İşletim Sistemlerine Giriş

Bellek Yönetimi
(Memory Management)

TLB(Translation Lookaside Buffers) Dönüşüm öngörü tablosu

Birçok sayfalama tasarımında, sayfa tabloları büyük boyutlu oldukları için bellekte tutulurlar. Bu tasarım potensiyel olarak performansı kötü etkiler.

Örneğin; bir yazmaçı diğerine kopyalayan komut sistemde sayfalama yoksa belleğe birkez erişir ve komutu alır.

Sayfalama ile sayfa tablosuna erişim için ek erişimler yapmalıdır. Her bellek adresine erişim 2 sayfa tablosuna erişilmesi ile mümkün olmaktadır.

TLB(Translation Lookaside Buffers) Dönüşüm öngörü tablosu

Birçok programın sayfalardan küçük bir gruba müracat ettiği belirlenmiştir. Bu yüzden sayfa tablolarının küçük bir parçası ağırlıklı okunmakta, kalan kısmı nadiren okunmaktadır.

Bu problemi çözebilmek için, sanal adresten fiziksel adrese sayfa tablosunu kullanmadan çeviri yapan bir donanım eklenmesi önerilmiştir. Bu donanıma TLB(Translation lookaside buffers) denilmektedir.

Genellikle MMU da bulunan ve küçük boyutlu bir tablodur.

TLB(Translation Lookaside Buffers) Dönüşüm öngörü tablosu

Geçerli	Sanal Sayfa	Değiştirilmiş	Koruma	Sayfa Çerçevesi
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(Translation Lookaside Buffers) Dönüşüm öngörü tablosu

TLB nin yönetimini MMU'nun kendisi yapabileceği gibi, işletim sisteminin de yaptığı sistemler vardır.

(RISC, SPARC, MIPS, Alpha) gibi sistemlerde yazılım ile TLB yönetimi yapılmaktadır.

Tersine çevrilmiş Sayfa Tabloları (Inverted Page Tables)

Şu ana kadar tanımladığımız sayfa tabloları sanal sayfa başına bir girdiye ihtiyaç duyar. Bunun nedeni bu tablolarda kullanılan indisin sanal sayfa numarası olmasıdır.

Eğer adres uzayı 2^{32} byte ise, sayfa başına 4096 byte kullanırsak, 1 milyon sayfa tablosu girdisine ihtiyacımız vardır.

64 bitlik bilgisayarların kullanıma geçmesiyle durum değişmektedir. Eğer adres uzayı 2^{64} byte ise, 4 Kb sayfalar ile 2^{52} girdili sayfa tablolarına ihtiyacımız vardır. Her girdide 8 byte bilgi saklandığını düşünürsek tablonun boyutu 30 milyon GB dan fazla olur.

Tersine çevrilmiş Sayfa Tabloları (Inverted Page Tables)

Tersine çevrilmiş sayfa tabloları çözüm için kullanılır. Bu tasarımda fiziksel bellekteki sayfa çerçevesi başına bir girdi bulunmaktadır.

Örneğin; 64 bit sanal adresleri olan bir sistemde, sayfa boyutları 4 KB seçilsin. Sistemin 256 MB RAM i varsa, 65535 adet girdiye ihtiyaç duyulmaktadır. Bu girdi sürecin hangi sanal sayfası sayfa çerçevesinde yerleştirilmiş onu tutmaktadır.

Bu yöntem ile bellek kullanım israfı önlensede, sanal-fiziksel çevrim işlemi daha zor olur.

Tersine çevrilmiş Sayfa Tabloları (Inverted Page Tables)

n süreci p sanal sayfasına başvurduğunda donanım p'yi indeks olarak kullanıp fiziksel adresi bulamaz. Bunun yerine tüm tersine çevrilmiş sayfa tablosunu (n,p) girdisi için tarar.

Bu arama her bellek başvurusunda yapılmalıdır.

Bu problemin çözümü için TLB kullanılmaktadır. Eğer TLB yoğun şekilde kullanılan sayfaların hepsini tutarsa, dönüştürme işlemi hızlı gerçekleşir.

