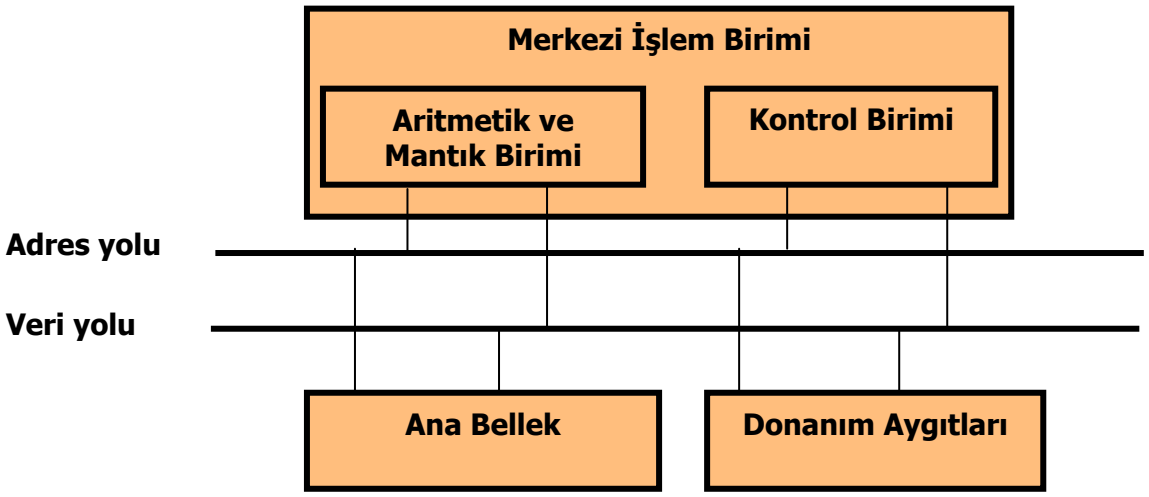
İŞLETİM SİSTEMLERİNE GİRİŞ

Von Neumann Mimarisi

Modern bilgisayar çalışma prensipleri, Von Neumann'ın 1945'de geliştirdiği mimariyi temel almaktadır.

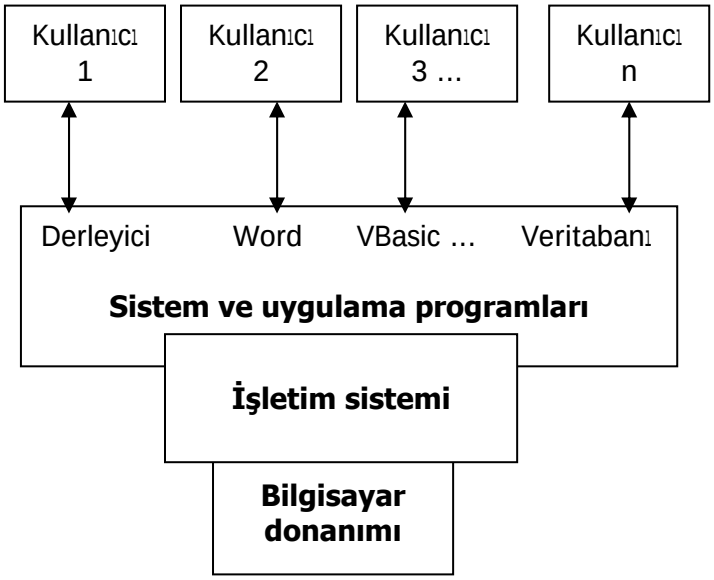


Şekil 1. Von Neumann Mimarisi

İşletim sistemi (Operating System) tanımı:

Bir işletim sistemini orkestrayı yöneten bir maestro gibi düşünebilirsiniz. Bir orkestrayı yöneten biri olmadığında, enstrümanlardan çıkan sesler birbiriyle uyumlu olmayacaktır. İşte işletim sistemi de bir bilgisayar sisteminin maestrosudur; bilgisayarın donanım elemanlarının birbiri ile haberleşmesini, birbirini tanımasını, kısacası birbiri ile uyumlu bir şekilde çalışmasını sağlar.

İşletim sistemi aynı zamanda, kullanıcı ve donanım, yazılım ve donanım ve son olarak yazılım ve yazılım arasındaki ara yazılımdır. Kullanıcı ve donanımın, donanım ve yazılımların ve birçok farklı yazılımın etkileşimini, birbirini anlamasını ve birbiri ile çalışabilmesini sağlar.



Şekil 2. Bir bilgisayar sisteminin genel görünümü

İşletim sistemi; bilgisayar sistemini oluşturan donanım ve yazılım nitelikli kaynakları kullanıcılar arasında kolay, hızlı ve güvenli bir işletim hizmetine olanak verecek biçimde paylaştırırken bu kaynakların kullanım verimliliğini en üst düzeyde tutmayı amaçlayan bir yazılım sistemidir.

Bir bilgisayar sistemindeki MİB, bellek, soyut bellek, G/Ç aygıtları ve dosyalar gibi kaynakları kontrol eden program modülleri topluluğudur. Bu modüller sistemin daha etkili ve performanslı kullanılmasını sağlar. Bunlar kullanıcı ve donanım arasında bir arabirim gibi çalışırlar.

Tarihsel Gelişim:

1. Eski Sistemler (Mainframe)
    a) 1950'lerdeki sistemlerdir. Tek kullanıcılıdırlar. İşletim sistemi kullanılmamıştır. Örn: Eniac da işletim sistemi yoktu
    b) Sadece tek bir iş: Aynı anda sadece bir iş yapılmaktaydı. Kısacası doğrusal çalışıyorlardı.

c) Bilgi girişi ve çıkışı için kartlar kullanılmaktaydı.

2. Toplu İşlem Sistemleri (Batch Systems):
    a) Monitor programları geliştirilerek otomatik olarak birden fazla işin arka arkaya yapılması sağlandı. Bir işe örnek olarak "şu programı çalıştır" veya "şu programı derle" komutları verilebilir.
    b) Kart okuyucular kullanılmaktaydı.

3. Zaman paylaşımlı sistemler (Timesharing Systems)
    a) Diskteki veya bellekteki farklı işleri aynı anda yapabilmektedir.
    b) Bu işleri yaparken bir donanımı belirli zamanlarda farklı işleri yapılması için tahsis ederdi. Örneği MİB farklı işleri aynı anda yapmıyor ama bu işlerin belli görevlerini yerine getiriyordu. Örneğin aynı anda 3 iş yapması gerekiyorsa öncelikle 1. işin ilk görevini sonra 2. işin ilk görevini ve 3. işin ilk görevini yapıyordu ve bu şekilde devam ediyordu. Kullanıcı bu işleri aynı anda yaptığını zannetse de aslında bu işlerin belli görevleri arasında değişim yapıyordu. İşte bu özelliğe 'Çok görevli (Multitasking)' denilmektedir. "Çoklu programlama (multiprogramming)" sistemleri ise aynı anda birden fazla programın çalıştırılması anlamına gelmektedir. Bu sistemlerde MİB zamanlaması (scheduling), bellek yönetimi gibi işlemler sistem tarafından yapılmaktaydı.
    c) Kart okuyucu olmadan kullanıcı ile iletişimde bulunabilmektedir.

4. Kişisel Bilgisayar sistemleri (Personal Computer Systems)
    a) Donanım ebatlarının küçülmesi ile masaüstünde yer alan bilgisayarlardır.
    b) G/Ç aygıtları kullanılmaktadır. Farklı teknolojiler adapte edilebilmektedir.
    c) İşletim sistem

5. Paralel veya Çok İşlemcili Sistemler (Paralell Systems):
    a) Birden fazla MİB'nin kullanılabildiği sistemlerdir. MİB'leri, belleği ve diğer donanım kaynaklarını paylaşırlar.
    b) Simetrik çoklu işleme: Her işlemci işletim sisteminin ayrı bir kopyası üzerinde çalışır. İhtiyaç olduğunda birbirleri ile haberleşirler.
    c) Asimetrik çoklu işleme: Her işlemci ayrı bir iş için tahsil edilmiştir. Master olan bir işlemci vardır ve görev ve iş dağılımlarını yapar, slave için işleri sıraya sokar ve zamanlar.

6. Dağıtık Sistemler (Distributed Systems):
    a) Bu sistemler internet ve ağ teknolojilerinin gelişmesi ile ortaya çıkmıştır. Bir ağ üzerinde kuruludur. Ancak bir ağdan farklı olarak ağ üzerinde gerçekleşen işlemler kullanıcıya görünmektedir. Kullanıcı, bir işlemci ve bir arayüz olduğunu düşünse de bu sistemler, değişik bilgisayar sistemleri üzerindeki verileri ve işlemleri bir bütün olarak işleyebilir ve çalıştırabilirler.
    b) Her işlemcinin kendine ait belleği vardır ve diğer işlemcilerle iletişim hatları üzerinden haberleşirler.
    c) Kaynak paylaşımı sağlanır. Bu kaynak MİB, diğer donanım elemanları olabileceği gibi veriler, dosyalar da olabilir.
    d) Dağıtık sistemlere; bilgisayar ağları, Sunucu-İstemci Sistemleri, Peer-to-Peer (Noktadan noktaya) ve Internet örnek olarak verilebilir.

7. Gerçek-Zamanlı Sistemler (Real-Time Systems):
    a) Çok kısa sürelerde örneğin milisaniyelerde çalışan sistemlerdir. Belli bir sistemi kontrol amacıyla kullanılır. Örneğin bir işlemciye çok yoğun bir veri akışı olduğunda veya bir uygulamanın kontrolü için ayrılan bir aygıt için kullanılır.

b) Bilimsel deneyleri kontrol eden, gösterim sistemleri, endüstriyel kontrol sistemleri vb. gerçek zamanlı sistemlerdir.

8. Gömülü Sistemler (Embedded Systems):
   a) Gerçek-zamanlı işletim sistemlerini çalıştıran sistemlerdir. Bunlarda genellikle bir kullanıcı arayüzü olmaz, sadece donanımları izleme ve yönetme işlemlerini yaparlar.
   b) Örneğin router'larda bulunan ayrı işletim sistemleri, firewall'lar, otomobil motorları, nükleer reaktörün soğutma işlemleri.

Modern bir işletim sistemleri yukarıdaki farklı işletim sistemlerinin özelliklerini kapsamaktadır.

İşletim Sisteminin Yerine Getirmesi Gereken Fonksiyonlar

Bir işletim sistemi şu fonksiyonları yerine getirebilmelidir:

1. İşleri sıraya koymalıdır
2. İşi kontrol eden dili yorumlayabilmelidir
3. Hata durumlarında ilgili işlemleri sonuçlandırmalıdır
4. Giriş/Çıkış işlemlerini sonuçlandırmalıdır
5. Kesmelerin gereğini yerine getirmelidir
6. İşlerde öncelik tanıyabilmelidir
7. Kaynakları kontrol etmelidir
8. Kullanıcıların birbirlerinin haklarına müdahalelerini önlemelidir
9. Bilgisayara birden fazla erişim sağlamalıdır
10. İyi bir ara yüzü olmalıdır
11. Bilgisayar kaynaklarının hesabını tutmalıdır.
12. Bilgileri uzun vadede saklamalıdır.

Bir İşletim Sisteminde İstenilen Özellikler

1. Etkinlik: İşletim sistemi, bir işi etkin ve verimli bir şekilde yapmalıdır.

2. İşler arasındaki zaman: Bir işi bitirip diğer işi başlama süresi kısa olmalıdır.

3. Kullanılmayan MİB zamanı: Merkezi işlem birimin kullanmadığı süre kısa olmalıdır. Yani mikroişlemciden belleğe aktarılacak veya bellekten alınacak bilginin erişim süresi kısa olmalıdır.

4. Toplu işlemler arasındaki zaman: Toplu işlem dosyalarının işlenmesi arasındaki süre kısa olmalıdır.

5. Cevap verme süresi: Sistemin cevap verme süresi kısa olmalıdır.

6. Az zamanda çok iş yapılmalıdır.

7. Güvenirlik: Sistem tamamen hatalardan arındırılmış olmalıdır.

8. Süreklilik: Sistem bakım yapılabilir ve dokümanı bol olmalıdır.

9. Düşük boyut: Sistem kendinden taviz vermeden küçük boyutta olmalıdır.

İŞLETİM SİSTEMLERİNE GİRİŞ - 2

Kaynakların Paylaşımı (Resource Sharing)

Sistem, sistem kaynaklarını belli bir hiyerarşi içinde kullanıcının hizmetine sunar. Bir işletim sisteminde paylaşılan kaynaklar ise şunlardır: Merkezi işlem birimi, Bellek, Giriş/Çıkış birimleri gibi donanım elemanları, Dosyalar ve klasörler, yazılımlar; uygulama programları, çeşitli programlar.

Kaynak paylaşımı yapılmasının nedenleri 4 faktör söylenebilir:

- Maliyet: Her kullanıcıya birbirinden bağımsız kaynak sağlamak zordur.
- Birinin geliştirdiği bir programı veya uygulamayı diğerleri de kullanabilir
- Aynı veri tabanı birden fazla kullanıcı tarafından kullanılabilir
- Bir programın birden fazla kullanıcı tarafından kullanılarak depolama birimlerinden tasarruf sağlanır ve geçersiz kaynak kullanımının önüne geçilebilir

İşletim sisteminin paylaşım işlemini gerçekleştirebilmek için ise şu görevleri yerine getirmesi gerekir:

a. Giriş/Çıkış İşlemleri: Kullanıcıdan bağımsız donanıma bağlı değişen G/Ç işlemlerini yerine getirme.
b. Bellek İşlemleri: Makinedeki fiziksel bellekten farklı olarak kullanılan soyut bir bellek imkanı sunmalıdır.
c. Dosya Sistemi: Soyut makinelerin çoğu program ve verilerin saklanması için bir dosya sistemi içerir. Kullanıcının bu bilgilere ulaşımı, adresler yerine sembolik isimlerle sağlanır.
d. Koruma ve hata kontrolü: Kullanıcıların birbirlerinin alanlarına müdahale etmeleri önlenmeli ve kişisel bilgilerin güvenli bir şekilde saklanması için koruma sağlanmalıdır.

e. Program kontrolü: Soyut bir makine kullanıcıya program ve işlemler üzerinde işlem yapmaya izin verir.
f. Etkileşim: İşletim sistemi kullanıcılar arasında karşılıklı etkileşim olanağı verir.

Kaynakların farklı kullanıcılar veya programlar arasında paylaştırılması farklı teknikler kullanılarak yapılabilir;

- Aynı anda çalışma (Concurrent execution): Bir bilgisayarın birden fazla programı aynı anda çalıştırmasıdır. Sistemde aynı anda farklı aktivitelerin yer almasıdır. Mesela birden fazla kullanıcı G/Ç işlemi yapabilmeli belleği kullanabilmelidir. Bu durumda işlerin senkronize edilmesi gerekir. Çünkü aslında sistemde gerçekleştirilecek işlemler aynı anda yapılmamakta, sıralı olarak yapılmaktadır. Yani birden fazla program mantıksal olarak aynı anda çalışmakta iken fiziksel olarak sıralı olarak çalışmaktadır. Örnek olarak çoklu programlama yaparken MİB'ni programların paylaşması verilebilir.
- Paralel çalışma (Paralel execution): Bir bilgisayarın birden fazla programı gerçekten aynı anda çalıştırmasıdır. Yani birden fazla program hem mantıksal hem de fiziksel olarak aynı anda çalışmaktadır.

