YZM 3102
İşletim Sistemleri

Yrd. Doç. Dr. Deniz KILINÇ
Celal Bayar Üniversitesi
Hasan Ferdi Turgutlu Teknoloji Fakültesi
Yazılım Mühendisliği

BÖLÜM – 8

Hafıza Yönetimi Bölümünde,

- Giriş
- Temel Donanım Yapısı
- Adres Bağlaması
- Mantıksal ve Fiziksel Adres
- Takaslama (Swapping)
- Bitişik Hafıza Atama
- Segmentation
- Paging

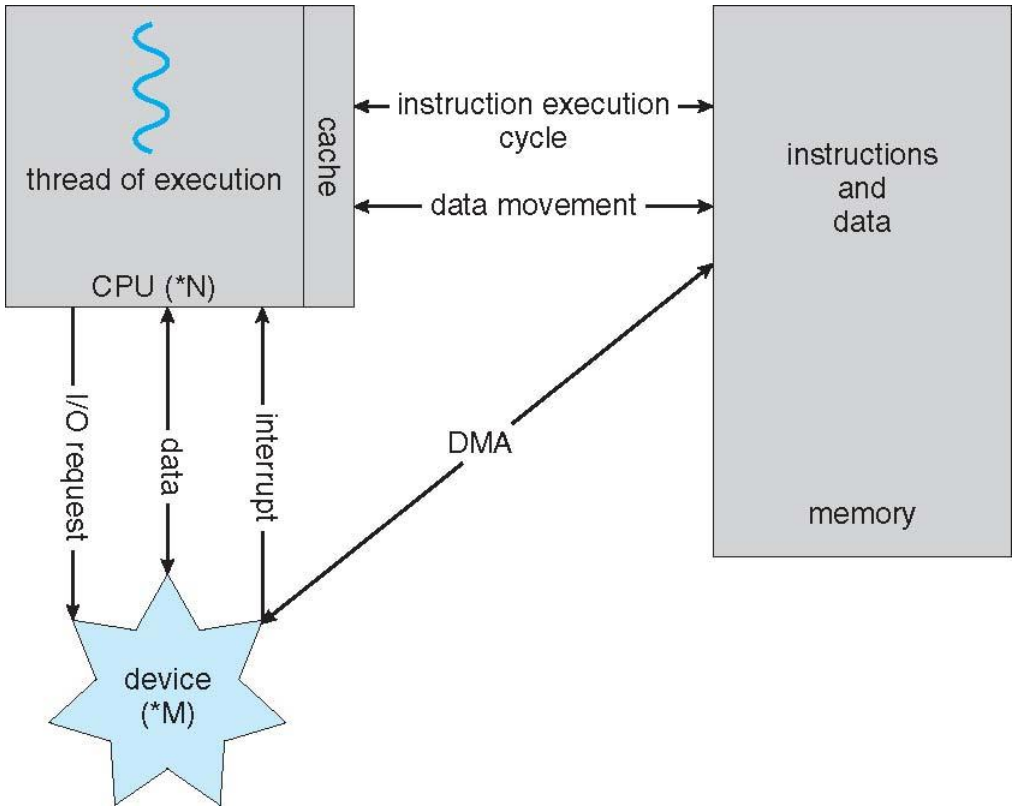
konularına değinilecektir.

Giriş

- Modern bilgisayar sistemlerinin çalışmasında **hafıza** merkezi role sahiptir.
- Hafıza, her birisi kendi adresine sahip olan çok sayıda byte alanına sahiptir. Diğer bir değişle hafıza **Word/byte dizisidir**.
- CPU, program counter (PC) değerine göre hafızadan bir komutu fetch eder.
- Hafızadan alınan komutlar, bir veya birden fazla parametre için hafıza erişimi (operand) gerektirebilirler.
- Komutun çalıştırılmasından sonra **elde edilen sonuç *hafızaya tekrar yazılabilir***.

Von Neumann Mimarisi

- Von Neumann mimarisine dayalı **komut işleme çevrimi** (**instruction - execution cycle**)



- Von Neumann mimarisi **tek bir veri yolu** üzerinden komut ve verilerin **iletişimini** yapan *işlemci, bellek, ve giriş/çıkış* birimlerinden oluşur.

Von Neumann Mimarisi (devam…)

- İlk olarak bellekteki komut getirilerek (**fetch**), komut kayıtçısına (**instruction register**) saklanır.
- Daha sonra komut **decode edilerek**; bellekteki **gerekli operandlar** bellekten getirilir ve bazı dahili kayıtçılarda saklanır.
- Operandlar üzerindeki **komutlar çalıştırıldıktan sonra**, işlem sonucu **tekrar** *belleğe yazılır*.
- **Dikkat:** Ana bellek birimi sadece bellek adres bilgilerini (stream) görür. Onların nasıl ve ne için oluştuğunu bilmez ve ilgilenmez.

Temel Donanım Yapısı

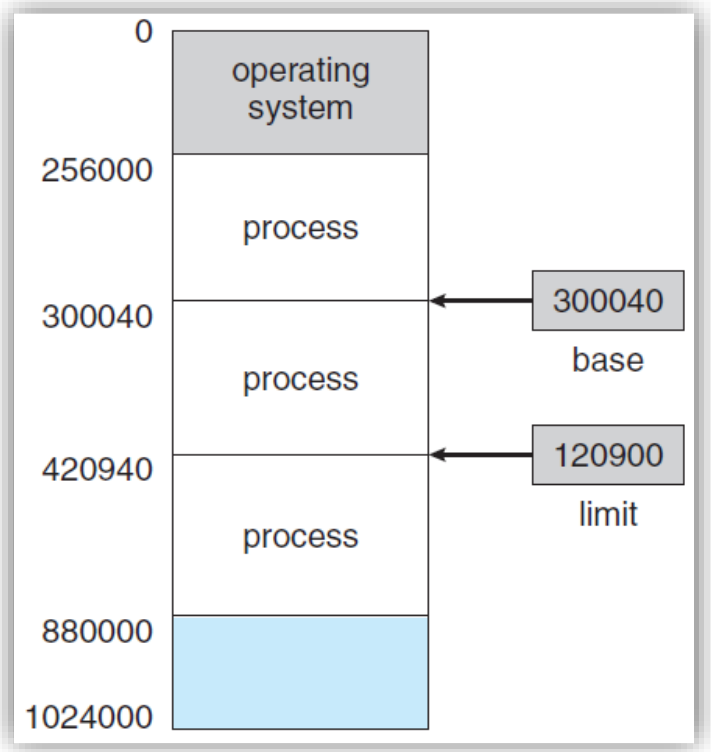
- Main memory ve genel amaçlı kayıtçılar (general purpose registers), **CPU tarafından adreslenen genel amaçlı kayıt alanlarıdır**.
- **Makine komutlarında hafıza** adresini parametre (operand) olarak alan **komutlar vardır**, ancak disk adresini alan komutlar yoktur.
- CPU'nun ihtiyaç duyduğu veri veya komut hafızada değilse, **öncelikle hafızaya alınmalıdır**.
- CPU içerisindeki register'lara genellikle bir cycle ile erişilebilmektedir.
- Hafızaya erişimi bus üzerinden yapılır ve register'a göre **oldukça uzun süre** gerektirir.
- **Hafıza ile CPU arasına** *çok daha hızlı* ve **CPU'ya yakın** bir **saklama alanı** oluşturulur(cache).

Temel Donanım Yapısı (devam…)

- Diğer kullanıcı proseslerin **başka bir prosese ayrılan alana erişiminin engellenmesi** gereklidir.
- Çok kullanıcılı sistemlerde **bir kullanıcı prosesine başka kullanıcının erişiminin** de **engellenmesi** gereklidir.
- Bu tür koruma işleri **donanımsal düzeyde** yapılır. **İşletim sistemi düzeyinde yapıldığında** *performans düşer*.
- Her proses **kendisine ait ayrı bir hafıza alanına** sahiptir.
- Böylelikle, prosesler **birbirinden ayrılmış olur** ve birden fazla proses eşzamanlı çalıştırılabilir.

