7 BELLEK YÖNETİMİ

Bellek Yönetimi

- Birden fazla prosese yer verilebilecek şekilde belleğin alt birimlere ayrılması
- Belleğin prosesler arasında atanması etkin olmalı:
  - en fazla sayıda proses

Bellek Yönetiminin Gerektirdikleri

- Yeniden yerleştirme (Relocation)
  - programcı çalışan programının bellekte nereye yerleşeceğini bilmez
  - koşan program ikincil belleğe atılıp, ana bellekte farklı bir yere tekrar yüklenebilir
  - Bellek referans adresleri fiziksel adres değerlerine dönüştürülmeli

Bellek Yönetiminin Gerektirdikleri

- Koruma
  - İzni olmadan bir proses bir başka prosesin bellek alanlarına erişemez
  - Programın yeri değişebileceğinden kontrol için programdaki gerçek adresler kullanılamaz
  - Çalışma anında kontrol edilmeli

Bellek Yönetiminin Gerektirdikleri

- Paylaşma
  - Birden fazla prosesin aynı bellek bölgesine erişmesi
    - program kodu
    - ortak veri alanı

Bellek Yönetimi Teknikleri

- Bölmeleme (Partitioning)
  - Sabit
  - Dinamik
- Basit sayfalama (Paging)
- Basit segmanlama (Segmentation)
- Sayfalamalı görüntü bellek (Virtual Memory)
- Segmanlamalı görüntü bellek

Sabit Bölmeleme

- Bölme boyları eşit
  - boş bölmeye, boyu bölme boyundan küçük ya da eşit prosesler yüklenebilir
  - tüm bölmeler doluysa proseslerden biri bellekten atılır
  - program bölmeye sığmayabilir ⇒ programcı program parçalarını birbirinin üzerini örtecek şekilde (overlay) yazar

Sabit Bölmeleme

- Bellek kullanımı etkin değil:
  - her program ne kadar boyu küçük de olsa tam bir bölmeyi elinde tutar ⇒ iç parçalanma (internal fragmentation)
  - eşit boyda olmayan bölmeler kullanılması sorunu bir derece çözer
- Maksimum aktif proses sayısı sınırlı
- İşletim sistemi tarafından gerçeklenmesi kolay
- Getirdiği ek yük az.

Yerleştirme Algoritmaları (Sabit Bölmeleme)

- Bölme boyları eşit
  - prosesin hangi bölmeye yerleştirileceği fark etmez
- Bölme boyları eşit değil
  - her prosesi sığacağı en küçük bölmeye
  - her bölme için kuyruk
  - bölme içi boş kalan yer miktarını en aza indirmek

    (IBM OS/MFT: multiprogramming with a fixed number of tasks)

Dinamik Bölmeleme

- Bölme sayısı ve bölme boyları sabit değil
- Proseslere sadece gerektiği kadar bellek atanır
- Kullanılmayan boş yerler yine de oluşur $\Rightarrow$ dış parçalanma
- Tüm boş alanın bir blok halinde olması için sıkıştırma kullanılır

  (IBM OS/MVT: multiprogramming with a variable number of tasks)

Yerleştirme Algoritmaları (Dinamik Bölmeleme)

- Hangi boş bloğun hangi proses atanacağına işletim sistemi karar verir
- En-İyi-Sığan Algoritması (Best-Fit)
  - Boyu istenene en yakın olan boşluk seçilir
  - Olası en küçük bölme bulunduğundan artan boş alan az $\Rightarrow$ sıkıştırmanın daha sık yapılanması gerekir

Yerleştirme Algoritmaları (Dinamik Bölmeleme

- İlk-Sığan Algoritması (First-fit)
  - En hızlı
  - Prosesler baş bölgelere yığılır $\Rightarrow$ boş yer ararken üst üste taranır

Yerleştirme Algoritmaları (Dinamik Bölmeleme

- Bir-Sonraki-Sığan Algoritması (Next-fit)
  - Son yerleştirilen yerden itibaren ilk sığan yeri bulur
  - Genellikle atamalar belleğin son kısımlarında yer alan büyük boşluklardan olur
    - En büyük boşluklar küçük parçalara bölünmüş olur
    - Sıkıştırma gerekir

