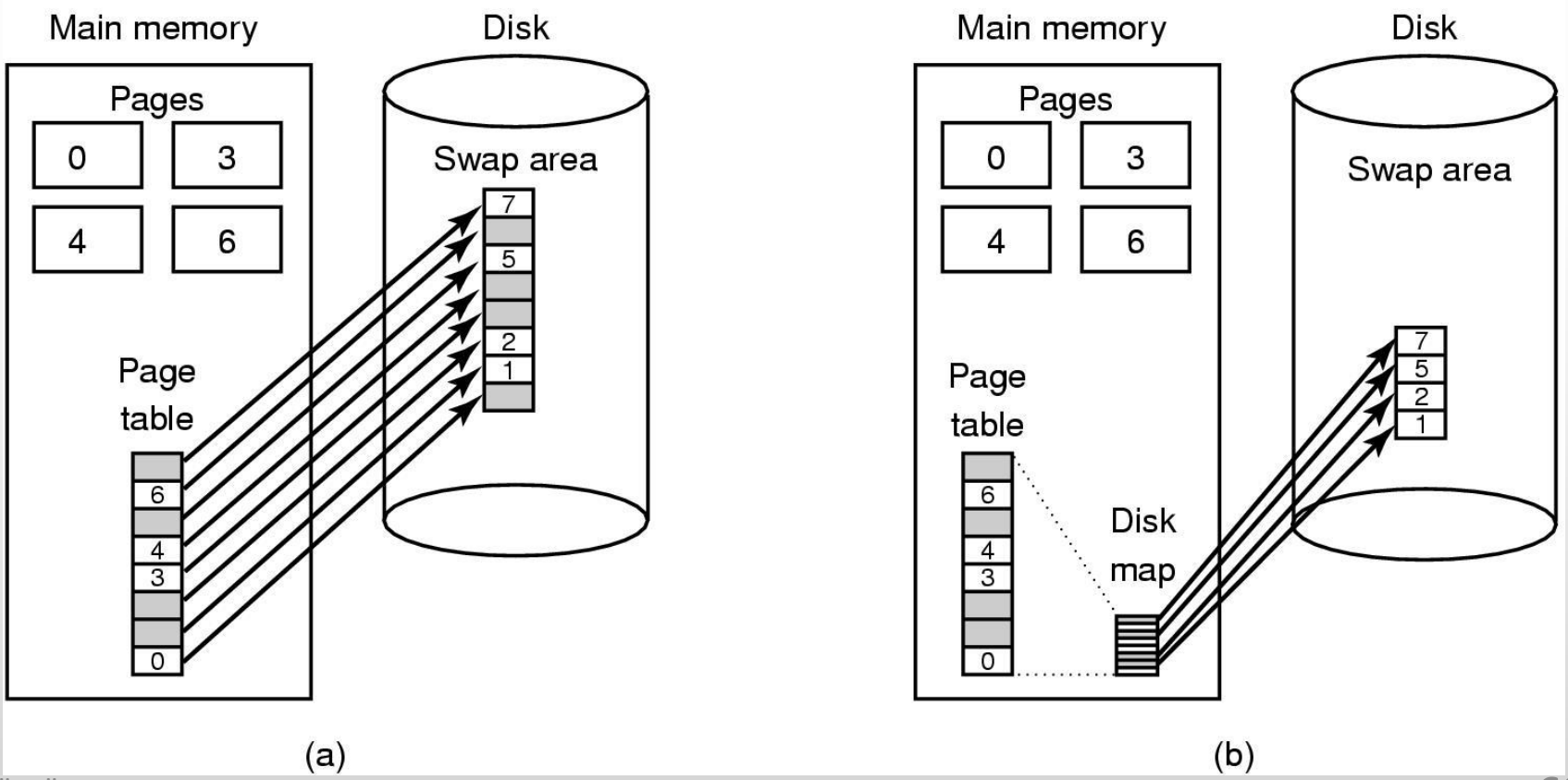
Yedekleme Deposu – Dinamik Yaklaşım

- Sayfa takas edildiğinde diskte nereye konur?

- Önceden disk alanı ayrılmaz.
- Gerektiğinde sayfaları içeri ve dışarı takas edilir.
- Bellekte disk haritasına ihtiyaç vardır

Takas Alanına Sayfalama

- (a) Statik takas alanına sayfalama (b) Sayfaları dinamik olarak yedekleme.