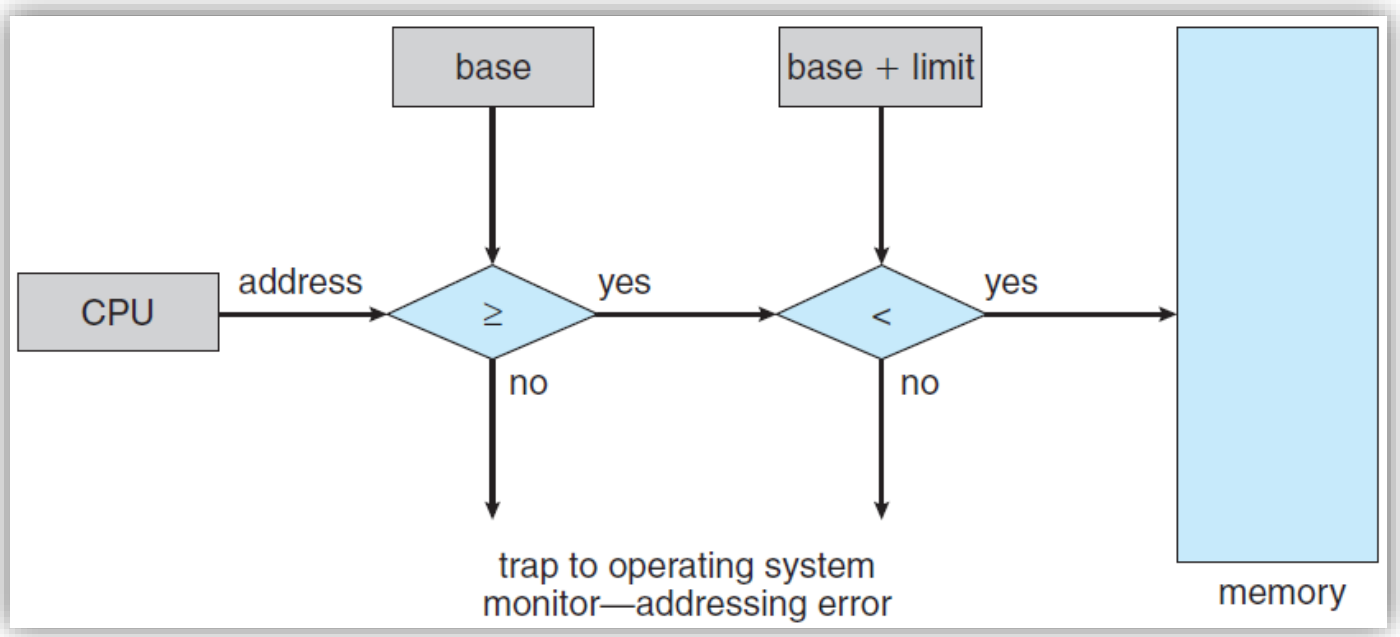
Temel Donanım Yapısı (devam…)

- Bir proses için **ayrılan alanın** **başlangıç adresi** (base register) ve **boyutu** (limit register) belirlenmelidir.



Temel Donanım Yapısı (devam…)

- Hafıza alanının korunması **donanımla gerçekleştirilebilir**.
- Kullanıcı modunda istek yapılan her hafıza adresinin base ile **base + limit** aralığında olduğu **kontrol edilir**.

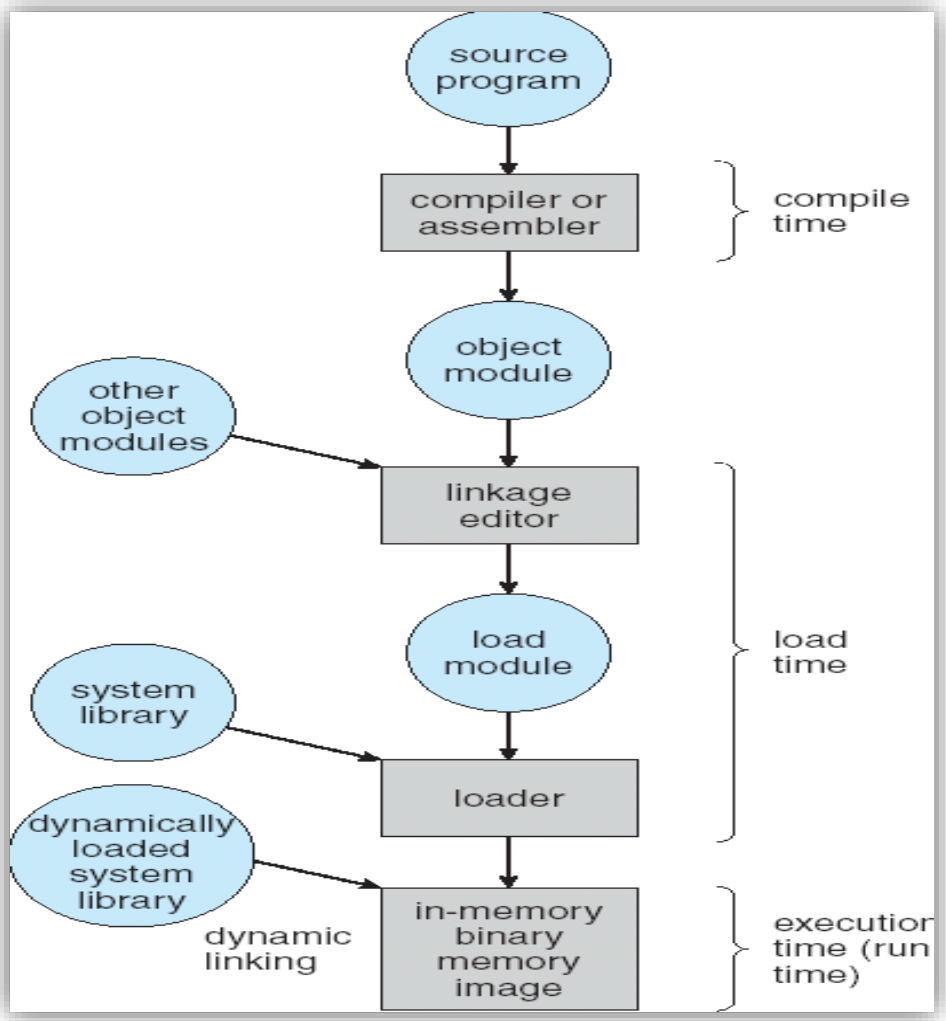


Adres Bağlaması (Binding)

- Bir program **disk üzerinde** **binary dosya olarak** bulunur.
- Bir programın **çalıştırılabilmesi için hafızaya alınması** gereklidir.
- Bir proses, **disk üzerinden → hafızaya alınmak** için kuyruğa alınır (**input queue**).
- Bir proses, **hafızaya yerleştikten sonra** **komutları çalıştırır** veya **hafızadaki veri üzerinde** işlem yapar.
- Prosesin çalışması **tamamlandığında** kullandığı **hafıza alanı boşaltılır**. *Sonraki program* yüklenir ve işletilir.
- Kullanıcı programı ***çalışmadan önce*** ve ***çalışması süresince*** **farklı aşamalardan** ve durumlardan geçer.

Adres Bağlaması (Binding) (devam…)

- Kaynak programda adres **genellikle semboliktir** (count).
- Compiler bu adresleri **yeniden yerleştirilebilir** (**relocatable**) adreslere **dönüştürür** (**Örn.: program başlangıcından itibaren 14.byte**).
- Linkage editör veya loader, bu adresleri **mutlak** (**absolute**) adreslere **dönüştürür** (Örn.: 74014).



Adres Bağlaması (Binding) (devam…)

Adres Atama Türleri

- Belleğe yerleştirilen programların adres atamaları (a)**programlama**, (b)**derleme (compile)**, (c)**yükleme (load)** ve (d)**çalışma (execution)** zamanlarında gerçekleşir.
- **(a)Program yazılırken** tüm adres atamaları belirlenmiş ise *programlama anında* adres atamaları yapılmış olur.
- **(b)Derleme anında** adres ataması, programda bulunan *sembolik adreslerin* derleyici tarafından **fiziksel adrese dönüşümü** ile sağlanır.
- *Başlangıç adresi değiştiğinde* programın **yeniden derlenmesi** gerekir.