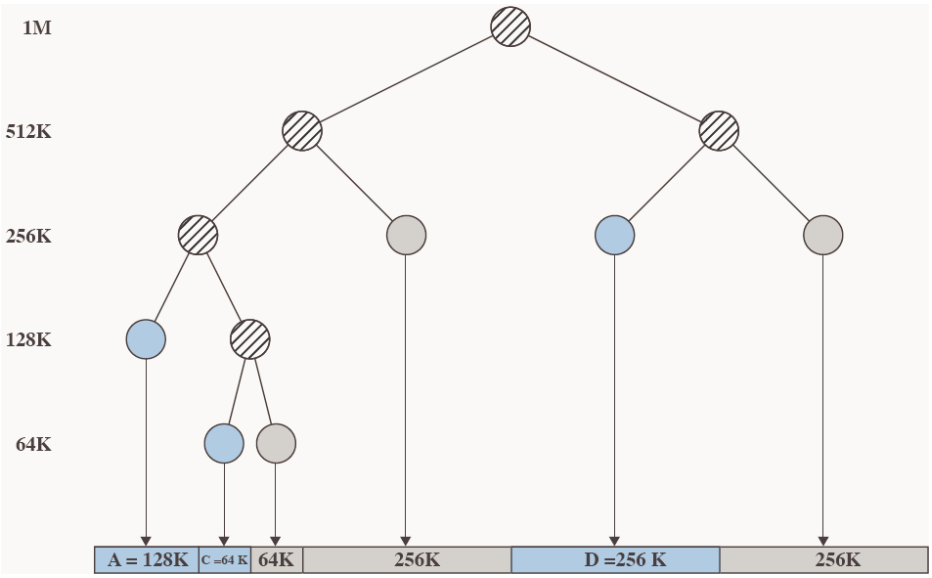
"Buddy" Yöntemi

- Tüm boş alan 2^U boyutunda tek bir alan olarak ele alınır
- s boyutundaki bir istek eğer $2^{U-1} < s <= 2^U$ ise tüm blok atanır
  - Aksi halde blok 2^{U-1} boyutunda iki eş bloğa bölünür (buddy)
  - s'den büyük veya eşit en küçük birim blok oluşturulana kadar işlem devam eder

Buddy Sistem Örneği

1 Mbyte block	1 M					
Request 100 K	A = 128K	128K	256K	512K		
Request 240 K	A = 128K	128K	B = 256K	512K		
Request 64 K	A = 128K	C = 64K	64K	B = 256K	512K	
Request 256 K	A = 128K	C = 64K	64K	B = 256K	D = 256K	256K
Release B	A = 128K	C = 64K	64K	256K	D = 256K	256K
Release A	128K	C = 64K	64K	256K	D = 256K	256K
Request 75 K	E = 128K	C = 64K	64K	256K	D = 256K	256K
Release C	E = 128K	128K	256K	D = 256K	256K	
Release E	512K			D = 256K	256K	
Release D	1M					

