Bölüm 9: Sanal Bellek

İşletim Sistemleri

Bellek Yönetimi

- Bir programın çalışabilmesi için;
  - diskten belleğe getirilmesi ve
  - bir süreç içerisine yerleştirilmesi gerekir.
- Ana bellek ve yazmaçlar, CPU'nun doğrudan erişebildiği alanlardır.
- Bellek birimi şu akışları görür:
  - adres + okuma isteği,
  - adres + veri + yazma isteği.
- Yazmaç erişimi bir CPU döngüsünde yapılır.
- Ana bellek erişimi, duraklamaya neden olacak kadar birçok döngü alabilir.
- Önbellek (*cache*), ana bellek ile yazmaçlar arasında bulunur.

Bellek Yönetimi

- Bellek, kısıtlı bir kaynak.
- Bu nedenle bellek hiyerarşisi var.
  - Önbellek (*cache*)(**hızlı**)
  - Ana bellek
  - Disk (**yavaş**)
- Bellek yöneticisi, bellek soyutlaması yapmak için hiyerarşi kullanır.

Bellek Yönetimi

- Bellek (RAM) önemli ve kısıtlı bulunan bir kaynaktır.
  - Programlar, genişleyerek kendilerine sunulan belleği doldururlar.
- Programcının istediği,
  - Belleğin korumalı, sonsuz büyük, sonsuz hızlı, ve kalıcı olması..☺
- Gerçekte olan
  - Akla gelen en iyi çözüm: bellek hiyerarşisi. ☹
  - Yazmaç, önbellek, bellek, disk, teyp.
- Bellek yöneticisi,
  - Belleği verimli bir şekilde yönetir.
  - Serbest bırakılan bellek alanlarını takip eder.
  - İhtiyaç halinde programlara tahsis eder.

Sanal Bellek - Takas

- Sistemde arka planda çalışan bir çok süreç var.
- Fiziksel bellek tüm süreçlere yetecek kadar büyük değil. ☹
  - Takas (*swap*)
    - Süreçleri belleğe getir, yürüt, takas ile diske geri götür.
  - Sanal bellek
    - Süreçleri kısmen bellekte olsalar bile çalıştır.
- **Takas** bir süreci birincil ve ikincil bellek arasında geçici olarak aktarır.
- **Talebe bağlı sayfalama** (*demand paging*),
  - Sürecin tamamını değil, yalnızca gerekli sayfaları belleğe getirir.

Sanal Bellek

- Programın yürütülebilmesi için bellekte olması gerekir,
  - Ancak, programın tamamı nadiren kullanılır.
  - *Hataları ele alan kod parçası, sıra dışı prosedürler, büyük veri yapıları.*
- Program kodunun tamamı aynı anda gerekli değildir.
- Kısmen yüklenmiş programlar yürütülebilir! ☺
  - Program artık fiziksel bellek limitiyle sınırlanamaz.
  - Program çalışırken daha az fiziksel bellek kullanır.
  - Aynı anda daha fazla program çalışabilir.
  - Programları belleğe yüklemek veya takas için daha az G/Ç gerekir.

Sanal Bellek

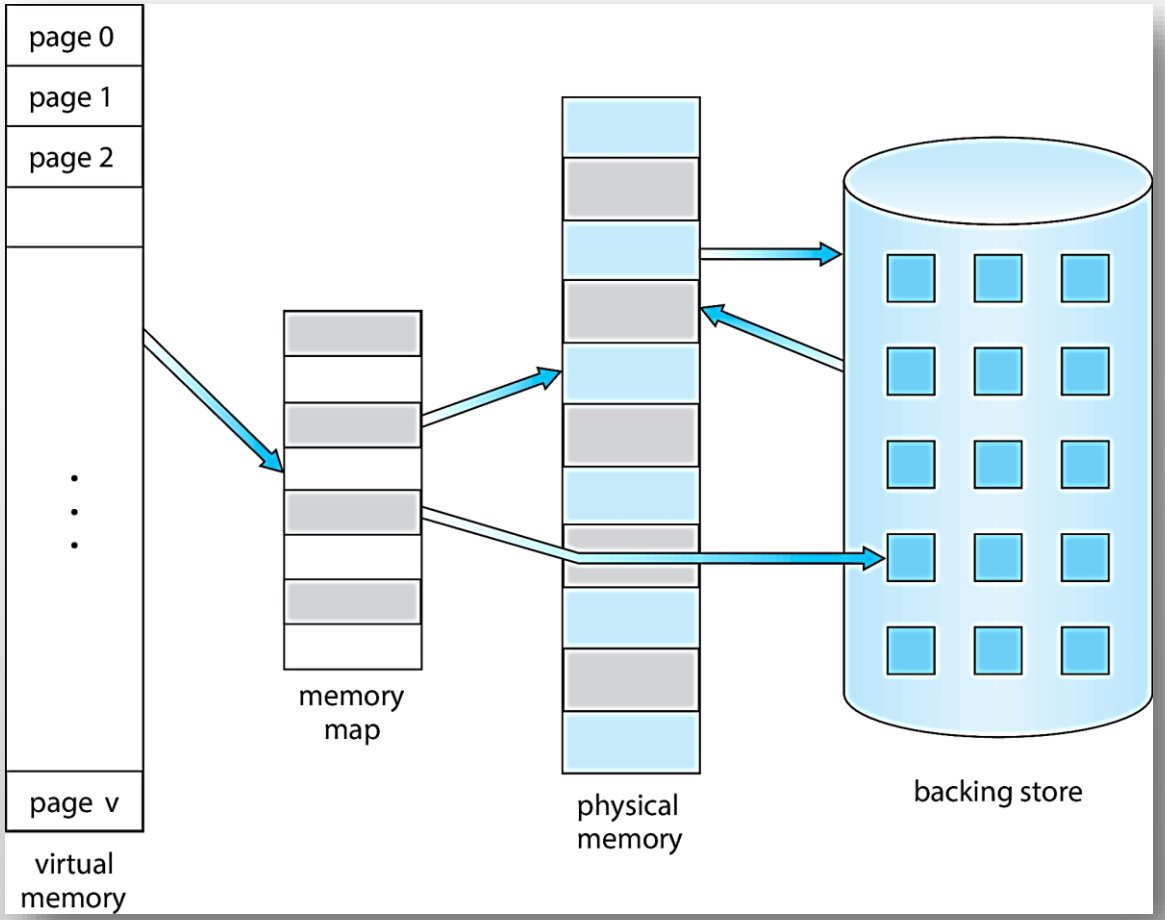
- Mantıksal belleğinin fiziksel bellekten ayrılması.
- Programın yürütülmesi için bir kısmının bellekte olması yeterli.
- Mantıksal adres alanı, fiziksel adres alanından çok daha büyük olabilir.
- Adres alanlarının birkaç süreç tarafından paylaşılmasına izin verir.

Sanal Bellek Adres Uzayı

- Sürecin bellekte yerleşimine dair mantıksal görünüm.
- Genellikle 0 adresinden başlar, alanın sonuna kadar bitişik adresler.
- Fiziksel bellek, sayfa çerçeveleri olarak düzenlenir.
- *MMU* mantıksal ve fiziksel adresleri eşler.
- Sanal bellek iki yolla uygulanır:
  - Talep olduğunda sayfalama (*demand paging*).
  - Talep olduğunda kesimleme (*demand segmentation*).

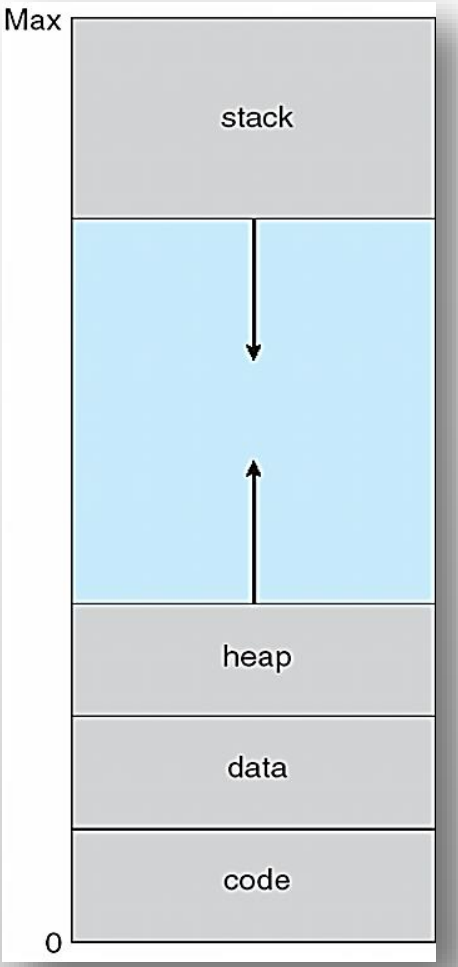
Sanal Bellek

- .

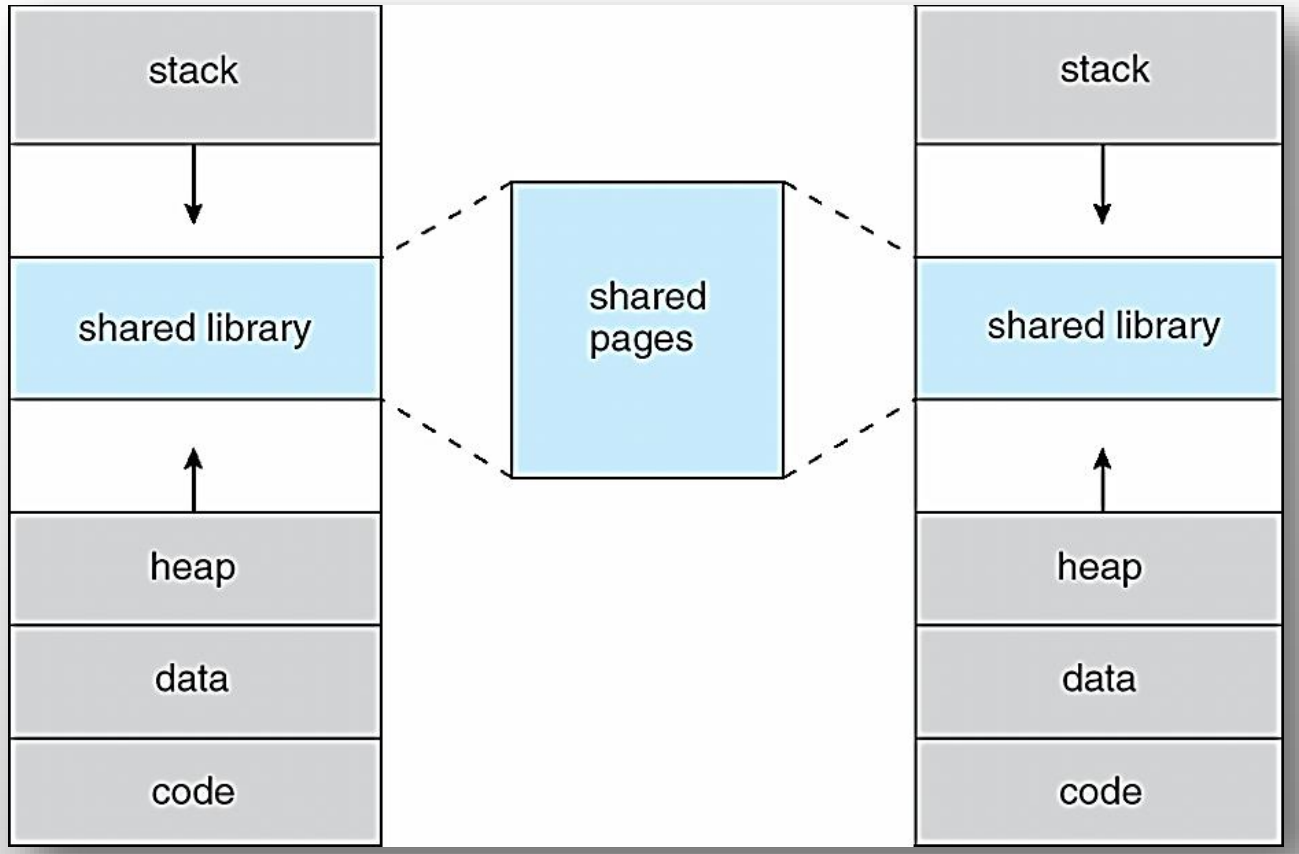


Sanal Bellek Adres Uzayı

- Yığıt maksimum mantıksal adreste başlar ve
  - büyüdükçe aşağı doğru iner.
- Kullanılmayan adres alanı, boşluk veya delik (*hole*).
- Yığıt veya yığın, bulunduğu sayfa dolana kadar fiziksel belleğe ihtiyaç yoktur.
- Sanal adres alanına eşleme yoluyla sistem kütüphaneleri paylaşılabilir.



