T.C.
FIRAT ÜNİVERSİTESİ
TEKNİK EĞİTİM FAKÜLTESİ
ELEKTRONİK VE BİLGİSAYAR BÖLÜMÜ

BİL391 İŞLETİM SİSTEMLERİ

(Ders Notları)

İbrahim TÜRKOĞLU

ELAZIĞ- 2006

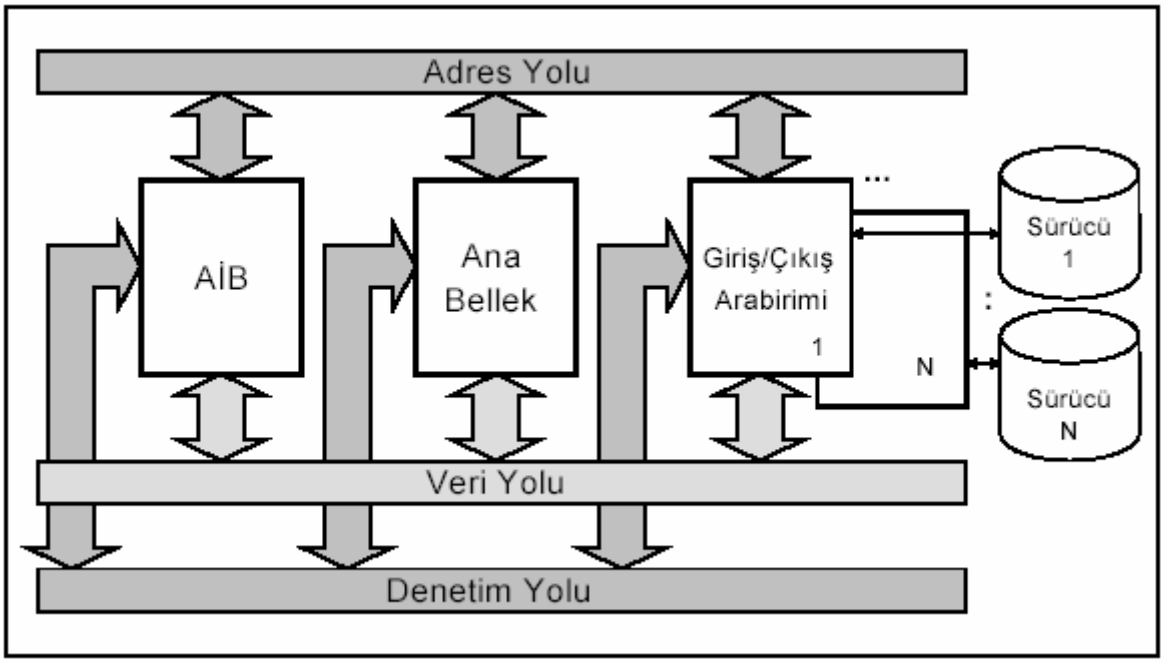
İÇİNDEKİLER

1. BİLGİSAYAR SİSTEMLERİNİN YAPISI

İşletim sistemleri, bir veya daha fazla sürecin (proses) donanım kaynaklarını kullanmasını sağlayarak, bilgisayar sistemi kullanıcılarına çeşitli hizmetler sunar.

1.1. Bilgisayar Sisteminin Temel Elemanları

- İşlemci (Ana işlem birimi – AİB)
- Ana Hafıza
  - Gerçek hafıza veya birincil hafıza olarak da bilinir.
  - Elektrik kesildiğinde bilgileri kaybolduğundan, uçucu bir hafızadır.
- G/Ç modülleri
  - İkincil hafıza sürücüleri
  - Haberleşme birimleri
  - Terminaller (uç birimler)
- Sistem BUS (Adres yolu, Veri Yolu, Denetim yolu)
  - Süreçler, hafıza ve G/Ç modülleri arasında haberleşmeyi sağlar.