Buddy Yönteminin Ağaç ile Temsili



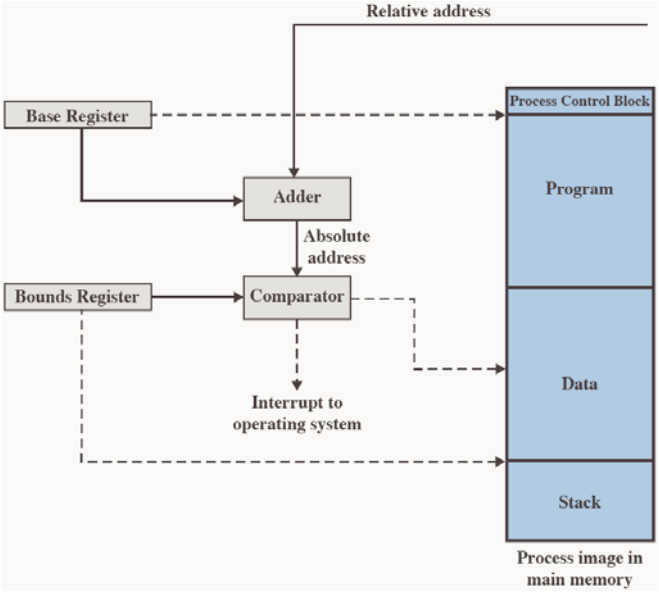
Yeniden Yerleştirme

- Proses belleğe yüklendiği zaman ancak mutlak bellek adresleri belirlenebilir
- Proses çalışması boyunca değişik bölmelere yerleşebilir (swap) $\Rightarrow$ Mutlak adresler değişebilir
- Sıkıştırma nedeni ile de proses farklı bölmelerde yerleşebilir $\Rightarrow$ farklı mutlak bellek adresleri

Adresler

- Mantıksal
  - belleğe erişimde kullanılan adres gerçek fiziksel adreslerden bağımsız
  - adres dönüşümü gerekir
- Bağıl
  - adres bilinen bir noktaya göre bağıl verilir
  - genellikle bu nokta prosesin başıdır
- Fiziksel
  - ana bellekteki gerçek konum adresi

Yeniden yerleştirme için gerekli donanım desteği



Saklayıcılar

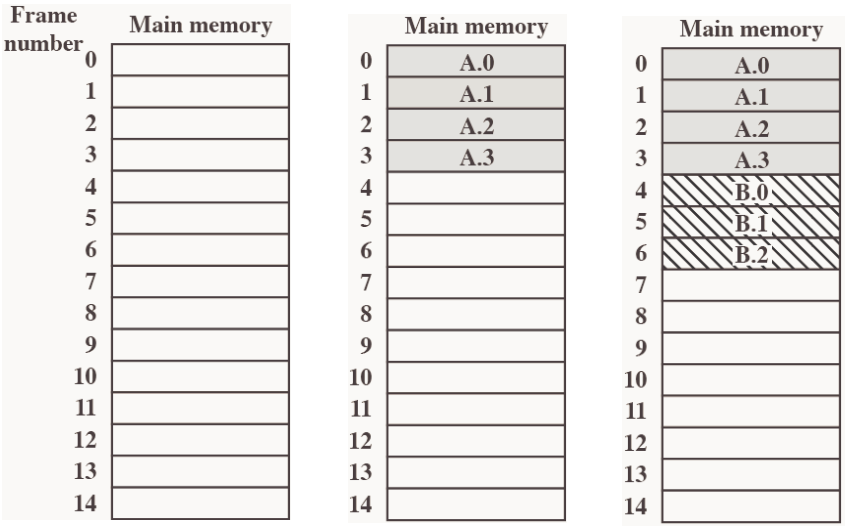
- Taban saklayıcısı
  - prosesin başlangıç adresi
- Sınır saklayıcısı
  - prosesin son adresi
- Bu değerler saklayıcılara proses belleğe yüklendiğinde yazılır

Sayfalama

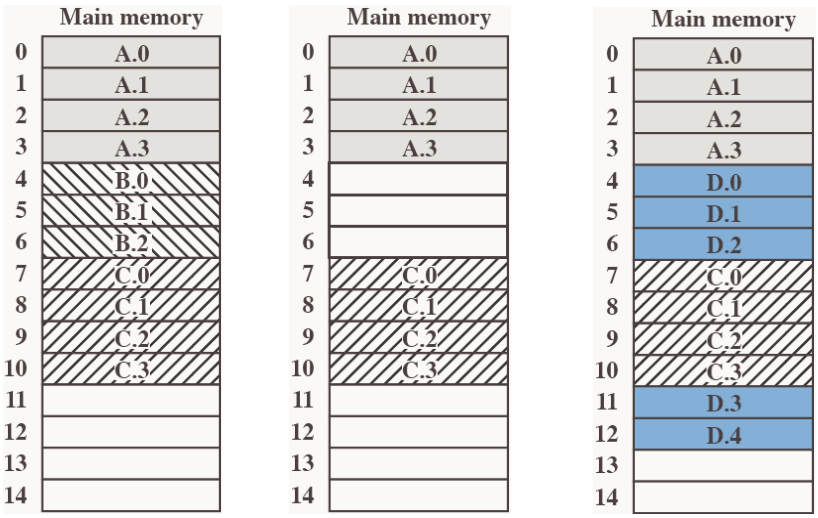
- Belleği küçük, eşit boylu parçalara böl. Prosesleri de aynı boyda parçalara ayır.
- Eşit boylu proses parçaları: *sayfa*
- Eşit boylu bellek parçaları: *çerçeve*
- İşletim sistemi her proses için *sayfa tablosu* tutar
  - prosesin her sayfasının hangi çerçevede olduğu
  - bellek adresi = sayfa numarası ve sayfa içi ofset adresi
- İşletim sistemi hangi çerçevelerin boş olduğunu tutar