Paylaşımlı Kütüphane



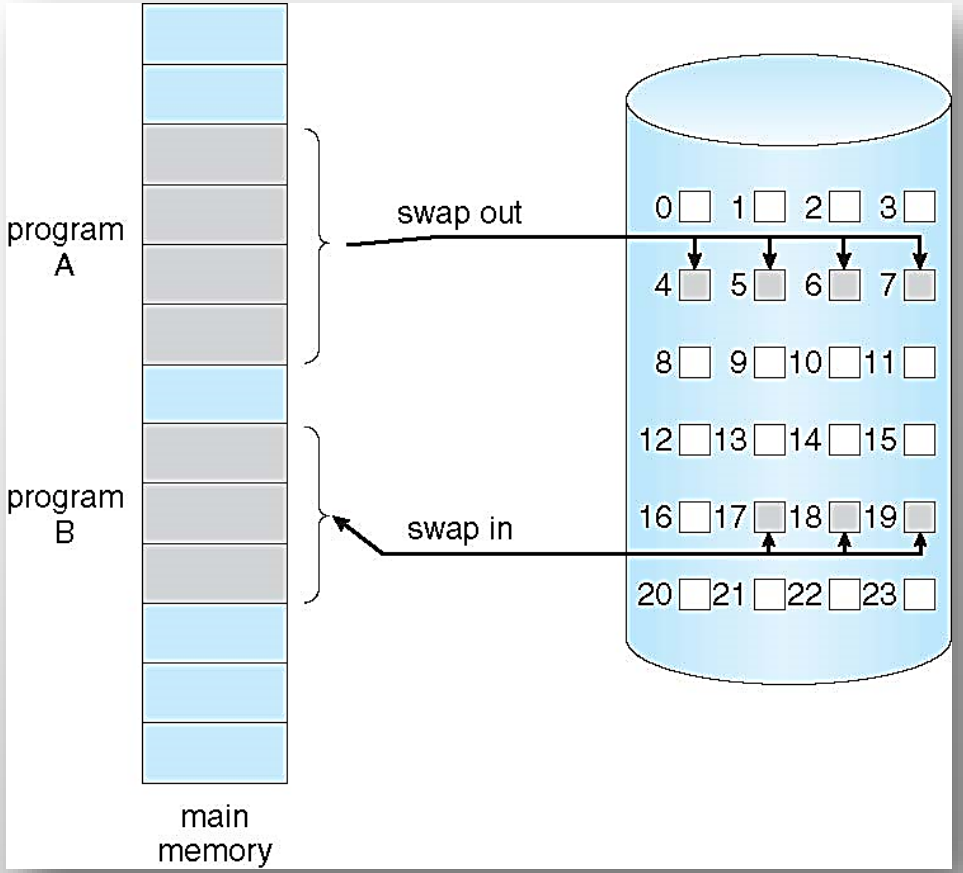
Sanal Bellek – Şişkin (Bloat) Yazılım Yönetimi

- Çok özellik taşıyan ve bellekte çok fazla yer kaplayan yazılım.
- Program belleğe sığmayacak kadar büyük olabilir.
- Programı parçalara ayırmak kötü bir fikir. ☹
- Sanal bellek
  - Her programın kendi adres alanı vardır.
  - Adres alanı, sayfa adı verilen parçalara bölünmüştür.
  - Her sayfa bitişik bir alandır ve bir fiziksel adres ile eşlenir.
  - Tüm sayfaların fiziksel bellekte olması gerekmez.
  - Gerekli bir sayfa bellekte olmadığında, işletim sistemi halleder. ☺

Talep Olduğunda Sayfalama

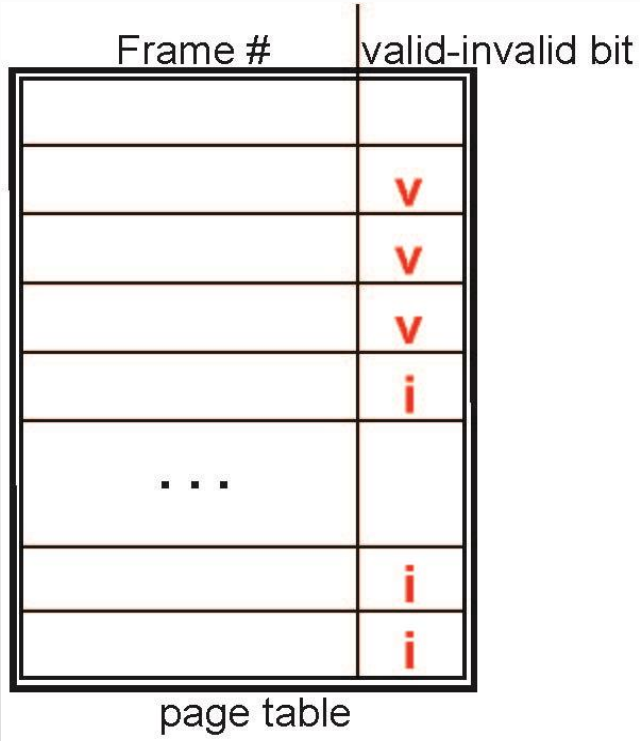
- Tüm süreç yükleme zamanında belleğe getirilebilir, veya
- Bir sayfa yalnızca gerektiğinde belleğe getirilebilir.
  - Daha az G/Ç gerekli, Gereksiz G/Ç yok, Daha az bellek gerekli.
- Takas (*swap*) sayfalamaya benzer.
  - Sayfa gerekli olduğunda referans verilir.
    - Referans geçersiz ise işlem iptal edilir.
    - Sayfa bellekte değil ise, belleğe getirilir.
- *Lazy swapper* - sayfa gerekmedikçe asla belleğe getirilmez.
- *Pager* - takas ile uğraşan birim.

Talep Olduğunda Sayfalama

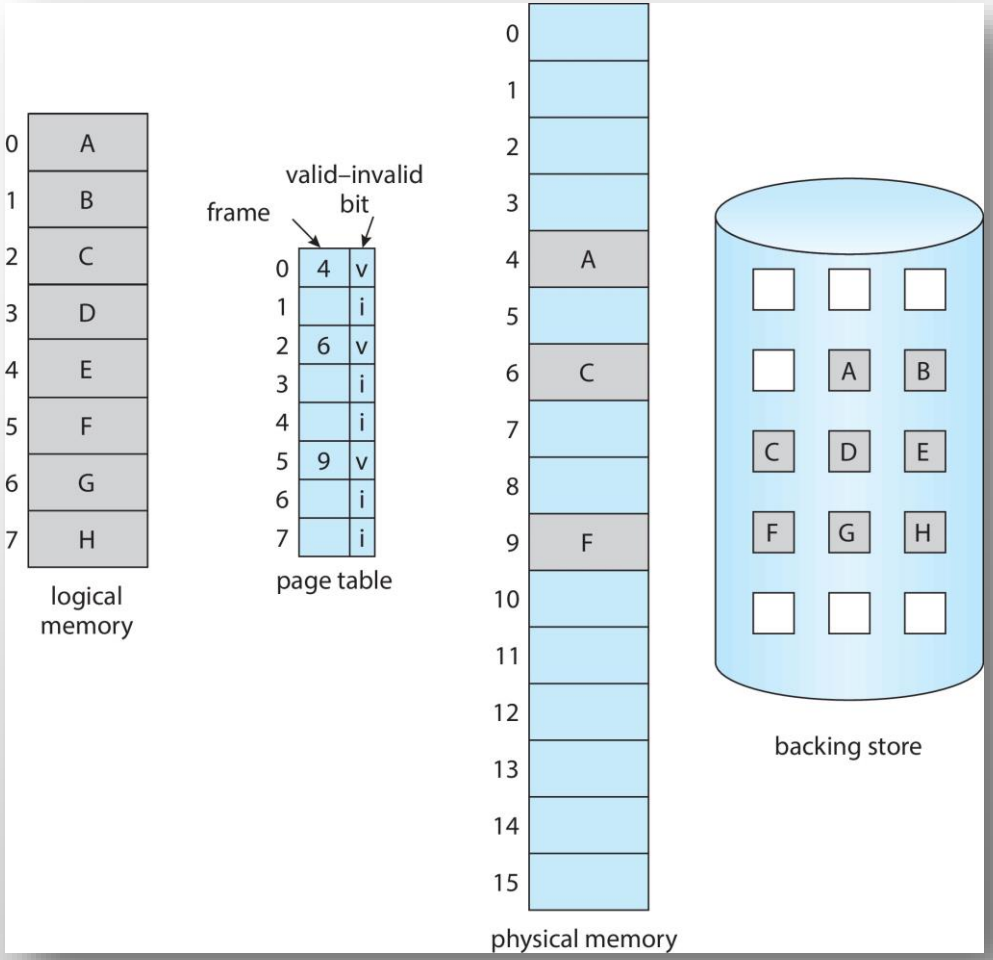


Geçerli Geçersiz Biti

- Sayfa tablosunda her satır için *geçerli-geçersiz* biti bulunur.
  - v - bellekte yerleşik (*valid*),
  - i - bellekte değil (*invalid*).
- Başlangıçta tüm çerçevelere *i* değeri atanır.
- *Geçerli-geçersiz* biti *i* ise → sayfa hatası.



Geçerli Geçersiz Biti



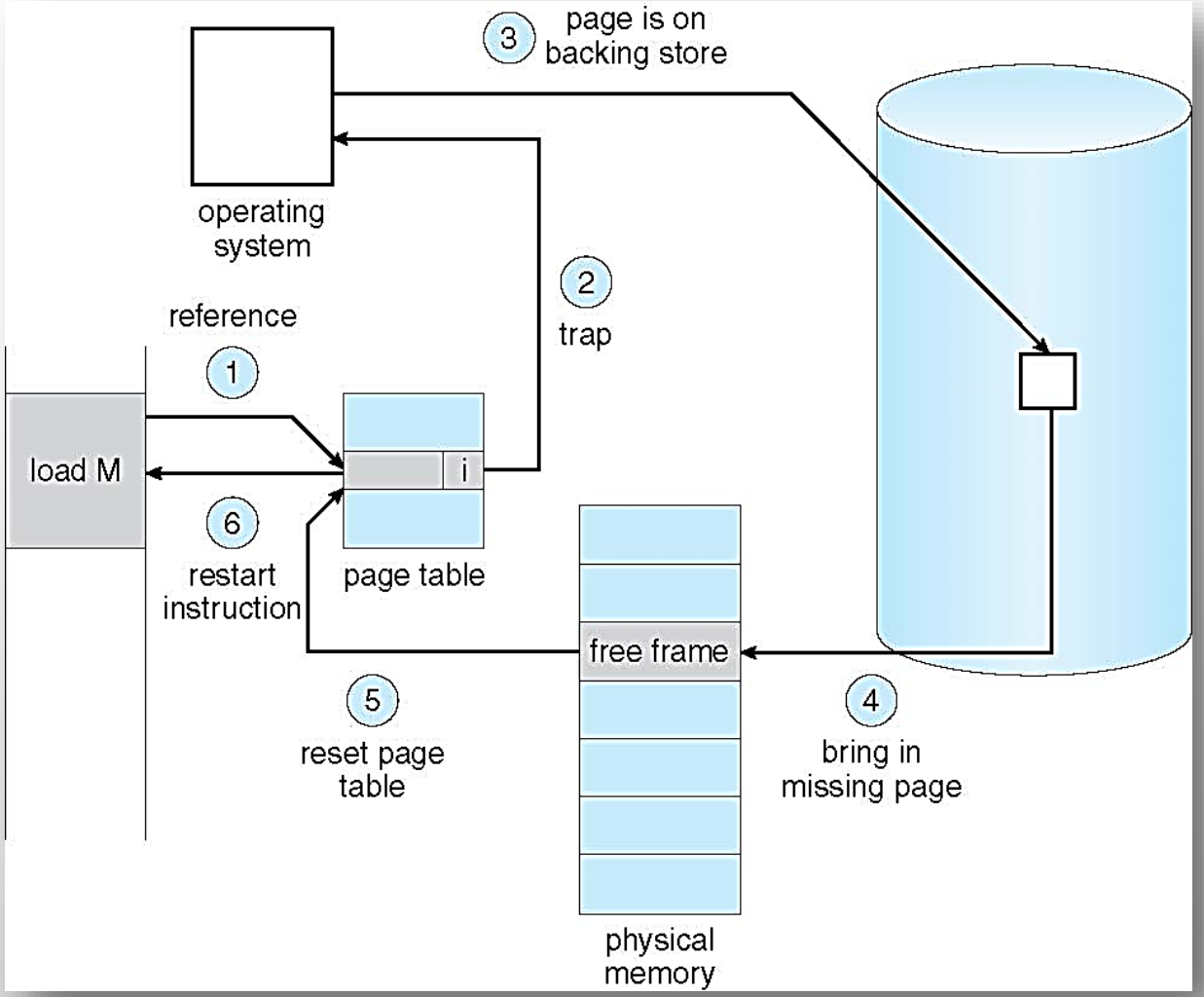
Hatalı Sayfa İşlemi

- *Mevcut/yok* biti, sayfanın bellekte olup olmadığını söyler.
- Adres bellekte yoksa ne olur?
- İşletim sistemine tuzak (*trap*).
  - Diske göndermek için bir sayfa seçilir.
  - Gerekli olan sayfa belleğe getirilir.
  - Komut yeniden başlatılır.