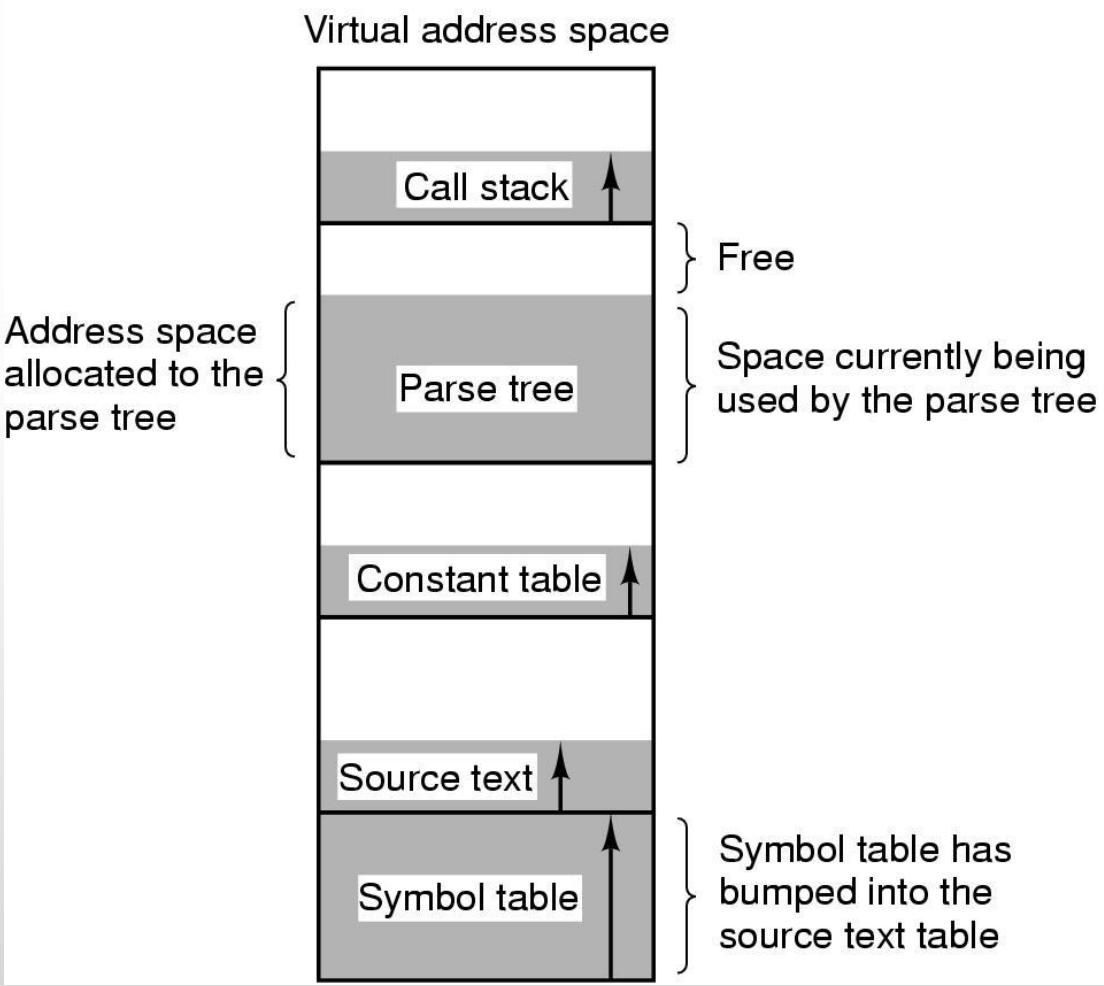


Kesimleme (segmentation)

- Bir derleyici, derleme ilerledikçe oluşturulan, aşağıdakileri içeren birçok tabloya sahiptir:
- Basılı listeleme için kaydedilen kaynak metin (toplu sistemlerde)(batch).
- Sembol tablosu – değişkenlerin adları ve nitelikleri.
- Kullanılan tamsayı, kayan noktalı sabitleri içeren tablo.
- Ayrıştırma (parse) ağacı, programın sözdizimsel (syntactic) analizi.
- Derleyici içinde prosedür çağrıları için kullanılan yığın.