Sayfalama

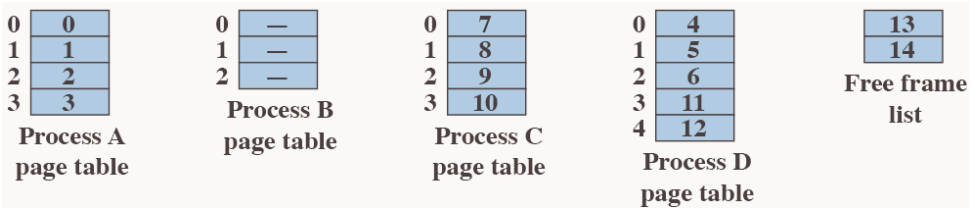
- Sabit bölmeleme benzeri
- Bölmeler küçük
- Bir prosese ilişkin birden fazla sayfa olabilir bellekte
- Sayfa ve çerçeve boylarının ikinin kuvvetleri şeklinde seçilmesi gerekli dönüşüm hesaplamalarını basitleştirir
- Mantıksal adres sayfa numarası ve sayfa içi kayıklık değerinden oluşur



15 available frames | Load process A | Load process A



Load process C | Swap out B | Load process D



Segmanlama

- Program segmanlara ayrılır. Tüm programların tüm segmanları aynı boyda olmak zorunda değil.
- Segman boyunun üst sınırı var
- Mantıksal adresler iki bölümden oluşur -segman numarası ve segman içi ofset
  - segman tablosundan segmanın adresi ve boyu alınır
- Segman boyları eşit olmadığından dinamik bölmelemeye benzer
- Bir program birden fazla segmandan oluşabilir

Segmanlama

- Bir programa ilişkin segmanlar bellekte ardışıl yerleşmek zorunda değil
- Segman tabloları var (yükleme adresi ve segman boyu bilgileri)
- Segmanlar programcıya şeffaf değil
- Kullanıcı / derleyici program text ve veri alanlarını farklı segmanlara atar.
- Maksimum segman boyu bilinmesi gerekir

7.2 GÖRÜNTÜ BELLEK

Program Koşması

- Bellek erişimleri dinamik olarak çalışma anında fiziksel adreslere dönüştürülür
  - Proses çalışması boyunca belleğe alınıp, bellekten atılabilir ve her seferinde farklı bir bölgeye yerleştirilebilir.
- Prosesin parçalarının bellekte birbirini izleyen bölgelerde olması gerekli değil
- Çalışma anında prosesin tüm parçalarının birden bellekte olması gerekmez

Program Koşması

- İşletim sistemi başlangıçta belleğe prosesin bir kısmını yükler
- Yerleşik set (resident set) - prosesin bellekte bulunan kısmı
- İhtiyaç duyulan bir bölge bellekte yoksa kesme oluşur
- İşletim sistemi prosesi bloke eder

Program Koşması

- İstenen mantıksal adresi içeren parçası belleğe yüklenir
  - İşletim sistemi tarafından disk G/Ç isteği oluşturulur
  - Disk G/Ç işlemi yürütülürken bir başka proses çalışır
  - Disk G/Ç işlemi tamamlanınca kesme oluşur. İşletim sistemi bekleyen prosesi *Hazır* durumuna getirir.