Hatalı Sayfa İşlemi

- Bellekte bulunmayan bir sayfaya erişim olduğunda,
  - Yürütme işletim sistemine geçer.
- İşletim sistemi sayfa tablosunu kontrol eder.
  - Geçersiz bir referans ise süreç iptal edilir.
- Sorun yalnızca erişilecek sayfanın bellekte olmaması ise,
  - Bellekte boş bir çerçeve (*frame*) bulunur.
  - Sayfayı çerçeve içine yerleştirmek için disk işlemi çizelgelenir.
  - Sayfa tablosu güncellenir.
  - Geçerli, mevcut (*valid*) biti atanır.
- Sayfa hatasına neden olan *komut* yeniden başlatılır.

Hatalı Sayfa İşlemi



Talep Olduğunda Sayfalama Adımları

- İşletim sistemi *yürütmeyi* eline alır.
- Yazmaçlar ve süreç durumu *kaydedilir*.
- Kesmenin bir *sayfa hatası*ndan kaynaklandığı belirlenir.
- Sayfa referansının *geçerli olup olmadığı* kontrol edilir.
- Sayfanın *diskteki konumu* belirlenir.
- Disk işlemi için *sırada* beklenilir.
- Diskte arama ve/veya *gecikme süresi* boyunca beklenilir.
- Sayfa *boş bir çerçeveye* aktarılmaya başlanır.

Talep Olduğunda Sayfalama Adımları

- Beklerken *CPU* başka bir sürece tahsis edilir.
- *Disk G/Ç* alt sisteminden kesme üretilir (*G/Ç tamamlandı*).
- Aktif süreç için yazmaç ve süreç durumu *kaydedilir*.
- Kesme kaynağının diskten olduğu belirlenir.
- Sayfanın bellekte olduğunu göstermek için *sayfa tablosu güncellenir*.
- *CPU*'nun tekrar tahsis edilmesi beklenir.
- Yazmaçlar, süreç durumu ve güncel sayfa tablosu *geri yüklenir*.
- Yarıda kesilen komut *tekrar başlatılır*.

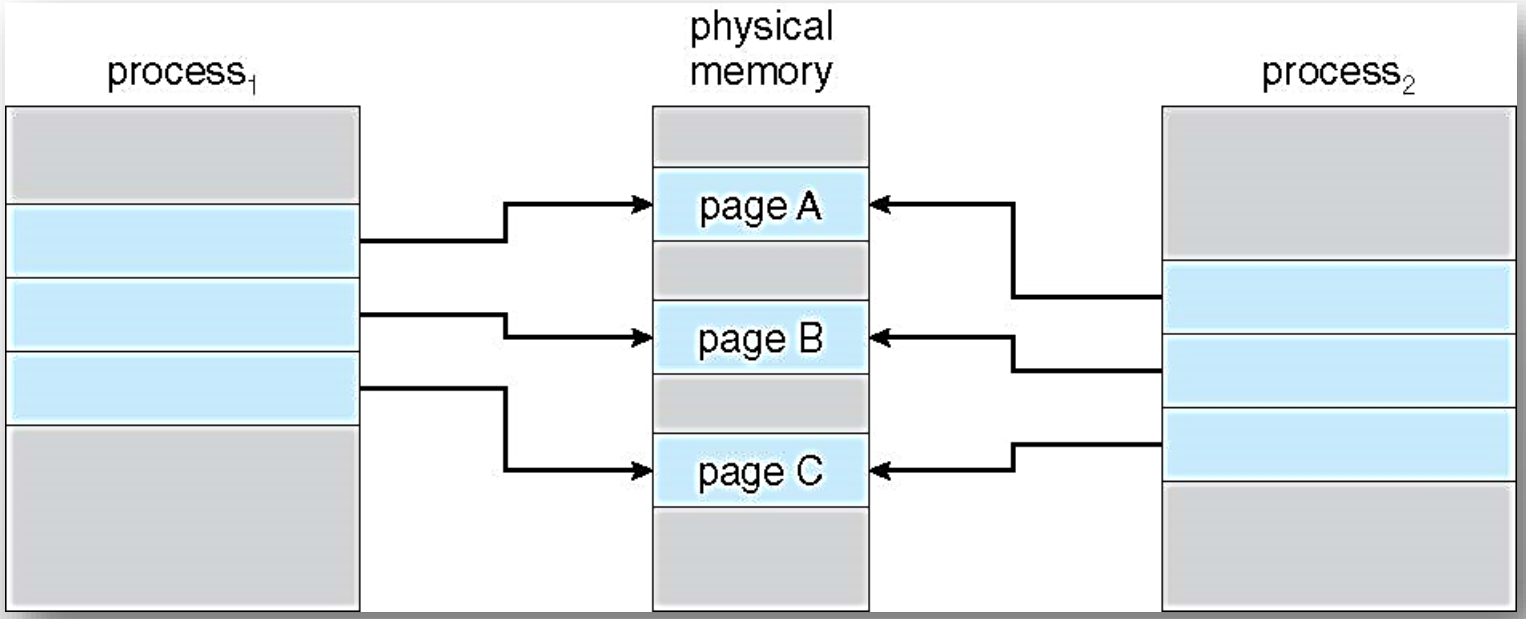
Bellek Eşlemeli (Memory Mapped) Dosyalar

- Süreç, bir dosyayı sanal adres alanının bir parçasına eşleyebilir.
- Paylaşımlı bellek (*shared memory*) yoluyla iletişim için kullanılır.
- Süreçlerin aynı dosyayı paylaşabilmesi sağlanır.
- Okuma ve yazma işlemleri yapılabilir.

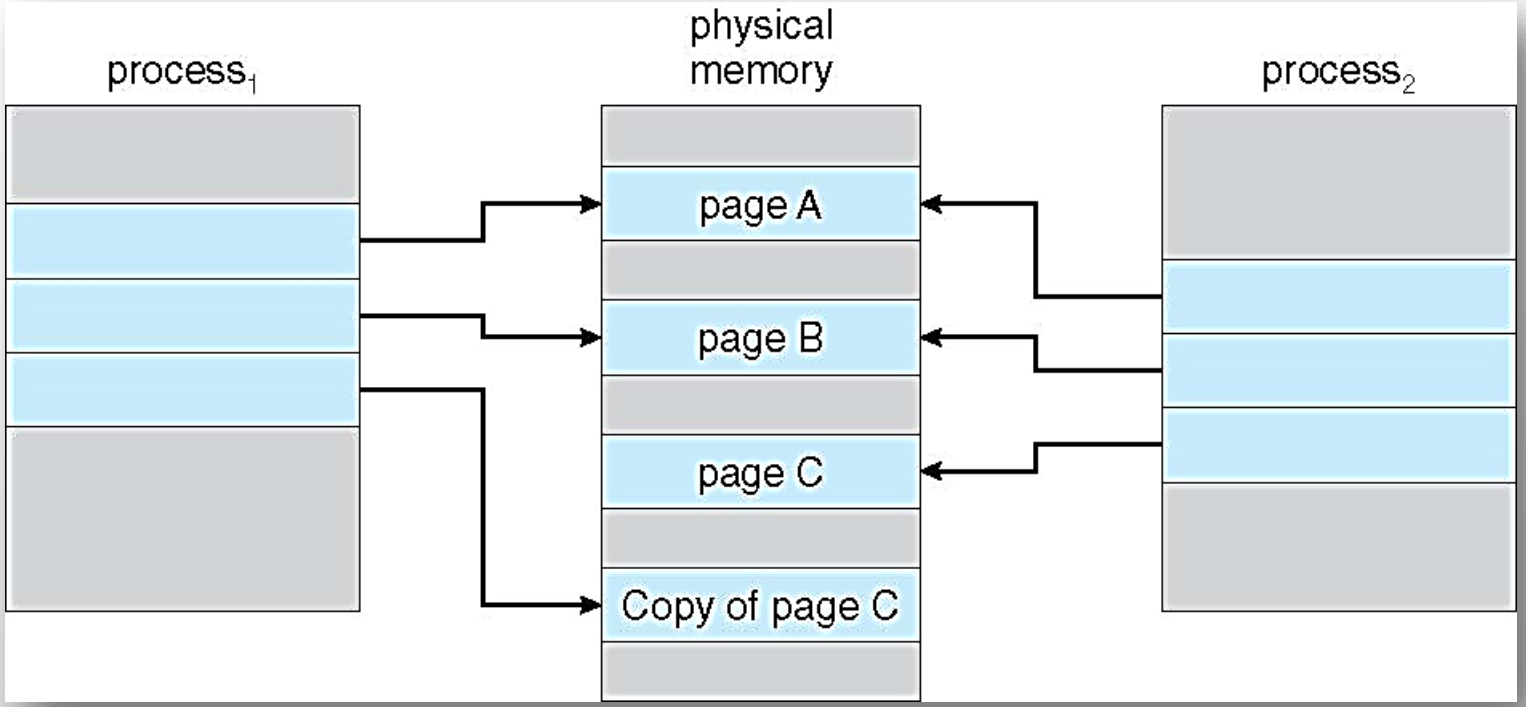
Yazarken Kopyalama (Copy-on-Write)

- Süreçlerin başlangıçta aynı sayfaları bellekte paylaşmasına izin verir.
  - Paylaşılan sayfa değiştirileceğinde, sayfa kopyalanır.
- Sadece değiştirilen sayfalar kopyalanır,
  - Süreçler az maliyetle oluşturulabilir.
- Boş sayfalar, talep üzerine bir havuzdan tahsis edilir.
  - Talepler için, havuzda her zaman boş çerçeveler tutulur.
  - Hata durumunda boş çerçeve aramak ve boşaltmak gerekmez.