Adres Bağlaması (Binding) (devam…)

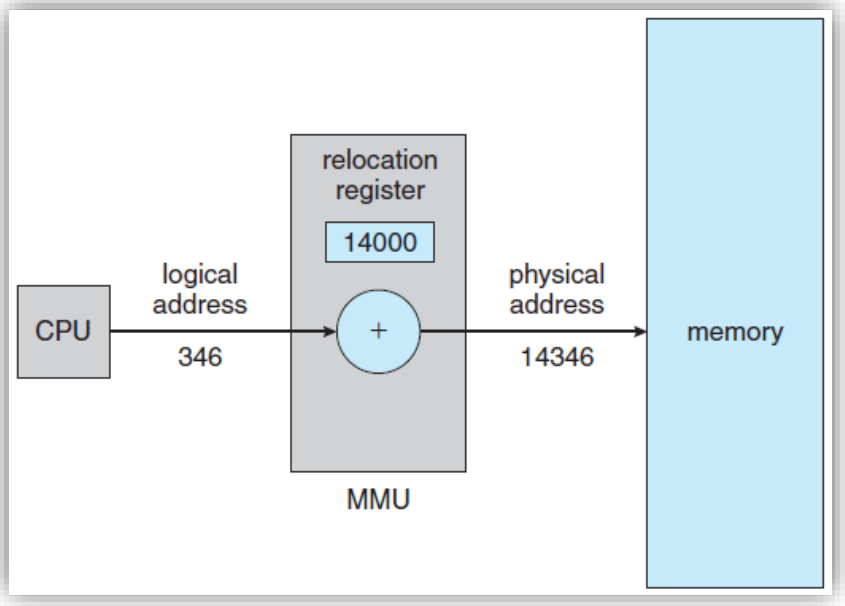
- **(c)Yükleme zamanında** adres ataması, derleyicinin ürettiği **göreceli adreslerin**, programın **yüklenmesi sırasında** **mutlak adrese** çevrilmesi ile gerçekleşir.
- Başlangıç adresi değiştiğinde *kodun sadece yeniden yüklenmesi* gerekir.
- **(d)Çalışma anında** adres atamasında ise, *yüklenen program* hala göreceli adresleri kullanmaktadır ve bu adresler işlemci donanımı tarafından **mutlak adreslere** dönüştürülür.
- Çalışma anında adres ataması, çalışma süresince bir bellek alanından diğerine taşınabilen prosesler için kullanışlıdır. Bu nedenle **işletim sistemlerinde** ***genelde bu yöntem tercih edilir***.
- Çalışma anında adres ataması yönteminde mantıksal adrese **sanal adres** de denir.

Mantıksal ve Fiziksel Adres

- CPU tarafından oluşturulan adres **mantıksal adres** (**logical address**) olarak adlandırılır.
- Hafıza biriminin gördüğü adres **fiziksel adres** (**physical address**) olarak adlandırılır.
- Compile-time veya load-time adres bağlama işlemleri **mantıksal veya fiziksel adres üretir**.
- Execution-time adres binding ise **sanal adrestir** (virtual address) (page number + offset).
- Run-time'da **sanal adresin → fiziksel adrese** dönüştürülmesi donanım bileşeni (**memory-management-unit, MMU**) tarafından yapılır.
- **Base-register** fiziksel adrese dönüştürme için kullanılan *donanım bileşenidir*.

Mantıksal ve Fiziksel Adres (devam…)

- Base-register (relocation register) değeri, CPU'nun hesapladığı adrese eklenir.



- Kullanıcı programı **hiçbir zaman** fiziksel adresi bilmez. **Mantıksal adres: [0, 0+max]** aralığında, **fiziksel adres: [R, R+max]** aralığındadır.

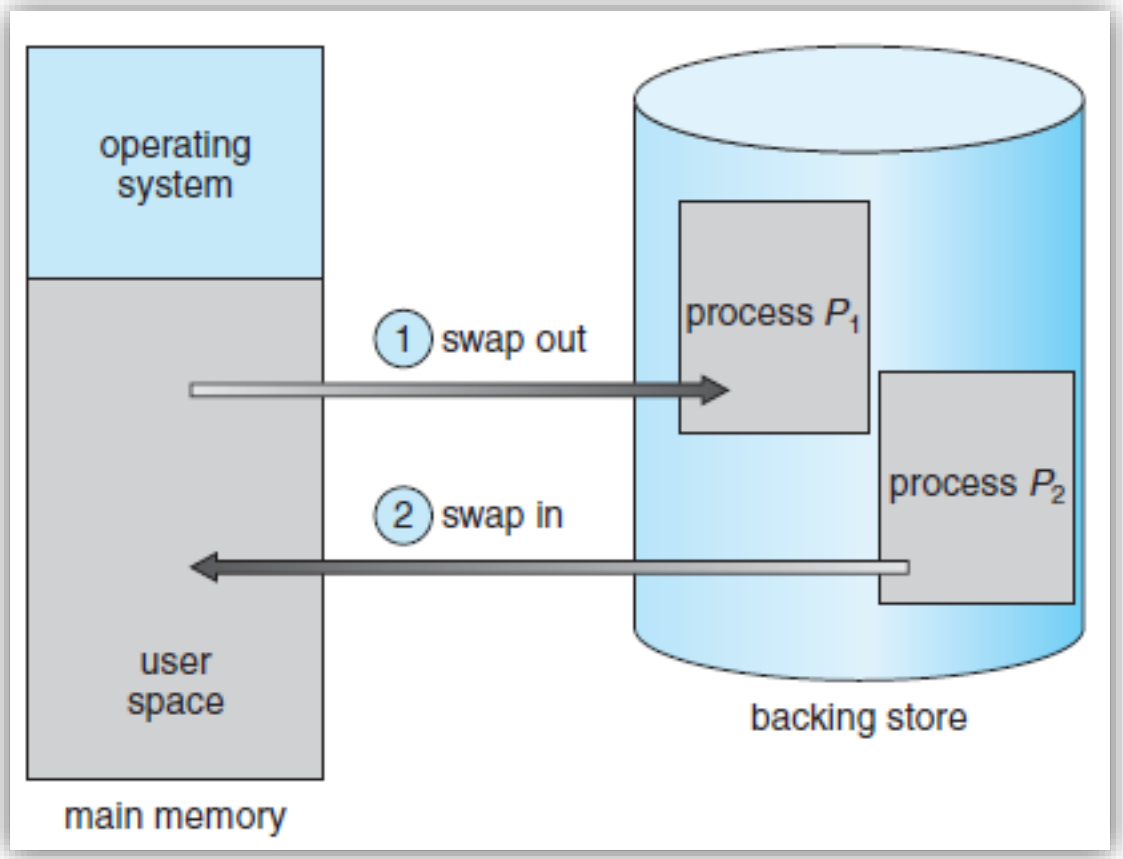
Dinamik Yükleme

- Bir programın tamamı hafızaya yüklenmez gerektikçe modül (blok) halinde yüklenir (dynamic loading).
- Önce main program hafızaya yüklenir ve çalıştırılır.
- Bir program parçası (routine) çalışırken başka routin'i çağırdığında, hafızada yüklü değilse loader tarafından yüklenir.
- **Çok büyük boyuttaki programların** çalıştırılması için hafıza **yönetimi açısından fayda sağlar**.
- **Sık kullanılmayan** rutin'lerin (hata yordamları) *hafızada sürekli bulunmasını engeller*.

Swapping – Takaslama

- Bir proses çalışmak için **hafızada olmak zorundadır**.
- Bir proses **geçici olarak diske** (**backing store**) aktarılabilir ve tekrar hafızaya alınabilir (**swapping**).
- Ready queue (hazır kuyruğu), **CPU'da çalıştırılmak üzere bekleyen prosesleri tutar**.
- CPU scheduler bir prosesi çalıştırmaya karar verdiğinde **dispatcher'ı çağırır**.
- Dispatcher, çalışacak prosesin **hazır kuyruğunda olup olmadığını kontrol eder** ve kuyrukta ise çalıştırır.
- **Kuyrukta değilse ve hafızada yeterli yer yoksa başka bir prosesi hafızadan atar** (swap out) ve **istenen prosesi yükler** (swap in).

Swapping – Takaslama (devam…)



Swapping – Takaslama (devam…)

- **İki prosesin yer değiştirmesi** **context-switch** işlemini gerektirir ve uzun süre alır.
- 100 MB'lık bir prosesin 50MB/sn hızındaki bir diske kaydedilmesi için 2sn gerekir. İki prosesin yer değiştirmesi 4sn süre alır.
- Proseslerin dinamik hafıza gereksinimleri için request_memory() ve release_memory() sistem çağrıları kullanılır.
- Bir prosesin **swap out yapılabilmesi için** tüm **işlemlerini bitirmesi zorunludur**.
- Bir proses **I/O kuyruğunda bekliyorsa** veya **başka bir işlem sonucunu bekliyorsa** **swap out yapılamaz**.
- ***Modern işletim sistemleri*** **hafıza** **eşik değerin altına düşmeden** **swapping yapmaz**.

Bitişik Hafıza Atama

- Main memory **hem işletim sitemini** hem de **kullanıcı programlarını** yerleştirmek zorundadır.
- Bitişik hafıza atama yönteminde **hafıza iki parçaya ayrılır**:
  a) İşletim sisteminin yerleştiği kısım
  b) Kullanıcı proseslerinin yerleştiği kısım
- İşletim sistemi **hafızanın başlangıcına** veya **sonuna** yerleşebilir.
- **Interrupt vector table** ***düşük adrese*** veya ***yüksek adrese*** yerleştirilebilir.
- İşletim sistemi ile **interrupt vector tablosu** genellikle ***aynı tarafa*** yerleştirilir.
- Bitişik hafıza atama yönteminde bir **proses için ayrılan alan** ***tek bölümden oluşur*** ve ***sonraki proses*** için ayrılan ***yere kadar devam edebilir***.