Prosesi Parçalara Ayırmanın Avantajları

- Ana bellekte daha fazla proses bulunabilir
  - Prosesin sadece gerekli parçaları yüklenebilir
  - Bellekte çok proses olduğundan en az birinin *Hazır* durumunda olması olasılığı yüksek
- Bir proses tüm ana bellekten daha büyük olabilir

Bellek

- Gerçek bellek
  - Ana bellek
- Görüntü bellek
  - Disk üzerinde oluşturulan ikincil bellek
  - Çoklu programlamayı daha etkin kılar.
  - Ana bellek boyu kısıtlarından programcıyı kurtarır.

Thrashing

- Bellekten atılan bir parçaya hemen ihtiyaç duyulması durumu
- İşlemci zamanı proses parçalarını ana bellek ve ikincil bellek arasında taşımakla geçer.
- Bu soruna karşılık, işletim sistemi, prosesin geçmişine bakarak hangi parçalara ihtiyaç duyacağını veya duymayacağını tahmin eder

Yerellik Prensibi

- Proses içi program kodu ve veri erişimleri birbirine yakın bölgelerde kalma eğilimindedir
- Kısa bir süre içinde prosesin sadece küçük bir alt kümesi gerekecektir
- Hangi parçaların gerekeceği konusunda tahminde bulunmak mümkün
- Etkin bir çalışma sağlamak mümkün

Görüntü Bellek İçin Gerekli Destek

- Donanım sayfalamaya ve segmanlı yapıya destek vermeli
- İşletim sistemi ana bellek ile ikincil bellek arasında sayfa ve/veya segman aktarımını etkin bir şekilde düzenleyebilmeli.

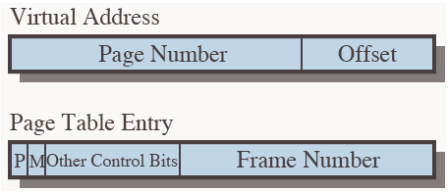
Sayfalama

- Her prosesin kendi sayfa tablosu var
- Tablonun her satırında sayfanın ana bellekte yer aldığı çerçeve numarası bulunur
- Sayfanın ana bellekte olup olmadığını gösteren de bir bit gerekli.