Yazarken Kopyalama



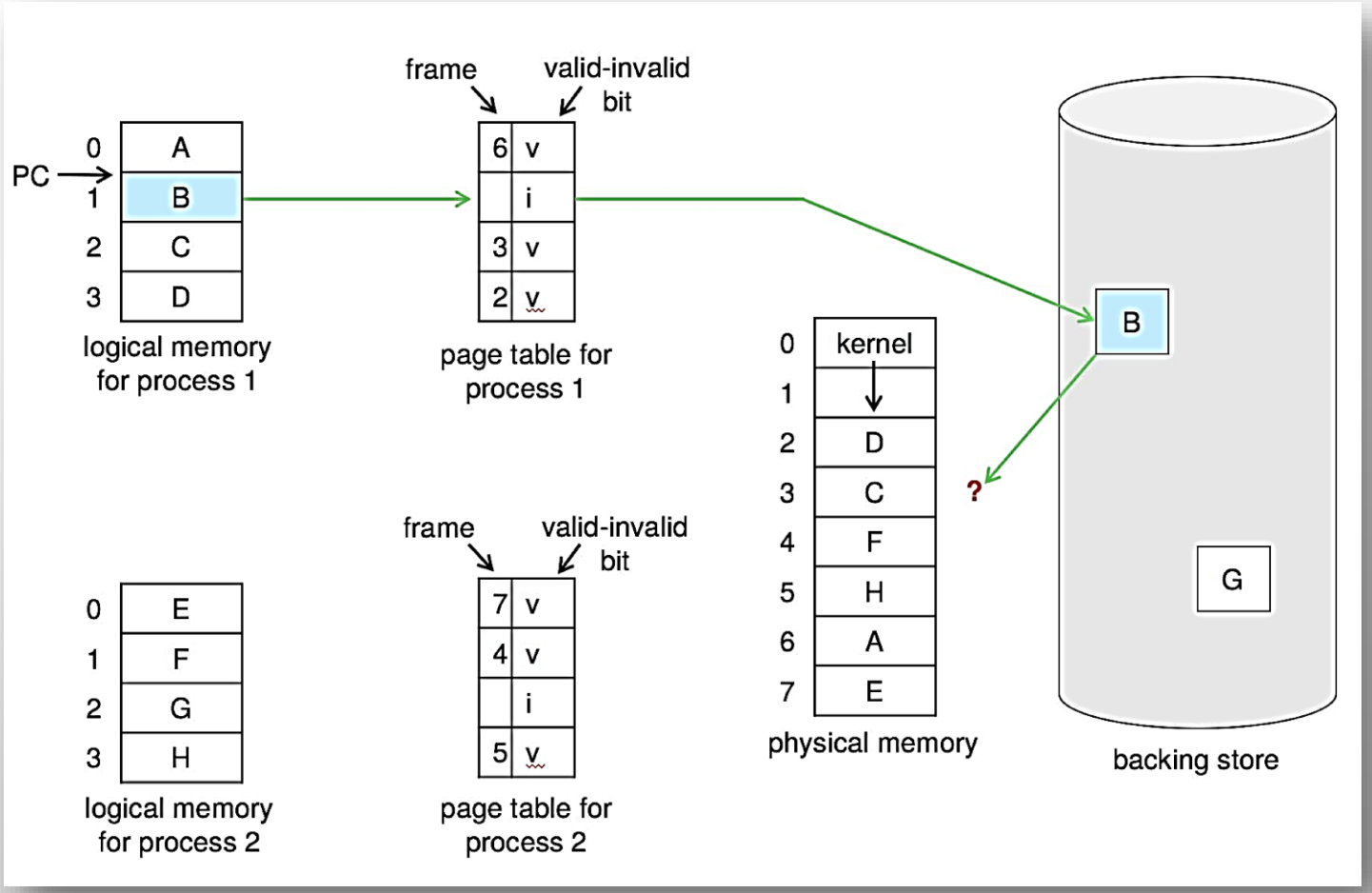
Yazarken Kopyalama



Boş Çerçeve Bulunamazsa

- Sayfalar süreçler tarafından kullanılmakta iken, yeni istekler gelebilir.
- Bellek kısıtlı, her bir sürece ne kadar tahsis edilmeli?
- **Sayfa takası**:
  - kullanımda olmayan sayfaların bulunması ve çıkarılması.
- **Algoritma**:
  - süreç sonlandırılsın mı?
  - sayfalar takas mı edilsin?
- **Performans**:
  - minimum sayfa hatasıyla karşılaşma. ☺
  - aynı sayfa birçok kez takas edilebilir.

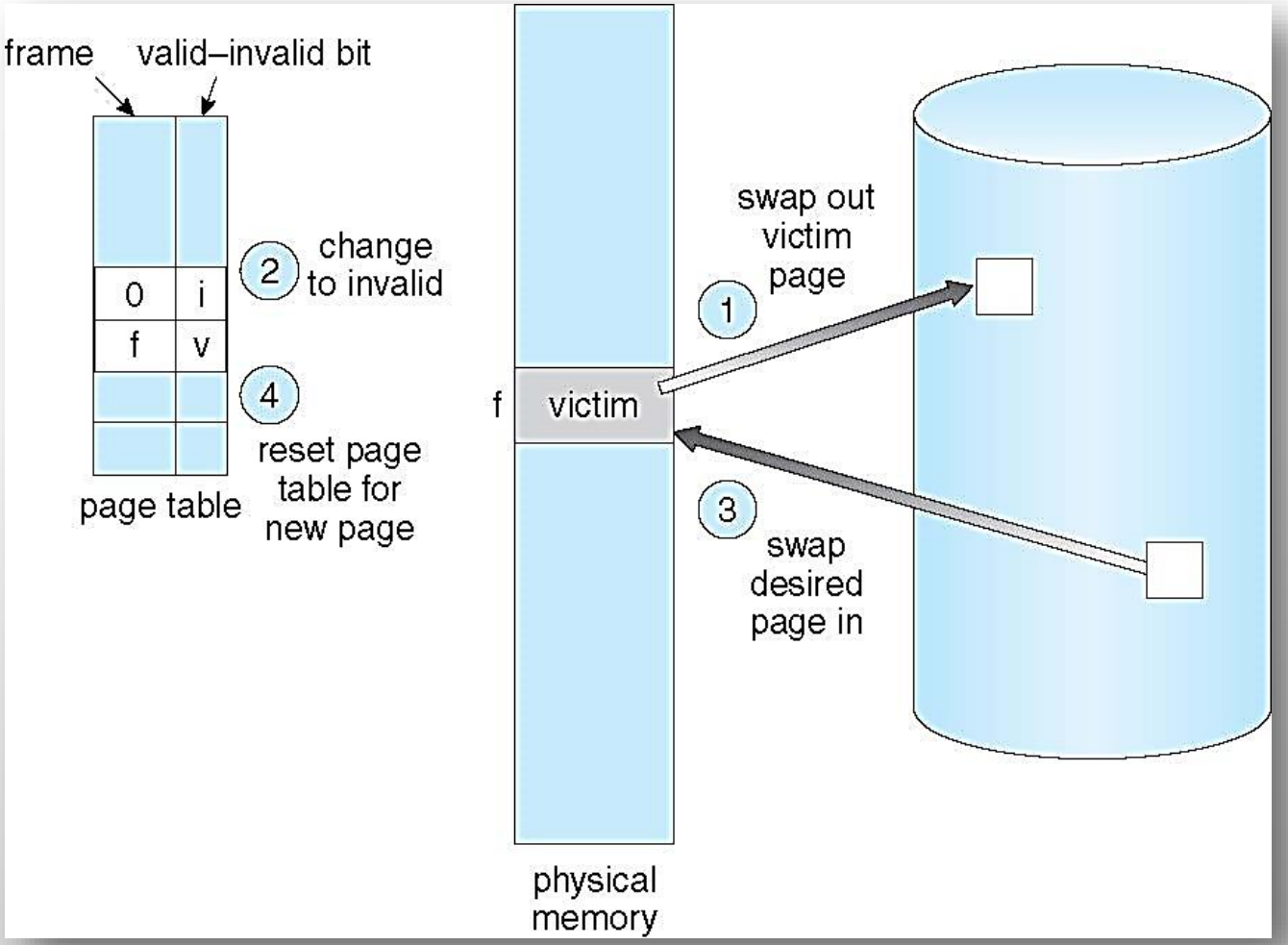
Sayfa Yer Değiştirme (Page Replacement)



Sayfa Yer Değiştirme

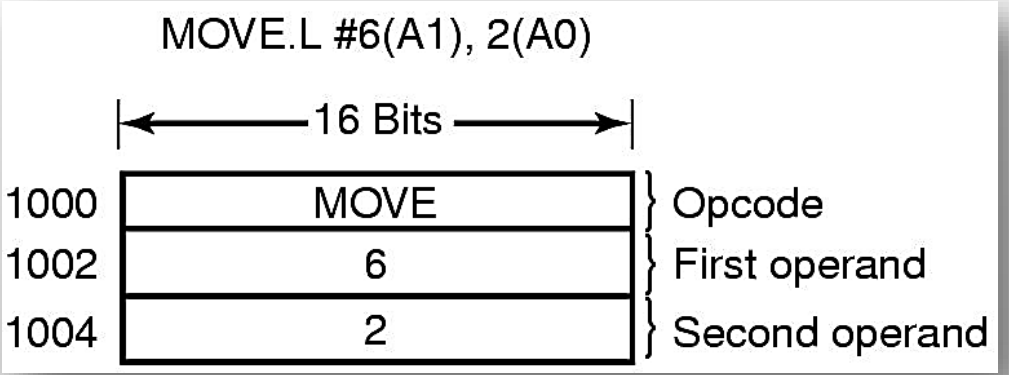
- Gerekli sayfanın diskteki konumu bulunur.
- Bellekte boş bir çerçeve aranır.
  - Boş bir çerçeve bulunursa kullanılır.
  - Bulunamazsa, bir kurban (*victim*) aranır. ☺
  - Kurban kirli (*dirty*) ise diske yazılır.
- İstenen yeni sayfa boş çerçeveye taşınır.
- Sayfa tablosu (*page table*) güncellenir.
- Tuzağa (*trap*) neden olan komut yeniden başlatılır.

Sayfa Yer Değiştirme



Sayfa Hatasına Neden Olan Bir Komut

- Komut tekrar hangi adresten başlatılır?
- Program sayacı, komutun hangi bölümünün hatalı olduğuna bağlıdır.
- İşletim sistemi 1002'de hata aldığında,
  - 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* (*miss rate*)
- EAT = Effective access time.
- EAT = $100(1-p) + (25)10^6 p = 100 + 24999900*p$.
- p = 0.001 ise, EAT = 25099.9 *ns.*
- EAT < 110 *ns* olması için, $p < 0.0000004$ olmalı.

Sayfa Yer Değiştirme Algoritmaları

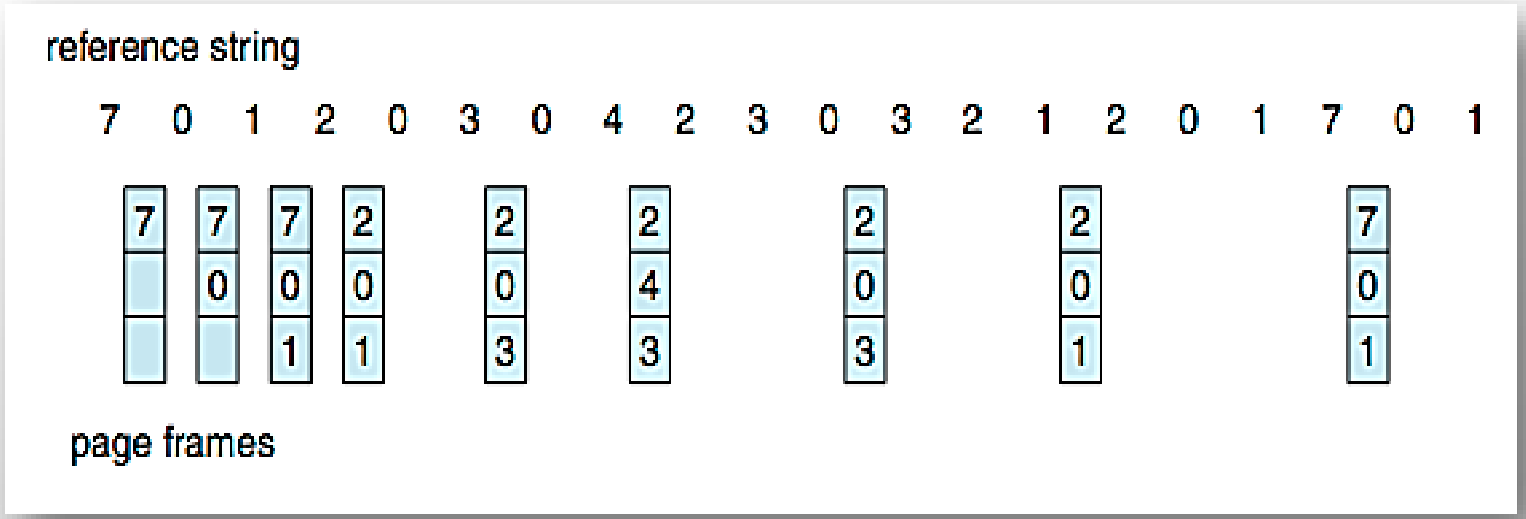
- Optimum (*optimal*)
- Yakın zamanda kullanılmayan (*not recently used*)
- İlk giren ilk çıkar (*first-in first-out*)
- İkinci şans (*second chance*)
- Saat (*clock*)
- En son kullanılan (*least recently used*)
- Çalışma kümesi (*working set*)
- Çalışma kümesi saat (*working set clock*)

Optimum Sayfa Değişimi

- Tarifi kolay, uygulanması imkansız.
- En uzun süre kullanılmayacak olan sayfa seçilir.
- Sayfalar kendisine erişimden önce çalıştırılacak komut sayısı ile etiketlenir.
- İhtiyaç halinde en yüksek etikete sahip sayfa çıkarılır.
- İşletim sistemi hangi sayfaya ne zaman erişileceğini bilemez (*crystal ball*).
- Algoritmaların performansını karşılaştırmak için kullanılır.