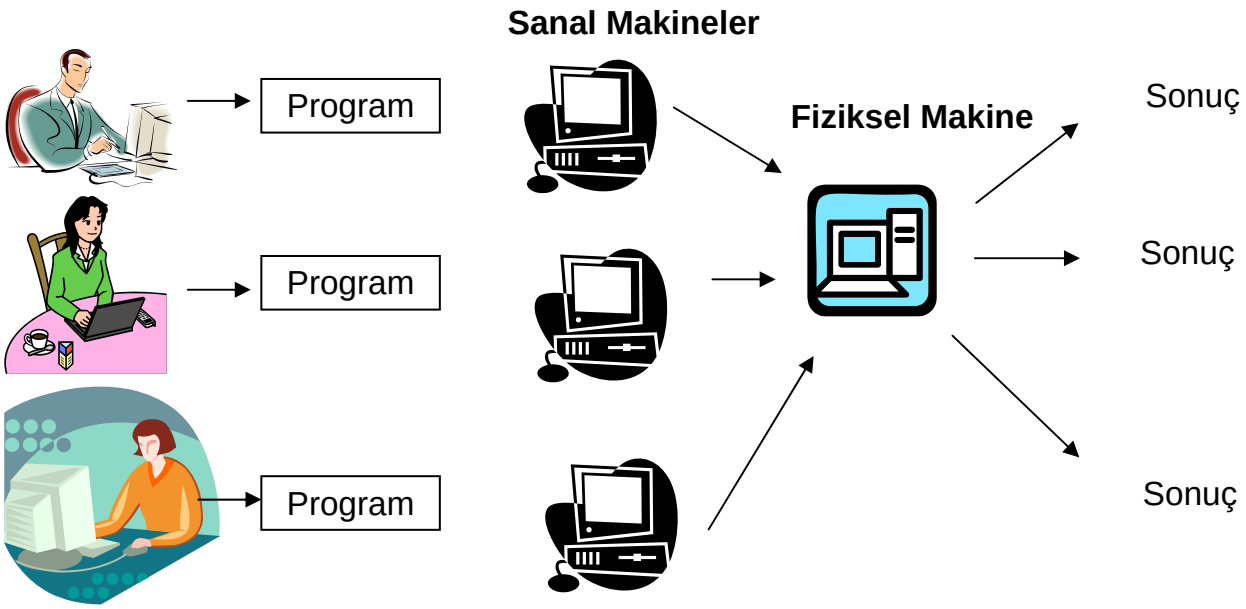
Aynı anda veya paralel çalışma sırasından bir bilgisayar kaynaklarını paylaştırmak zorundadır. Bu paylaşımlar iki türlü olabilir;

- Saydam paylaşım (Transparently sharing): Kaynakların paylaşımının işletim sistemi tarafından yürütmesidir. Burada kullanıcı kaynakların paylaştırıldığından habersizdir. Bu paylaşım türünün gerçekleştirilebilmesi için soyut makineler kullanılmaktadır.

- Açık paylaşım (Explict sharing): İşlemlerin genel kaynakları kendi politikalarına göre kullanmalarıdır. İşletim sistemini kullanıcıya makinedeki kaynakları paylaştırmasına da izin vermektedir.

Soyut/Sanal bir makine (Abstract Machine) ve Saydam Paylaşım

İşletim sisteminin ikinci görevi ise kullanıcıya veya kullanıcılara yalnızca kendine tahsis edilmiş bir makine varmış gibi çalışma ortamının sağlanmasıdır. Her bir soyut makine fiziksel makinenin bir simülasyonudur. Bir fiziksel makine birden fazla soyut makineyi aynı anda çalıştırabilir. Her program kendi soyut makinesinde çalışır. Bu işlemleri yapabilmek için fiziksel makine donanımlarını saydam paylaşım tekniğini kullanarak soyut makineler arasında paylaştırır. Soyut makine tarafından çalıştırılan programa genellikle 'process (işlem)' denilmektedir.



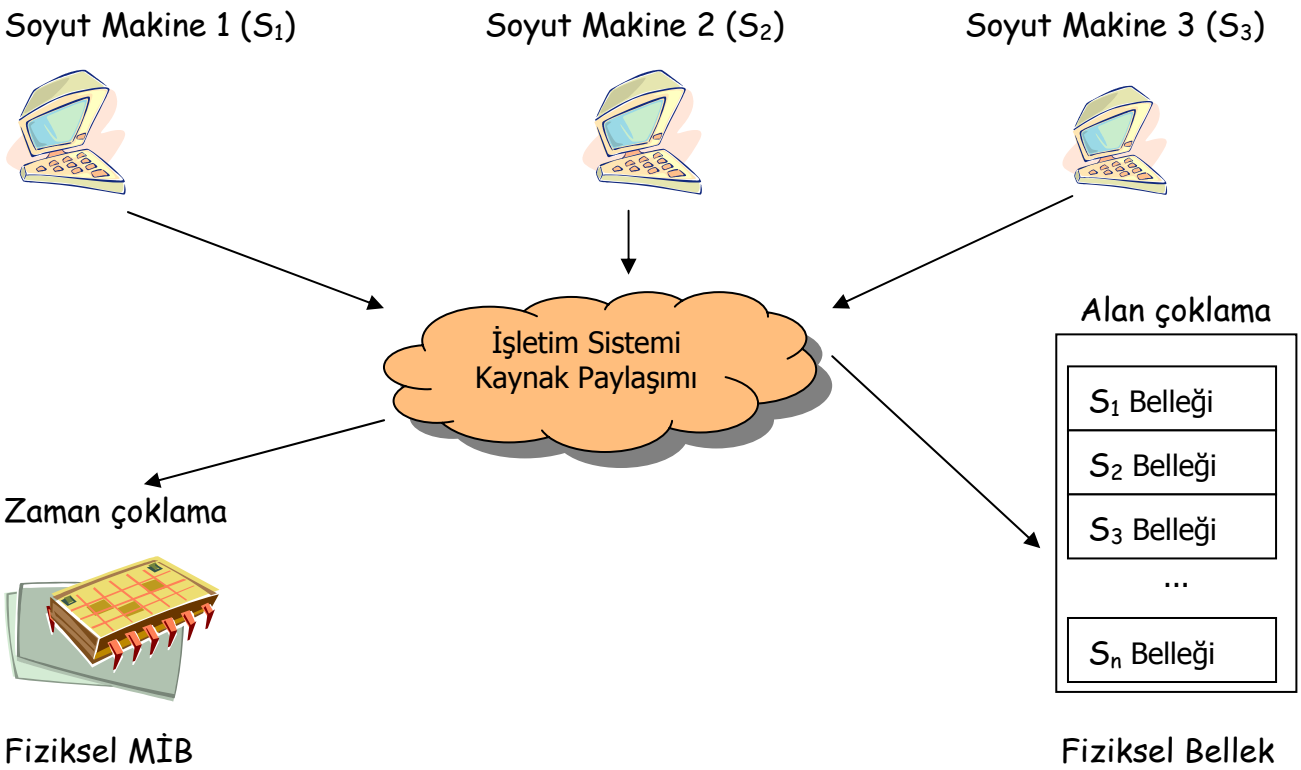
Soyut makinelerin oluşturulması için saydam paylaşım tekniğinin 2 tür paylaşımı kullanılmaktadır;

a. Alan çoklama paylaşımı (Space-multiplexed sharing): Bir kaynak bir veya daha fazla bölüme ayrılır, ardından her bir bölüm bir işlem'e atanır. Bu

paylaşım türüne, bellek ve hard disk'in belli bölümlere ayrılarak her bölümün farklı işlemler için ayrılması örnek olarak verilebilir.

b. Zaman çoklama paylaşımı (Time-multiplexed sharing): Bu paylaşım türünde bir kaynak bölümlere ayrılmaz, bunun yerine kaynak bir işlem tarafından belli bir süre kullanılır yani o işleme adanır ardından diğer işlem bu kaynağı belli bir süre kullanır. Örneğin MİB'ni işlemlerin belirli süreler için kullanılması bu duruma bir örnektir.

İşletim istemi bu iki paylaşım türünü beraber kullanabilir. Örneğin MİB'i için zaman çoklama, hard disk ve bellek için ise alan çoklama paylaşım türlerini kullanabilir. Bu paylaşımlar sayesinde aynı andan birden çok program çalışabilmektedir kısacası çoklu programlama özelliği kullanılabilmektedir. N tane soyut makine olduğunu düşünürsek;



Şekil 2. Çoklu programlama (Multiprogramming)

Çoklu programlama bir işlemcinin dolayısı ile bir bilgisayar sistemin performansını arttırır.

Açık paylaşım (Explict sharing)

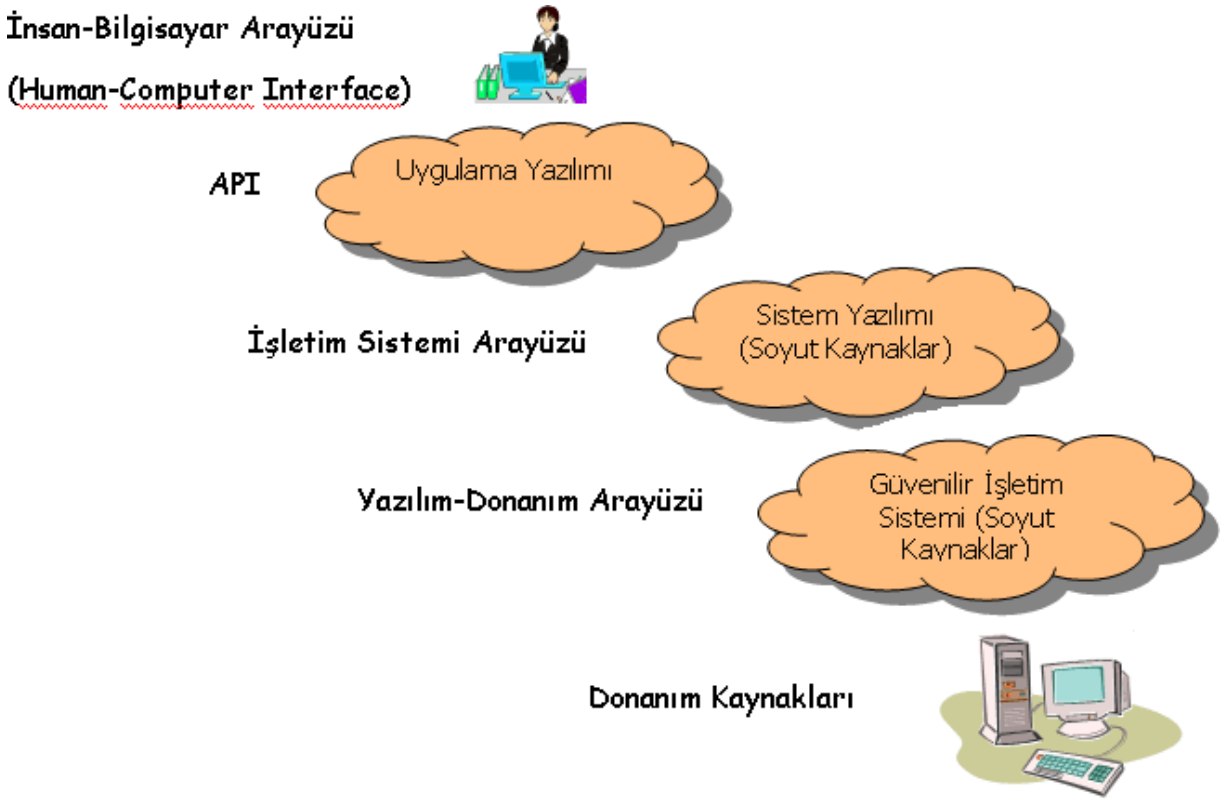
İşlemlerin genel kaynakları kendi politikalarına göre kullanmalarıdır. Bu paylaşımda zaman veya alan çoklama paylaşımı yapılsa da dikkat edilmesi gereken 2 önemli nokta vardır;

- Kaynak yalıtımı (Resource isolation): Sistem yerleşim politikasına göre kaynaklara ulaşımı ayırabilmelidir. İşletim sistemi, bir soyut makine tarafından kullanılan kaynağa diğer yetkisiz işlemlerin ulaşmasını engellemelidir. Örneğin bellek yalıtım mekanizmasında, bellek belli bölümlere ayrılarak her bir bölüm ayrı bir soyut makine tarafından kullanılmaktadır. Bu durumda sistem, bir soyut makineye ayrılmış bir bellek bölümüne başka bir soyut makinenin müdahale etmesini engellemelidir. Aynı şekilde MİB'nin işlemler tarafından sıralı olarak paylaşılmasında da, bir işlem MİB'ni kullanırken diğer işlemlerin bellek alanlarına MİB yoluyla müdahale etmemelidir.
- İşbirliği yaparak paylaşım (Cooperatively sharing): Sistem işlemlerin kaynakları istenildiği takdirde işbirliği yaparak paylaşmalarına izin verebilmelidir. Örneğin bir işlem diğer bir işlemin sonucunu kullanabilmeli veya belirli bir bellek bölgesindeki bilgiler iki işlem tarafından kullanılması gerektiğinde buna izin verebilmelidir. Fakat bu paylaşım son derece dikkatli yapılması gerekmektedir.

Kaynak paylaşımını yapmak ile görevli işletim sistemi yazılımı bu paylaşımı algoritma veya başka bir hata (bug) yüzünden gerçekleştiremediğinde ise hatalar oluşmaktadır. Oysa güvenilir ve gelişimi tamamlanmış bir işletim sisteminde bu tür hataların olması beklenmez. Güvenilir bir işletim sisteminde donanım kaynaklarının doğru bir biçimde paylaştırılması ve birbirinden yalıtılması gerekmektedir. Bir işletim sisteminin soyutlamaları, sistem çağrı arayüzü (system call interface) olarak da bilinen

işletim sisteminin arayüzü kullanılarak gerçekleştirilebilir. Tüm sistem yazılımları bir uygulama programlama arayüzü – API (Application Programming Interface) yoluyla ulaşılabilinir. API; bir yazılım veya sistem yazılım parçasının programlama arayüzüdür. Veritabanları, VBasic editörü gibi. Programcılar uygulama programlama arayüzlerini kullanırlarken, işletim sistemi, sistem çağrı arayüzünü kullanır. Uygulama programlama arayüzleri sistem yazılım arayüzlerine ulaşarak bunların kullanıcı tarafından kullanılabilmesini sağlar. Microsoft Windows sistem çağrı arayüzünü "Win32 API" olarak adlandırmıştır.

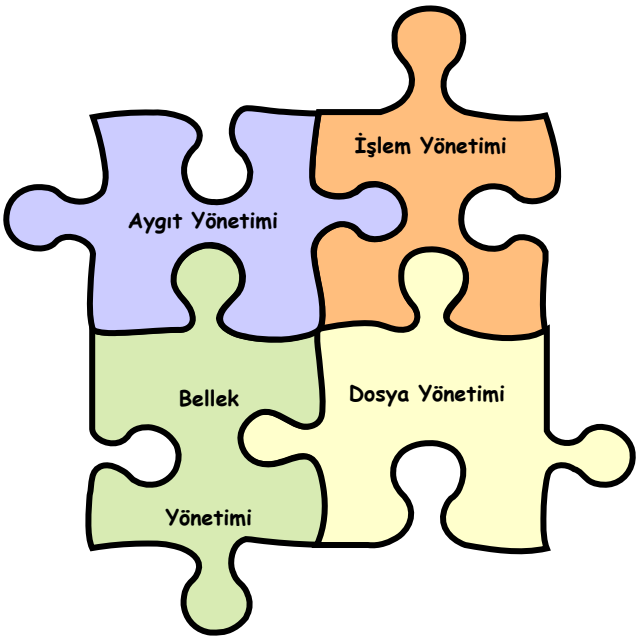
Uygulama yazılımı, sistem yazılımı ve işletim sistemi arasında bir hiyerarşi vardır. İşletim sistemi, yazılım-donanım arayüzünü kullanarak işletim sistemi arayüzüne, sistem yazılımı, işletim sistemi arayüzünü kullanarak API'ye ve uygulama yazılımı da API'yi kullanarak insan-bilgisayar arayüzünde gerçekleşecek olan yazılımı oluşturur.



Şekil 3. Uygulama yazılımı, Sistem yazılımı ve İşletim Sistemi

Bir İşletim Sisteminin Fonksiyonları Açısından Mantıksal Yapısı

Bir işletim sisteminin, temel olarak 4 bileşeni bulunmaktadır;

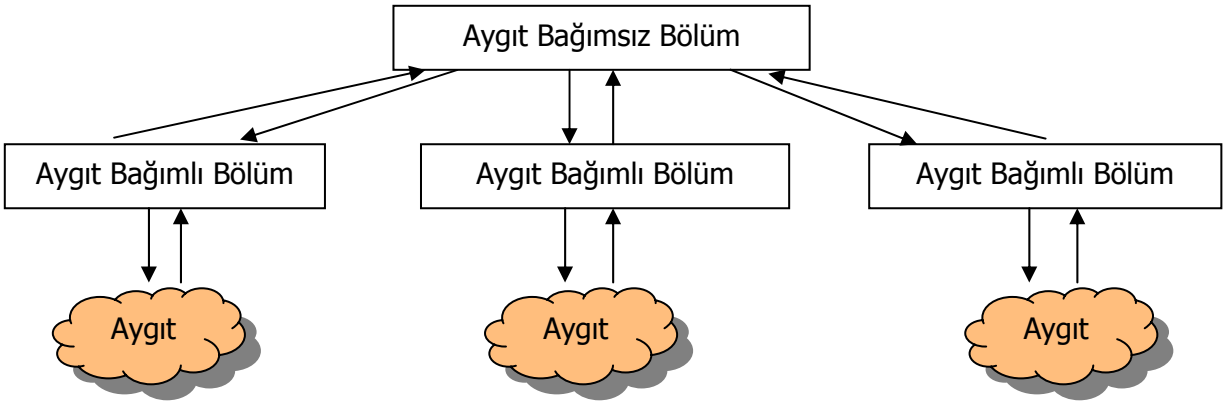


- Aygıt yönetimi (Device management)
- İşlem yönetimi (Process management)
- Bellek yönetimi (Memory management)
- Dosya yönetimi (File management)

Aygıt yönetimi (Device management)

Bir işletim sistemi donanım aygıtlarının yönetiminden sorumludur. Birçok işletim sistemi yazıcı, disk gibi donanım aygıtlarına genelde aynı şekilde yönetirken işlemci ve belleğin yönetiminde farklı yaklaşımlar kullanmaktadır.