Bitişik Hafıza Atama (devam…)

- Bir prosese, **hafıza alanı atama** ***işletim sistemine göre farklı şekillerde*** yapılabilir: **a)fixed-sized** ve **b)variable partitions**.

a) Hafıza, çok sayıda, sabit boyutta küçük parçaya ayrılabilir (**fixed-sized partitions**) ve **her parça bir prosesi içerebilir** (**multiple partition**).
   - Multiprogramming sistemlerde, **eşzamanlı çalışan program sayısı partition sayısına bağlıdır**.
   - Bir partition boşaldığında, hazır kuyruğunda bekleyen bir proses seçilerek partition atanır.
   - IBM OS/360 işletim sistemi kullanmıştır günümüzde kullanılmamaktadır.

Bitişik Hafıza Atama (devam…)

b) Değişken parçalı (**variable-partition**) yönteminde işletim sistemi hafızanın **boş ve dolu olan parçalarını** bir **tabloda tutar.**
   - Bu yöntemde, her prosese **farklı boyutta parça ayrılabilir**.

Bitişik Hafıza Atama (devam…)

- Bir proses sisteme girdiğinde **ihtiyaç duyacağı hafıza alanı ayrılabilirse** **hafızaya yüklenir** ve **CPU'yu beklemeye başlar**.
- Bir proses **sonlandığında**, **ayrılan hafıza alanı serbest bırakılır**.
- **Herhangi bir anda**, işletim sisteminde ***kullanılabilir hafıza*** blokları listesi ile proseslerin giriş kuyruğu kümeleri vardır.
- Kuyruğun başındaki proses için kullanılabilir yeterli alan yoksa beklenir veya **kuyruktaki prosesler taranarak** ***boş alana uygun olan varsa*** seçilir.
- Hafızadaki boş alanların **birleştirilmesi**, **prosese uygun alanın oluşturulması**, **serbest bırakılan alanların birleştirilmesi** işlemleri **dynamic storage allocation problem** olarak adlandırılır.

Bitişik Hafıza Atama (devam…)

- **Dynamic storage allocation** problemi için **3 farklı çözüm** kullanılabilir:
  1. **First fit:** Yeterli boyuttaki **ilk boş alan atanır** ve listede kalan kısım aranmaz.
  2. **Best fit:** Yeterli boyutta alanların **en küçüğü seçilir**. ***Tüm liste aranır***.
  3. **Worst fit:** Yeterli boyuttaki alanların **en büyüğü seçilir**. ***Tüm liste aranır***.
- Simülasyonlarda, alan atama süresinin **first fit ile**, hafıza alanı kullanma verimliliğinin best fit ile daha iyi olduğu görülmüştür.
- First fit yöntemi, **best fit ve worst fit'e göre *daha kısa sürede*** atama ***gerçekleştirmektedir***.

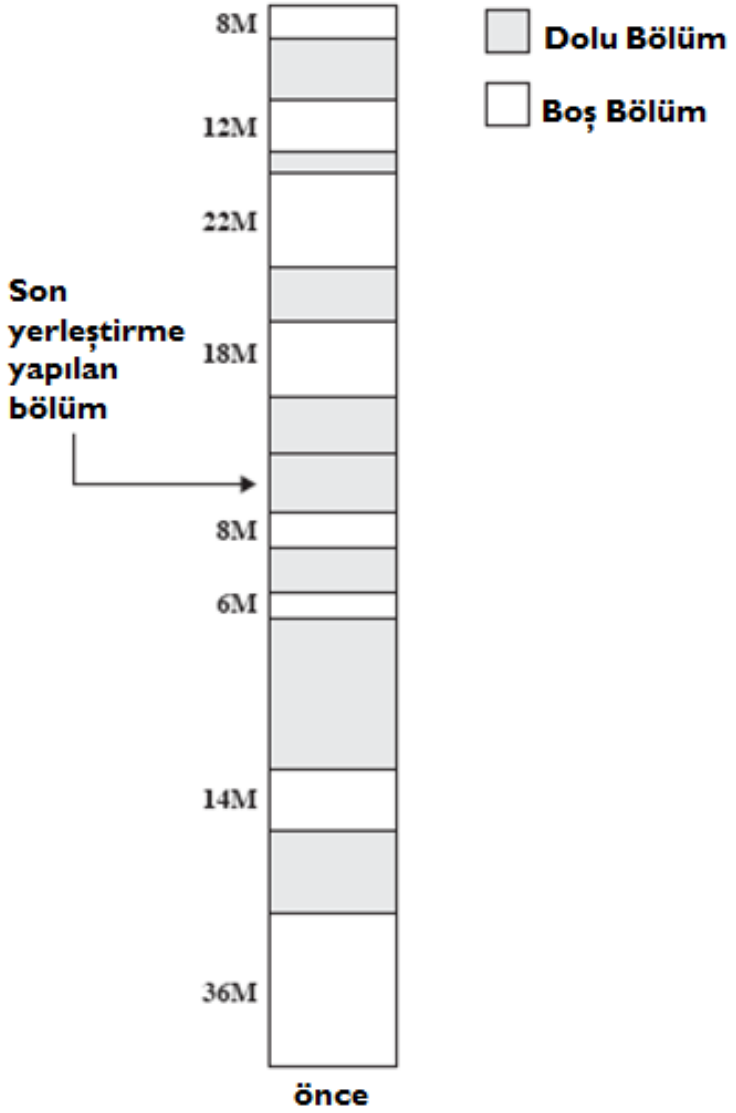
Bitişik Hafıza Atama (devam…)

- Prosesler hafızaya yüklenirken ve atılırken **hafıza alanları sürekli parçalanır** (**fragmentation**).
- Bir proses için ***yeterli alan olabilir***, ancak bunlar **küçük parçalar halinde dağılmış durumda olabilir**.
- **En kötü durumda** her iki proses arasında ***boş kısım olabilir***.
- First fit ile yapılan istatistiksel analize göre, N tane kullanılmış blok için **N/2 tane boş blok** oluşur.
- Bu durumda hafızanın ***1/3 kısmı kullanılamaz***. Buna **%50 kuralı (50-percent rule)** denir.
- Fragmentation **çözümünde** ***küçük bloklar yer değiştirilerek büyük blok elde edilir*** (fazla süre gerektirir).
- **Segmentation** ve **paging** yaklaşımları *fragmentation çözümünde* **etkindir**.

Bitişik Hafıza Atama (devam…)

Örnek 1

- Ana bellekteki boş ve dolu bölümlerin yan tarafta göründüğü gibi olduğunu varsayınız.
- 16M'lık bir yerleştirme isteğini **first fit (ilk uygun)**, **best fit (en iyi uygun)** ve **worst fit (en büyük uygun)** yer algoritmasına göre belleğe yerleştiriniz.



Bitişik Hafıza Atama (devam…)

Örnek 2

- Ana bellekte sırayla yukarıdaki boş bölümlerin olduğunu varsayınız.
- 24K, 20K ve 18K'lık bellek kullanım istekleri için sırasıyla hangi boşlukların kullanılacağını ve oluşacak parçalanmaları (fragmentation) aşağıdaki algoritmaları kullanarak belirtiniz.
  - First Fit Algoritması
  - Best Fit Algoritması
  - Worst Fit Algoritması