Optimum Sayfa Değişimi

- Sayfa hatası (*page fault*) 9 kez,
- Sayfa değiştirme (*page replacement*) 6 kez.



Yakın Zamanda Kullanılmayan Sayfa Değişimi

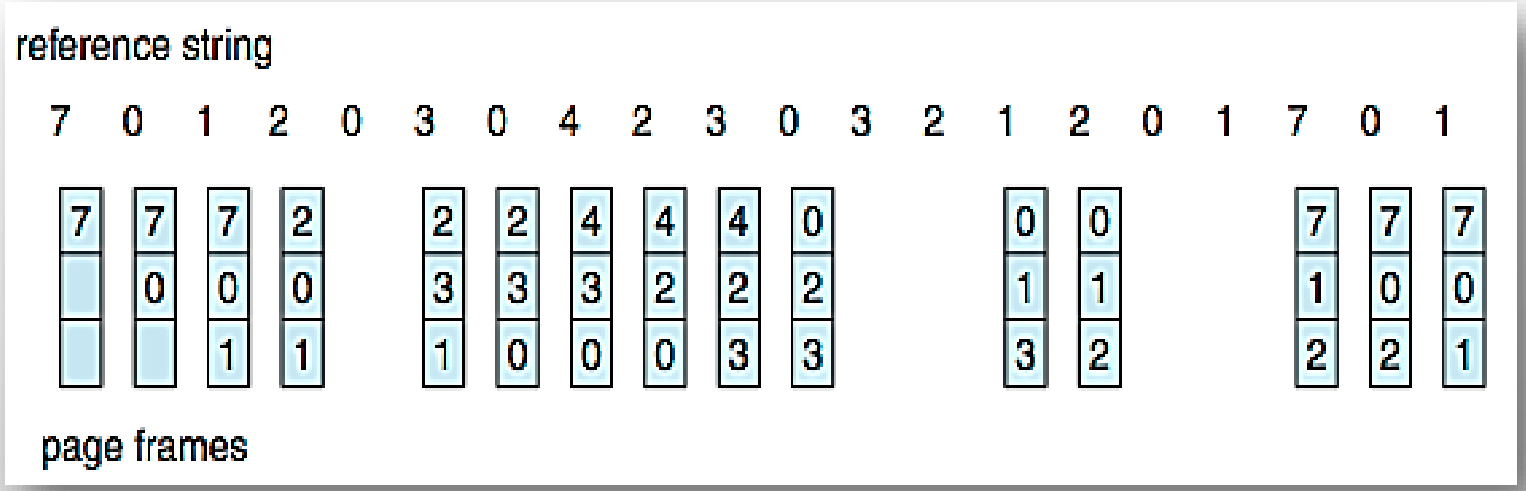
- Sayfalara kullanım zamanlarına göre öncelik değerleri atanır.
- Sayfa çıkarılacağında en düşük öncelikli sayfa seçilir.
- Her bir sayfa için *R (referenced)* ve *M (modified) bitleri* bulunur.
- Süreç başladığında, bitlere 0 atanır, periyodik olarak, *R biti* temizlenir.
- Sayfa dört farklı durumda olabilir.
  - Erişilmedi, değiştirilmedi.
  - Erişilmedi, değiştirildi.
  - Erişildi, değiştirilmedi.
  - Erişildi, değiştirildi.

İlk Giren İlk Çıkar Sayfa Değişimi

- FIFO (first in first out).
- Zamana göre sıralı sayfa listesi tutar.
  - En son erişilen sayfa listenin sonunda durur.
- İhtiyaç halinde en eski sayfa, yani sıranın başındaki tahliye edilir.
- Uygulaması kolay bir algoritma. ☺
- En eski sayfa en yoğun kullanılan sayfa olabilir!
- Kullanım bilgisini dikkate almaz.

İlk Giren İlk Çıkar Sayfa Değişimi

- *En eski olanı* değiştirir.
- Kötü performans ama basit.
- Sayfa hatası 15 kez, Sayfa değiştirme 12 kez.



İkinci Şans Sayfa Değişimi

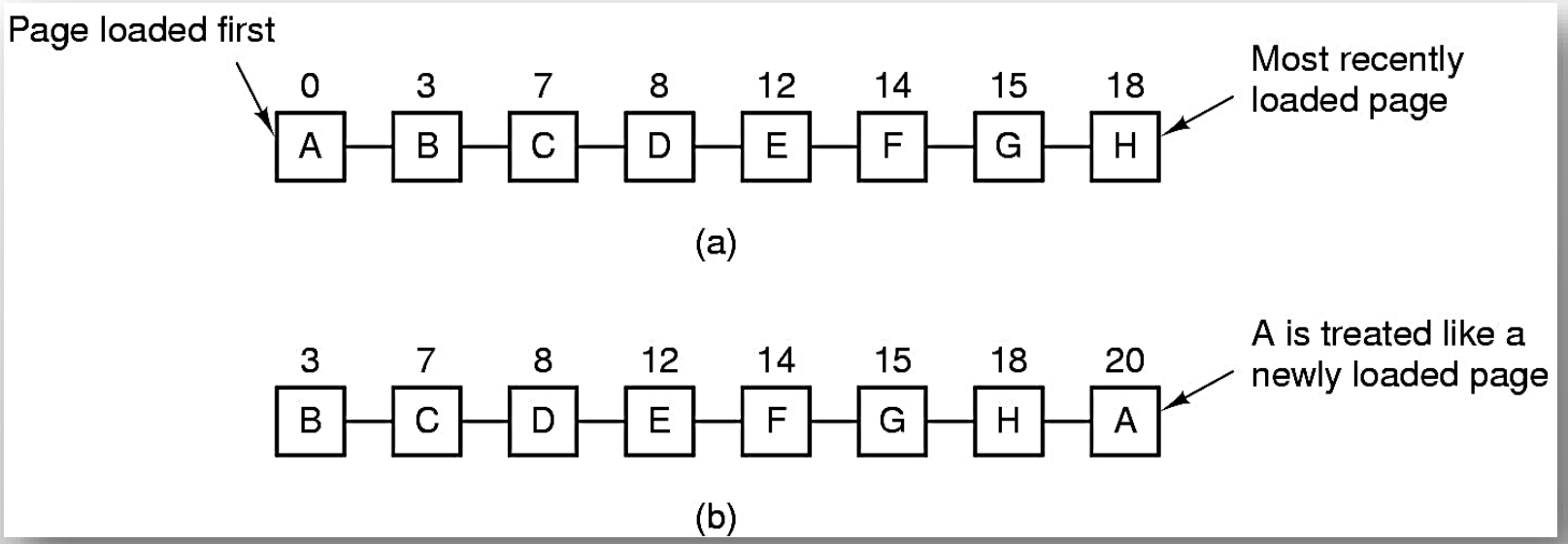
- *İlk giren ilk çıkar* yönteminin gelişmiş bir versiyonu.
- Sık kullanılan sayfaların değiştirilmesi önlenir.
  - *R (referenced)* bitine bakılır.
  - *R* değeri *0* ise, sayfa eski ve erişilmemiştir.
  - *R* değeri *1* ise, sayfaya ikinci bir şans verilir.
  - Eğer tüm sayfalara erişildiyse, *İlk giren ilk çıkar* gibi çalışır.

İkinci Şans Sayfa Değişimi

(a) FIFO mantığında sıralı sayfalar.

(b) Sayfa hatası varsa, *A*'nın *R* biti 1 ise, *A*'ya *ikinci şans* verilir.

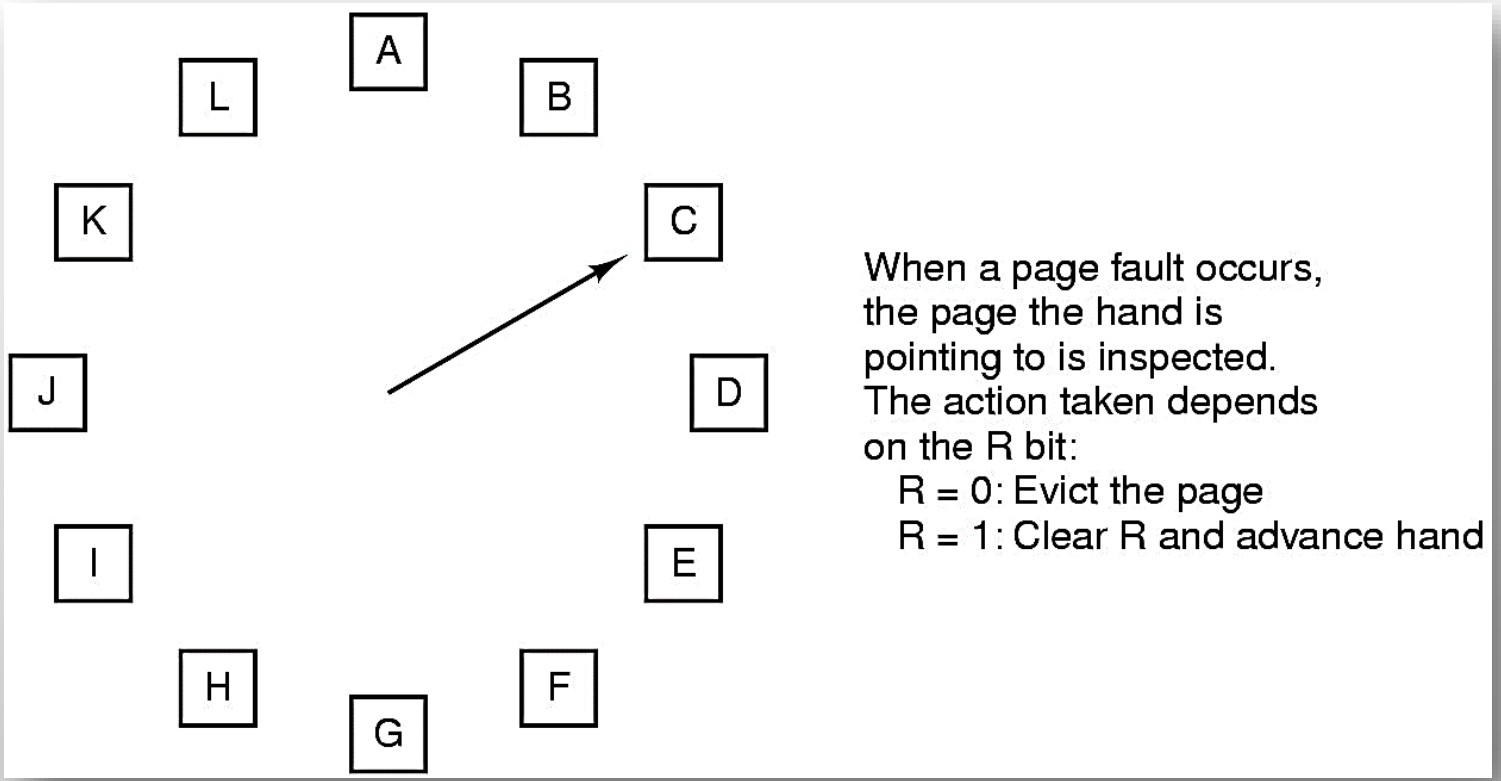
Sayılar: sayfaların gelme zamanlarını gösterir.



Saat Sayfa Değişimi

- *İkinci şans* algoritmasına benzer, işleyişi farklıdır.
- *İkinci şans* algoritması, sayfaları listede sürekli olarak hareket ettirir.
- Alternatif olarak sayfa çerçeveleri *saat* şeklinde dairesel bir listede tutulur.
- Sayfa tablosundaki bir işaretçi (*saat işareti*) kullanılarak sayfalar işlenir.