Aygıt yönetiminin aygıt bağımlı ve bağımsız olmak üzere ki bölümü bulunmaktadır. Bağımlı olan bölüm'e aygıt sürücüsü (device driver) de denilmektedir. İşletim sisteminin her bir aygıt için kullandığı ayrı bir sürücü vardır. Aygıt yönetiminin bağımsız olan bölümü ise aygıt bağımlı bölümün yürüteceği yazılım ortamını temsil etmektedir. Örneğin aygıt bağımsız alan, sistem çağrı arayüzündeki çağrıları aygıt sürücüsüne iletmektedir. Aygıt bağımsız alan genellikle aygıt yönetiminin küçük bir bölümüdür, büyük bölümü sürücülere ayrılmıştır.



Şekil 4. Aygıt yönetimi

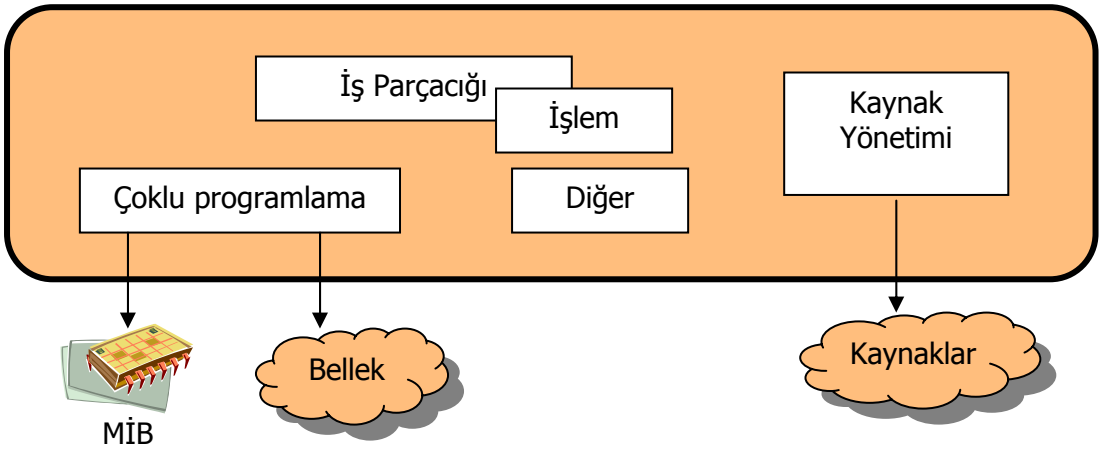
Aygıt yönetiminin bu şekilde ikiye ayrılması ile bilgisayara yeni bir donanım eklemek çok kolay hale gelmiştir. Öncelikle işletim sistemi tasarımcısı, aygıtın hangi bölümünün bağımlı ve hangisinin bağımsız olacağını belirler. Bağımsız olan bölüm temel işletim sistemi içerisinde, bağımlı olan bölüm ise aygıt sürücüsü içerisinde uygulamaya geçirilir. Bu; aygıt yönetimin bağımsız bölümünün; bir aygıta okuma ve/veya yazma işlemlerini yürüten sistem çağrılarını içerdiği anlamına gelmektedir. Örneğin yazıcı sürücüsü bir yazıcı ile ilgili tüm yazılımları içermektedir. Bu yazıcı bilgisayara bağlanarak sürücüsü yüklendiğinde aygıt yönetiminin bağımsız olan bölümü işletim sisteminin içerisinde hali hazırda bulunduğu için bu yazıcı kullanıcı tarafından hemen kullanılabilir.

İşlem Yönetimi (Process Management)

Birçok işletim sistemi işlem ve iş parçacığı (Thread) ve kaynak yönetimini birlikte ele almaktadır. İş parçacığı, bilgisayarda en düşük kaynağa ihtiyaç duyan bir program parçasıdır. Genellikle bir işlem ile birlikte kullanılır. İlgili işleme ayrılmış disk alanı, dosyalar, bellek gibi kaynaklarını kullanarak çalışır. Çoklu kullanım

(multithreading) ise bir işlemin birden fazla iş parçacığına bölünerek aynı anda çalıştırılmasıdır. Bir işlem birden fazla iş parçacığından oluşmaktadır.

Bir işletim sisteminin işlem yönetimi; birden fazla işlem ve iş parçasına aynı makinenin kaynaklarını paylaşmaları, işlemlerin eşzamanlı olarak çalışabilmeleri için zamanlamanın sağlanması gibi görevleri vardır. İşlem yönetimi; işlemlerin kaynaklara ulaşması sırasında nasıl bir kaynak yalıtımı yapacağı, bir kaynağı paylaşması gereken birden fazla işlem olduğunda hangi politikaları kullanarak bu kaynağı paylaştıracağı gibi soruları cevaplamaya çalışmaktadır. Bunları yaparken de bellek yönetimi ile birlikte çalışarak belleğin bu işlemler, iş parçacıkları arasında paylaştırılmasını sağlar.

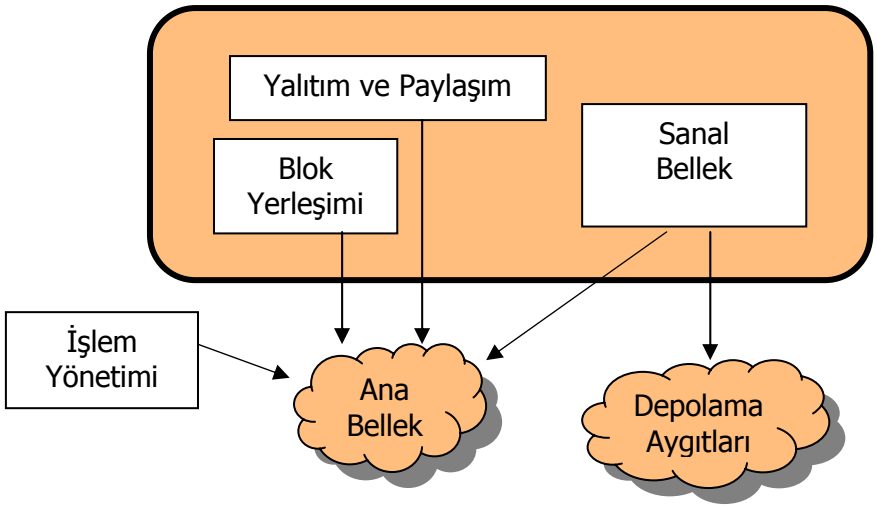


Şekil 5. İşlem yönetimi

Bellek Yönetimi (Memory Management)

Bellek yönetimi işlem yönetimi ile birlikte çalışarak ana bellekte işlemlerin yerleşimini sağlamaktadır. Her işlem bir bellek bölgesi istemekte ve bellek yönetimi de bu işlemlerin çalışması için kaynak yalıtımını da sağlayarak gerekli bellek bölümünü ayırmaktadır. Böylece bellek yönetimi bellekteki blokların paylaştırılması için gerekli

stratejileri uygulamaktadır. Modern bellek yönetimleri sanal bellek (virtual memory) sağlayarak fiziksel bellekten çok daha büyük bir bellek alanının kullanılmasını sağlamaktadır. Eğer işletim sistemi sanal belleği destekliyorsa bellek yönetiminin bir kısmı aygıt ve dosya yönetimleri ile birlikte çalışarak belleği yönetir.



Şekil 6. Bellek yönetimi

Dosya Yönetimi (File Management)

Dosya yönetimi, bellek ve aygıt yönetimi ile birlikte çalışarak dosyaların hard disk ve CD-ROM gibi depolama birimlerine yazılmasını sağlamaktadır. İşletim sistemi bu yönetimi yapabilmek için dosya sistemlerini kullanmaktadır. Bu dosya sistemlerine örnek olarak FAT, FAT32, NTFS, EXT2 verilebilir.

İŞLETİM SİSTEMİ KATMANLARI (Çekirdek, kabuk ve diğer temel kavramlar)

Bir işletim sisteminin yazılım tasarımında ele alınması gereken iki önemli konu bulunmaktadır;

1. Performans: İşletim sistemi, makine kaynaklarını (özellikle MİB zamanı ve bellek alanı) en etkili şekilde kullanılmasını sağlayacak şekilde tasarlanmalıdır. Makinenin donanımsal performansını en iyi şekilde kullanabilmelidir.
2. Kaynakların özel kullanımı: İşletim sistemi, kaynakların yalıtımını sağlamalıdır, diğer bir deyişle bir işlemin diğer işleme ait kaynaklara olan müdahalesine veya bu işleme ait bilgilerin silinmesine izin vermeyen bir koruma mekanizması geliştirmelidir. Her işletim sisteminin kaynaklara ulaşımı yönetmede kullandığı stratejiyi belirleyen bir güvenlik politikası vardır.

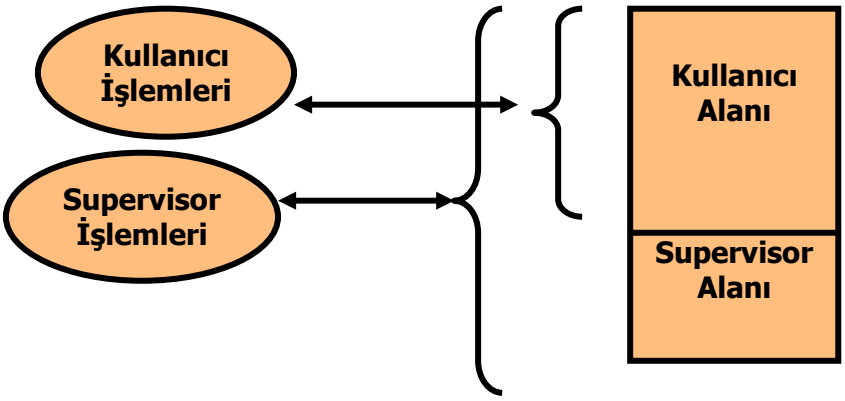
Her işletim sisteminin tasarımında olan üç temel unsur ise şunlardır;

1. İşlemci modları
2. Çekirdek (Kernel)
3. Sistem servislerini uyarma metodu

İşlemci Modları

İşlemcide, bir programın çalışma yeteneğini gösteren bir mod biti bulunmaktadır. Bu bit 'supervisor (kernel)' veya 'kullanıcı' modunu belirlemede kullanılır. İşlemci supervisor modda iken donanımsal her tür komutu çalıştırırken kullanıcı modunda ise bazı komutları çalıştırabilir. Supervisor moda çalışan komutlara supervisor, öncelikli veya korunmuş komutlar denilmektedir. İşletim sistemi programları supervisor moda çalışırken diğer tüm yazılımlar kullanıcı modunda çalışmaktadır. Örneğin giriş/çıkış işlemleri supervisor moda çalışmakta, kullanıcı modunda yer alan bir program herhangi bir giriş/çıkış işlemi yapılmasını istediğinde bunu işletim sisteminin yapmasını istemektedir.

Sistem aynı zamanda mod bitini kullanarak bellek alanları tanımlar. Eğer mod biti supervisor modda olacak şekilde ayarlandığında işlemcide çalışan işlem hem belleğin supervisor hem de kullanıcı alanlarına ulaşabilir. İşlemci kullanıcı modunda çalışırken ise bu işlem bellekte sadece kullanıcı alanına ulaşabilir. İşletim sistemlerinde bu alanlara kullanıcı ve sistem (supervisor, çekirdek veya korunmuş) alanı denilmektedir.

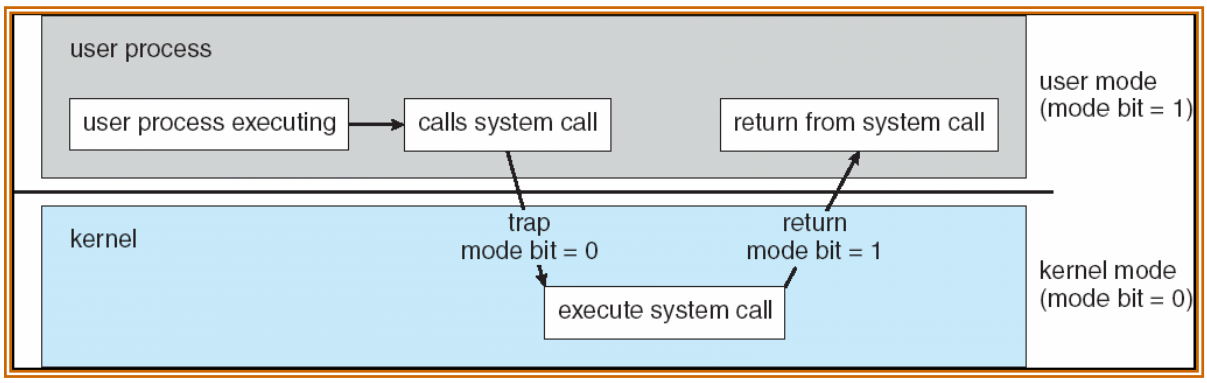


Şekil 1. Supervisor ve Kullanıcı Belleği

Genel olarak; mod biti işletim sisteminin koruma haklarından biridir. İşletim sistemi supervisor modda çalışmakta ve kullanıcı moduna göre belleğe ve öncelikli komut kümesine ulaşmakta daha fazla haklara sahip olmaktadır.

İşlemci supervisor moda geçtiğinde işletim sisteminin kodlarını çalıştırmaktadır. Kullanıcı modundaki bir işlem işletim sistemini çağırdığında işlemci hemen supervisor moda, mod bitini kullanarak geçer ki bu duruma supervisor çağrı (veya sistem çağrısı) denilmektedir. Örneğin; Word'de büyük bellek gerektiren bir dosya açınca başka işlemlerin alanlarına müdahale edilir. Bunu önlemek amacıyla yeni bir alan bu dosya için eklenmelidir. Burada supervisor çağrı yapılmıştır.

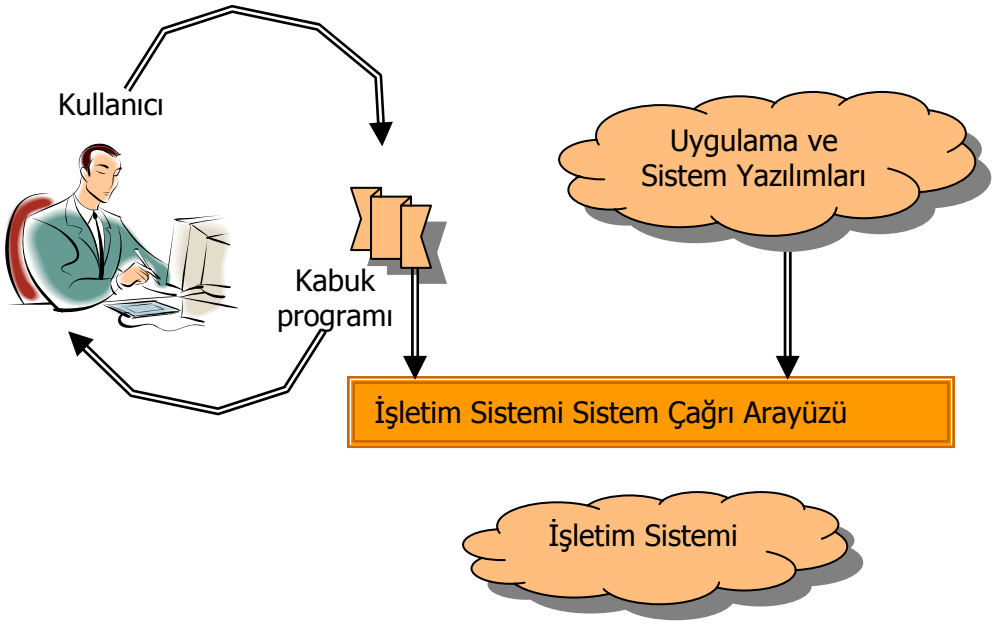
8086/8088 gibi eski işlemcilerde mod biti bulunmadığı için supervisor ve kullanıcı komutları arasında bir ayrım yapılmamaktadır. Bu da kaynakların paylaşımını ve yalıtımını güçleştirmekteydi.