TLB de bir ıska(miss) gerçekleşirse yani aranan sayfa bulunmaz ise, inv.p.table da bir arama yapılmalıdır.

Tersine çevrilmiş Sayfa Tabloları (Inverted Page Tables)

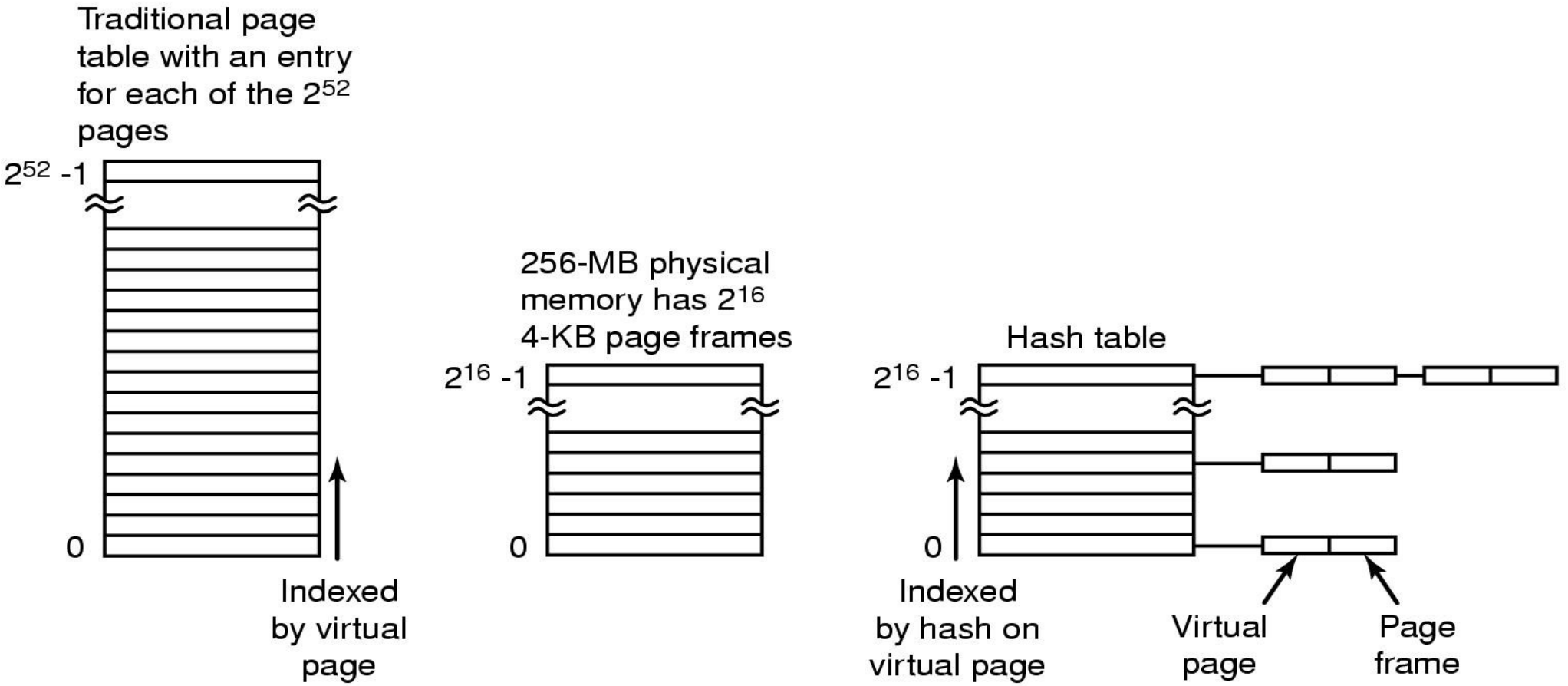
Arama işleminin her durumda yapılması zorunlu olduğu için, aramayı hızlandırmak amacı ile HashTable kullanılmaktadır.

Tüm sanal tablolar birbirlerine hash lenmiş olarak zincir yapısında bulunurlar.

Seçilen hash tablosunun tablo boyutu fiziksel adres boyutu kadar seçilerek arama işlemi hızlandırılmış olur.

Sayfa çerçevesi bulunur bulunmaz TLB'ye eklenir.