Segmentation

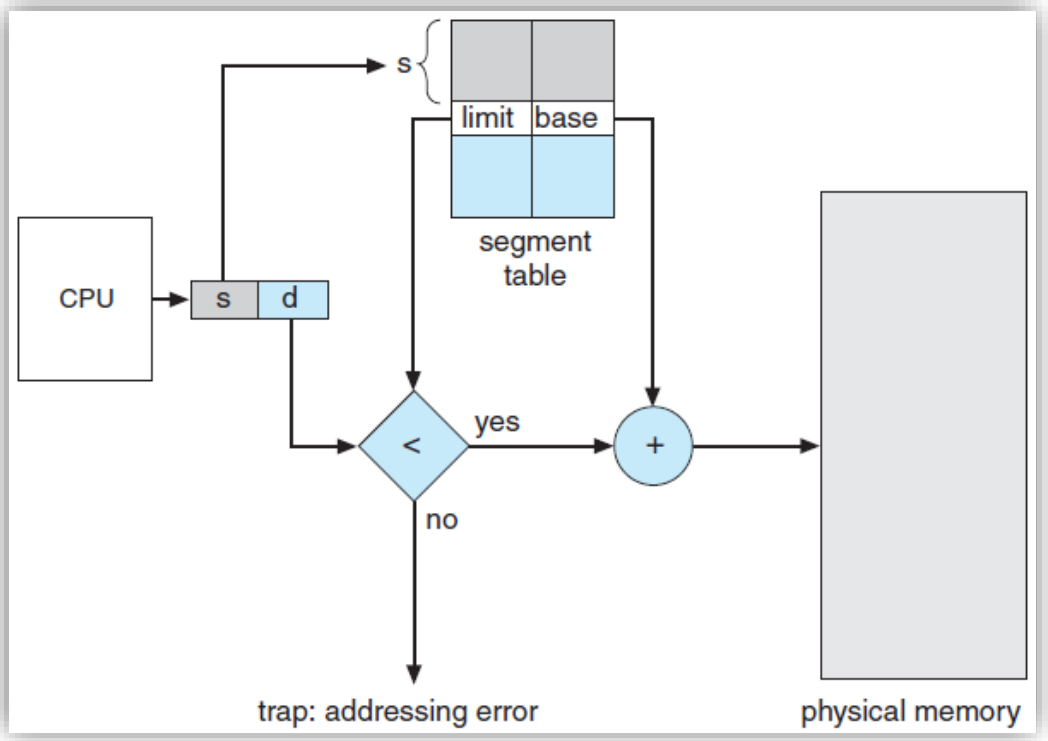
- Segmentation yaklaşımda, her segment bir **isme** ve **uzunluğa** sahiptir.
- Bir **mantıksal adres:** segment adı ile offset (segment içerisindeki konumu) değerini belirler.

<segment number (ad), offset>

- Bir C derleyicisi aşağıdakiler için ***ayrı ayrı segment*** oluşturabilir:
  - Program kodu
  - Global değişkenler
  - Heap (nesneler yerleştirilir)
  - Stack (threadler kullanır, lokal değişkenler, call/return)
  - Standart C kütüphanesi
- Derleme sırasında, derleyici **segment atamalarını** gerçekleştirir.

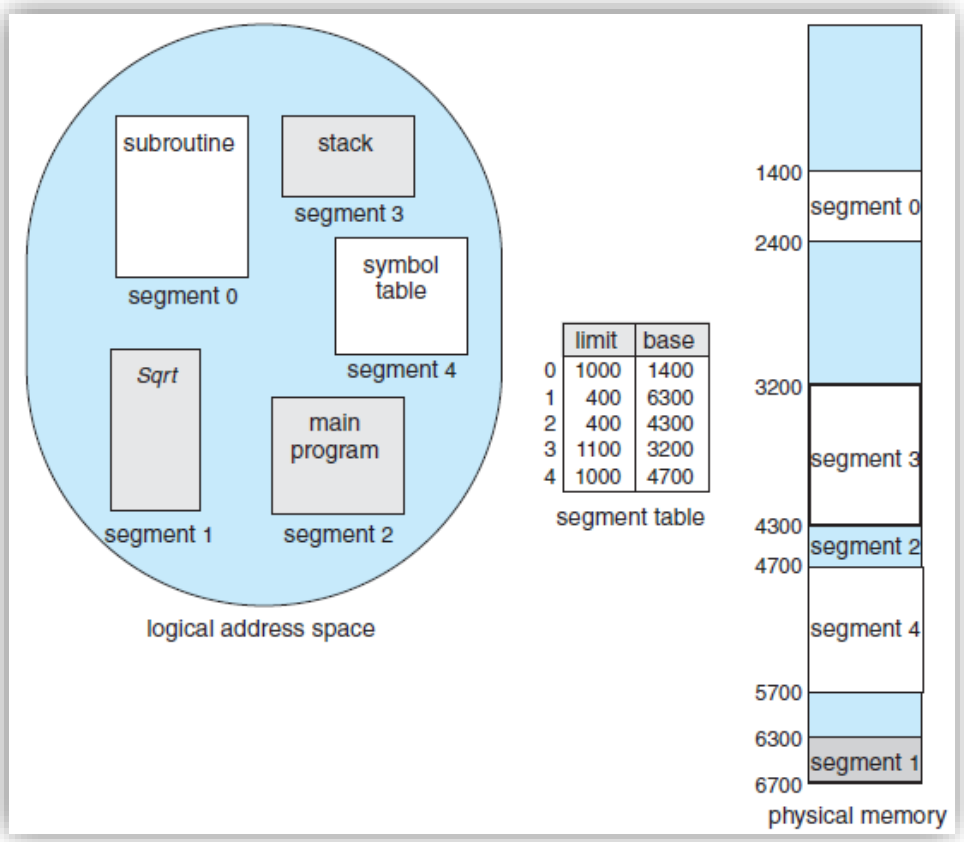
Segmentation (devam…)

- Segment ve offset adresiyle **<u>iki boyutlu adresleme</u>** yapılır.
- Hafıza adresleri ***tek boyutludur*** ve ***dönüştürme işlemi*** gereklidir.



Segmentation (devam…)

- Aşağıda **5 segment vardır** ve aşağıdaki gibi yerleştirilmiştir.



Segmentation (devam…)

Soru:

- Segment tablosunun yandaki gibi olduğunu varsayınız.
- Aşağıdaki mantıksal adreslerin (segment no, ofset) **fiziksel adres** karşılıklarını bulunuz.

a)(0, 198)
b)(2, 156)
c)(1, 530)
d)(3, 455)
e)(0, 252)

Seg.No	Taban Adr.	Uzunluk
0	660	248
1	1752	422
2	222	198
3	996	604

- Fiziksel adresi hesaplamak için aşağıdaki formül kullanılabilir:

Fiziksel Adres(FA)=Kesim Taban Adresi (TA) + Ofset (d)

Segmentation (devam…)

Seg.No	Taban Adr.	Uzunluk
0	660	248
1	1752	422
2	222	198
3	996	604

Cevap:

- İlk olarak her mantıksal adresin hangi segmentte olduğu segment tablosu kullanılarak bulunmalı ve ofset değeri bu segmentin uzunluğu ile karşılaştırılmalıdır.
- Eğer ofset küçükse fiziksel adres hesaplanmalı, değilse bellek erişim hatası oluşturulmalıdır.

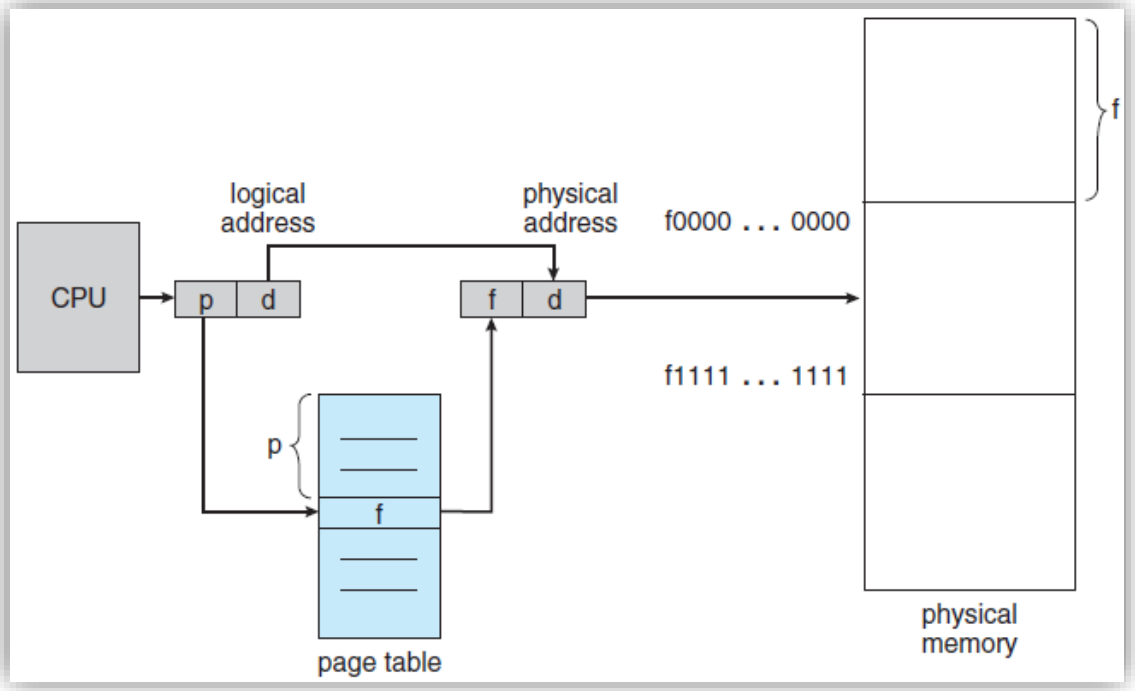
	Mantıksal adres	Fiziksel adres	
a)	(0, 198)	198 < 248	660 + 198 = 858
b)	(2, 156)	156 < 198	222 + 156 = 378
c)	(1, 530)	530 ≮ 422	bellek erişim hatası
d)	(3, 455)	455 < 604	996 + 455 = 1451
e)	(0, 252)	252 ≮ 248	bellek erişim hatası