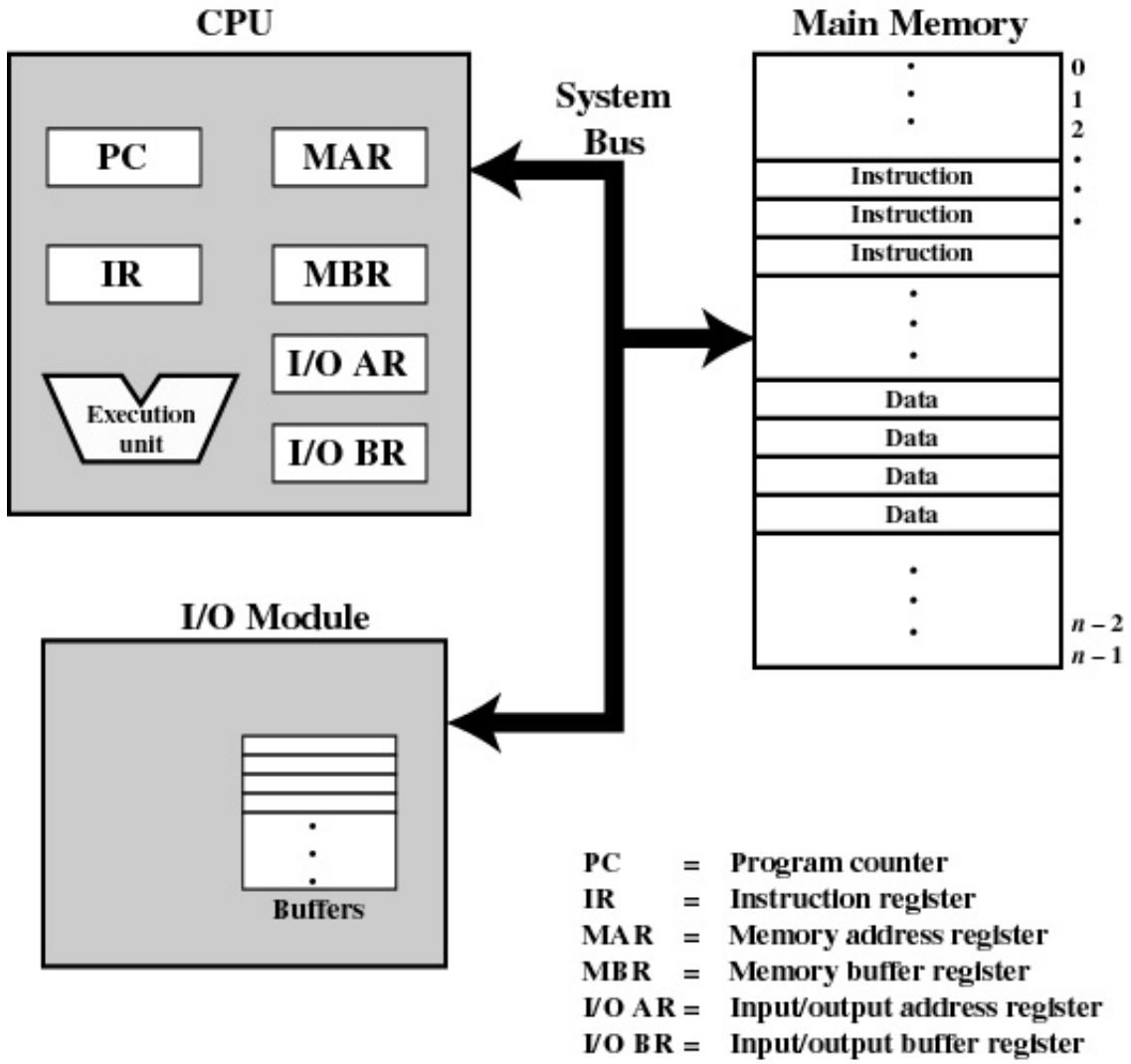
Şekil. Bilgisayar Sistemi

1.2. Bilgisayar Sisteminin Üst Seviye Bileşenleri

A. İşlemci (CPU) kayıtçıları

- Kullanıcıya görünen kayıtçılar:
  - Bunlar ile programcı, kayıtçı kullanımını en iyi şekilde organize ederek ana hafıza kullanımı minimize eder.
  - Makine dili (asembly) tarafından kullanılabilir.
  - Tüm programlar (uygulama programları, sistem programları) ile kullanılabilirler.
  - Kayıtçı türleri : Veri ve adres kayıtçıları
    - Veri Kayıtçısı : Verileri üzerinde tutar.
    - Adres Kayıtçıları: Adresleri üzerinde tutar.
      - İndisli :Bir adrese ulaşmak için taban bir değere bir indis eklemede kullanılır.
      - Segment pointer (Segment göstericisi): Hafıza segmentlere bölündüğü zaman, hafızaya bir segment ve ofset değeri ile ulaşılabilmesini sağlar.
      - Stack pointer (Yığın göstericisi) : Yığının en üstüne işaret eder.