Tersine çevrilmiş Sayfa Tabloları (Inverted Page Tables)



Sayfa Yerdeğiştirme Algoritmaları (Page Replacement Algorithms)

Sistemde bir sayfa hatası(page fault) görülürse, işletim sistemi bellekten bir sayfayı kaldırmak için seçmelidir. Seçilen ve bellekten çıkarılan sayfa yerine, yeni sayfa gelecektir.
Eğer bellekten kaldırılacak olan sayfa bellekte iken değiştirilmiş ise, diskteki kopyanın güncel olması için tekrar bellekten diske yazılmalıdır. Değiştirilmemiş ise diske yapılmasına gerek yoktur.

Her sayfa hatasında yoğun olarak kullanılmayan sayfanın çıkarılması, sistemin performansını arttırır. Yoğun kullanılan bir sayfa çıkarılırsa çok kısa bir süre sonra bu sayfa tekrar istenebilir.

Sayfa Yerdeğiştirme Algoritmaları (Page Replacement Algorithms)

Sayfa yer değiştirme algoritmaları üzerinde çok fazla teorik ve deneysel çalışma yapılmıştır.

Sayfa yer değiştirme algoritmaları bilgisayarda farklı alanlarda kullanılmaktadır. (Örn: cache, web sunucularında, ...)

En iyi sayfa yer değiştirme algoritması

En iyi sayfa yer değiştirme algoritması kolaylıkla tanımlanır fakat uygulanamaz.

Algoritma şudur: sayfa hatasının görüldüğü anda, bellekte sayfa kümeleri bulunmaktadır. Bu sayfalardan bir tanesine bir sonraki komutta(instruction) başvurulacaktır.

Bu komutu içermeyen sayfalara belkide 10,100,1000,... komut sonrasına kadar erişim gerçekleşmeyecektir.

Her sayfa erişim işlemi yapılmadan kaç komut geçeceği ile etiketlenebilir.

En iyi sayfa değiştirme algoritması

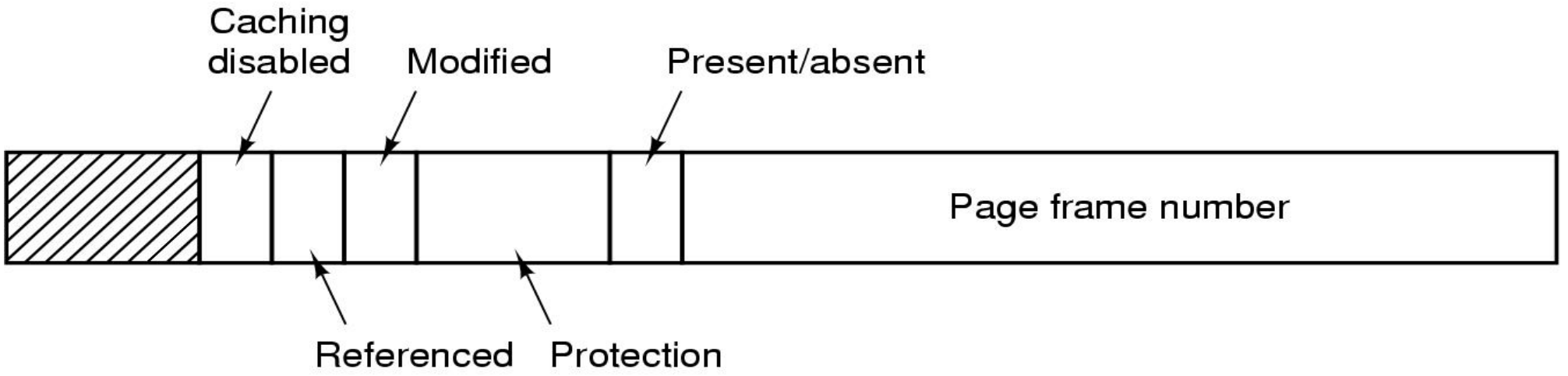
En iyi sayfa algoritması en yüksek etiketli sayfanın kaldırılması gerektiğini söyler.

Elimizde 8 milyon komut sonra ve 6 milyon komut sonra erişilecek iki sayfa var ise, birinci sayfanın bellekten çıkarılması daha mantıklıdır.