Paging

- Segmentation ile bir prosese atanan fiziksel adres alanının **bitişik olmamasına** ***izin verilir***.
- Paging ile segmentation'da olduğu gibi proseslere **bitişik olmayan** *hafiza adresleri* atanabilir.
- Paging yönteminde,
  - **Fiziksel hafıza:** **frame** adı verilen **küçük bloklara** bölünür.
  - **Mantıksal hafıza:** ise **aynı boyutta page** adı verilen **bloklara** bölünür.
- Bir proses çalıştırılacağı zaman, **kaynak kodu** **diskten alınarak** → **hafızadaki frame'lere** yerleştirilir.
- Mantıksal adres alanı ile fiziksel adres alanı *birbirinden ayrıştırılmış durumdadır*.

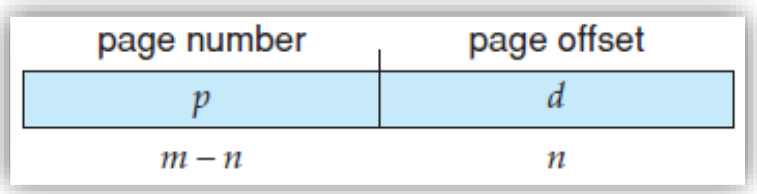
Paging (devam…)

- CPU tarafından oluşturulan her adres **iki parçaya ayrılır:** **page number(p)** ve **page offset(d)**.
- **Page number:** page table içerisindeki ***indeks değeri*** için kullanılır.
- **Page table:** her sayfanın fiziksel hafızadaki ***base adresini*** içerir.



Paging (devam…)

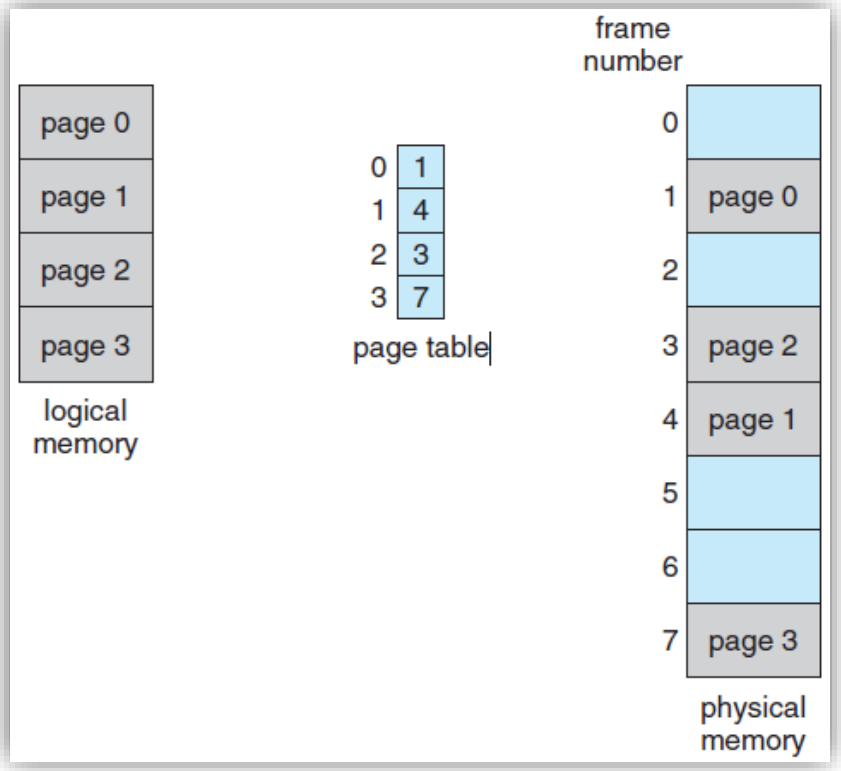
- **Mantıksal adres boyutu 2^m**, **fiziksel adres (offset) boyutu 2^n** byte ise, **sayfa numarası** için soldaki m-n bit, offset değeri için sağdaki n bit alınır.
- Aşağıda örnek mantıksal adres **görülmektedir**.



- **p:** fiziksel bellekteki her bir sayfanın taban adresini tutan sayfa tablosundaki göstergedir.
- **d:** taban adresi ile birleştirilerek fiziksel bellekte sayfanın içerisindeki yerin belirlenmesinde kullanılır.

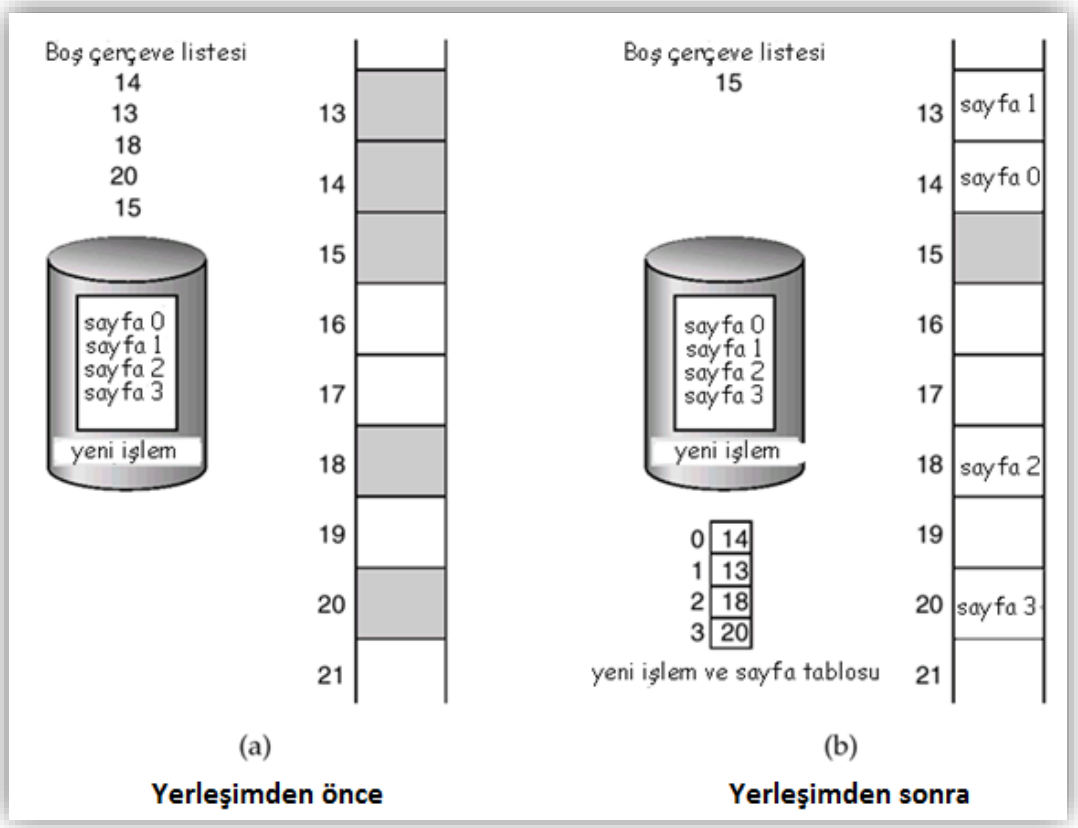
Paging (devam…)

- Page size, **donanım tarafından mikroişlemci mimarisinde** belirlenir.
- Page size 512 byte ile 1 GB arasında olabilir.



Paging (devam…)

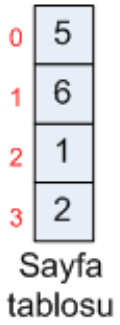
- İşletim sistemi bu yöntemi kullanırken, **boş çerçevelerin** de listesini tutar.