- Kontrol ve durum kayıtçıları : İşletim sistemi aracılığı ile işlemci tarafından kullanılarak, programların düzenli olarak çalışmasını sağlamak için işlemciyi kontrollü bir şekilde çalıştırır. Bunlar:
  - Program Sayacı (Pogram Counter, PC): Çalıştırılacak olan bir komutun adresini içerir.
  - Komut Kayıtçısı (Instruction Register, IR) : Çalıştırılacak komutu içerir.
  - Program durum sözcüğü (status word, PSW): Şartlı işlemleri, kesme kullanımı ve yönetim/kullanıcı modunu kontrol eder. Şöyleki; yapılan işlemlerin sonucunda işlemci donanımı tarafından PSW bitler kurulur. Bu bitlere, bir program tarafından erişilebilir fakat değiştirilemezler. Örnek: pozitif, negatif, sıfır ve taşma sonuçları gibi

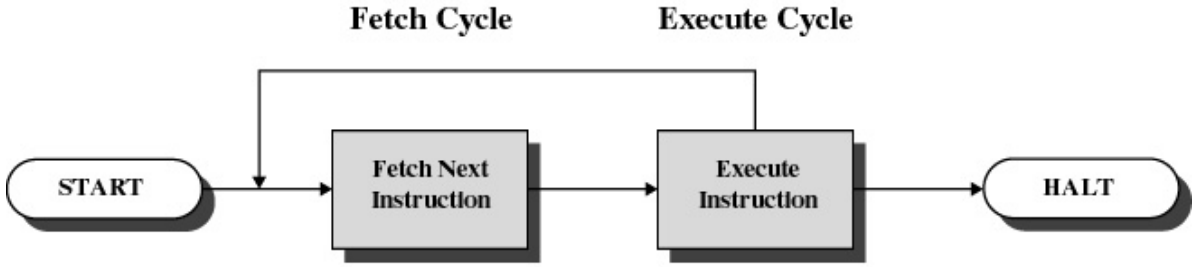


Şekil. Üst seviye bileşenler

B. Komut alma ve yürütme stratejisi

1.İşlemci, PC nin gösterdiği hafıza adresinden çalıştırılacak olan komutu alır.

2.Sonrasında PC bir sonraki alınacak olan komutun adresini tutmak üzere içeriğini artırır.