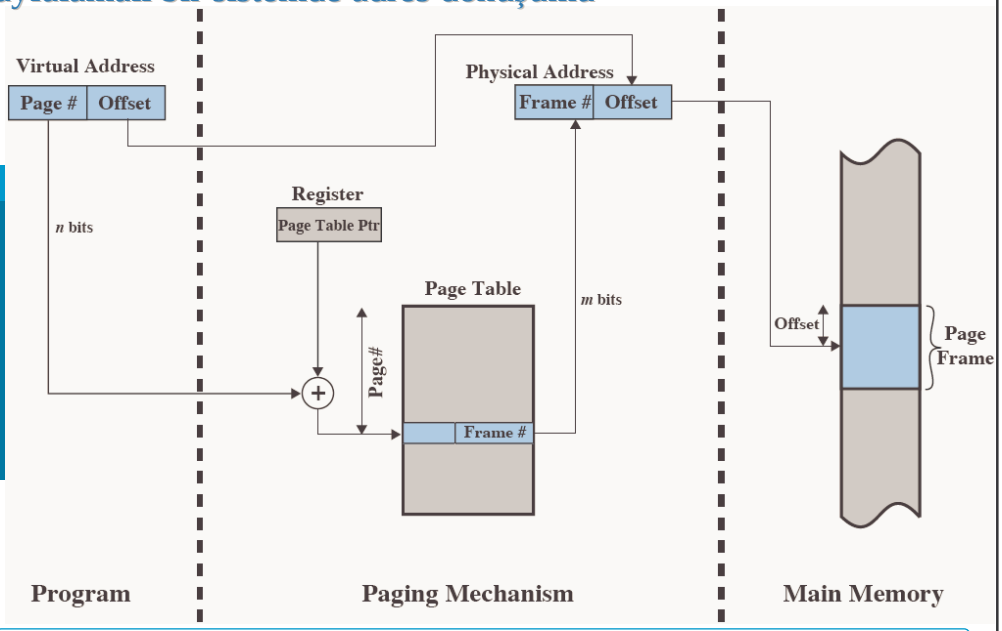
Sayfa Tablosundaki Değişti Biti

- Sayfanın belleğe yüklendikten sonra değişip değişmediğini gösterir
- Değişiklik yoksa ana bellekten ikincil belleğe alınırken yeniden yazmaya gerek yok

Sayfa Tablosu Kayıtları



Sayfalamalı bir sistemde adres dönüşümü



Sayfa Tabloları

- Sayfa tablosunun tamamı çok yer gerektirebilir.
- Sayfa tabloları da ikincil bellekte saklanır
- Koşan prosesin sayfa tablolarının bir kısmı da ana belleğe alınır.

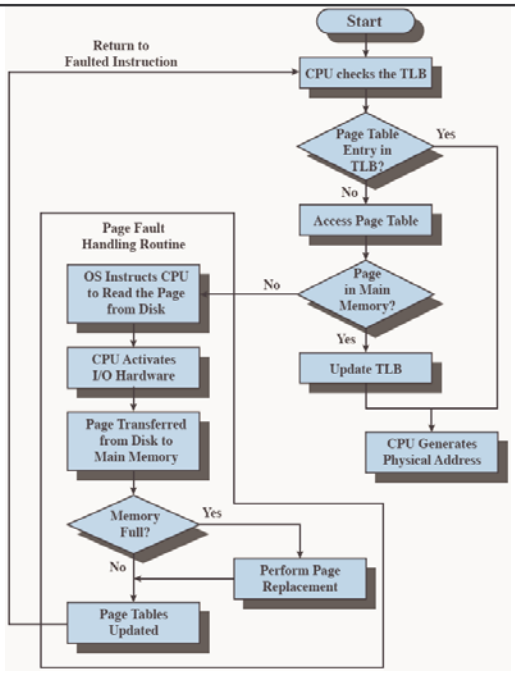
Translation Lookaside Buffer

- Her görüntü bellek erişiminde iki fiziksel bellek erişimi olabilir:
  - sayfa tablosunu getirmek için
  - veriyi getirmek için
- Bu problemin çözümünde sayfa tablosu kayıtlarını tutmak için hızlı bir cep bellek kullanılır:
  - TLB - Translation Lookaside Buffer