Bu algoritmanın gerçekleştirilememe nedeni: işletim sisteminin sayfa hatası olduğunda sayfaların bir sonraki erişilme zamanlarını bilmemesidir.

Son zamanlarda kullanılmayan sayfanın yer değiştirilmesi algoritması (The not recently used page replacement algorithm)

İşletim sisteminin hangi sayfaların kullanılıp kullanılmadığı hakkında kullanışlı istatistikler sağlamak için, her sayfa için 2 bit kullanılmaktadır. **R**(Referenced)başvuruldu, **M**(Modified)değiştirildi bitleri. R biti bu sayfaya erişildiği zaman 1 lenir. M de bu sayfaya yazma gerçekleştiği zaman 1 lenir.



Son zamanlarda kullanılmayan sayfanın yer değiştirilmesi algoritması (The not recently used page replacement algorithm)

Bu bitler her bellek başvurusunda güncellenmelidir, bu yüzden donanım ile yapılmalıdır. Birkez bir bit 1 olduğunda işletim sistemi onu 0 yapmadan 1 kalır. Eğer donanımda bu bitler yok ise şu şekilde benzetim yapılır. Bir süreç çalışmaya başladığında, tüm sayfa tablosu girdileri bellekte değil olarak işaretlenir. Bir sayfaya başvuru yapıldığında sayfa hatası(page fault) görülür.
İşletim sistemi kendi tablolarındaki sayfa bilgilerini tutan tabloda R bitini günceller ve sayfa tablosu girdisini doğru tabloyu gösterecek şekilde günceller. Bu sayfayı sadece okunabilir yapar. Daha sonra bu sayfaya yazılmak istenildiğinde başka bir hata meydana gelir.

Son zamanlarda kullanılmayan sayfanın yer değiştirilmesi algoritması (The not recently used page replacement algorithm)

İşletim sistemi bu sayede yazılma işleminide anlar ve sayfanın M bitini 1 ler ve sayfayı R/W olabilir yapar.

P ve M bitleri kullanılarak basit bir sayfa yer değiştirme algoritması (PRA) oluşturulabilir. Bir süreç çalışmaya başladığında tüm sayfaları için R ve M biti 0 lanır. Periyodik olarak (her saat kesmesinde) R biti 0 lanır.Bu işlem son zamanlarda kullanılmayan sayfanın belirlenmesi için gereklidir.

Bir sayfa hatası meydana geldiğinde, işletim sistemi sayfaları araştırır ve sayfaları 4 kategoriye ayırır. Bu kategoriler R ve M bitinin durumlarıdır.

Son zamanlarda kullanılmayan sayfanın yer değiştirilmesi algoritması (The not recently used page replacement algorithm)

Sınıf 0: başvurulmamış ve değiştirilmemiş = 0 0
Sınıf 1: başvurulmamış, değiştirilmiş = 0 1
Sınıf 2: başvurulmuş, değiştirilmemiş = 1 0
Sınıf 3: başvurulmuş, değiştirilmiş = 1 1