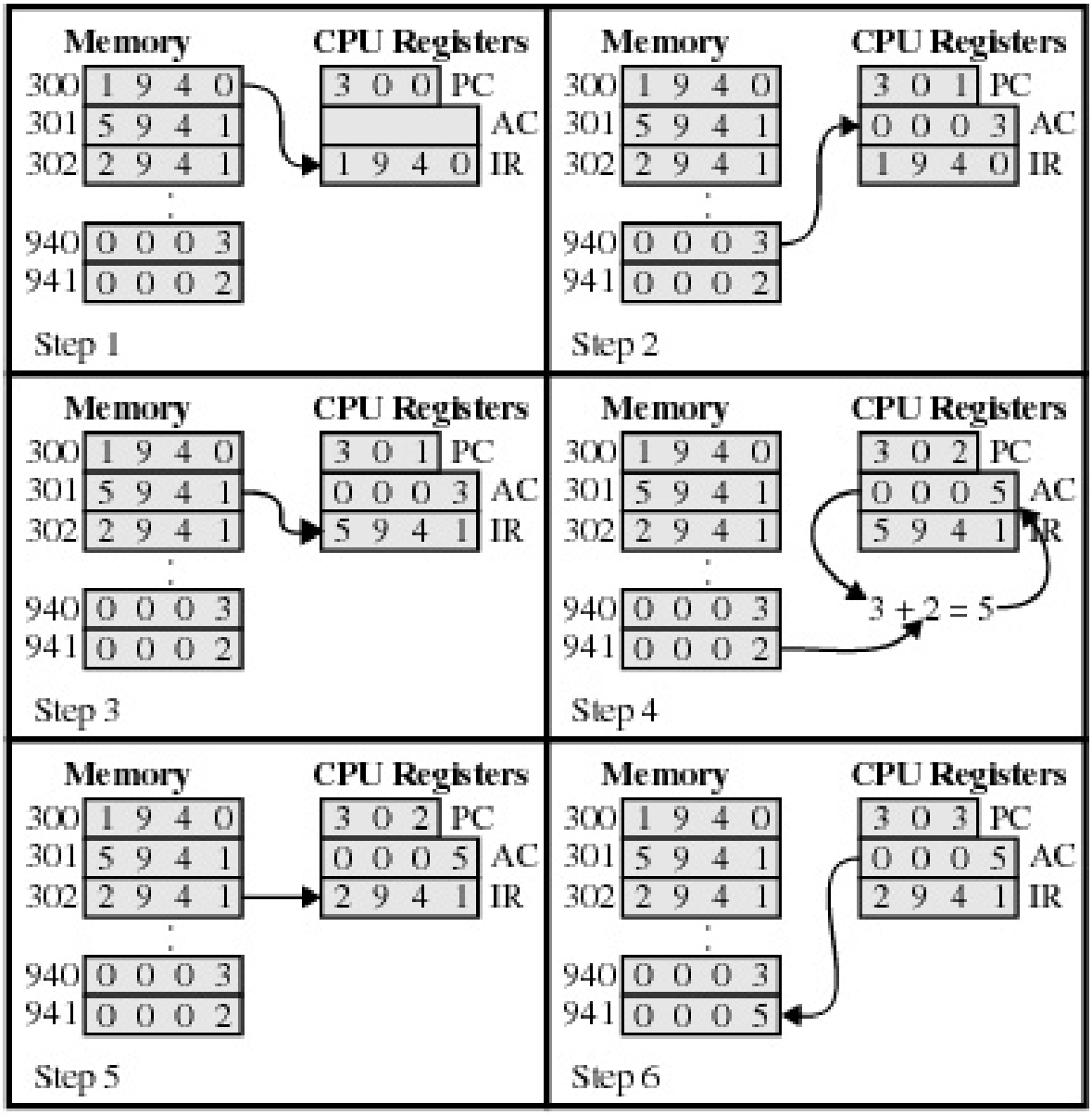
Şekil. Bir komut saykılı

C. Komut kayıtçısı

Hafızadan alınan komutlar, komut kayıtçısına (IR) yerleştirilir. Komut türleri:

- İşlemci – Hafıza : İşlemci ve hafıza arasında veri transferini sağlarlar.
- İşlemci - G/Ç : Bir çevresel birim ile veri transferi yapılamasını sağlarlar.
- Veri İşleme : Veri üzerinde aritmetik ve lojik işlemler yapılmasıdır.
- Kontrol : Yürütüm sırasını değiştirirler.

D. Program yürütüm örnekleri



E. Doğrudan hafıza erişimi (Direct Memory Access, DMA)

Hafıza ile G/Ç birimleri arasında doğrudan veri değişimlerinin yapılmasıdır. İşleyiş tarzı:

- İşlemci G/Ç modülünü hafızaya yazma veya hafızadan okuma ile yetkilendirir.
- Değişim esnasında işlemci sorumluluğu devreder.
- Bu değişim esnasında, işlemci diğer işleri yapabilir.

F. Kesmeler

İşlemcinin, normal yürütüm sırasını değiştirmek ve gereksinim duyulan başka bir işi varsa onu yerine getirmek için kesme programları kullanılır. Kesme sürecinde kontrolü kesme programı devralır. Kesmeler işletim sistemlerinin genel bir parçasıdırlar. Kesmeler:

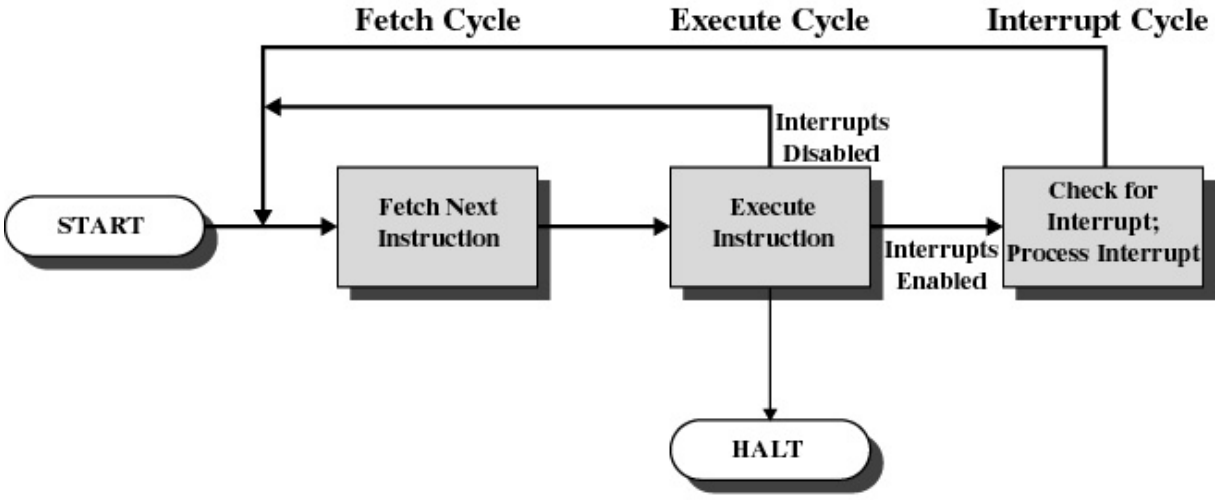
- İşlemcinin etkinliğini artırır.
- Bir G/Ç işlemi yapılırken, işlemcinin diğer komutları yürütmesine olanak verir.
- Bir sürecin geçici olarak durdurulmasına (sonradan çalıştırılmak üzere) sebep olurlar.

Kesme türleri:

- Program
  - Aritmetik taşma
  - Sıfıra bölme
  - İllegal komut yürütme
  - Kullanıcının hafıza alanının dışına çıkması
- Zamanlayıcı (Timer)
- G/Ç
- Donanım hatası

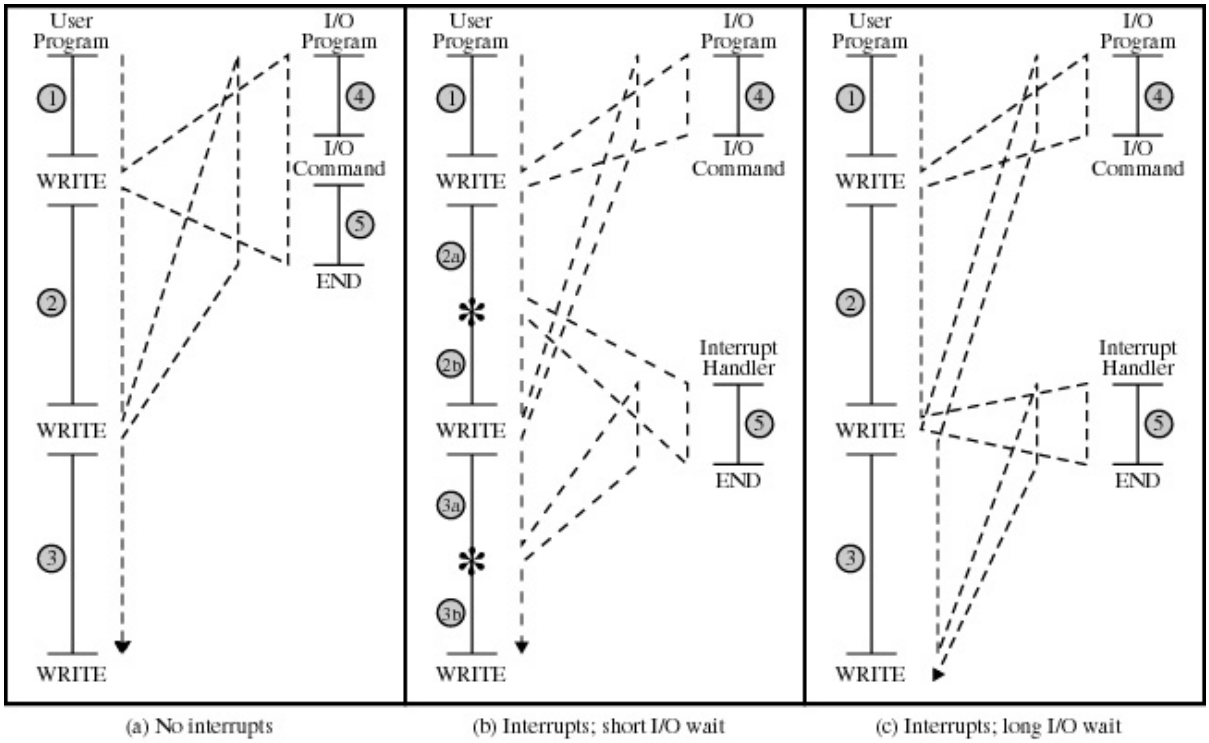
Kesme saykılının yürütümü:

- İşlemci, kesmeleri kontrol eder.
- Eğer kesme yoksa, mevcut program içinde işlemci bir sonraki komut alır.
- Eğer kesme varsa, mevcut çalışan program olduğu yerde bırakılır (çalışmasını bitirmeyi beklemeksizin) ve kesme programı idareyi ele alır.