Saat Sayfa Değişimi

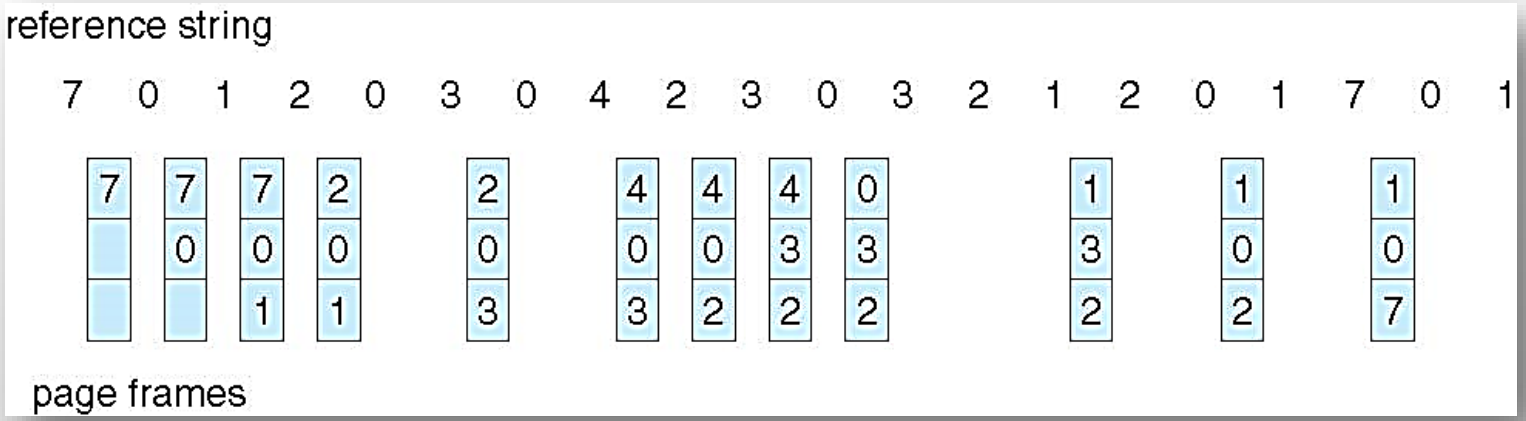


En Son Kullanılan Sayfa Değişimi

- *Optimum* algoritmaya yakın bir yaklaşım.
- *Uzun süre kullanılmayan sayfalar, bir daha kullanılmaz* mantığı.
- Sayfa hatası olduğunda, en uzun süre kullanılmayan sayfa çıkarılır.
- Süreyi kaydetmek için,
  - Bellekteki sayfaların yer aldığı bir bağlı liste veri yapısı,
  - Donanım sayacı, veya
  - Bir matris kullanılır.

En Son Kullanılan Sayfa Değişimi

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

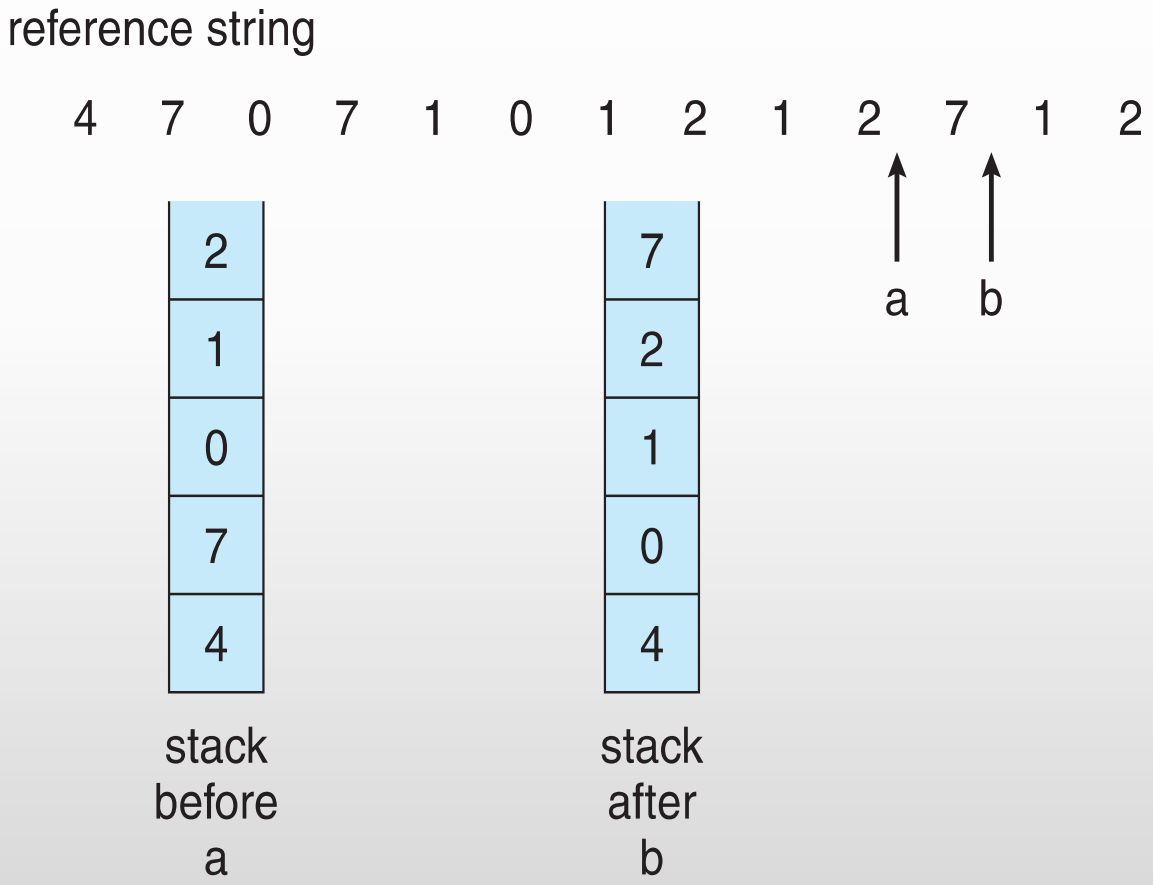


En Son Kullanılan Sayfa Değişimi

- Maliyetli bir algoritma, verimli bir şekilde gerçeklenmesi zor.
- Sayfa için fazladan bir sayaç (*kullanım süresi*) kullanılır. (*donanımsal*)
  - En eski sayfa değiştirilir.
  - Sayaç için *donanım desteği* gerekir.
  - Sayfa hatasında, en eski olanı bulmak için tüm sayaçlar kontrol edilir.
- Bir yığında sayfa numaraları saklanır. (*yazılımsal*)
  - Yığının en altındaki sayfa değiştirilir.
  - Sayfa çerçevelerin boyutunda bir *yığın gerekli*.
  - Sayfa zaten yığının içindeyse, önce çıkarılır, sonra yığına geri konur.
  - Değilse, yığına konur. (*push*)

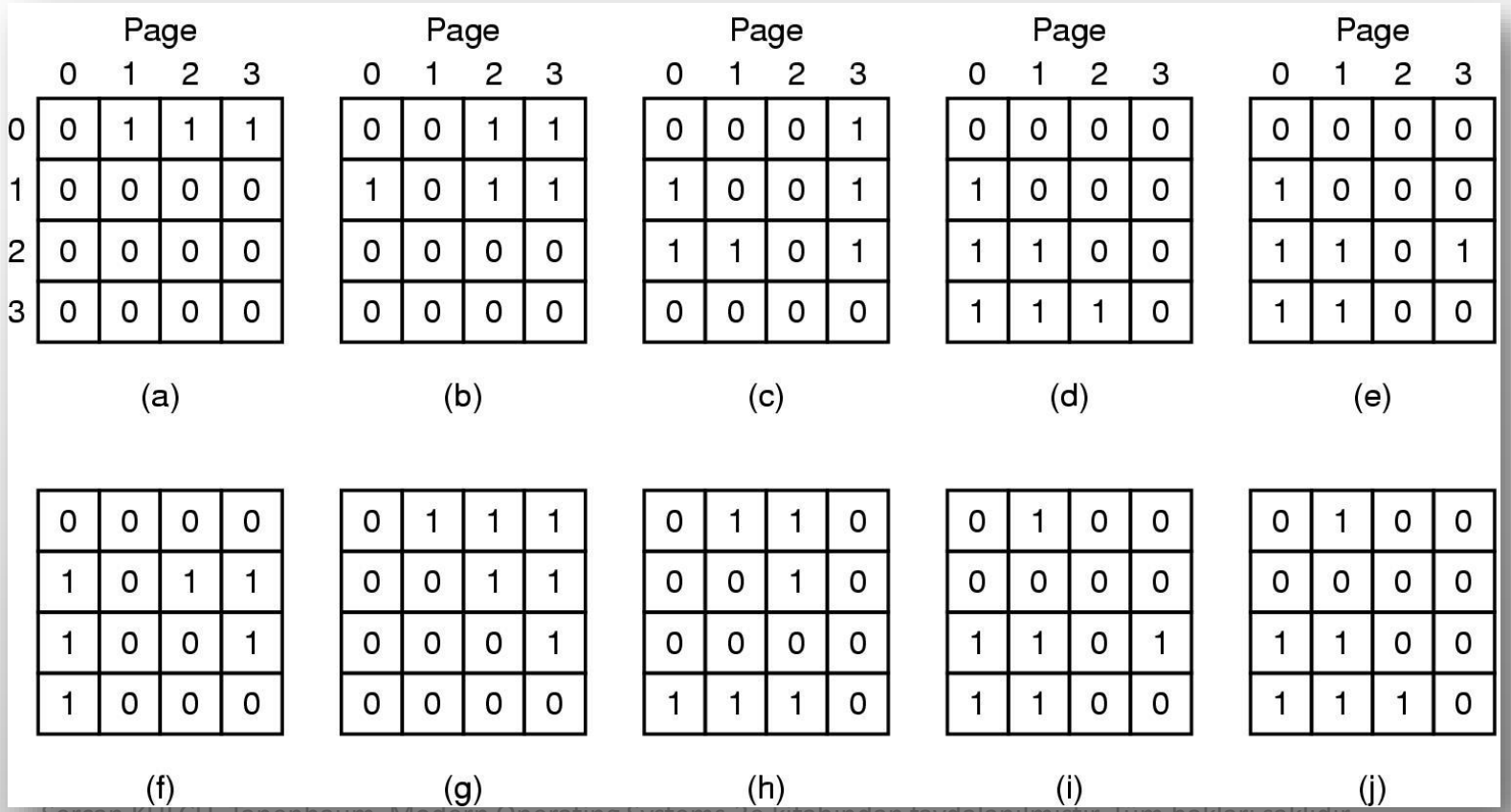
En Son Kullanılan Sayfa Değişimi (Yığın)

- .



En Son Kullanılan Sayfa Değişimi (Donanımsal)

- Sayfalara 0, 1, 2, 3, 2, 1, 0, 3, 2, 3 sırasıyla erişildiğinde matrisin içeriği.