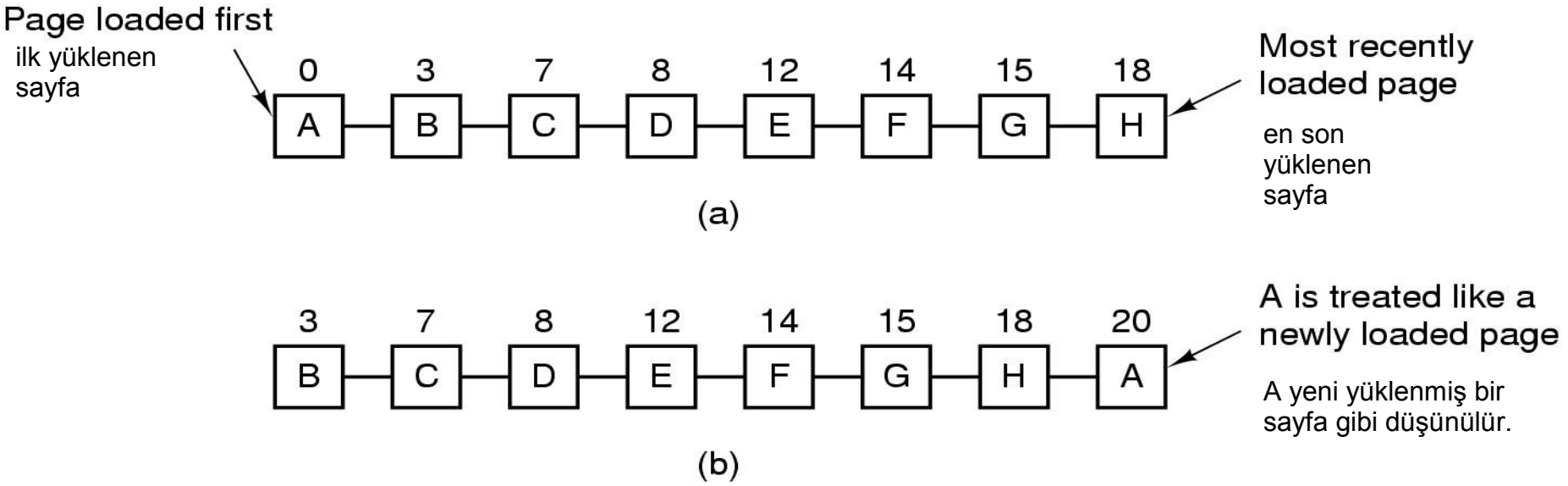
1. sınıf hiçbir zaman oluşmaz gibi görünsede 3. sınıfda R=0 landığında 1. sınıf oluşur.
NRU (Not Recently Used) algoritması boş olmayan en düşük numaralı sınıflardan bir sayfayı rastgele seçer.

İlk gelen ilk çıkar algoritması
FIFO(First in First Out)

İşletim sistemi bellekteki sayfaları geliş sırasına göre bağlı listede tutar. Listenin başında ilk gelen sayfa, sonunda da en son gelen sayfa bulunur. Her zaman baştaki sayfa çıkarılır ve yeni gelen sayfa sona eklenir.