Şekil 2. İşletim sisteminin 2 mod arası geçişi

Çekirdek

İşletim sisteminin supervisor modda çalışan ve diğer parçaları için temel servisleri sağlayan en önemli parçasıdır. İşletim sisteminin uzantıları kullanıcı modunda çalışır ve daha sınırlı haklara sahip olur. Çekirdekte çalışan işletim sistemi fonksiyonları ise belleğe ve çekirdeğin diğer bölümlerine ulaşmada daha fazla haklara sahiptir. Kabuk (shell) veya diğer adıyla komut yorumlayıcısı ise kullanıcının sisteme verdiği komutları anlayan ve çalıştıran bir programdır. Kabuğun genellikle bir arayüzü bulunmaktadır; örneğin DOS'taki C:> nin göründüğü komut istemi arayüzü ve kullanıcının girdiği DIR komutu. Çekirdek ve kabuk bazı işletim sistemlerinde ayrı iken bazılarında da sadece kavramsal olarak ayrılmıştır.



Şekil 3. Kabuk (shell)

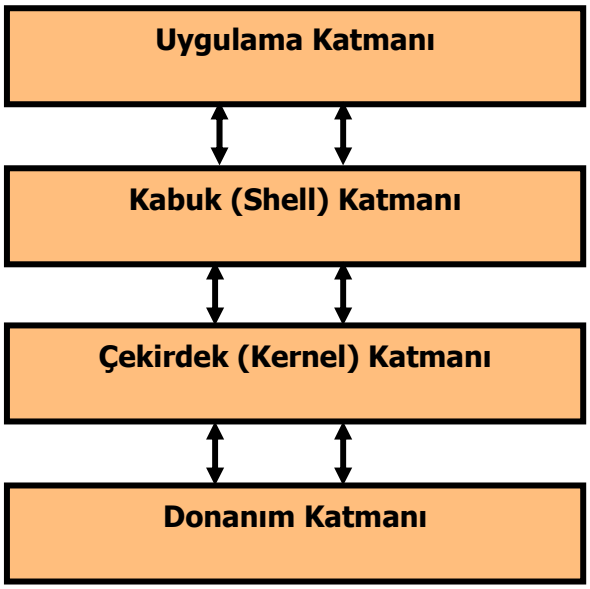
Monolitik çekirdekler (monolithic kernel), 1970–1990 arasında kullanılan ilk çekirdeklerdir. Burada tüm yazılımlar, sürücüler işletim sisteminin çekirdeğinde yer almaktadır. Örnek olarak Unix verilebilmektedir. Çekirdek büyük olmasına karşın her tür fonksiyonu içerdiği için genelde hızlıdır. Monolitik çekirdeklerin boyutlarının çok büyük olduğu düşüncesi modüler yapıda olan mikro çekirdekleri (microkernel) yaratmıştır. Bu çekirdeklerde sadece en önemli işletim sistemi fonksiyonları bulunduğu için oldukça küçük boyutta olmaktadır. Yeni bir donanım eklendiğinde onun sürücüsü de çekirdeğe tanıtılmaktadır. Bu çekirdeklere örnek olarak MS-DOS verilebilir.

Sistem servislerini uyarma metodu

Kullanıcı işlemlerinin işletim sisteminden belli servisleri (program çalıştırma, giriş/çıkış ve dosya işlemleri, ağ erişimi gibi) sağlaması istendiğinde oluşan bir durumdur. Bu bir sistem fonksiyonunun çağrılması veya MİB'ne bir mesaj gönderilmesi (message passing) ile gerçekleşmektedir. Sistem çağrıları, işletim sistemi ve işlemler arasında bir arayüzdür. Bu çağrılar genellikle Assembly dili komutları şeklindedir. C ve C++ gibi bazı programlama dilleri bunu direkt olarak yapabilmektedir. Microsoft Windows ise bunu Win32 API ile gerçekleştirmektedir.

Temel İşletim Sistemi Katmanları

Bir işletim sisteminde yer alan katmanlar şunlardır; Donanım, çekirdek, kabuk ve uygulama katmanı.



Şekil 4. İşletim Sistemi Katmanları

Uygulama Katmanı: Kullanılan her tür program bu katmanda yer almaktadır. Örneğin Word, Excel vb...

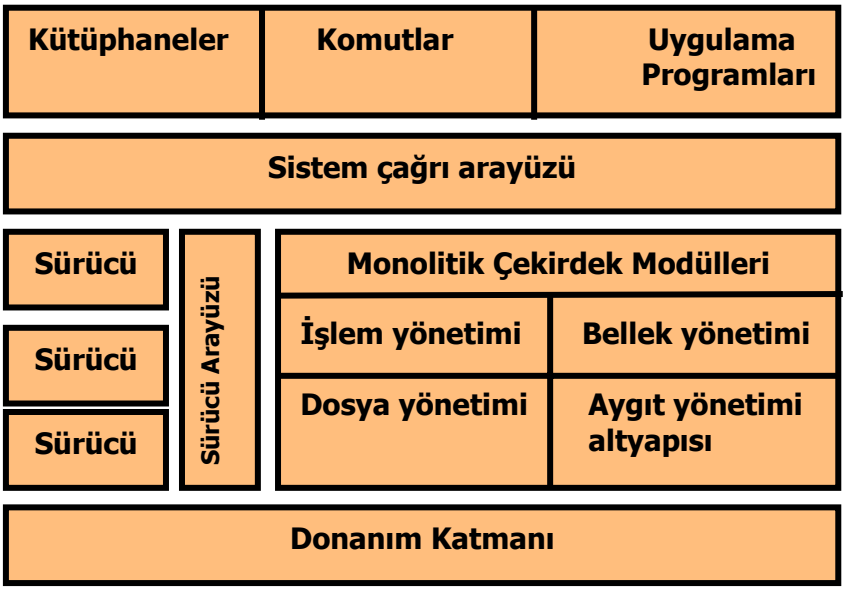
Kabuk Katmanı: Genellikle kullanıcı arayüzü de denilen kullanıcı ile bilgisayarın iletişimini sağlayan arabirimdir. Buna MS-DOS'da komut istemi arayüzü (C:>), Linux'de root olarak giriş yapıldığında #, kullanıcı olarak giriş yapıldığında $ ile görünen arabirim örnek olarak verilebilir. Linux'de birden fazla kabuk bulunmakta ve chsh komutu ile bunlar arasında geçiş yapılabilmektedir. Bu katman; uygulama katmanında kullanıcın verdiği bir komutu alarak çekirdek katmanına iletmektedir.

Çekirdek katmanı: Kabuk katmanından gelen komutlar doğrultusunda donanım katmanı ile iletişime geçerek gerekli işlemleri yürüten kısımdır.

Donanım katmanı: Ekran kartı, ses kartı gibi donanım elemanlarının bulunduğu kısımdır.

UNIX İşletim Sistemi Katmanları

UNIX, 1969 yılında, Ken Thompson tarafından Bell Laboratuvarlarında geliştirilmiş, çok kullanıcılı, çok görevli yapıyı destekleyen bilgisayar işletim sistemidir. Günümüzde Windows tabanlı sistemlerden sonra en çok kullanılan ve kökleri UNIX'e dayanan işletim sistemi GNU/Linux'tur.

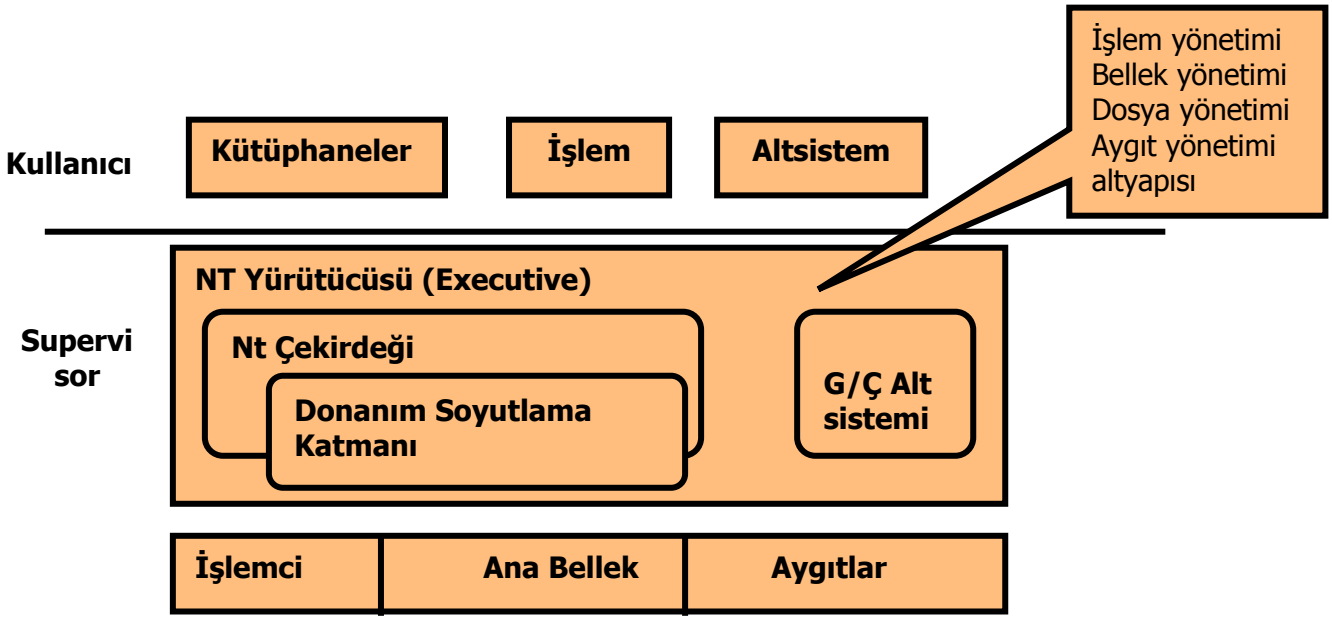


Şekil 5. Unix işletim sistemi mimarisi

Unix işletim sistemi çekirdeğinde, işletim sisteminin temel fonksiyonlarını yapması için gerekli işlem, bellek, dosya yönetimi bulunmaktadır. Aygıt yönetimi öncelikli olarak çekirdeğin içinde yer almaktaydı. Ancak daha sonra tek bir donanımın eklenmesinin bile bütün çekirdeğin tekrar derlenmesini gerektirmesi nedeni ile aygıt yönetimi yine çekirdeğin içinde olmak üzere ayrı bir bölüm haline getirilmiştir. Sistem çağrı arayüzü yani kabuk kullanıcının uygulama katmanında gerçekleştirdiği işlemleri ve komutları alıp yorumlayarak çekirdeğe iletmekte ve çekirdekte direkt olarak donanım ile iletişime geçerek gerekli işlemleri yapmaktadır.

Windows NT/2000/XP Katmanları

Windows NT mimarisi donanım soyutlama katmanı, NT çekirdeği, NT yürütücüsü ve birçok alt sistemden oluşmaktadır. Donanım soyutlama katmanı, bir donanımın birçok detayının soyutlanması (örneğin kesme adresleri) ile işletim sisteminin donanımsal adresleri kullanması yerine farklı soyutlamaları kullanmasını sağlamaktadır. NT çekirdeği donanıma en yakın işletim sistemi mekanizmalarını örneğin kesmelerin ele alınması ve iş parçacıklarının zamanlanması gibi işlemleri yürütmektedir. NT yürütücüsü ise NT çekirdeğinin en üstünde işlemler ve kaynaklarla uğraşan ve işlem, bellek ve dosya yönetimini yapan ve aygıt alt sistemini içeren bölümdür. Sistem çağrı yorumlayıcısı Win32 API tarafından gerçekleştirilmektedir. G/Ç alt sistemi, sistem çekirdek fonksiyonlarından ayrı bir alt sistem olarak tanımlanmış ve aygıt sürücülerini içermektedir. NT alt sistemleri ise kullanıcı tarafında NT çekirdeğinin bazı işlemleri yapabilmesi için gerekli servisleri sağlamaktadır.



Şekil 6. Windows NT/2000/XP işletim sistemleri mimarisi

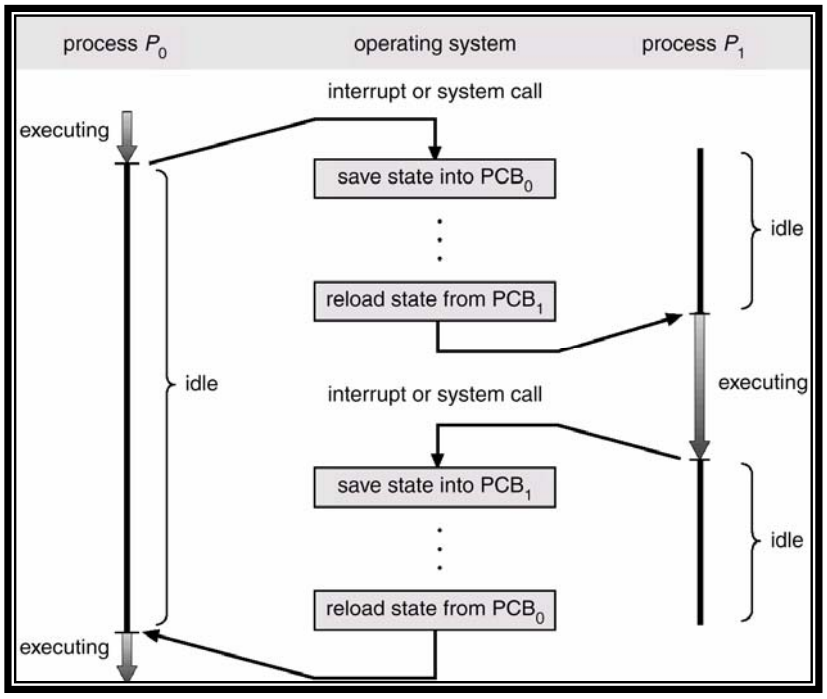
İşlem Yönetimi (İşlem, Semafor ve Kilitlenme)

İşlem (process) belli bir işlemi yerine getiren program parçasıdır. Ancak bir program parçasında sadece kodlar vardır oysa bir işlem çalışma sırasında gerekli olan tüm bilgileri bünyesinde barındırmaktadır. Bir işlemin yapısında; işlemin yazılı olduğu metin bölümü, bir sonraki işletilecek olan komutu gösteren sayaç ve işlemci kaydedicilerinin içeriği olan program sayacı (program counter) bölümü, yerel değişkenler, dönüş adresleri gibi geçici bilgileri tutan yığın (stack) bölümü, genel değişken vb. bilgileri tutan veri bölümü bulunmaktadır. Bazı işlemlerde işlemin çalıştığı sürece kullandığı bir alan (heap) de bulunmaktadır.

İşlem Kontrol Bloğu (Process Control Block): Her işlem işletim sistemi tarafından işlem kontrol bloğu tarafından gösterilir, yer alan öğeler ise şunlardır;

Şekil 1. İşlem Kontrol Bloğu

- İşlem durumu: İşlem yeni, çalışıyor, bekliyor, hazır veya bitti şeklinde olabilir.
- Program sayacı: İşlemde çalıştırılacak bir sonraki komutu gösterir.
- MİB kaydedicileri: İşlemci mimarisini göre kaydedicilerin sayısı ve türü değişmektedir. İşlemde kullanılan kaydediciler (AX, BX vb.) içerisindeki bilgiler bir kesme geldiğinde mutlaka saklanmalıdır (Şekil 2).
- MİB zamanlanma bilgisi: Burada işlemin önceliğini, zamanlama kuyruğundaki işaretçisi ve diğer zamanlama parametreleri ile ilgili bilgiler bulunmaktadır.
- Bellek yönetimi bilgisi: İşletim sistemi tarafından kullanılan bellek sistemleri; taban ve limit (tavan) kaydedicileri, sayfa tablosu veya bölüm tablosu gibi bilgileri içermektedir.
- Hesap bilgileri: İşlem tarafından kullanılan MİB ve diğer parametrelerin kulanım zamanlarını içermektedir.
- G/Ç durum bilgisi: İşlem tarafından kullanılan G/Ç aygıtlarının listesi, açılan dosyaların listesi vb. bilgileri içermektedir.