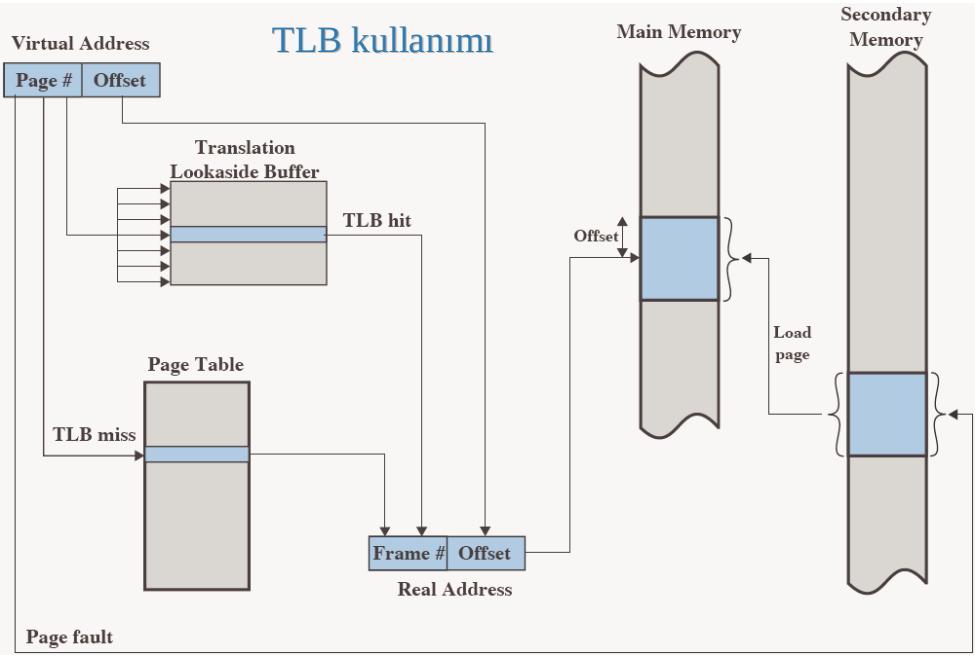
Translation Lookaside Buffer

- En yakın zamanda kullanılmış olan sayfa tablosu kayıtlarını tutar
- Ana bellek için kullanılan cep bellek yapısına benzer bir işlev görür

TLB kullanım akışı



TLB kullanımı



Sayfa Boyu

- Sayfa boyu küçük olursa iç parçalanma daha az
- Küçük sayfa boyları olursa proses başına gereken sayfa sayısı artar.
- Proses başına fazla sayfa olması sonucunda sayfa tablosu boyları büyür.
- Sayfa tablosu boyunun büyük olması sonucu tablonun ikincil bellekte tutulan kısmı daha büyük
- İkincil belleklerin fiziksel özellikleri nedeniyle daha büyük bloklar halinde veri aktarımı daha etkin ⇒ sayfa boyunun büyük olması iyi

Sayfa Boyu

- Sayfa boyu küçük olunca bellekteki sayfa sayısı artar
- Zaman içinde proseslerin yakın zamanda eriştikleri sayfaların büyük kısmı bellekte olur. *Sayfa hatası* düşük olur.
- Sayfa boyu büyük olunca sayfalarda yakın zamanlı erişimlere uzak kısımlar da olur. Sayfa hataları artar.

Sayfa Boyu

- Birden fazla sayfa boyu olabilir.
- Büyük sayfa boyları program komut bölümleri için kullanılabilir
- Küçük boylu sayfalar iplikler için kullanılabilir
- Çoğu işletim sistemi tek sayfa boyu destekler

Örnek Sayfa Boyları

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1024 36-bit word
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 kbyes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
PowerPc	4 Kbytes
Itanium	4 Kbytes to 256 Mbytes

Alma Yaklaşımları

- Alma yöntemi
  - Sayfanın belleğe ne zaman alınması gerektiğini belirler
  - "*İsteğe dayalı sayfalama*" kullanılıyorsa ancak sayfaya erişim olduğunda belleğe getirir
    - sayfa hatası başta daha yüksek
  - "Önceden sayfalama" yöntemi kullanıldığında gerektiğinden daha fazla sayfa belleğe alınır
    - Diskte birbirini izleyen konumlarda yer alan sayfaları birlikte belleğe getirmek daha etkin

Yerine Koyma Yaklaşımları

- Yerleştirme yöntemi
  - Hangi sayfanın yerine konacak?
  - Bellekten atılacak sayfa yakın zamanda erişilmesi olasılığı düşük olan bir sayfa olmalı.
  - Çoğu yöntem bir prosesin gelecek davranışını eski davranışına dayanarak kestirmeye çalışır.

Yerine Koyma Yaklaşımları

- Çerçeve kilitleme
  - Bir çerçeve kilitliyse yerine başkası yerleştirilemez
  - İşletim sistemi çekirdeği
  - Kontrol yapıları
  - G/Ç tamponları
  - Her çerçeveye bir kilit biti atanması

Temel Yerine Koyma Algoritmaları

- Optimal yöntem
  - Bir sonraki erişimin olacağı zamanın en uzak olduğu sayfanın seçilmesi
  - Gelecek olaylar hakkında kesin bilgi olması imkansız

Temel Yerine Koyma Algoritmaları

- En Uzun Süredir Kullanılmamış (Least Recently Used (LRU)) yöntemi
  - En uzun zamandır erişim olmamış olan sayfayı seçer
  - Yerellik prensibine göre yakın zamanda da bu sayfaya erişim olmayacaktır
  - Her sayfada en son erişim zamanı bilgisi tutulur. Ek yük getirir.