Kesimleme (segmentation)

- Tek boyutlu adres uzayı.

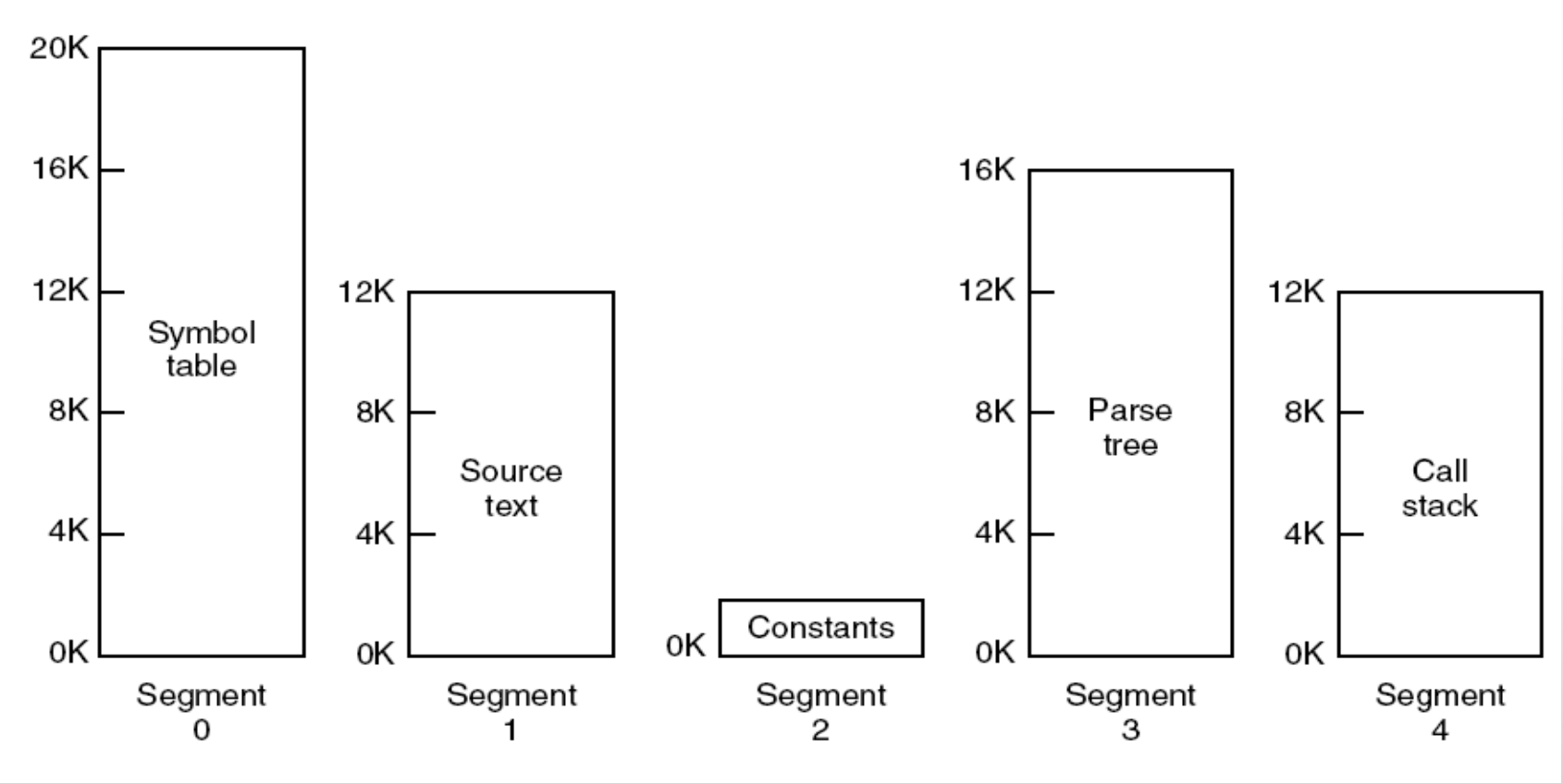


Kesimleme Avantajları

- Büyüyen ve küçülen veri yapılarının ele alınmasını basitleştirir
- Segment n'nin adres alanı (n,yerel adres) biçimindedir, burada (n,0) başlangıç adresidir
- Segmentleri diğer segmentlerden ayrı olarak derleyebilir
- Kütüphaneyi bir segmente koyabilir ve paylaşabilir
- Farklı segmentler için farklı korumalara (r,w,x) sahip olabilir

