Giriş

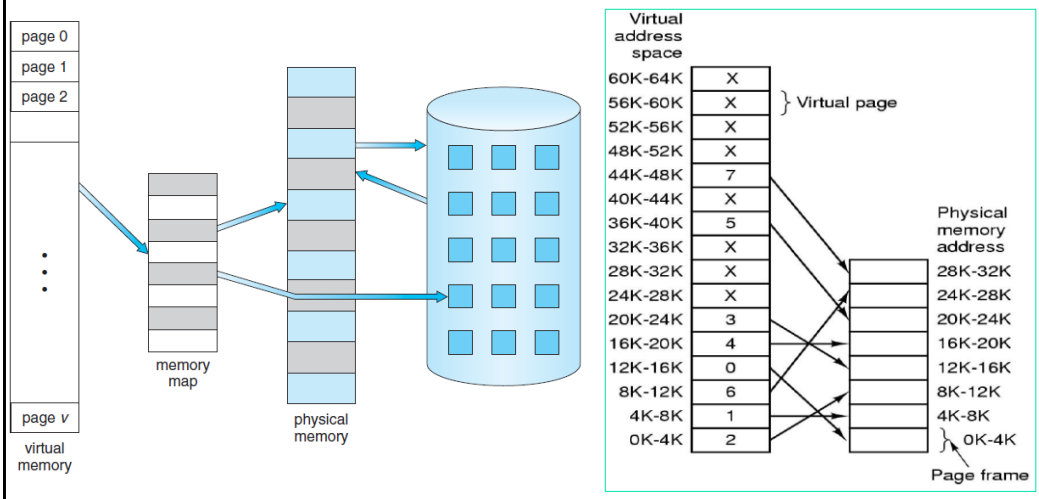
- **Sanal bellek (virtual memory) yöntemi, process'lerin tamamının hafızaya yüklenmeden çalıştırılmasına izin verir.**
- Günümüzdeki programlar fiziksel hafızanın kapasitesinden daha büyük olabilmektedir.
- **Sanal bellek, programcıların hafıza limitlerinden soyutlanmasını sağlar.**
- Sanal bellek, dosyaların paylaşımını ve paylaşılmış hafıza oluşturulmasını kolaylaştırır.
- Bir programın çalışması için tamamının hafızaya yüklenmesine ihtiyaç yoktur:
  - **Nadiren hata yapan programlarda hata yönetimi kodlarının yüklenmesine gerek yoktur.**
  - **Dizi değişkenlerinin aktif kullanılan boyutları nadiren 10 * 10'dan büyük olmaktadır.**

Giriş

- **Bir programın tamamının çalışması gerektiğinde bile, tümü aynı anda gerekmeyebilir.**
- Bir programın hafızaya parçalı bir şekilde alınarak çalıştırılması aşağıdaki faydaları sağlar:
  - **Bir programın toplam boyutu fiziksel hafızanın kapasitesinden fazla olabilmektedir.**
  - **Her kullanıcı programı, aynı anda küçük fiziksel hafıza alanı kullandığından, çok sayıda program eş zamanlı çalıştırılabilir.**
- Sanal bellek, programcı için hafıza alanı limitini ortadan kaldırır.