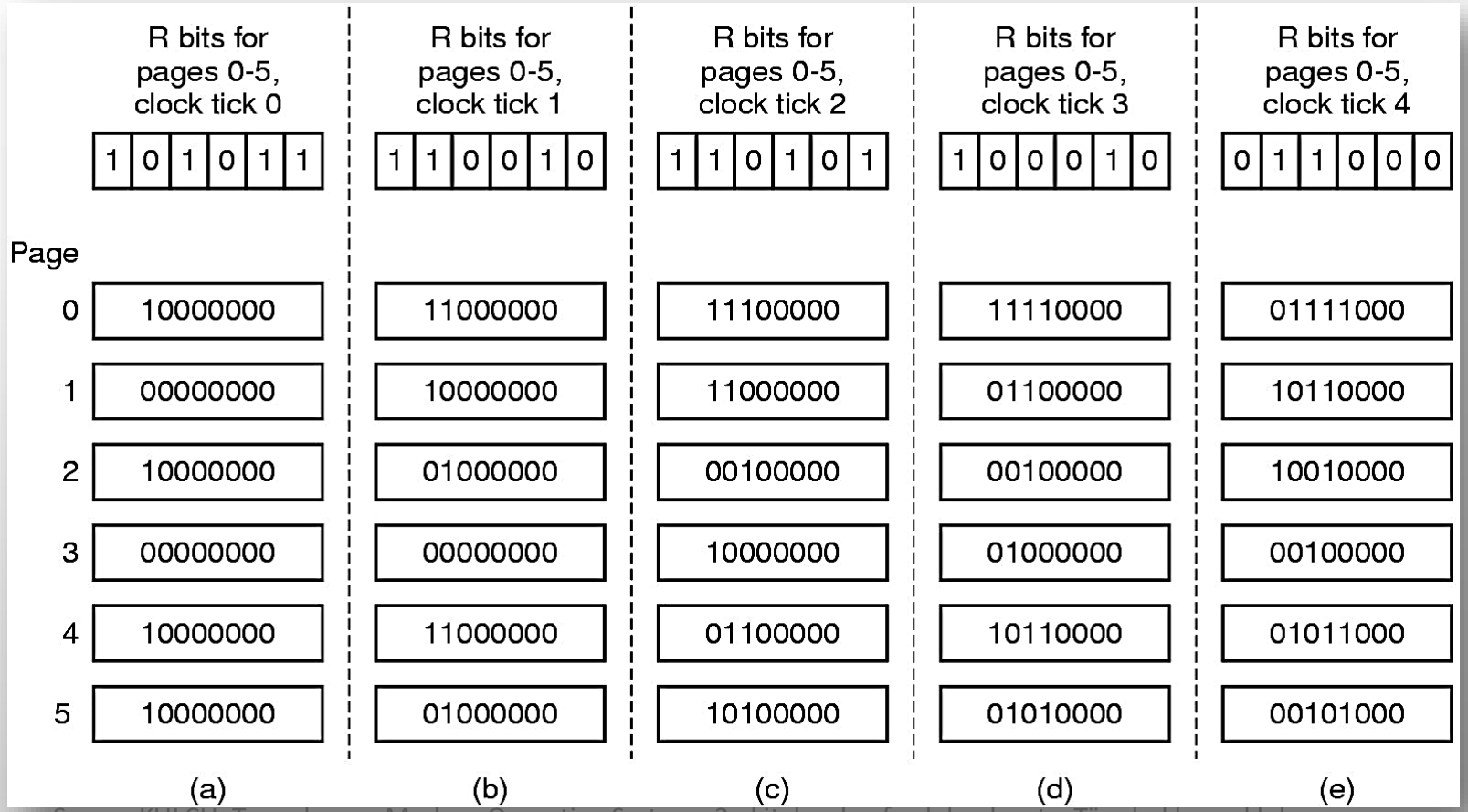


Sık Kullanılmayan Sayfa Değişimi

- *En son kullanılanı değiştir* algoritmasına benzer. (*Not frequently used*)
- Sayfaların erişim zamanları kaydedilir.
- Sayfa hatası olduğunda, en düşük değerli olan çıkarılır.
- Uzun bir günlük (*log*) tutmak gerekir. ☹

Sık Kullanılmayan Sayfa Değişimi

- Altı sayfa ve beş zaman dilimi için algoritma çıktısı.

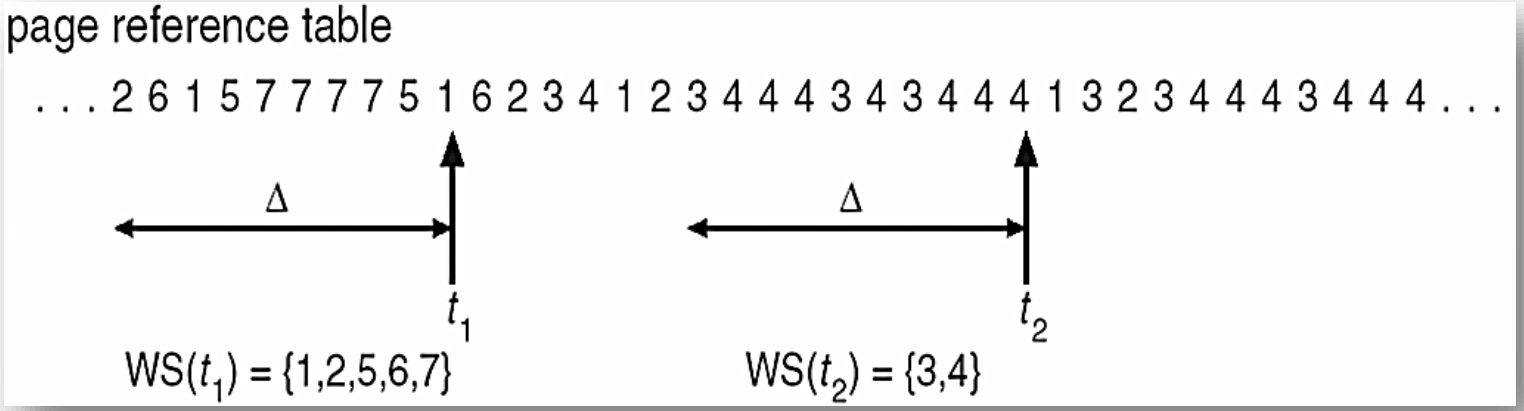


Çalışma Kümesi Sayfa Değişimi

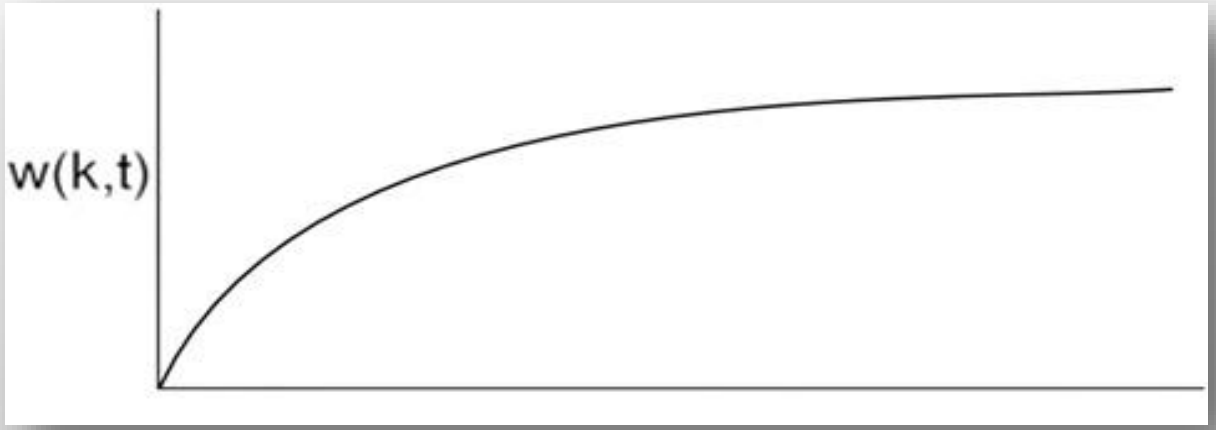
- Sayfaları önceden değil talep üzerine yükler.
- Süreç koşarken, sayfalarının nispeten küçük bir kısmına erişir.
- Çalışma kümesi (*working set*):
  - Bir sürecin herhangi bir anda kullanmakta olduğu sayfalar.
  - Bir sonraki aşamaya kadar, çok fazla sayfa hatasına neden olmaz.
- Bir süreç bellekten çıkarılıp, tekrar belleğe alındığında,
  - Önce çalışma kümesi yüklenir,
  - Böylece *sayfa hata* sayısı, büyük ölçüde azaltılır.

Çalışma Kümesi Sayfa Değişimi

- *Çalışma kümesi*, *k* adet en son erişilen sayfa.



- *w(k,t)* fonksiyonu, *t* anında çalışan kümenin boyutu.

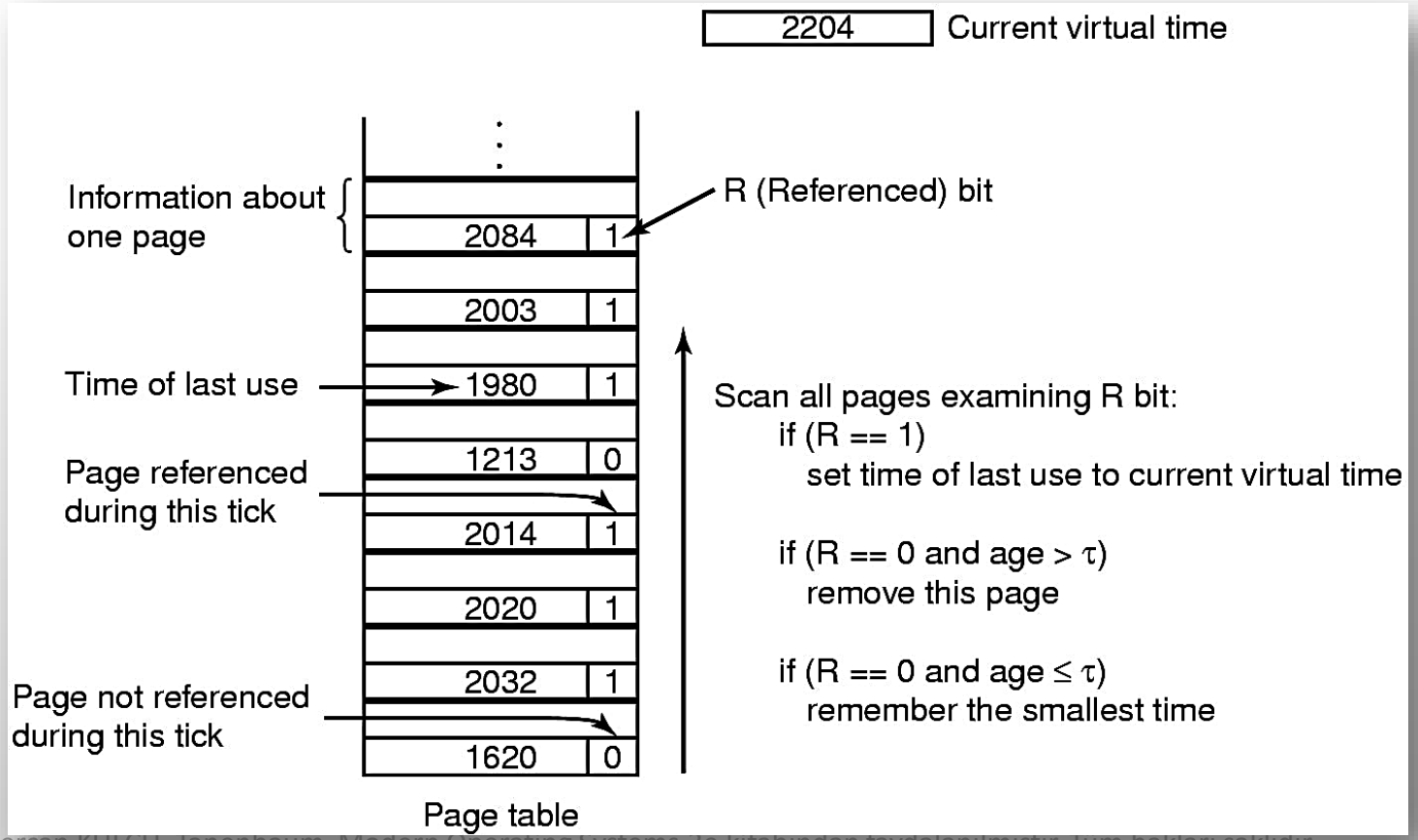


Çalışma Kümesi Sayfa Değişimi

- Sayfa hatası olduğunda, çalışma kümesinde olmayan sayfa çıkarılır.
- Çalışma kümesi için,
  - bir yazmaç tutulur ve
  - maliyetli olacak her erişimde değeri bir kaydırılır (*shift*).
- Her bellek erişiminde bellekteki sayfalar takip edilir. Masraflı. ☹
- Her *k* erişim, bir çalışma kümesiyle sonuçlanır.
- *k* adet bellek erişimi yerine sürecin koşma süresi (*tau*) kullanılabilir.
- *t* koşma süresi boyunca erişilen son *k* sayfanın kaydı tutulur.
- tau: time of actively used.