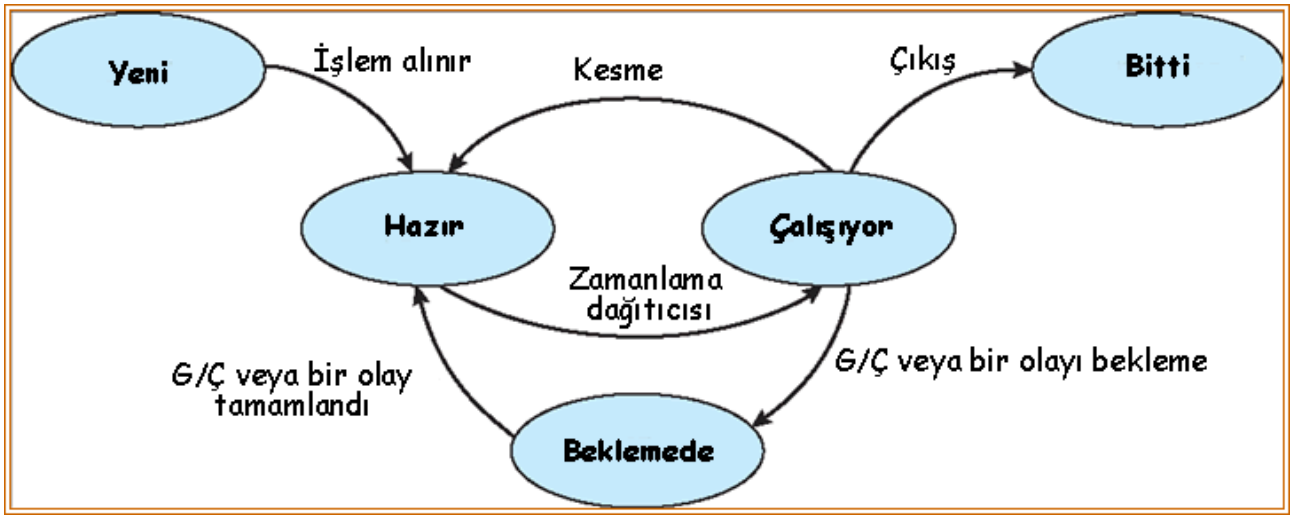


Şekil 2. MİB'nin işlemden işleme geçiş aşamaları

İşlem durumu (Process state):

Bir işlemin hangi durumda olduğunu gösteren ve işlemin ömrü boyunca değişken bir durumu bulunmaktadır. Bunlar;

- Yeni: İşlemin oluşturulduğunu gösterir.
- Çalışıyor: İşlemin komutları yürütülmektedir.
- Bekliyor: İşlem bir olayın gerçekleşmesini beklemektedir. Örneğin bir G/Ç işlemi.
- Hazır: İşlem bir işlemciye atanmak için beklemektedir.
- Bitti: İşlem çalışmasını bitirdi.



Şekil 3. İşlem durumları

İş Parçacıkları (Threads)

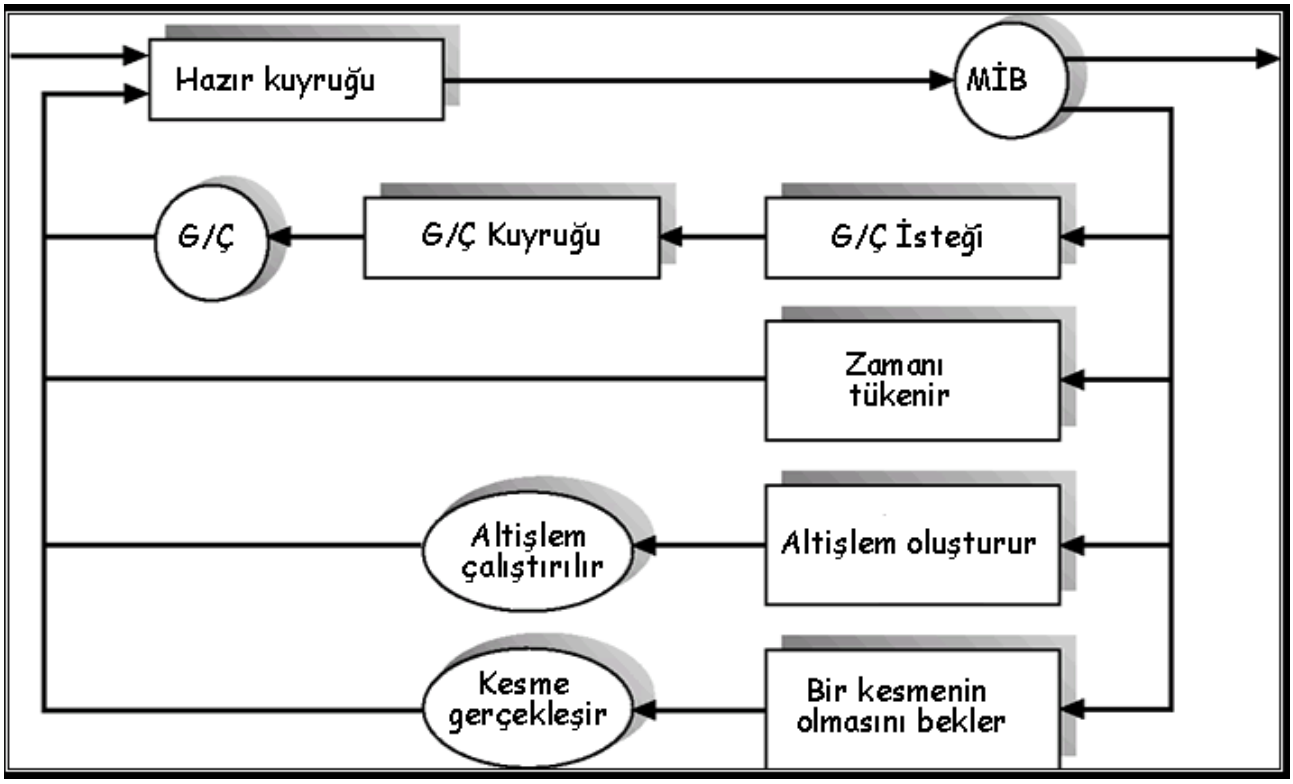
İş parçacığı, işlemin bir parçasıdır. Örneğin bir işlemin bir kelime işlemci programının çalışmasını yürüttüğünü düşünelim. Burada kullanıcı örneğin bir satırdaki yazıları seçerek kalın yapmak istediğinde bu faaliyete iş parçacığı denilmektedir. Modern işletim sistemleri aynı işlem içerisinde birden fazla iş parçacığının yürütülmesine izin vermektedir.

İşlemlerin Zamanlanması (Process Schedulers)

Tüm işlemler iş kuyruğuna (Job queue) alınmaktadır. Çalıştırılmaya hazır olan veya bekleyen işlemler hazır kuyruğana (ready queue) alınarak MİB'ne gönderilir. 'Dağıtıcı (Dispatcher)' çalıştırılacak olan bir işlemi seçerek MİB'ne gönderdikten sonra aşağıdaki olaylar meydana gelebilir;

- İşlem bir G/Ç isteğinde bulunabilir ve böylece G/Ç kuyruğuna yerleştirilir.
- İşlem alt işlemler oluşturabilir ve bunların bitmesini bekleyebilir.
- İşlem bir kesme oluştuğu için MİB'de çalışması durdurularak kesmenin bitiminin ardından kuyruğa alınarak MİB'ne gönderilebilir.

İlk iki durumda işlem hazır durumdan bekliyor durumuna getirilir ardından işlemler gerçekleştikten sonra hazır durumuna getirilerek kuyruğa tekrar girer. Bu döngüde bir işlem bitirilinceye, tüm kuyruklardan çıkıncaya ve kullandığı kaynakları bırakıncaya kadar kalır.



Şekil 4. İşlemlerin kuyruğa alınarak zamanlanması

Kilitlenme (Deadlock)

Sonsuz döngü anlamına da gelmektedir. Değişik işlemlerin/iş parçacıklarının birbirlerinin alanlarını kullanmalarından ortaya çıkar. Windows'da sıklıkla karşınıza çıkan mavi ekranın sebebi! Örneğin bir işlem bir kaynağı (örneğin A dosyasını) kullanırken diğer bir kaynağı (örneğin B dosyasını) kullanmak istemektedir. Ancak aynı zamanda başka bir işlem B dosyasını kullanırken A dosyasını kullanmak istemektedir. Bu durumda her iki işlem de istedikleri dosyaları alamayacakları için bu işlemler bloklanacaklardır.

Semafor (Semaphore):

Latincede 'deniz feneri' anlamına gelen Semafor, negatif olmayan bir tamsayı olup işlemin başlangıç deyiminden itibaren wait ve signal işlemleri ile değeri değiştirilebilen bir değişkendir.

Signal (S) : S = S + 1 (1 arttırma)

Wait (W) : W = W – 1 (1 azaltma)

$$Vol(sem) = C(sem) + ns(sem) - nw(sem)$$

Vol (sem): Semafor'un değeri. Bu değer 0'a eşit veya büyük olmalıdır.

C(sem) : Semafor'un başlangıç değeri

ns(sem): Semafor üzerindeki signal operasyon sayısı

nw(sem): Semafor üzerindeki wait operasyon sayısı

$Vol(sem) \leq 0$

$Nw(sem) \leq ns(sem) + C(sem)$

İki ayrı işlem aynı anda bellekte yürütülürken kaynaklara aynı anda erişmeye çalışabilirler. Bu durumda kullanılan semafor, sistem kilitlenmelerine engel olur. Bir semafor'un paylaşılabilmesi onun başlangıç değerine bağlıdır. Başlangıç değeri 'n' olan bir semafor 'n' işlem tarafından kullanılır.

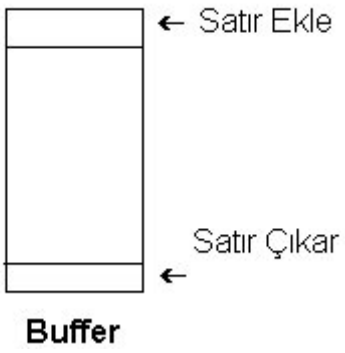
Wait ve signal işlemleri yanlış yerleştirilirse sistem hemen kilitlenir. Semaforlar işlemler arası iletişimi sağlayarak sonsuz döngülere ve sistem kilitlenmelerine engel olurlar. Örneğin; X ve Y başlangıç değerleri 1 olan iki semafor olsun. P_0 ve P_1 ise bu iki semaforu kullanan işlemler olsun.

P_0	P_1	
.	.	
wait (S)	wait (Q)	
wait (Q)	wait (S)	
.	.	Deadlock
.	.	(işlemler bloklanır)
.	.	
signal (S)	signal (Q)	
signal (Q)	signal (S)	

Bloklama ve bloğu açma (Blocking and Unblocking): Bir wait işlemi eğer ilgili semaphore değeri 0 ise o işlemin bloke edilmesine neden olur. Semaphore'un değeri signal ile 1 olunca bloke kaldırılarak işlem çalışabilir hale getirilir.

Örnek: İki işlemden bir tanesinin bir buffer'a çeşitli değerler aktardığını, diğerinin ise buradan bu bilgileri aldığını farz edelim. Buna verilecek bir örnek bir satır uzunluğundaki bir karakter dizisini her seferinde buffer'a atan bir işlem (process) ile her seferinde buffer'dan bir satır alıp bunu ekrana yazan bir işlem olabilir. Burada bir başka farz ediş ise tampon (buffer) büyüklüğünün kısıtlı olmasıdır (n satır tutulabildiği). Bu durumda bu iki işlem arasında senkronizasyon gereklidir:

1. Eğer buffer dolu ise buraya bilgi aktarılmayacaktır.
2. Eğer buffer boş ise buradan bilgi alınmayacaktır.



$0 \leq d - e \leq N$

d: Buffer'a atılan satır sayısı

e: Buffer'dan alınan satır sayısı

N: Space avaible (Boş alan sayısı)

Q: Item avaible (dolu alan sayısı yani eleman sayısı)

Program for Producers

Repeat indefinitely

Begin

Produce item;

Wait (space avaible);

Wait (buffer maniplation);

Deposit item in buffer ;

Signal (buffer maniplation);

Signal (item avaible);

End.

Yukarıdaki program bir üretici programdır. Buffer'a eleman ekler. Elemanı üretir, boş alan oluncaya kadar bekler, bufferı kullanıp kullanamayacağını sorgular

(Buffer = 0), üretilen satırı buffer'a yerleştirir, buffer'ı serbest bırakır ve eleman sayısını 1 arttır.

```
Program for consumers;
Repeat indefinitely
Begin
Wait (item avaible);
Wait (buffer maniplation);
Extract item in buffer ;
Signal (buffer maniplation);
Signal (space avaible);
End.
```

Yukarıdaki program bir tüketici programdır. Eleman sayısı 0'dan büyük olana kadar- bufferda eleman oluncaya kadar bekler, bufferı kullanıp kullanamayacağını sorgular (Buffer = 0), elemanı alır, buffera ulaşımı sağlar (Buffer maniplation'ı bir arttırır), boş yer sayısını bir artırır.

Bu iki işlem bellekte aynı anda çalışmalarına rağmen buffer maniplation ile aralarındaki iletişim sağlanmış ve kilitlenmeler önlenmiştir. Bir anda ancak bir işlem buffera erişmektedir.

Monitör Programları

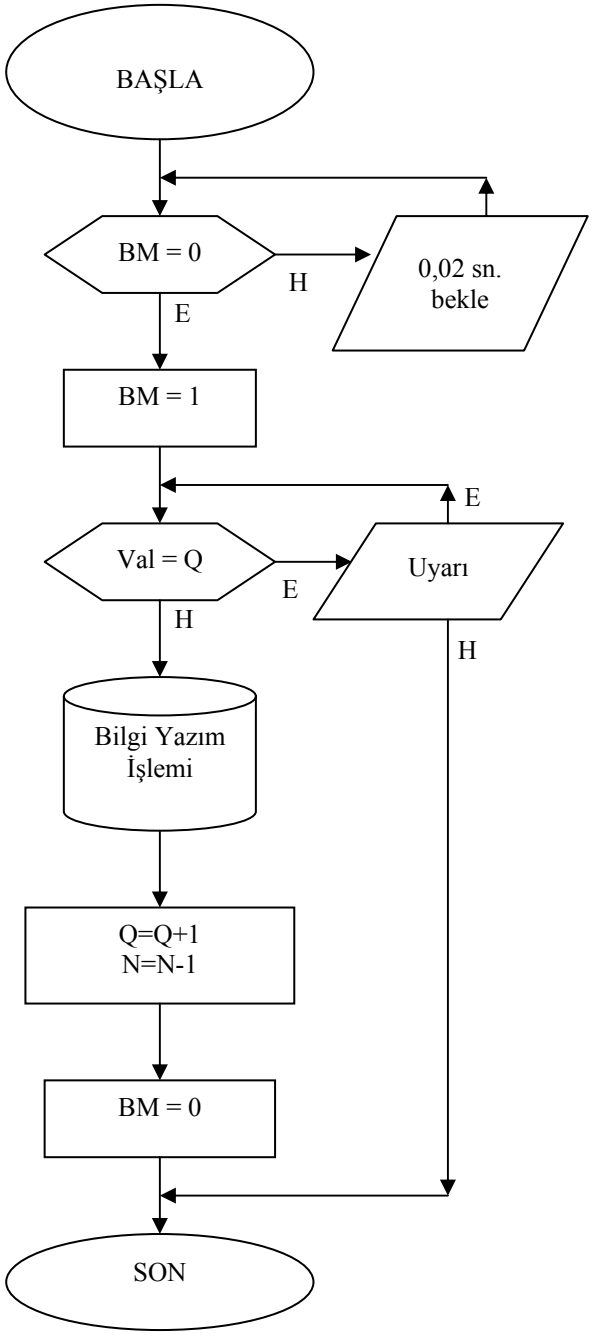
Monitör programları paylaşılan nesneye ulaşmada meydana gelebilecek problemleri ortadan kaldırmaya yönelik olarak geliştirilmiştir. Yani izleyici programlardır.