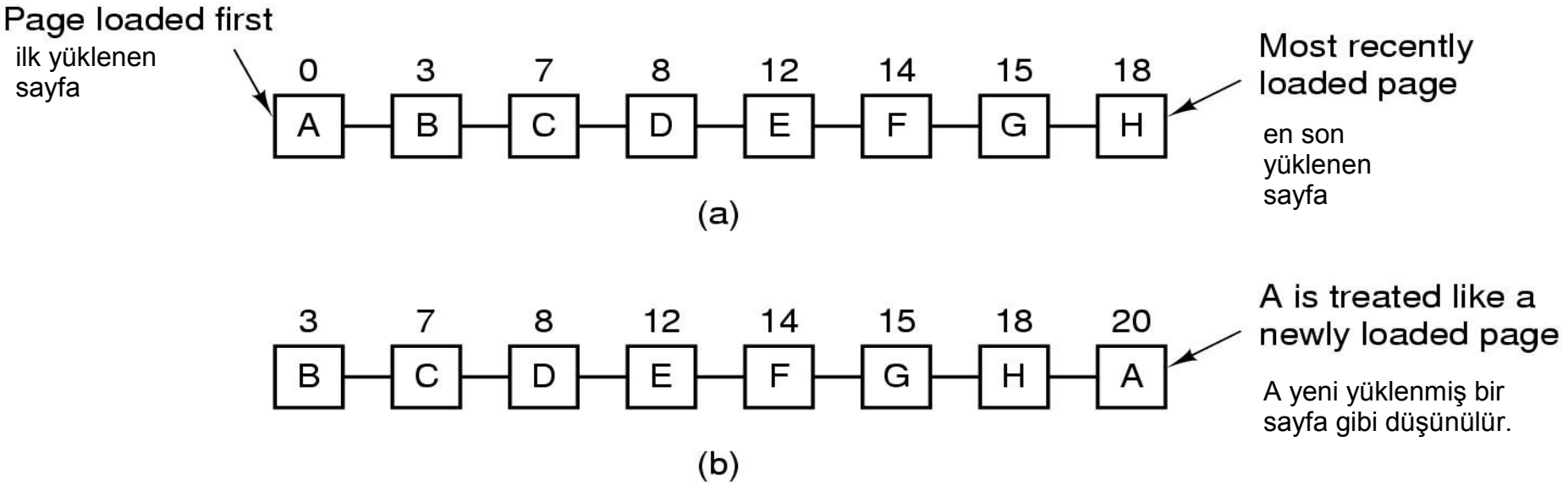
İkinci Şans Yerdeğiştirme Algoritması

FIFO temellidir. Eski sayfaların R bitlerinin kontrolü ile yoğun kullanılan bir sayfanın çıkartılması engellenir. Eğer R=0 ise bu sayfa hem eski hemde kullanılmayan bir sayfa olduğu için çıkartılabilir. Eğer R biti 1 ise bu bit 0 lanır ve bu sayfa listenin sonuna konulur ve yüklenme zamanı yeniymiş gibi düşünülerek güncellenir.



İkinci Şans Yerdeğiştirme Algoritması

20. saniyede bir sayfa hatası oluşursa A nın R si 0 ise sayfa tahliye edilir. M=1 ise diskteki veri güncellenir. A nın R=1 ise listenin sonuna eklenir, R=0 ve yüklenme zamanı 20 yapılır. Arama B den devam eder.



Saat Sayfa Yerdeğiştirme Algoritması (The clock page replacement algorithm)

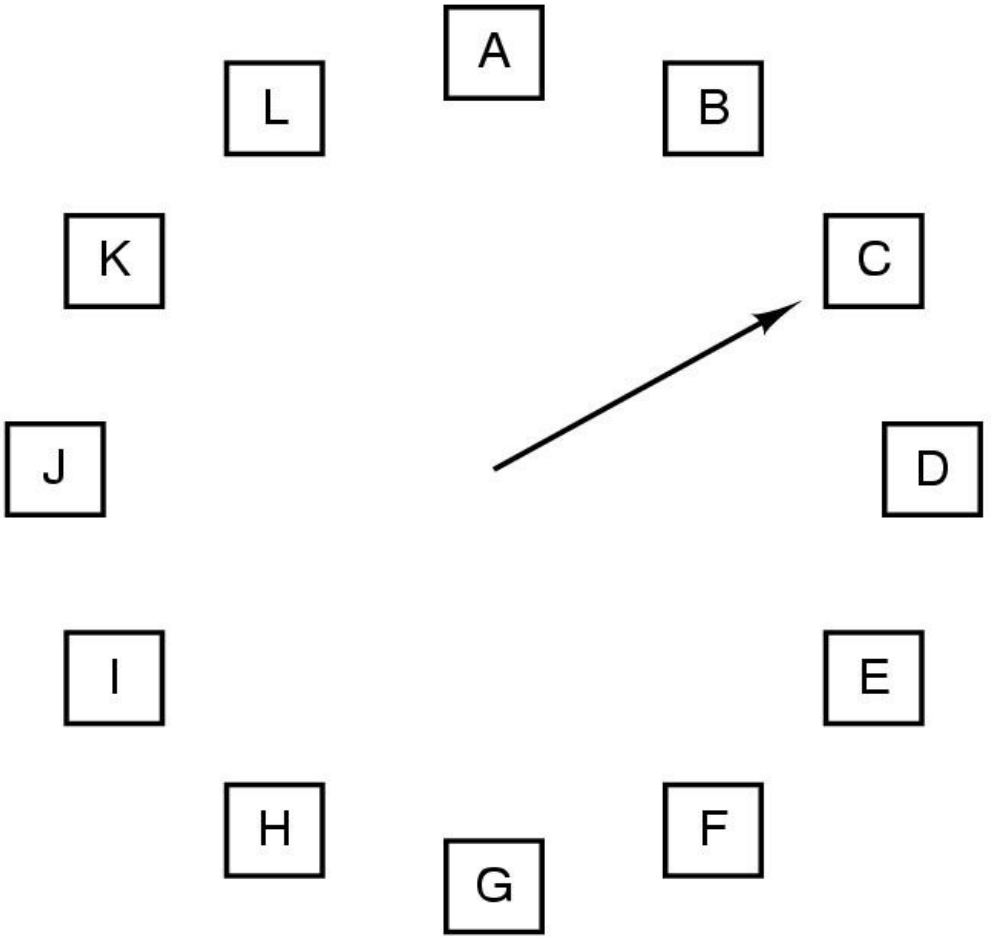
İkinci şans algoritması akla uygun olmasına rağmen verimsizdir. Bunun nedeni sayfaların liste üzerinde yer değiştirmesidir.