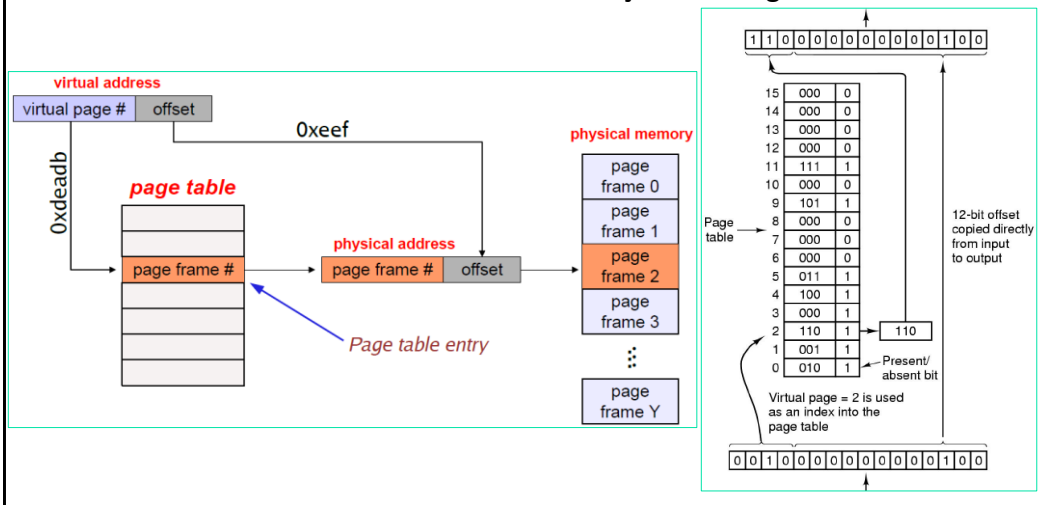
Giriş

- Sanal bellek, **programcıya fiziksel hafızadan çok büyük bir alan sağlar.**



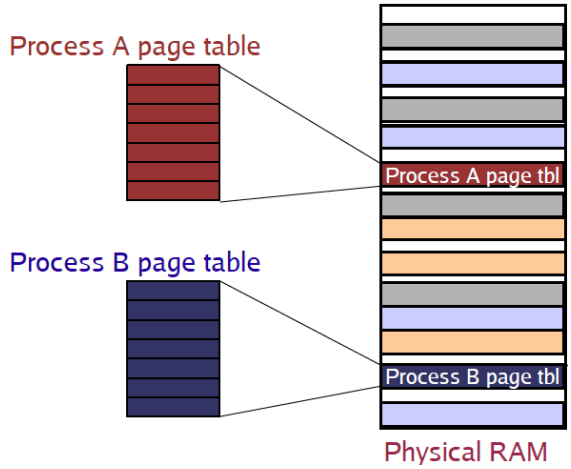
Giriş

- Sanal bellek adresinin fizikel adrese dönüştürülmesi gereklidir.



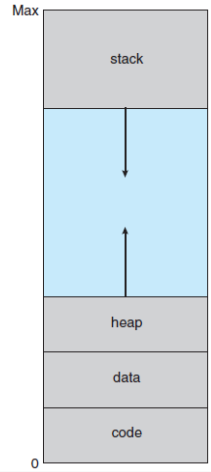
Giriş

- Process'ler için page table fiziksel hafızada saklanır**.**



Giriş

- Bir pocess'in sanal adres alanı, hafızaya nasıl yüklendiğini gösteren mantıksal bir görünümüdür.
- Bir process'in **mantıksal adresi 0 ile başlar ve bitişik bir alandır.**

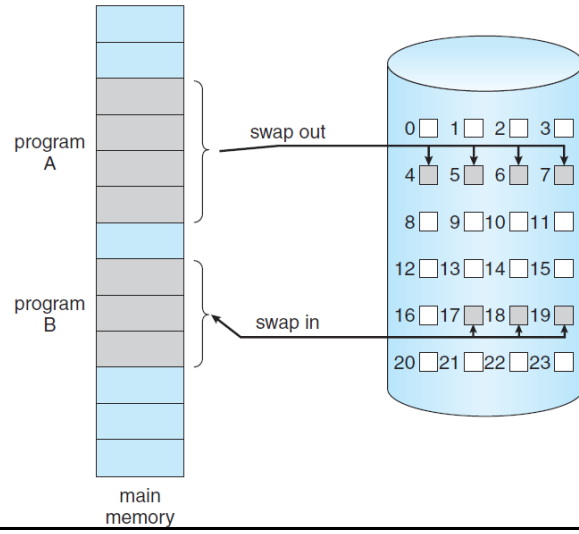


Demand paging

- **Programlara ait sayfaların ihtiyaç olduğunda yüklenmesine demand paging denilmektedir.**
- Demand paging yöntemi **sanal bellek sistemlerinde yaygın kullanılmaktadır.**
- Demand paging ile programın çalışması süresince **kullanılmayan sayfalar fiziksel hafızaya yüklenmez.**
- Demand paging sistemi **disk üzerindeki process'lerin hafızaya swapping ile yüklenmesini gerçekleştirir.**
- **Bir process içindeki bir sayfa gerekmedikçe hafızaya yüklenmez (lazy swapper).**
- **Pager process içindeki sayfaların yüklenmesini gerçekleştirir.**

Demand paging

- Hafızadaki **sayfaların bitişik disk aralığına** aktarımı şekilde görülmektedir.