1. Paylaşılan nesneyi oluşturan veri
2. Bu nesneye ulaşmak için kullanılan bir grup procedure
3. Nesneyi başlangıç durumuna getiren bir program parçasından oluşmaktadır.

Daha önceki buffer örneği için;

1. Buffer alanı ve buraya işaret eden pointerler
2. Depolama ve çıkarma işlemleri için kullanılan iki procedure
3. Buffer işaretçilerini eski haline getiren bir program parçası

Monitör programları, wait ve signal işlemleri yanlış gerçekleştirildiğinde oluşan sistem kilitlenmeleri engeller. Semaforlarla işlemler arası kombinasyonu sağlarlar ve wait ve signal işlemleri arasındaki senkronizasyonu sağlarlar.



Program başlatılır.

BM bayrağının değeri kontrol edilir. BM=1 ise program bu bayrağın değeri sıfır olana kadar bekler.

BM bayrağı, buffer'ın diğer programlar tarafından kullanılmasını önlemek üzere 1 yapılır.

Semafor değeri ile dolu alan sayısı karşılaştırılır. Değerlerin birbirine eşit çıkması buffer üzerinde boş alan olmadığı anlamına gelir. Bu durumda program bir uyarı mesajı verir. Seçeneneğe göre işlem devam eder.

Program buffer üzerindeki uygun alana konumlanarak verileri yazar.

Semafor'un değeri 1 arttırılır.

BM bayrağı tekrar sıfır yapılır.

Program sonlandırılır.

Mehmet ÇEVRİK
010535012

BELLEK YÖNETİMİ

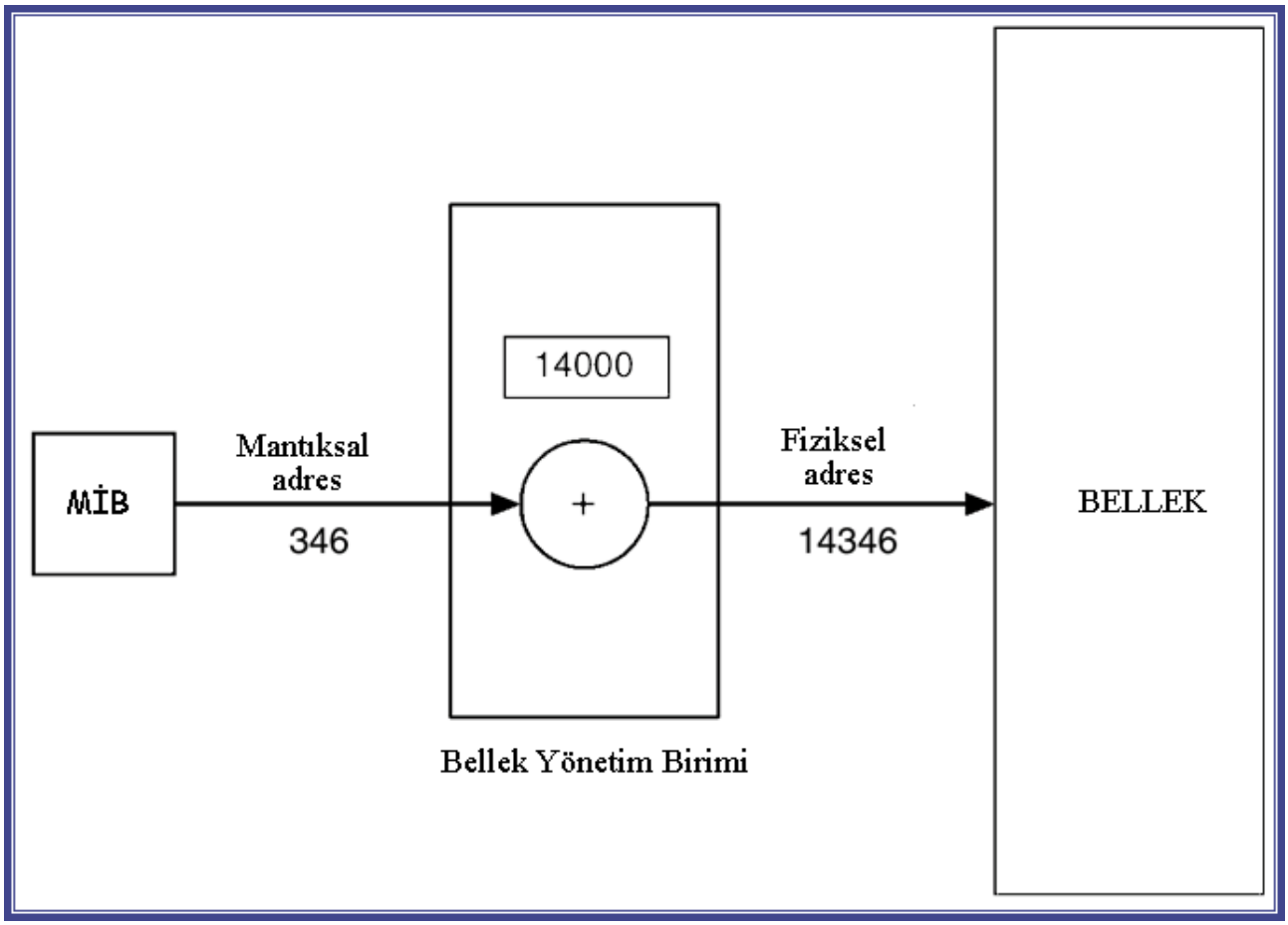
Genellikle bir program hard diskte ikili kodda çalıştırabilir dosya olarak tutulmaktadır. Bu program belleğe bir işlem yardımıyla getirilerek yerleştirilir. Bellek yönetimine göre bu işlem çalışırken bellek ve hard disk arasında gidip gelebilir. Diskteki işlemler belleğe getirilerek çalıştırılmak için beklerken oluşturdukları forma giriş kuyruğu denilmektedir.

İşletim sisteminin bellek yönetiminin unsurları şunlardır:

- Bellekteki herhangi bir işlemi başka bir yere aktarabilmelidir.
- Birden fazla işlem veya kullanıcı olduğunda bir kullanıcını diğer kullanıcını alanlarına girmeleri önlenmelidir.
- Kullanıcılar arası kaynak paylaşımını sağlamalıdır.
- Belleğin mantıksal alanlara bölünmesini sağlayarak bilgiye erişimi kolaylaştırmalıdır.
- Belleğin yetmediği durumlarda fiziksel başka bellek alanlarını yani hard diskleri kullanabilmelidir.

Mantıksal ve Fiziksel Adres

MİB tarafından oluşturulan adrese genellikle mantıksal (veya sanal) adres, bellekte yer alan adrese ise fiziksel adres denilmektedir. Mantıksal adresi MİB tarafından belirlenen bir kullanıcı işlemi belleğe yerleşirken fiziksel adresi belirlenerek yerleştirilir ki bunun için de tekrar yerleştirme (relocation) da denilen taban (base) kaydedicisi kullanılır. Bu taban kaydedicisinin değeri ile mantıksal adres toplanarak elde edilen adres ise fiziksel adrestir.



Şekil 1. Taban kaydedicisini kullanarak belleğe dinamik yerleşim

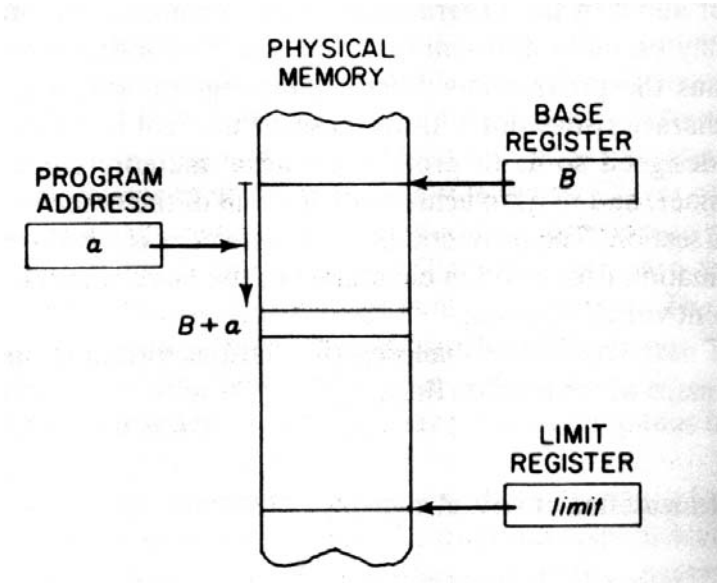
Taban ve Taban Kaydedicileri

Belleğe işlemlerin yerleşimi Tavan (Limit) ve Taban (Base) kaydedicileriyle donanımsal olarak sağlanır. Tavan kaydedicisi uzunluğu belirtmektedir ve taban ve tavan kaydedicisi toplanarak işlemin bittiği adres bulunabilir. Taban adresi değiştirilerek bir işlem bellekte yer değiştirilir. Örneğin;

a: Program adresi (O anda bulunulan adres)

b: Taban adresi (Başlangıç adresi, base adress) olsun.

Şekil 2'de bu işlemin belleğe yerleşimi görülmektedir.

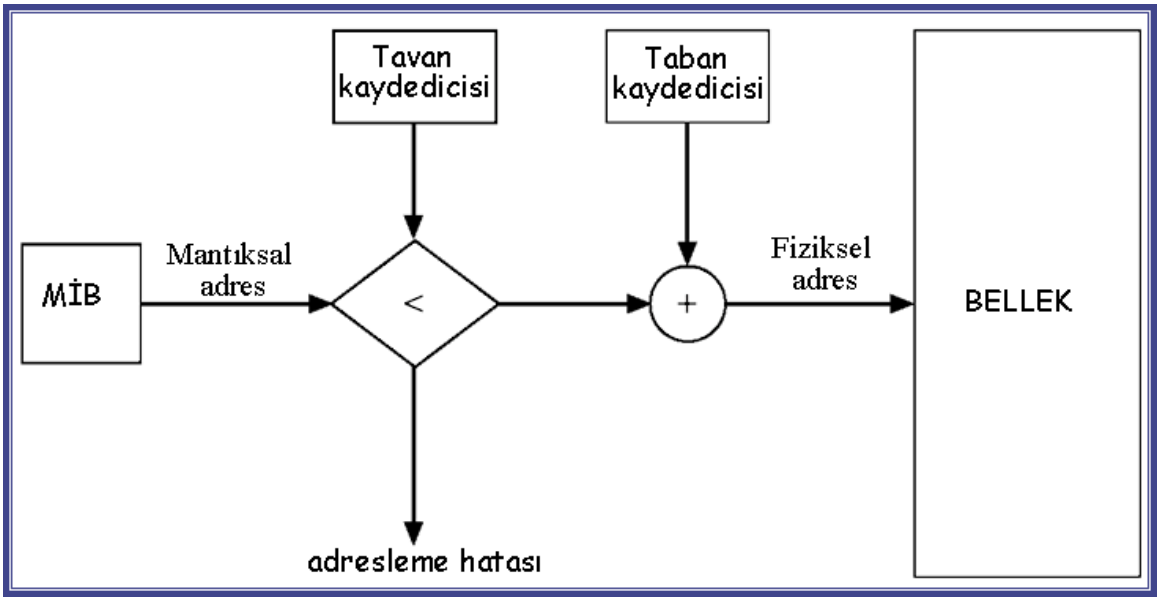


Şekil 2. Taban ve tavan kaydedicileri için adres haritası

Bu adres haritası aşağıdaki kriterlere göre donanımsal olarak adreslenir;

eğer a < 0 ise bellek koruma hatası
a' = B + a
eğer a' > Tavan ise bellek koruma hatası
a' burada istenilen bellek yeridir.

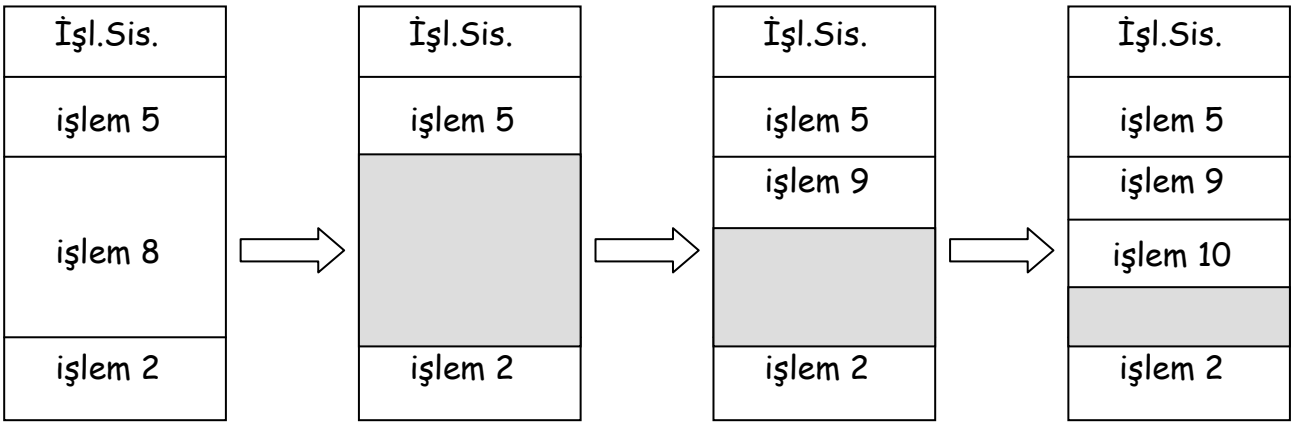
Bellek koruma: Yerleştirilecek olan işlemin bellekte limit kaydedicisinin değerini aşması durumunda bellek koruma hatası ortaya çıkmaktadır.



Şekil 3. Bellek koruma hatası

Bitişik Yerleşim (Contiguous Memory Allocation)

Ana bellekte hem işletim sisteminin hem de kullanıcı işlemlerinin yer almaktadır. Bu nedenle belleğin en etkin şekilde kullanılması gerekmektedir. Bellekte kullanıcı işlemleri boşluklara (hole) bloklar halinde yerleştirilmektedir. Yeni bir işlem geldiğinde bu belleğe işlemin yeteceği kadar bir alan ayrılarak konumlandırılmakta ve işletim sistemi boş alanlar ve dolu alanlar ile ilgili bilgileri kayıt altına almaktadır.



Şekil 4. Bitişik bellek yerleşimi

İşletim sistemi bellekteki n adet boşluğun bulunduğu listeden işlem için en uygun boşluğu nasıl belirleyeceğine ise 3 strateji kullanarak karar vermektedir.

- İlk uyan (First-fit): Listeden işlem için yeterli büyüklükte olan ilk boşluğun seçilerek işlemin yerleştirilmesidir.
- En iyi uyan (Best-fit): Listeden işleme uygun büyüklükteki boşlukların aranarak bulunması ve bunların içinden en küçüğünün seçilerek işlemin yerleştirilmesidir.
- En kötü uyan (Worst-fit): Listeden araştırma yapıldıktan sonra en büyük boşluğun seçilerek işlemin yerleştirilmesidir.

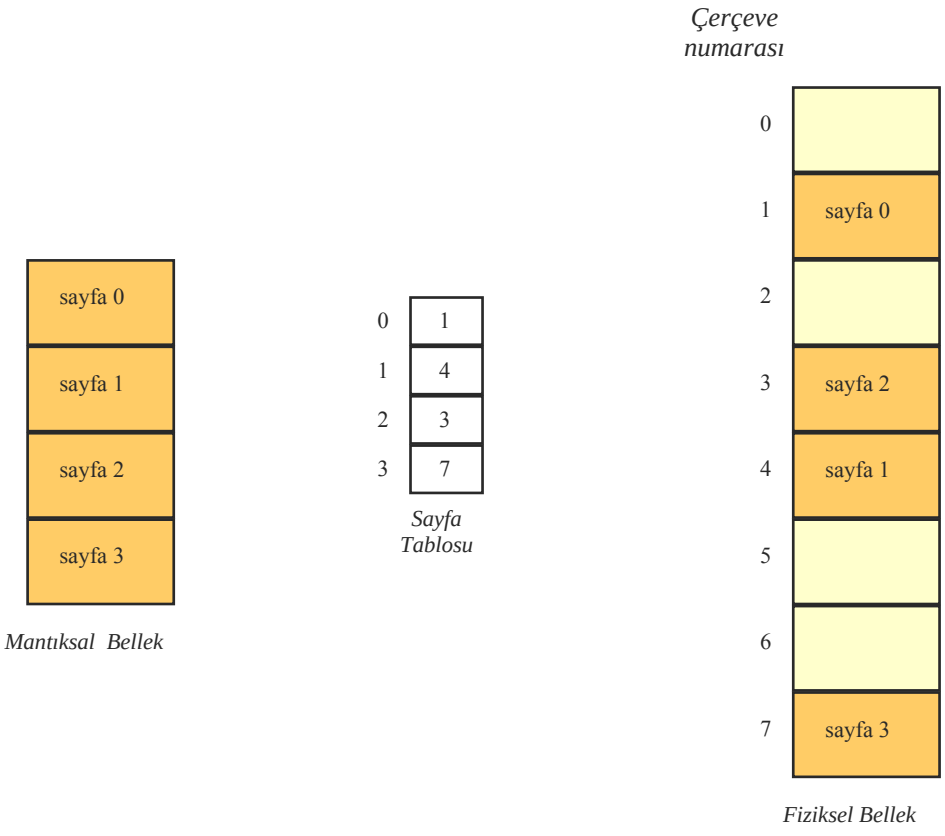
İşlemin belleğe yerleştirilme hızı ve belleğin etkili kullanımı açısından ilk uyan ve en iyi uyan stratejileri en kötü uyan stratejisinden çok daha kullanışlıdır.