Paging (devam…)

Örnek

- ***4 bayt uzunluğunda*** **sayfalardan** oluşan 32 baytlık bir bellek olduğunu varsayınız. Sayfa tablosu yanda verilmiştir. Aşağıdaki mantıksal adresler için fiziksel adresleri bulunuz.
  a) 0
  b) 3
  c) 4
  d) 13

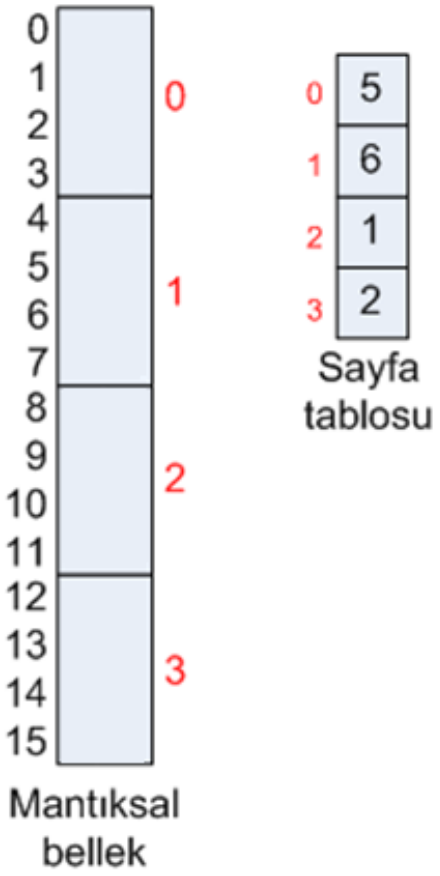


- Fiziksel adresi hesaplamak için ***aşağıdaki formül*** kullanılabilir:
- ***Fiziksel adres (FA) = Çerçeve Numarası (ÇN) * Çerçeve Boyutu (ÇB) + Ofset (O)***

Paging (devam…)

Örnek (devam…)

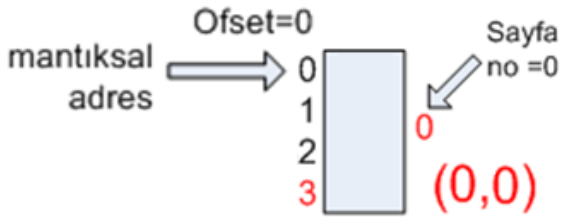
- İlk olarak **mantıksal belleğin** çizilmesi gerekir.
- Ardından **sayfa boyutuna** göre **belleğin bölünmesi** ve her bir mantıksal adresin sayfa numarası ve sayfa ofsetinin hesaplanması gerekir.
- Sayfa numarası, sayfa tablosundan çerçeve numarasını bulmak için kullanılacaktır.



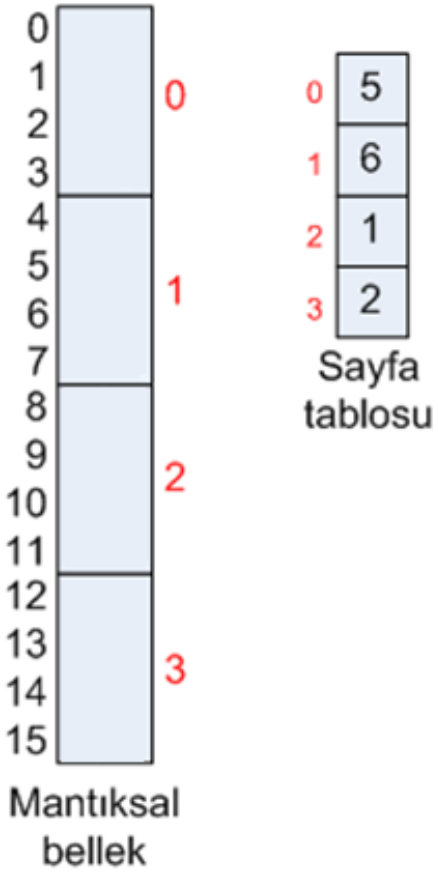
Paging (devam…)

Örnek (devam…)

a) 0



- sayfa numarası=0
- sayfa ofseti=0
- 0 numaralı sayfanın bulunduğu çerçeve numarası=5
- FA=ÇN*ÇB+O
- FA=5*4+0=20

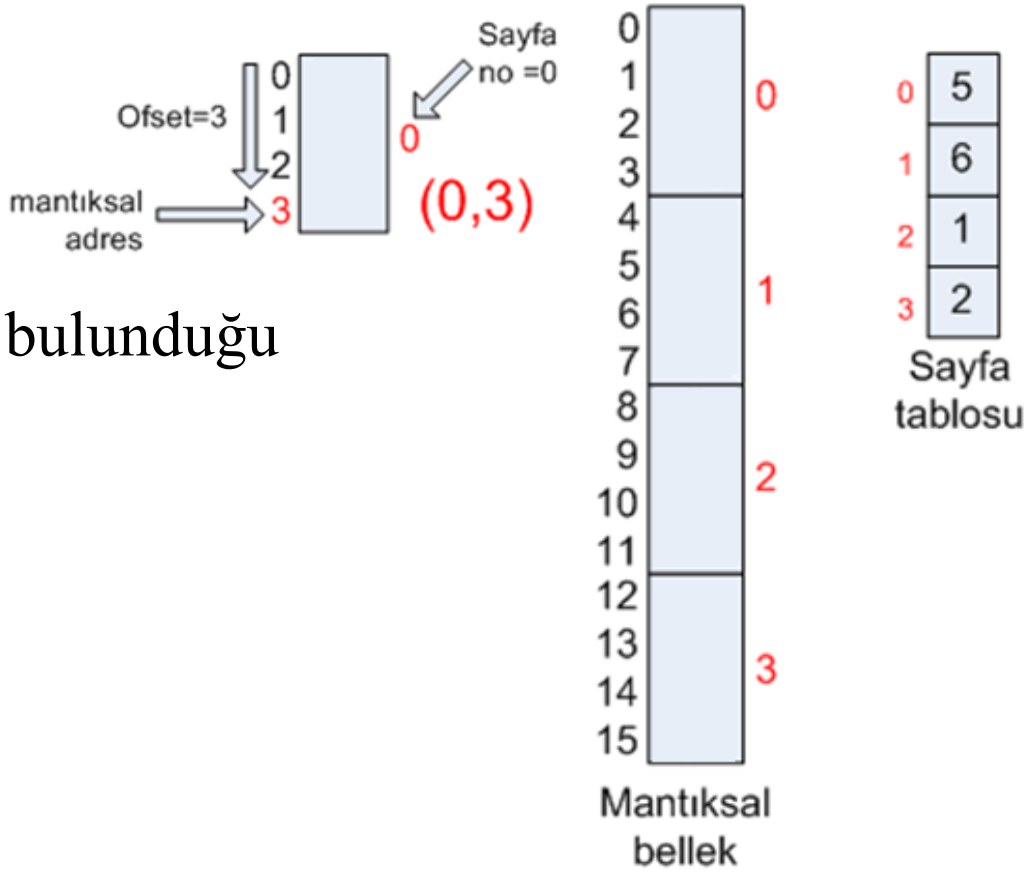


Paging (devam…)

Örnek (devam…)

b) 3

- sayfa numarası=0
- sayfa ofseti=3
- 0 numaralı sayfanın bulunduğu çerçeve numarası=5
- FA=ÇN*ÇB+O
- FA=5*4+3=23