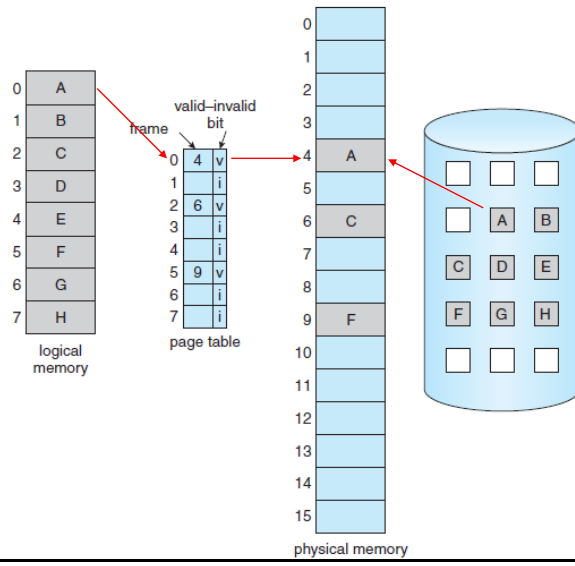


Demand paging

- Bir process swap in yapıldığında, **pager, swap out oluncaya kadar hangi sayfaların kullanılacağını tahmin eder.**
- Process'in tamamını yüklemek yerine, **gerekli sayfalar hafızaya yüklenir.**
- **Böylelikle gerekli hafıza alanı ve swap süresi azaltılmış olur.**
- Bir sayfanın hafızada mı yoksa diskte mi olduğunu tutmak için **donanımsal bileşen gerekir.**
- **valid** ve **invalid** şeklinde **bir bit** sayfanın bulunduğu yeri (hafızada olup olmadığı) **belirlemek için kullanılabilir.**

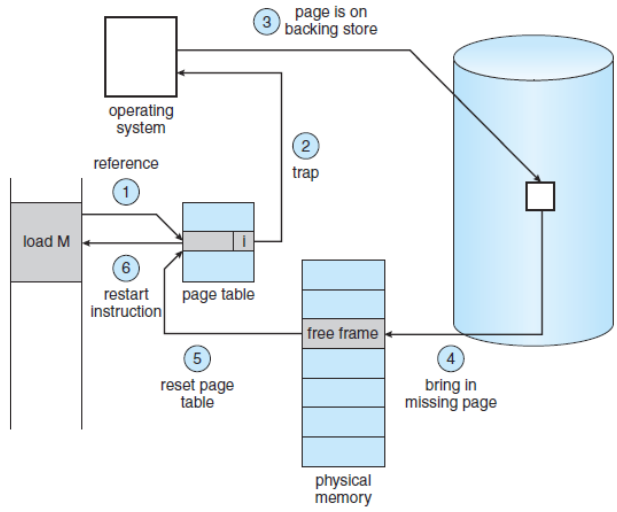
Demand paging

- **Invalid olan sayfaya process hiç erişmezse, invalid olmasının etkisi olmaz.**



Demand paging

- **Hafızada olmayan (invalid)** bir sayfaya erişime page fault (sayfa hatası) denir. İşletim sistemi sayfayı hafızaya aktarır.



Demand paging

- Bir sayfa hafızada bulunamadığında aşağıdaki işlemler gerçekleştirilir:
  - İstenen bloğun hafızada olmadığı belirlenir.
  - Çalışmakta olan process kesilir.
  - Hafızada boş bir frame belirlenir.
  - Disk üzerinden istenen sayfa hafızadaki boş frame'e aktarılır.
  - Sayfa tablosu değiştirilerek ilgili sayfanın hafızaya yüklendiği belirtilir.
  - Kesilen instruction ile process çalışmaya devam eder.
- Bir process başladığında hiçbir sayfa hafızada olmayabilir. **Process hemen page fault üretir ve bu sayfa hafızaya alınır.**
- Bir sayfanın ihtiyaç duyulmadığı sürece hafızaya alınmamasına pure demand paging denir.
- **Bazı programlar bir instruction ile çok sayıda sayfaya erişebilirler (bir instruction çok data) ve** çok sayıda page fault oluşur**.**