Temel Yerine Koyma Algoritmaları

- İlk Giren İlk Çıkar (First-in, first-out (FIFO))
  - Prosese atanmış sayfa çerçevelerini çevrel bir kuyruk olarak ele alır.
  - Sayfalar sıralı olarak bellekten atılır
  - Gerçeklenmesi en basit yöntem
  - Bellekte en uzun kalmış sayfanın yerine konur. Ancak bu sayfalara yakın zamanda erişim olabilir!

Temel Yerine Koyma Algoritmaları

- Saat yöntemi
  - Kullanım biti adını alan ek bit
  - Sayfa belleğe yüklendiğinde kullanım bitine 1 yüklenir
  - Sayfaya erişimde kullanım biti bir yapılır
  - Çevrel kuyruk tutulur. İşaretçi en son belleğe alınan sayfanın bir sonrasını gösterir.
  - Bellekten atılacak sayfa belirlenirken bulunan kullanım biti 0 olan ilk sayfa seçilir.
  - Atılacak sayfa belirlenirken 1 olan kullanım bitleri de sıfırlanır.
  - Kullanım biti 0 olan yoksa ilk tur tamamlanır, ikinci turda daha önce sıfır yaptıklarının ilki seçilir.

Temel Yerine Koyma Algoritmaları

- Sayfa tamponlama
  - Bellekten atılan sayfa şu iki listeden birisine eklenir:
    - sayfada değişiklik olmadıysa boş sayfalar listesine
    - değişiklik yapılmış sayfalar listesine

Temizleme Yaklaşımları

- İsteğe dayalı temizlik
  - sayfa ikincil belleğe ancak seçilirse atılır
- Önceden temizlik
  - sayfalar gruplar halinde ikincil belleğe atılır

Temizleme Yaklaşımları

- En iyi yaklaşım sayfa tamponlama ile
  - Atılacak sayfalar iki listeden birisine konulur
    - Değişenler ve değişmeyenler
  - Değişenler listesindeki sayfalar periyodik olarak gruplar halinde ikincil belleğe alınır
  - Değişmeyenler listesindekiler yeniden erişim olursa yeniden kullanılır ya da çerçevesi başka sayfaya verilirse kaybedilir

LINUX'ta Bellek Yöntemi

- Görüntü Bellek Adresleme
  - 3 seviyeli sayfa tablosu yapısı şu tablolardan oluşur: (her tablo bir sayfa boyunda)
    - sayfa kataloğu
      - her aktif prosesin bir sayfa kataloğu var.
      - boyu bir sayfa
      - her kayıt orta aşama sayfa kataloğunun bir sayfasına işaret
      - her aktif proses için bellekte yer almalı

LINUX'ta Bellek Yöntemi

- orta aşama sayfa kataloğu
  - birden fazla sayfadan oluşabilir
  - her kayıt sayfa tablosunda bir sayfaya işaret eder
- sayfa tablosu
  - birden fazla sayfadan oluşabilir
  - her kayıt prosesin bir sanal sayfasına işaret eder

LINUX'ta Bellek Yöntemi

- görüntü adres 4 alandan oluşur
  - en yüksek anlamlı alan sayfa kataloğundaki bir kayda indis
  - ikinci alan orta aşama sayfa kataloğundaki bir kayda indis
  - üçüncü alan sayfa tablosundaki bir kayda indis
  - dördüncü alan seçilen sayfadaki offset adresi
- platformdan bağımsız olarak 64 bitlik adresler

LINUX'ta Bellek Yöntemi

- Sayfa atama
  - buddy sistemi kullanılır
- Sayfa yerine koyma
  - sayfa yöntemine dayalı bir yaklaşım
  - kullanım biti yerine 8 bitlik yaş alanı var
  - yaşlı olan (uzun zamandır erişilmemiş) sayfalar öncelikle bellekten atılır