Sayfalama (Paging)

Toplam bellek alanının eşit büyüklükte sayfalara bölünmesine denir. Faydaları ise şunlardır:

- Adres çakıştırma işlemi: Bir program adresinde yer alan atıfın hangi sayfaya yapıldığını tespit etmek ve eğer bu sayfa hala bellekte ise bunun hangi bellek sayfası olduğunu tespit etmektir. Bu işlem bilgilere erişimde hız kazancı sağlar.
- Sayfaların gerektiğinde ikincil bellek alanından ana belleğe aktarılması ve bunun tersi işlem yapılması bellekte yer kazancı sağlar.

Fiziksel bellek üzerindeki aynı uzunluktaki bloklara 'çerçeve (frame)', mantıksal bellek üzerindeki aynı uzunluktaki bloklara ise 'sayfa (page) denilmektedir. Sayfaların uzunlukları donanım tarafından belirlenmektedir. Genellikle 2'nin katları; 512 byte, 16 MB kullanılmaktadır.

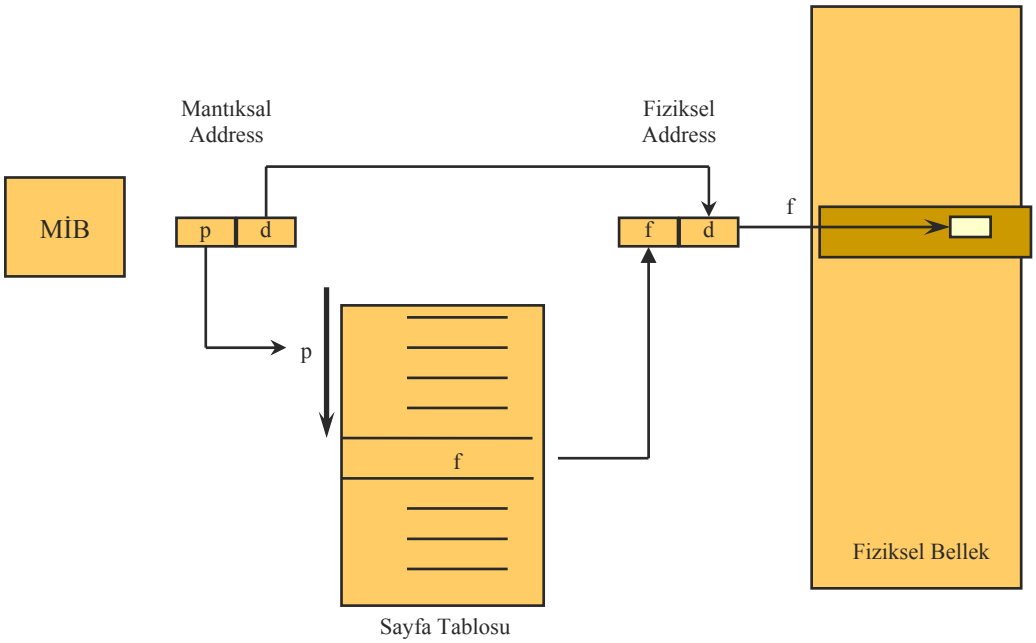


Şekil 5. Fiziksel ve mantıksal belleğin sayfalama modelleri

MİB'nin oluşturduğu adres iki bölümden oluşmaktadır;

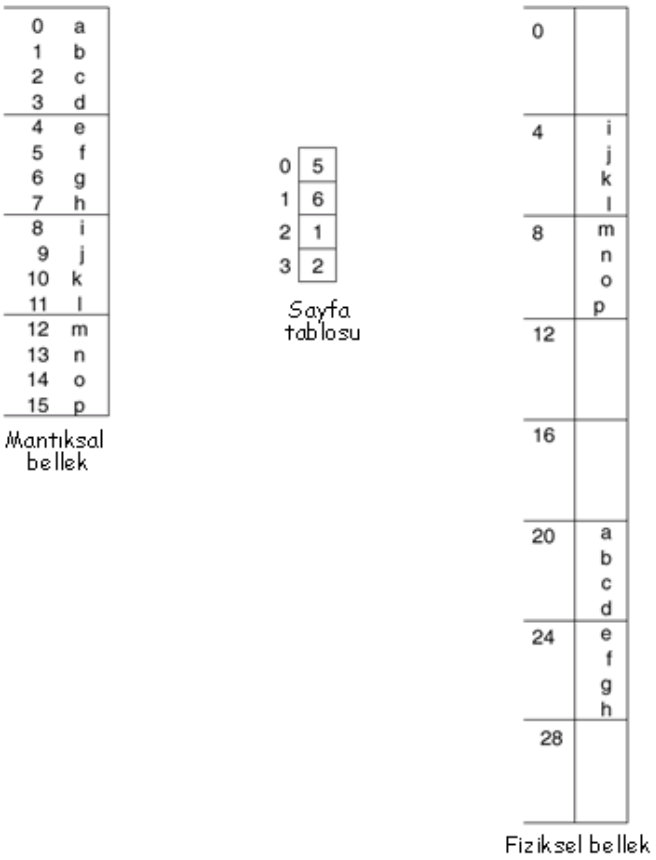
a) sayfa numarası (p): fiziksel bellekteki her bir sayfanın taban adresini tutan sayfa tablosundaki gösterge.

b) sayfa ofseti (d): taban adresi ile birleştirilerek fiziksel bellekte sayfanın içerisindeki yerin belirlenmesinde kullanılır. Sayfa tablosunda hem sayfa hem de ofset adresi bulunmaktadır.



Şekil 6. Sayfalama

Sayfalama örneği; 4 byte uzunluğunda sayfalardan oluşan 32 byte'lık bir bellek olduğunu düşünelim.

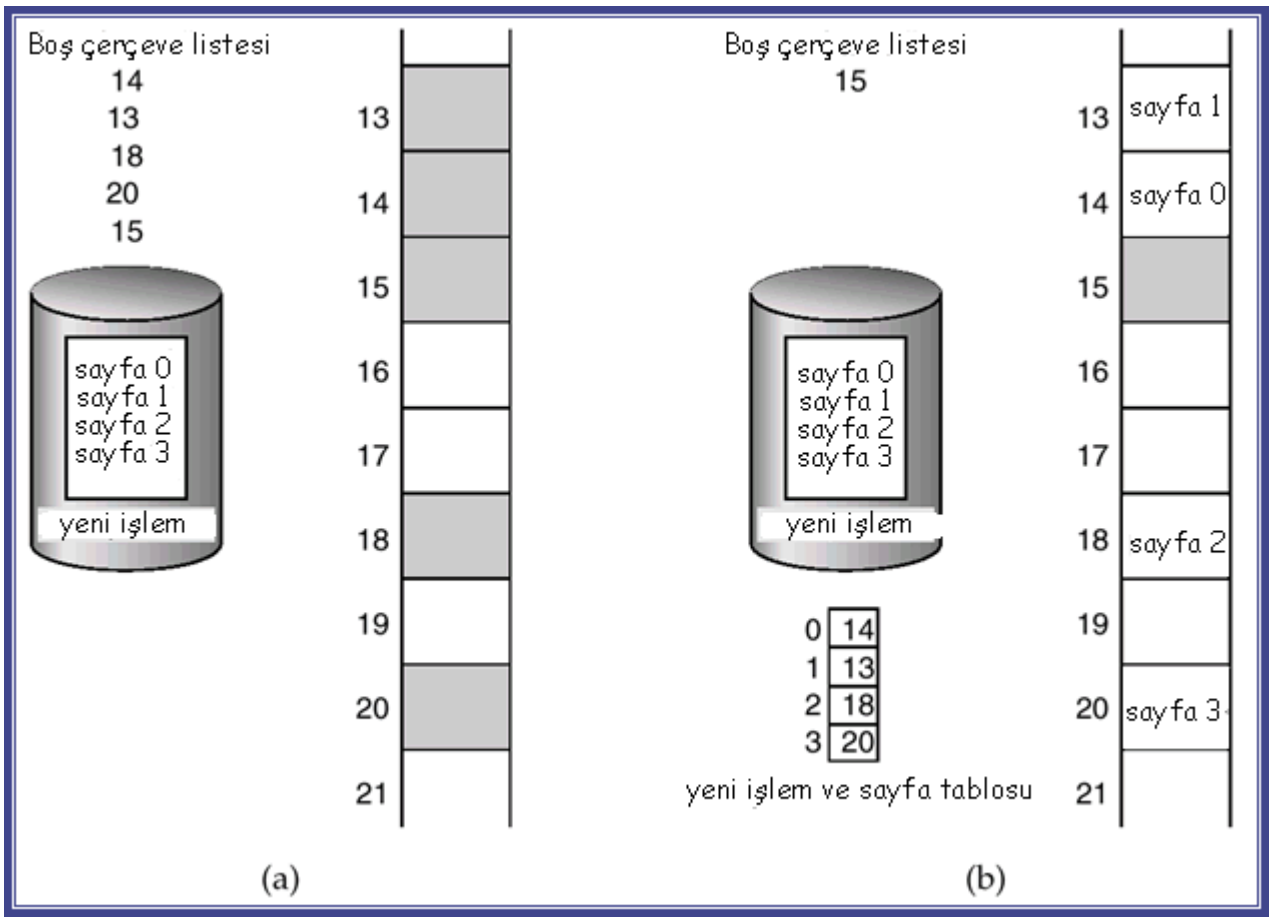


Şekil 7. 4 byte'lik sayfalardan oluşan 32-byte'lık bellekte sayfalama örneği

- Mantıksal adresi 0'da sayfa ve ofset adresi 0'dır. Sayfa tablosuna bakıldığında sayfa 0'ın çerçeve 5'de olduğu görülmektedir. Böylece mantıksal adres 0, fiziksel bellekte 20'ye ((5*4))+0) karşılık gelmektedir.
- Mantıksal adres 3 (sayfa 0, ofset 3) fiziksel bellekte 23'ü ((5*4))+3) adreslemektedir.
- Mantıksal adres 4 (sayfa 1, ofset 0), sayfa tablosuna bakıldığında çerçeve 6'dadır ve fiziksel bellekte 24'ü ((6*4))+0) adreslemektedir.
- Mantıksal adres 13, fiziksel adreste 9'u ((2*4)+1) adreslemektedir.

Modern işletim sistemlerinde; sayfaların uzunlukları genelde 4 KB ile 8 KB arasındadır, bazı sistemlerde bu artabilmektedir. Örneğin Solaris 8 KB ve 4 MB sayfalar kullanabilmektedir. Genellikle sayfa tablosunun içindeki her bir veri 4 byte'lıktır.

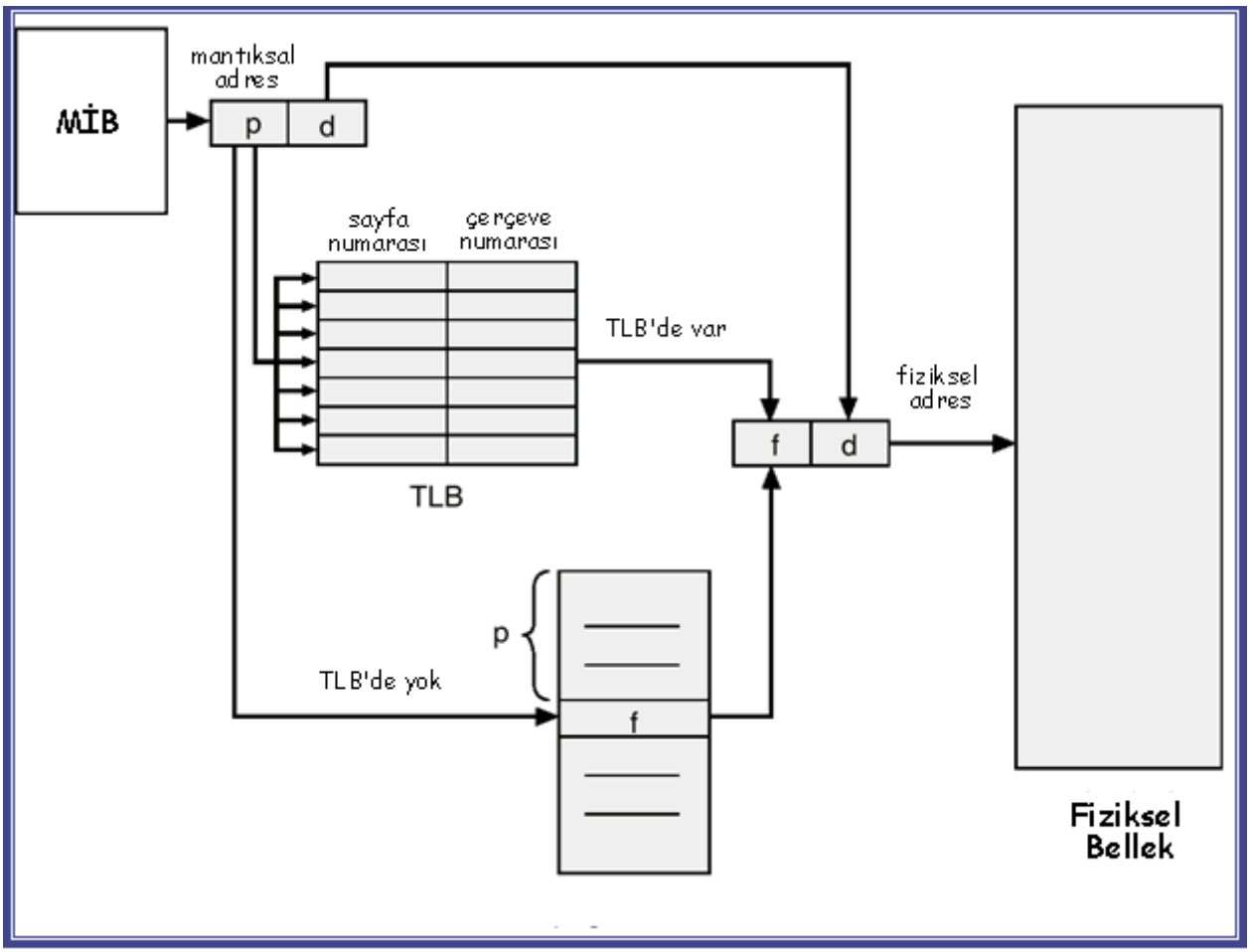
Bir işlem sistem tarafından çalıştırıldığında, işlemin kullanmak istediği kadar sayfa sayısına göre bellekte gerekli sayıda çerçeve ayrılmaktadır. İlk sayfa ayrılan bir çerçeveye yerleştirilir ve bu çerçevenin numarası sayfa tablosuna konulur, diğer bir sayfa bir çerçeveye yerleştirildiğinde o çerçevenin numarası da sayfa tablosuna işlenmektedir.