Page replacement

- **Her sayfa ilk çağrıldığında bir kez page fault oluşur.**
- Bir process 10 sayfadan oluşuyorsa, **çalışması sırasında genellikle yarısını kullanır.**
- Bu yüzden, demand paging I/O gereksinimini azaltır.
- **Multiprogramming ile daha çok process'i çalıştırabiliriz (over-allocating).**
- Process'lerden bazıları tüm sayfaları kullanmak isteyebilir.
- İşletim sistemi **yeni sayfa için mevcut sayfalardan birisini swap out yapabilir (page replacement).**

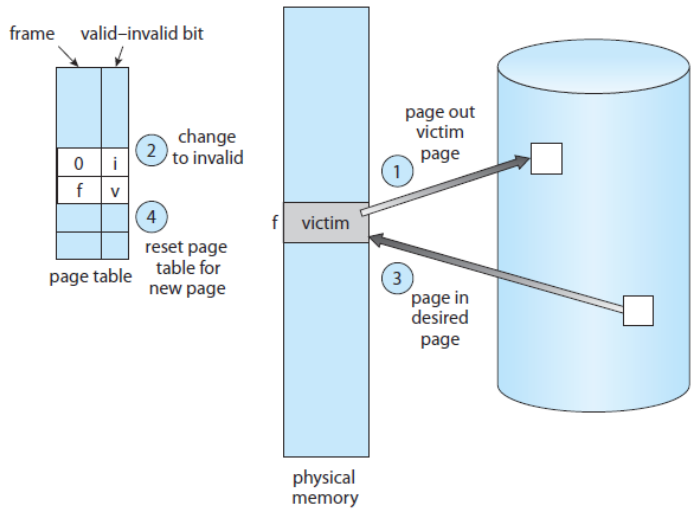
Page replacement

Temel page replacement algoritması

- Eğer boş frame yoksa **şu anda kullanılmayan bir frame seçilir ve swap out (disk swap space'e yazılır) yapılır.**
- Swap out yapılan frame için page table invalid yapılır.
  - **İstenen sayfa disk üzerinde bulunur.**
    - Boş frame varsa, bu frame kullanılır.
    - Boş frame yoksa, page-replacement algoritması ile bir frame seçilerek kullanılır.
    - Seçilen frame disk'e yazılır ve page table değiştirilir.
  - **Disk'ten istenen sayfa okunarak bu frame'e yazılır ve page table değiştirilir.**
  - **Page fault olan process çalışmaya devem eder.**

Page replacement

Temel page replacement algoritması



Page replacement

Temel page replacement algoritması

- Page fault olması halinde boş frame yoksa, **iki kez page fault süresi kadar beklenir (swap out ve swap in).**
- Hafızada kaldığı süre içerisinde **değişmeyen sayfaların hafızaya yazılmasına gerek yoktur.**
- Her sayfa için modify bit (dirty bit) ile değişip değişmediği tutulur.
- Page replacement algoritması ile **seçilen sayfa değişmişse hafızaya yazılır.**
- Bir process için **kaç tane frame'in hafızaya alınacağına** **frame-allocation algorithm** ile karar verilir.
- **Boş frame olmadığında ise hafızadan atılacak frame'e** **page-replacement algorithm** ile karar verilir.

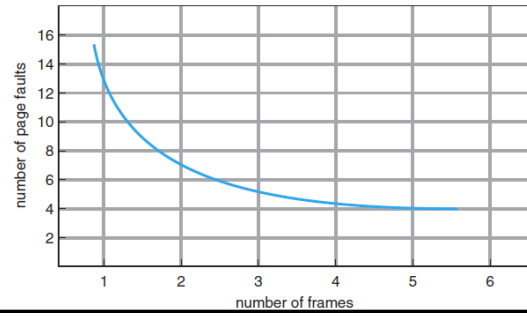
Page replacement

Temel page replacement algoritması

- Her sayfanın 100 byte olduğu durum için aşağıdaki **hafıza referansları** ve **ihtiyaç duyulan sayfalar verilmiştir.**

0100, 0432, 0101, 0612, 0102, 0103, 0104, 0101, 0611, 0102, 0103, 0104, 0101, 0610, 0102, 0103, 0104, 0101, 0609, 0102, 0105	1, 4, 1, 6, 1, 6, 1, 6, 1, 6, 1