Çalışma Kümesi Sayfa Değişimi

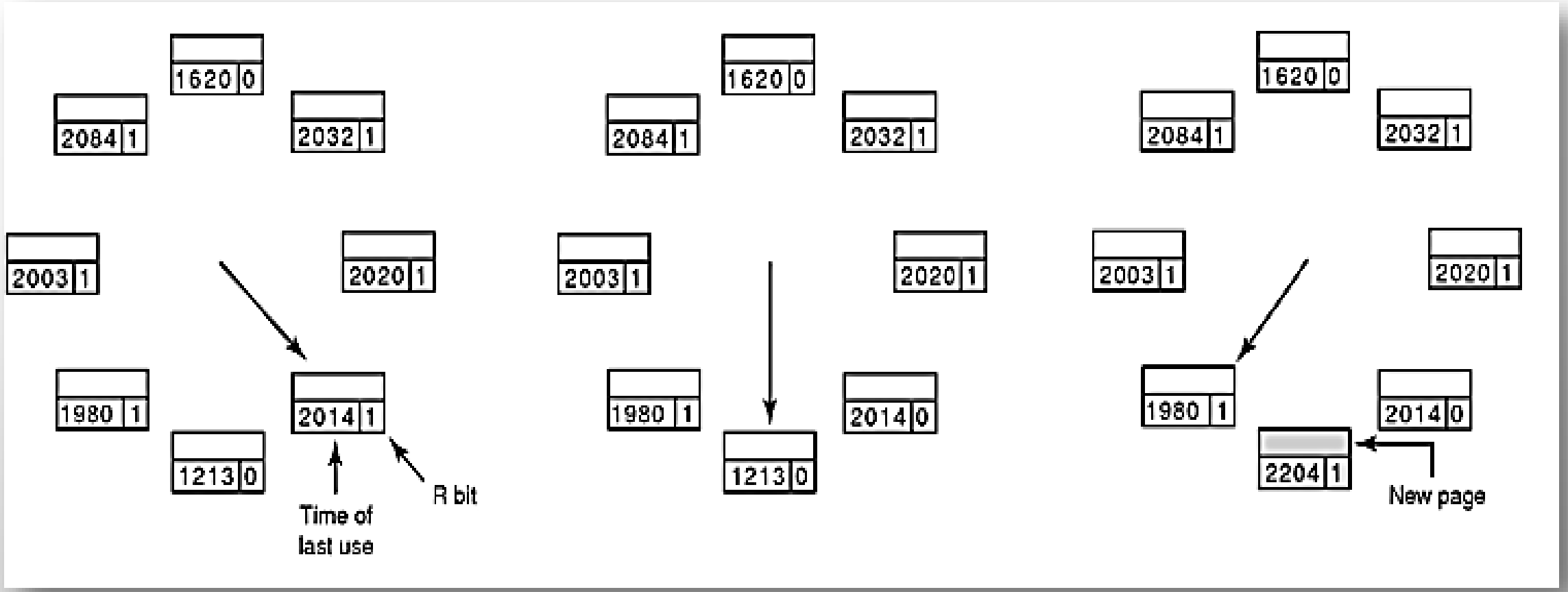
- *R (referenced)* biti periyodik olarak temizlenir.



Çalışma Kümesi Saat Sayfa Değişimi

- Çalışma kümesi algoritması, sayfa hatasında sayfa tablosunu tarar.
- *Working Set Clock*: basit ve etkili.
  - Sayfa çerçeveleri dairesel bir listede tutulur.
  - *R* biti 1 ise,
    - sayfaya erişilmiştir.
    - *R*'ye 0 ata. eli ilerlet.
  - *R* biti 0 ise,
    - *M* biti 0 ise ve (yaş > tau) ise yer değiştir.
    - *M* biti 1 ise, eli ilerlet ve diske geri yazmayı çizelgele.

Çalışma Kümesi Saat Sayfa Değişimi



Çalışma Kümesi Saat Sayfa Değişimi

- İki durumda da el başlangıç noktasına kadar gelir.
- En az bir yazma çizelgelendi ise,
  - İşaretçi temiz bir sayfa bulana kadar hareket eder.
- Hiçbir yazma çizelgelenmedi ise,
  - Tüm sayfalar çalışma kümesindedir.
  - Temiz bir sayfa seçilir.
  - Temiz sayfa yoksa,
    - Geçerli sayfa kurban (*victim*), diske geri yaz.

Özet

Algoritma	Yorum
Optimum	Uygulanabilir değil, kıyaslama amaçlı.
Yakın Zamanda Kullanılmayan	Çok kaba.
İlk Giren İlk Çıkar	Önemli sayfaları çıkarabilir.
İkinci şans	İlk Giren İlk Çıkar'ın gelişmiş versiyonu.
Saat	Gerçekçi.
En Son Kullanılan	İyi performans, uygulanması zor.
Sık Kullanılmayan	En Son Kullanılan'a yakınsar, adil.
Çalışma Kümesi	Uygulanması maliyetli.
Çalışma Kümesi Saat	İyi, verimli bir algoritma.

Yerel ve Genel Tahsis Politikaları

- Genel tahsis:
  - Bir süreç, tüm çerçeveler arasından yedek bir çerçeve seçer.
  - Bir süreç diğeri ile çerçeve paylaşabilir.
  - Yüksek verim, süreçlerin yürütme süresi büyük ölçüde değişebilir.
- Yerel tahsis:
  - Süreçler yalnızca kendi çerçeve kümesinden seçim yapabilir.
  - Süreç başına daha tutarlı performans elde edilir. ☺
  - Bellek verimli kullanılmayabilir. ☹

Yerel ve Genel Tahsis Politikaları

(a) Orijinal konfigürasyon. (b) Yerel tahsis. (c) Genel tahsis.

	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

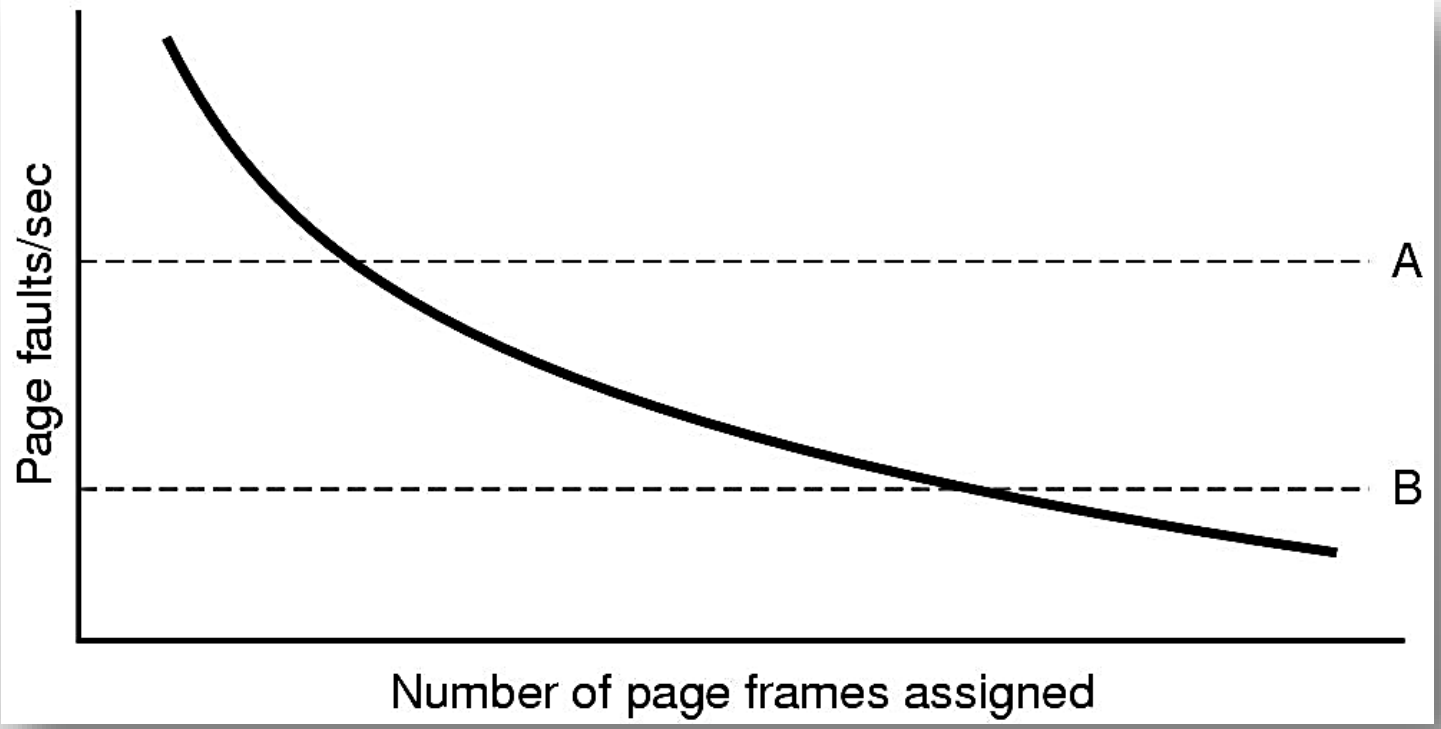
(b)

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

(c)

Yerel ve Genel Tahsis Politikaları

- .



Belady Anomalisi

- Daha çok çerçeve, daha iyi performans beklenir.!
- Erişim sıralaması: 1 2 3 4 1 2 5 1 2 3 4 5
- 4 çerçeve: sayfa hatası: 10 kez, sayfa yer değiştirme: 6 kez

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 kez, sayfa yer değiştirme: 6 kez

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 Tablosu Boyutu

- Ek maliyet = *se/p + p/2*
  - *p*, sayfa boyutu,
  - *s*, süreç boyutu,
  - *e*, sayfa tablosunun satır boyutu.

- *p* = √*(2se)*
  - *s* = 1 *MB*, *e* = 8 *bayt*, *p* = 4 *KB* en uygun.
  - 1 *KB* tipik.
  - 4-8 *KB* yaygın.

Temizleme İlkesi

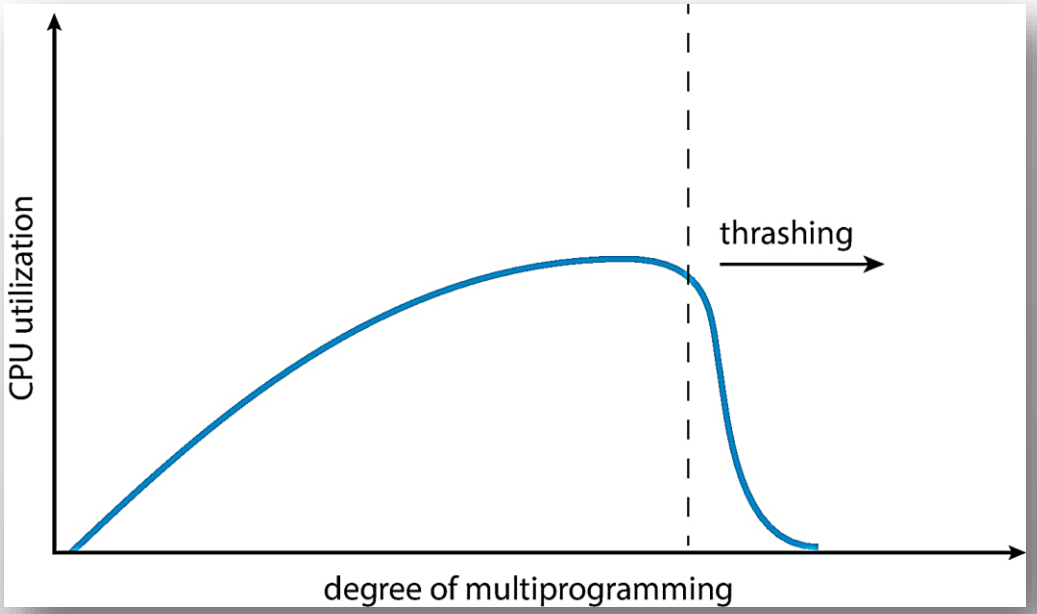
- İhtiyaç duyulduğunda kurban aramak yerine,
  - tahliye edilecek sayfaları bulmak için arka plan programı olmalı.
- Arka plan (*daemon*) programı periyodik olarak uyanır.
- Boşta çerçeve azsa, kullanılmayan çerçeveleri serbest bırakır.
- Çerçeveler, tahliye edilmeden önce,
  - temiz (değiştirilmemiş) olmalı.

Boşa Çabalama (Thrashing)

- Bir sürecin yeterli sayfası bellekte değil ise,
  - Sayfa hatası oranı yüksektir.
  - Sayfa takası için sayfa hatası üretir.
  - Mevcut çerçeve değiştirilir.
  - Ancak hızlı bir şekilde takas edilmeli.
    - CPU verimi düşer.
    - İşletim sistemi süreç sayısının artması gerektiğini düşünür.
    - Sisteme eklenen bir süreç daha.

Boşa Çabalama (Thrashing)

- Süreç, sayfaları takas etmekle meşgul.



Ön Sayfalama (Prepaging)

- Süreç başlangıcında sayfa hatasını azaltmak için kullanılır.
- Sürecin ihtiyaç duyacağı sayfaların bir kısmı hazırlanır.
- Ancak, hazırlanan sayfalar kullanılmaz ise,
  - G/Ç ve bellek boşa harcanmış olur.

SON