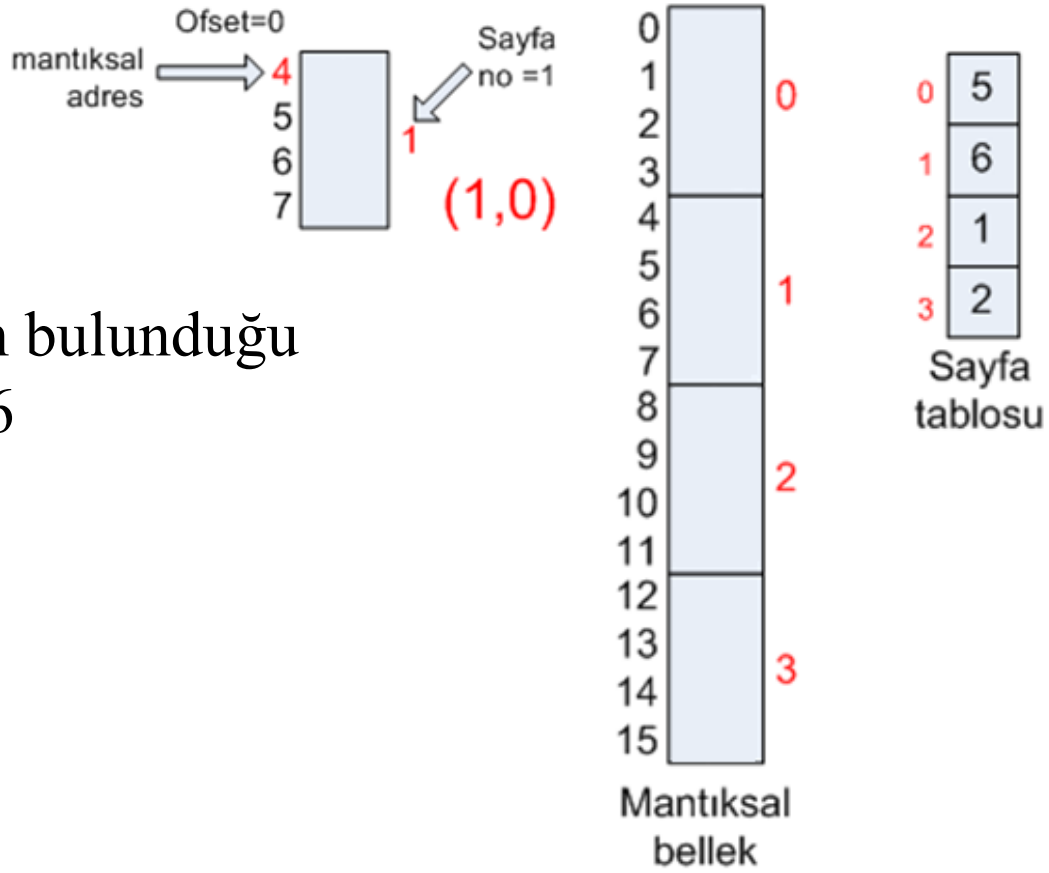


Paging (devam…)

Örnek (devam…)

c) 4

- sayfa numarası=1
- sayfa ofseti=0
- 1 numaralı sayfanın bulunduğu çerçeve numarası=6
- FA=ÇN*ÇB+O
- FA=6*4+0=24

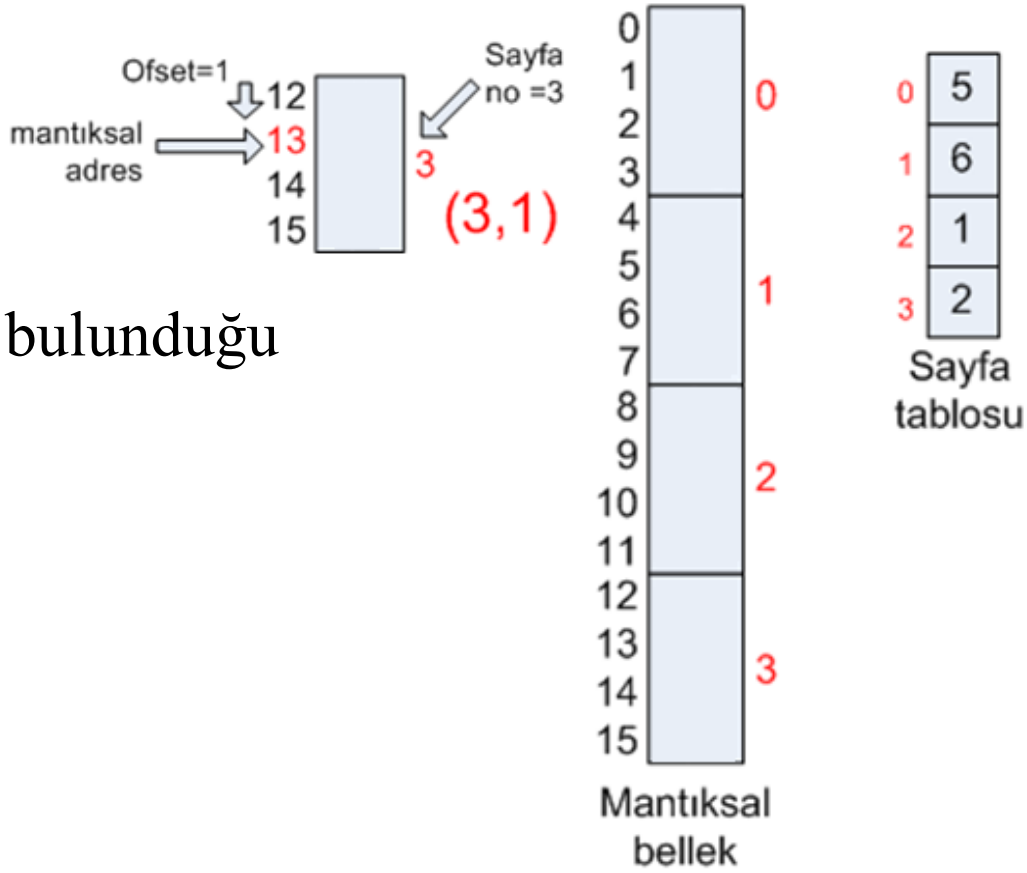


Paging (devam…)

Örnek (devam…)

d) 13

- sayfa numarası=3
- sayfa ofseti=1
- 3 numaralı sayfanın bulunduğu çerçeve numarası=2
- FA=ÇN*ÇB+O
- FA=2*4+1=9



Paging (devam…)

- Paging ile fragmentation oluşabilir.
  - Page size = 2048 byte
  - Proses size = 72766 byte
  - Gerekli alan = 35 sayfa + 1086 byte
  - 1086 byte 36. sayfaya yerleştirilir.
  - Kullanılmayan alan 2048 –1086 = 962 byte
- 36. sayfada 962 byte boş alan kalır.
- En kötü durumda 1 byte kalır ve ayrı sayfaya yerleştirilir.
- Boş alan 2048 –1 = 2047 byte olur.

Paging (devam…)

- 32-bit CPU'da genellikle 32-bit ile page table adresi verilir.
- 2^{32} adet fiziksel page frame bulunur.
- Bir frame boyutu 4 KB (2^{12}) ise,Toplam adreslenebilir fiziksel hafıza = $2^{32} * 2^{12} = 2^{44}$ olur (16 TB).
- Bir proses sisteme çalışmak için geldiğinde, ***gerekli sayfa sayısı belirlenir***.
- Prosesin her sayfası bir frame'e ihtiyaç duyar.
- Toplam **n sayfa varsa**, **en az** **n tane frame** ***boş olmalıdır***.
- Her sayfa bir frame'e yerleştirilir ve frame numarası page table'a kaydedilir.
- Programcı prosesin adresini **tek ve bitişik** olarak görür. ***Frame eşleştirmesini*** **işletim sistemi yapar**.

Paging (devam…)

Translation look-aside buffer (TLB)

- Her işletim sistemi **page table** saklamak için ***kendine özgü yöntem*** kullanır.
- Bazı işletim sistemleri **her proses için ayrı page table** kullanır.
- Her page table için pointer ***ayrı bir register'da*** tutulur.
- Her page table için ***bir grup register*** oluşturulur.
- Modern bilgisayarlarda **page table çok büyüktür** (Örn.: 1 milyon giriş).
- Bu durumda page table için ***register oluşturulması*** **mantıklı değildir**.
- Page table'ın **hafızada tutulması** halinde, her adres değişikliğinde hafıza erişimi gerekli olur (**performans düşer**).
- Mikroişlemcilerde, **küçük boyutta** ve **hızlı donanımsal önbellek (translation look-aside buffer)** ile bu problem çözülür.

Paging (devam…)

Translation look-aside buffer (TLB)

- TLB içerisindeki her giriş satırı page number ile frame number değerlerini tutar.
- Bir mantıksal adres geldiğinde, ***page number*** **tüm TLB** içerisinde **aranır** (full associative).
- Page number değeri **bulunursa**, ilgili satırdaki frame number değeri alınarak *hafızada ilgili sayfaya gidilir*.
- TLB içerisinde **bulunamayan** page number için **page table'a** gidilir.
- Page table'dan alınan **frame number** ile **page number** değeri **TLB'ye kaydedilir**.
- TLB **dolu ise** ***replacement algoritması*** (least recently used - ***LRU***, round robin, random) ile seçilen **satır silinerek** ***yerine yazılır***.

İYİ ÇALIŞMALAR…

Yararlanılan Kaynaklar

- **Ders Kitabı:**
  - Operating System Concepts, Ninth Edition, Abraham Silberschatz, Peter Bear Galvin, Greg Gagne
- **Yardımcı Okumalar:**
  - İşletim Sistemleri, Ali Saatçi
  - Şirin Karadeniz, Ders Notları
  - İbrahim Türkoğlu, Ders Notları
- **M. Ali Akcayol, Gazi Üniversitesi Bilgisayar Mühendisliği Bölümü**
- **BTEP205 - İşletim Sistemleri**