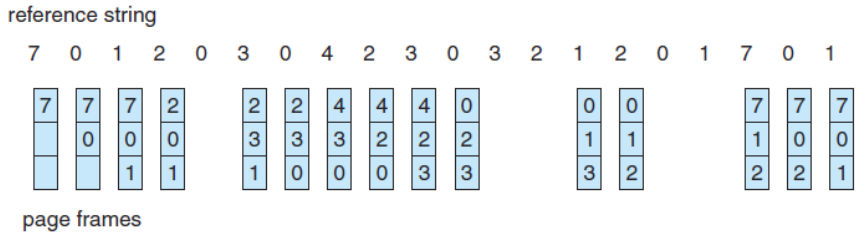
- **Hafızaya yüklenen frame sayısı ile page fault sayısı ters orantılı değişir.**



Page replacement

FIFO page replacement

- **FIFO algoritmasında ilk gelen frame atılarak yeni gelen buraya yazılır.**
- Şekilde process için 3 sayfa ayrılmıştır.
- Yeni gelen frame'ler en eski olan atılarak yerine yazılmaktadır.



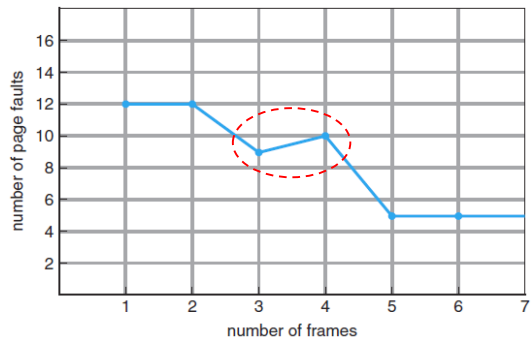
- **Şekilde 15 page fault olmaktadır.**
- Seçilen sayfa aktif kullanılıyorsa swap out yapılması uygun değildir (o sayfa için hemen tekrar page fault oluşur).

Page replacement

FIFO page replacement

- FIFO algoritmasında bazı durumlarda **daha fazla frame ayrılması halinde daha fazla page fault olabilmektedir** (Belady's anomaly).
- Aşağıdaki hafıza erişim serisi için page fault sayıları grafikte verilmiştir.

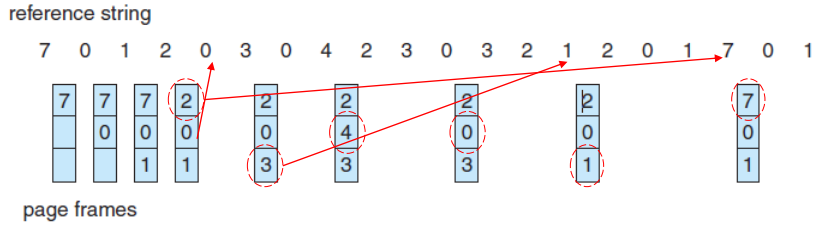
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



Page replacement

Optimal page replacement

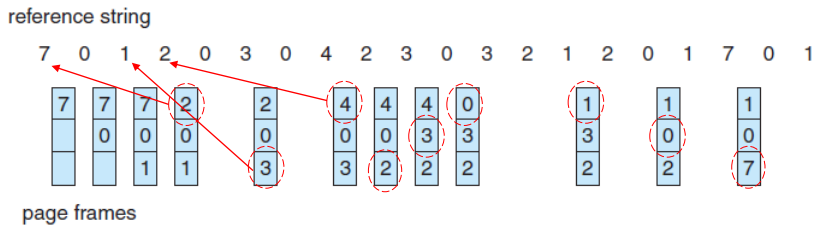
- Optimal page replacement algoritmasında, **en uzun süre kullanılmayacak sayfa ile** yer değiştirme yapılır.
- Aşağıdaki hafıza erişim serisinde **9 page fault ile yer değiştirme yapılmaktadır.**



Page replacement

LRU page replacement

- Least recently used (LRU) algoritmasında, **en uzun süre kullanılmamış** olan sayfa atılır.
- Kullanılmama süresini tutmak için **counter kullanılabilir.**
- Kullanılan sayfa stack kullanılarak her defasında top eleman yapılır. **Stack'in en altındaki sayfa uzun süre kullanılmayan sayfadır.**



Page replacement

LRU page replacement

- Least recently used (LRU) algoritması için **stack kullanılabilir.**