Şekil 8. Boş çerçeveler (a) Yerleşimden önce ve (b) Yerleşimden sonra

Burada en önemli sorun; sayfa tablosunda çerçeve adresini ararken kaybedilen süredir. Bunu önlemek için özel, küçük ve hızlı bir ön bellek olan TLB (Translation look-aside buffer) kullanılmaktadır. Bu hızlı bir bellek alanıdır. Burada bir anahtar ve değer bulunmaktadır. Bulunmak istenen bilgi için tüm anahtarlara aynı anda hızlı bir biçimde bakılmaktadır. TLB sayfa tablosu ile birlikte kullanılmaktadır. TLB sayfa tablosunun sadece bir kısmını kapsamaktadır. MİB tarafından mantıksal bir adres tanımlandığında bu sayfa numarası TLB'ye iletilir. Sayfa numarası burada bulunursa çerçeve adresi hızlı bir biçimde kullanılarak belleğe ulaşılır. Eğer sayfa numarası TLB'de bulunmazsa (TLB'de eksik olarak bilinir) sayfa tablosuna gidilerek çerçeve adresi bulunur ve bu TBL'ye eklenir ki böylece bir daha ihtiyaç olduğunda bu adres

hızlı bir biçimde bulunabilir. Eğer TLB alanı dolarsa, işletim sistemi buradaki en az kullanılan değerleri bularak bunların yerine yeni değerleri ekleyecektir.

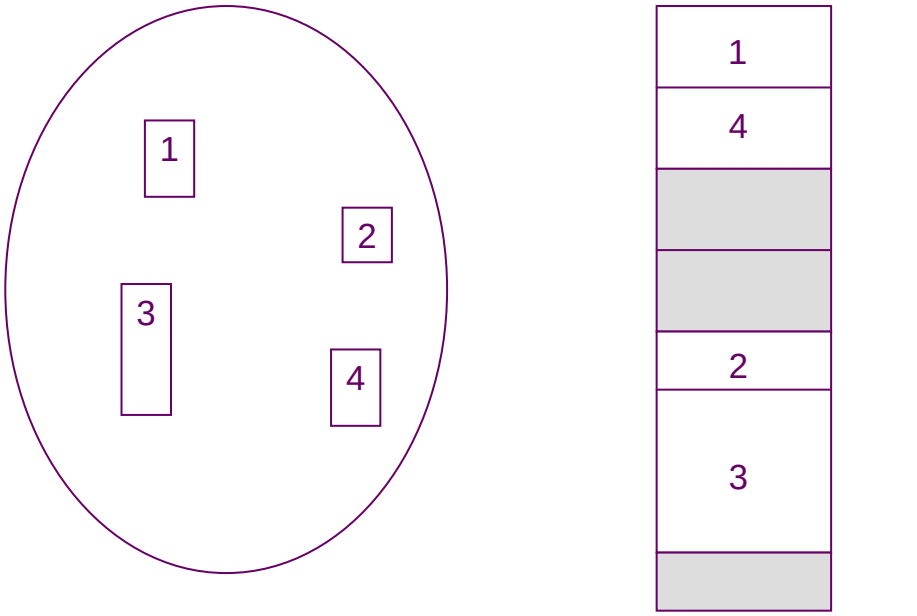


Şekil 9. TLB ile sayfalama

Bölümleme (Segmantation)

Bellek yönetiminin önemli bir yönü, kullanıcının belleği kullanması ile gerçek fiziksel belleğin birbirinden çok farklı olmasıdır. Daha önceki sayfalardan da anlaşılacağı gibi kullanıcının belleği gördüğü biçim ile gerçek bellek birbirinden çok farklıdır. Örneğin bir program yazıldığında bu programın verileri, değişkenleri yığını ayrı bölümler halinde bellekte yer aldığını düşünen kullanıcının belleği gördüğü bu biçimini bölümleme desteklemektedir. Kullanıcı bir program yazdığında, temel olarak

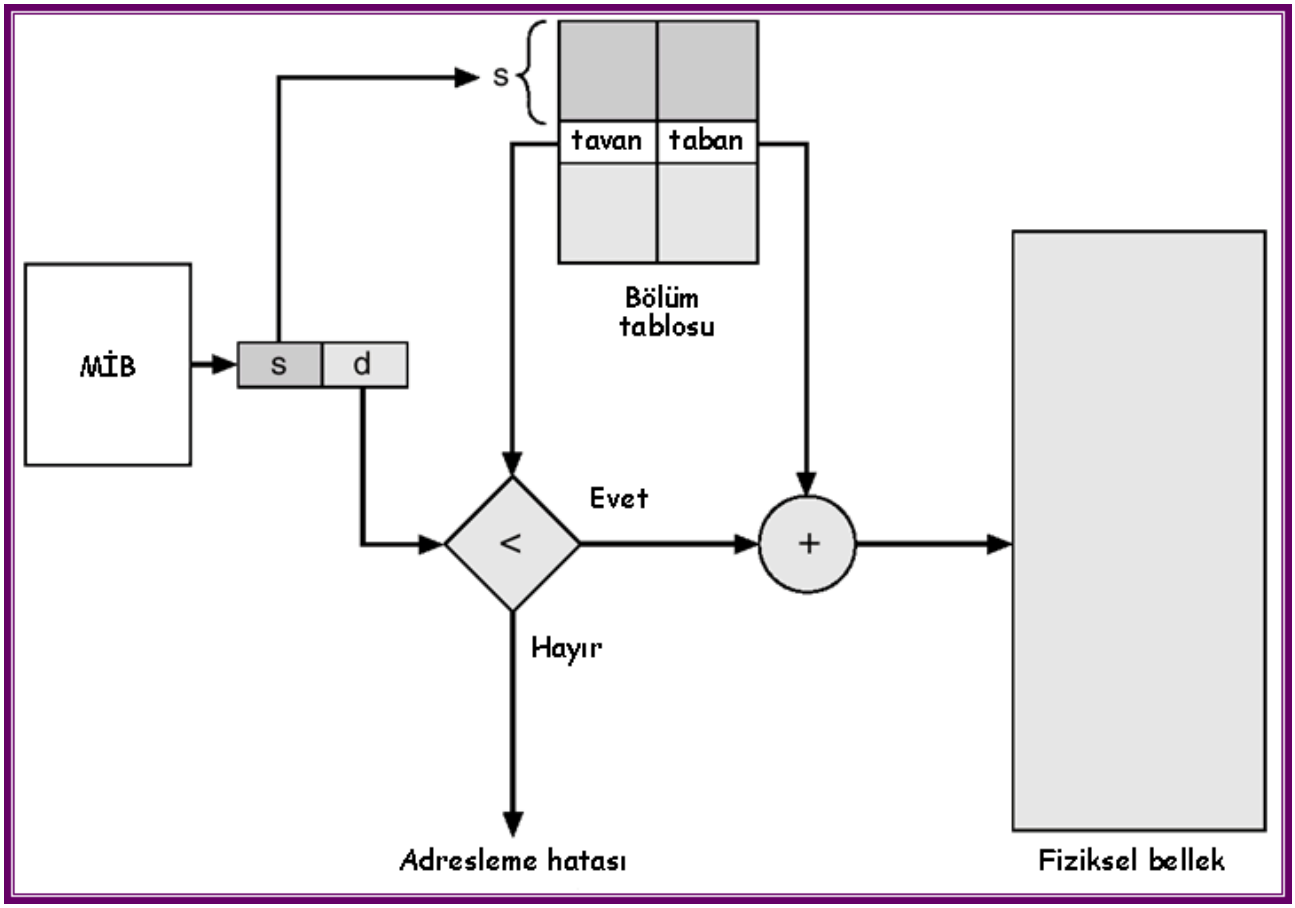
kodların saklandığı kod bölümü, verilerin saklandığı veri bölümü ve alt programlara dallanmada kullanılan yığın bölümü oluşturulmaktadır.



Şekil 10. Kullanıcı ve fiziksel bellek alanlarının gösterimi

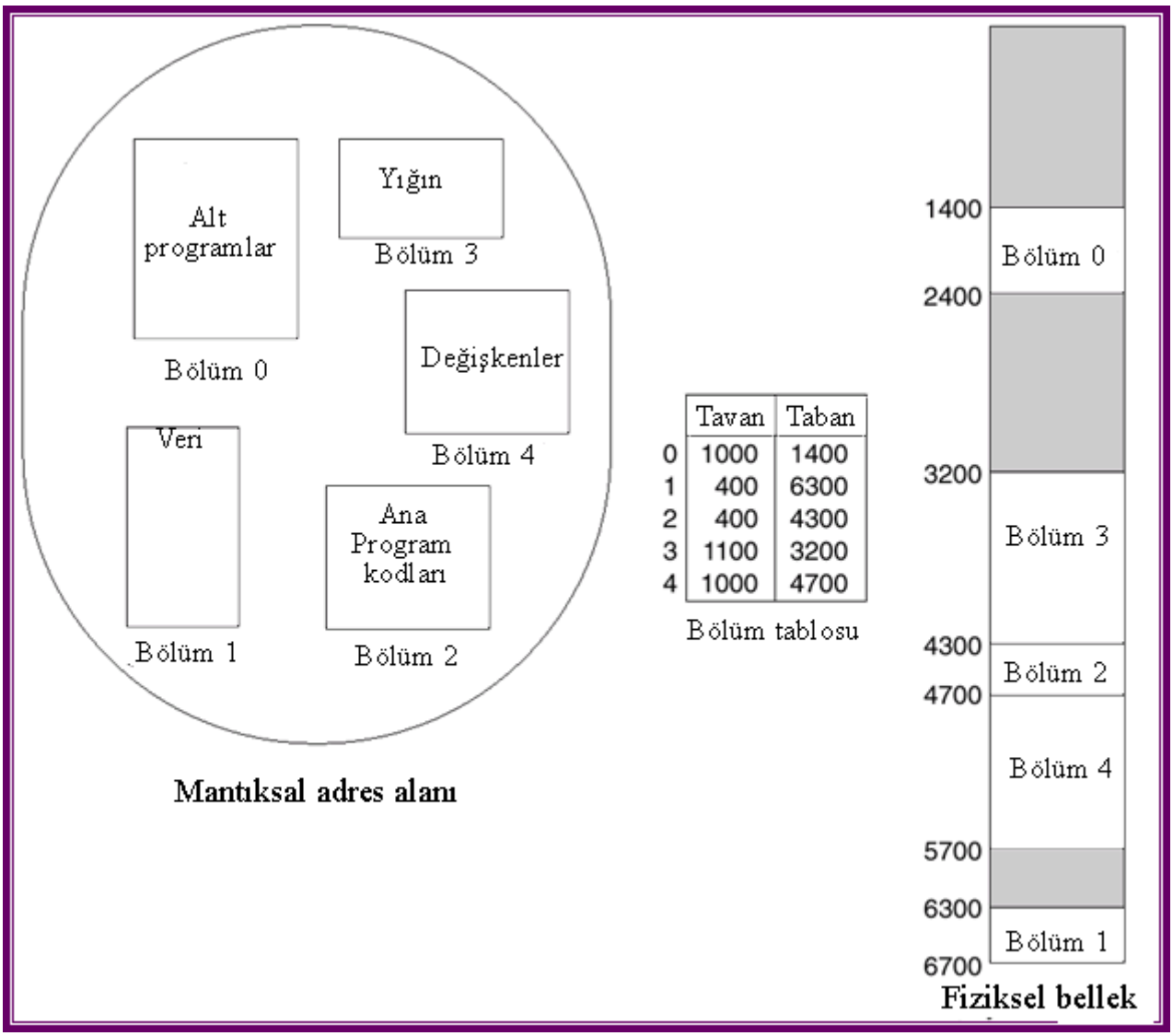
Bölümlemede; mantıksal adres alanı bölümlerden oluşmaktadır. Adres hem bölüm numarası hem de bölüm içerisindeki adresi belirten ofset numarasından meydana gelmektedir (<bölüm numarası, ofset>). Fiziksel bellek tek boyutlu olmasına karşın burada adres iki boyutlu olarak gösterilmekte ve bu da adresin fiziksel belleğe ulaşım için çevrilmesi gerektiği anlamına gelmektedir. Bölüm tablosunda bölümün başlangıç yerini tutan bölüm taban ve bölümün uzunluğunu yani bitiş yerini belirlemede kullanılan bölüm tavan kaydedicileri bulunmaktadır. MİB tarafından oluşturulan mantıksal adreste s: bölüm numarasını ve d: bölümdeki ofset adresi göstermektedir. Ofset adres taban ve tavan kaydedicisi değerleri arasında olmalıdır. Eğer değeri fazla olursa mantıksal adresin bölümü aşması nedeniyle işletim sistemi devreye girmektedir. Eğer

değeri bu belirtilen sınırlar arasında ise bölüm taban kaydedicisi ile bu adres toplanmakta ve bu şekilde istenilen byte'a ulaşılabilinmektedir.



Şekil 11. Bölümleme

Bölümleme için şu örnek verilebilir; bir program yazdığımız düşünelim. Bu program, belli bölümler oluşmaktadır: ana program kodları, alt programlar, yığın, veri ve değişkenlerin olduğu bölümler olsun. Bunlar bölüm tablosu kullanılarak belleğe yerleştirilmektedir. Bu bölümlerin bitiş adresleri ise taban ve tavan kaydedici değerlerinin toplanması sonucu elde edilmektedir. Şekil 12'de bu örneğin gösterimi bulunmaktadır.



Şekil 12. Bölümleme için örnek

Sayfalama ve bölümleme arasındaki farklar

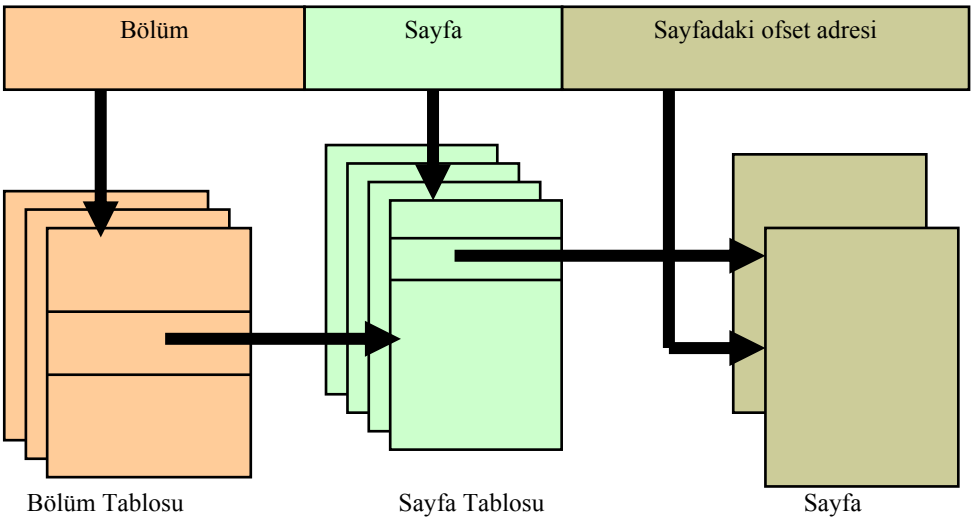
1. Bölümlemenin amacı adres alanının mantıksal olarak dilimlenmesidir. Sayfalama ise belleğin fiziksel olarak dilimlenip bir düzeyli bir bellek (sabit disk, bellek farkı gözetmeden erişim) oluşturulması amacına yöneliktir.

2. Sayfalar makine donanımına bağlı olarak sabit boyuttadır. Bölümler ise kullanıcı tarafından belirlenecek boyuttadır. Yani program adresinin ilgili bölüm kaydedicisi sınırları çerçevesindedir.

3. Program adresinin sayfa ve ofset numaralarına ayrılması donanımın bir işlevidir. Ofset numarasının sınırı aşması otomatik olarak sayfanın çevrilmesine sebep olur. Oysa bölüm ve ofset numarasında bir sınır aşma hadisesi söz konusu değildir. Bu durumda bellek erişim hatası yani erişilmemesi gereken yere erişim hatası oluşur.

Bölümlemenin sayfalama ile birlikte kullanılması

Hem sayfalama ve hem de bölümlemenin avantajları ve dezavantajları bulunmaktadır. Bu nedenle bu ikisi birlikte de kullanılabilir. Örneğin 80*86 bölümleme ve sayfalamayı birlikte kullanmaktadır. Bölümler kendi içerisinde sayfalara ayrılmaktadır. Mantıksal adres üç parçadan oluşmaktadır; bölüm, sayfa ve ofset adresi. Öncelikle MİB tarafından oluşturulan mantıksal adresten bölüm numarası alınarak bölüm tablosuna bakılmaktadır. Bölümün gerçek adresi bulunduktan sonra o bölüme ait sayfa tablosuna bakılmaktadır. Buradan sayfanın çerçeve adresi elde edilerek bellekte bölüm sayfa ve ofset adres toplanarak gerçek adres elde edilmektedir.



Şekil 13. Bölümleme ve sayfalamanın birlikte kullanımı