Daha iyi bir yaklaşım, tüm sayfaları bir saat biçimli dairesel listede tutmaktır.

Saatin kolu en eski sayfayı işaret eder. Sayfa hatası olduğunda saat kolunun gösterdiği sayfa araştırılır. R ye göre bir eylem gerçekleştirilir.

*R=0 ise sayfayı çıkar.
*R=1 ise, R=0 yap ve saat kolunu bir ilerlet.

Saat Sayfa Yerdeğiştirme Algoritması (The clock page replacement algorithm)



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 uzun zamandır erişilmemiş sayfa yer değiştirme algoritması (LRU-The Least Recently Used page replacement algorithm)

En iyi algoritmaya yaklaşık bir algoritmadır. Ana fikri son zamanlarda erişilmiş olan sayfalara büyük olasılıkla sonraki birkaç komutçada erişilecektir. Tam tersi erişilmeyenlerde erişilmeyecektir. Bir sayfa hatası olduğunda en uzun zamandır kullanılmayan sayfa dışarı atılır. Bu stratejiye LRU (Least Recently Used) sayfalama denilir.

LRU yu tam olarak gerçekleştirebilmek için, tüm sayfaların bilgisini tutacak bir bağlı listeyi bellekte tutmak gereklidir. Bu listenin başında en son kullanılan sayfa, sonunda en uzun zamandır kullanılmayan sayfa olmalıdır. Liste her bellek başvurusunda güncellenmelidir. Listeden sayfanın bulunması, silinmesi ve başa alınması işlemleri zaman almaktadır.

En uzun zamandır erişilmemiş sayfa yer değiştirme algoritması (LRU-The Least Recently Used page replacement algorithm)

Bu işlem zaman almasına rağmen, özel donanımlar ile LRU yu gerçekleştirmenin yolları vardır:

donanımda 64 bitlik C sayacı(counter) olsun. Bu her komuttan sonra arttırılsın.
Bununla birlikte her sayfa tablosu girsininin de sayac boyutu kadar yani 64 bitlik alanları olsun.
Her bellek başvurusundan sonra, başvurulan sayfanın sayaç alanına o anki sistemin C sayacının değeri atılsın.