Kesimleme (segmentation)

- Bölümlere ayrılmış bellek



Sayfalama Ve Kesimleme Karşılaştırılması

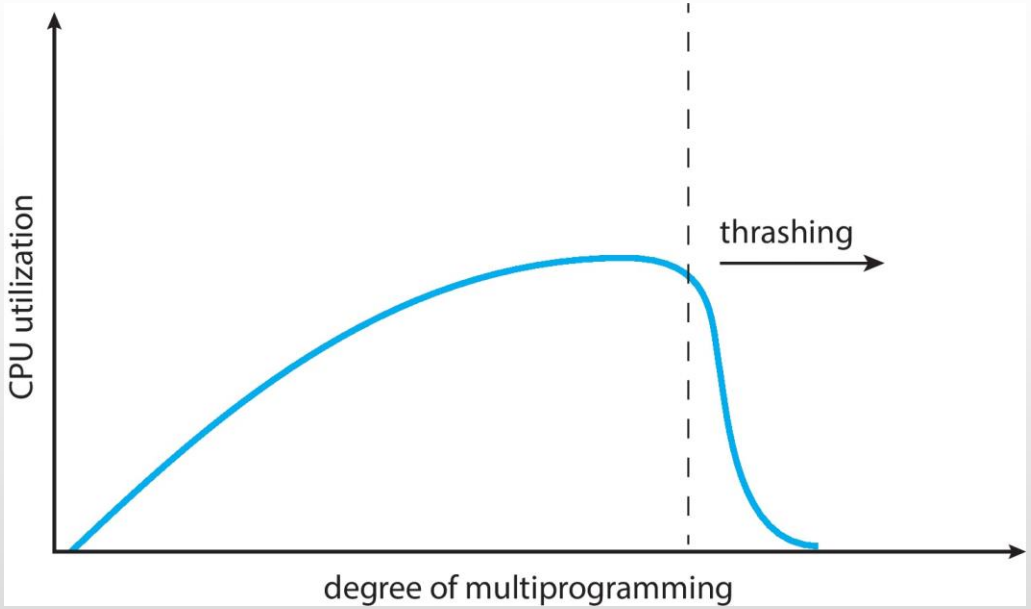
Durum	Sayfalama	Kesimleme
Programcı bu tekniğin kullanıldığının farkında olmalı mı?	Hayır	Evet
Kaç tane doğrusal adres alanı var?	1	Çok
Toplam adres alanı, fiziksel belleğin boyutunu aşabilir mi?	Evet	Evet
Prosedürler ve veriler ayırt edilebilir ve ayrı ayrı korunabilir mi?	Hayır	Evet
Boyutları değişkenlik gösteren tablolar kolayca yerleştirilebilir mi?	Hayır	Evet
Prosedürlerin kullanıcılar arasında paylaşılması kolaylaştırılmış mı?	Hayır	Evet
Bu teknik neden icat edildi?	Daha fazla fiziksel bellek almadan geniş bir doğrusal adres alanı elde etmek için	Programların ve verilerin mantıksal olarak bağımsız adres alanlarına ayrılmasına izin vermek için

Boşa Çabalama (Thrashing)

- Bir sürecin yeterli sayfası yoksa, sayfa hatası oranı çok yüksektir
  - Sayfa almak için sayfa hatası üretir
  - Mevcut çerçeveyi değiştirir
  - Ancak hızlı bir şekilde değiştirilen çerçeveye ihtiyaç duyar
  - Bu durum şunlara yol açar
    - Düşük CPU kullanımı
    - İşletim sistemi süreç sayısını artırması gerektiğini düşünür
    - Sisteme eklenen bir süreç daha

Boşa Çabalama (Thrashing)

- Bir süreç, sayfaları değiş tokuş etmekle meşgul



Ön Sayfalama

- Süreç başlangıcında oluşan çok sayıda sayfa hatasını azaltmak için
- Erişmeye çalışmadan önce, bir sürecin ihtiyaç duyacağı sayfaların tümü veya bir kısmı hazırlanır
- Ancak önceden sayfalanmış sayfalar kullanılmıyorsa G/Ç ve bellek boşa harcanmış olur

SON