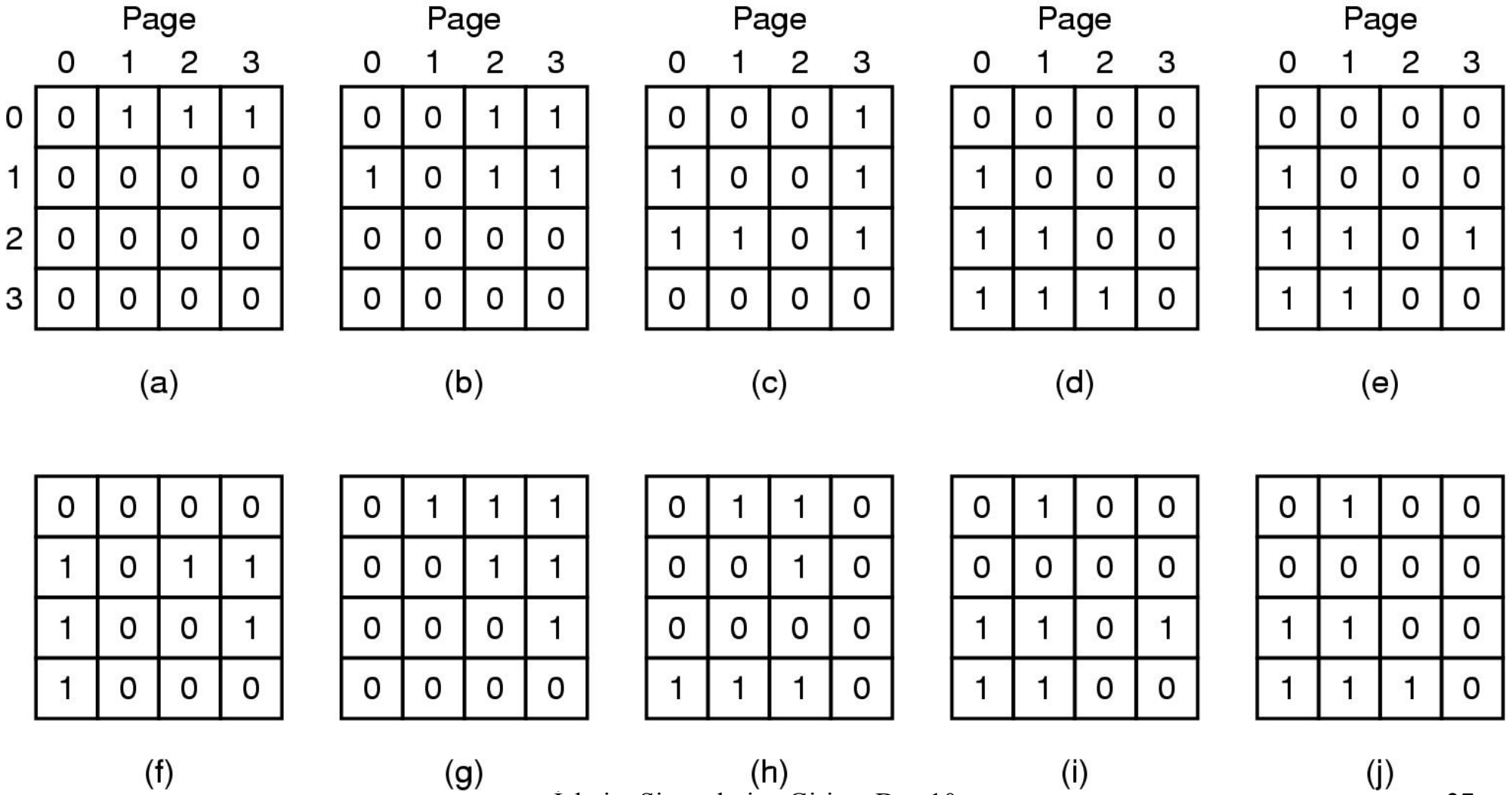
Bir sayfa hatası görüldüğünde, işletim sistemi sayfa tablosundaki tüm sayaçları tarar ve en küçük olanı bulur. Bu sayfa en uzun zamandır erişilmeyen sayfadır.

En uzun zamandır erişilmemiş sayfa yer değiştirme algoritması (LRU-The Least Recently Used page replacement algorithm)

İkinci metot:
n sayfa çerçeveli makine için, LRU donanımı nxn bitlik bir matris tutsun. Başlangıçta matrisin tüm elemanları 0 olsun. Bir k sayfasına başvurulduğunda donanım k satırındaki tüm bitleri 1 daha sonra k sütunundakileri 0 yapsın. Her hangi bir anda bu matrisin en küçük değerli satırı en uzun süredir erişilmemiş sayfadır.

En uzun zamandır erişilmemiş sayfa yer değiştirme algoritması (LRU-The Least Recently Used page replacement algorithm)

Şu sırada sayfalara erişim yapılsın: **0 1 2 3 2 1 0 3 2 3**



En uzun zamandır erişilmemiş sayfa yer değiştirme algoritması (LRU-The Least Recently Used page replacement algorithm)

İkinci metot:
n sayfa çerçeveli makine için, LRU donanımı nxn bitlik bir matris tutsun. Başlangıçta matrisin tüm elemanları 0 olsun. Bir k sayfasına başvurulduğunda donanım k satırındaki tüm bitleri 1 daha sonra k sütunundakileri 0 yapsın. Her hangi bir anda bu matrisin en küçük değerli satırı en uzun süredir erişilmemiş sayfadır.

İşletim Sistemlerine Giriş

Bellek Yönetimi
(Memory Management)