Şekil. Kesmeli bir komut saykılı

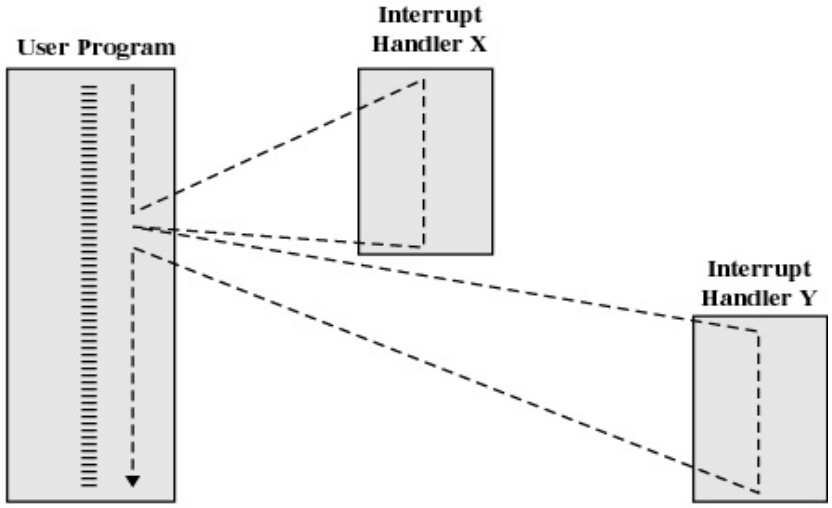
Kesme ve kesmesiz programların yürütüm akışları:



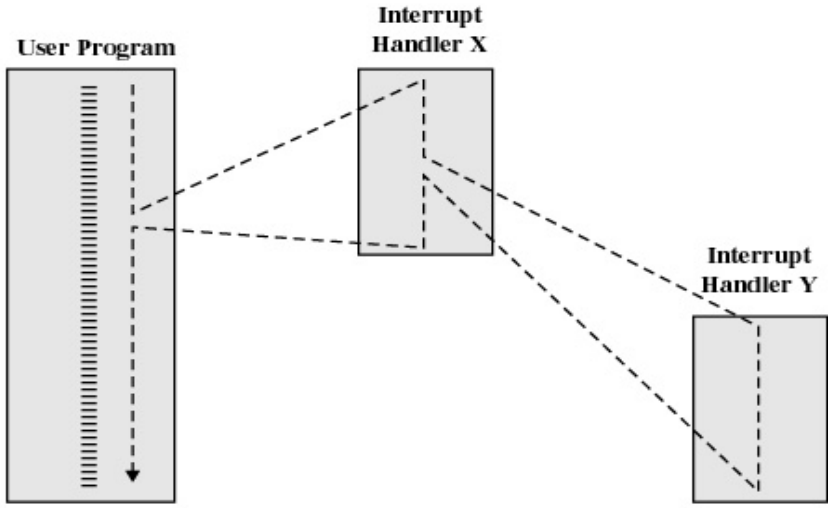
G. Çoklu Kesmeler

Bir kesme süreci başlamışken, gelebilecek diğer kesmeler kabul edilmez. İşlemci yeni gelebilecek herhangi bir kesme isteğini dikkate almaz. Kesme devre dışı kaldığında, işlemci kaldığı yerden görevini tamamlayabilir. İşlemci kesmelere izin verinceye kadar kesmeler devreye girmezler. Kesme kullanıcısının programı rutini tamamlandıktan sonra, işlemci ilave kesmeleri kontrol eder.

En yüksek öncelikli (imtiyazlı) kesme, düşük imtiyazlı kesmenin beklemesine neden olur. Yada düşük öncelikli kesme devrede ise kesilmesine neden olur.



(a) Sequential interrupt processing



(b) Nested interrupt processing

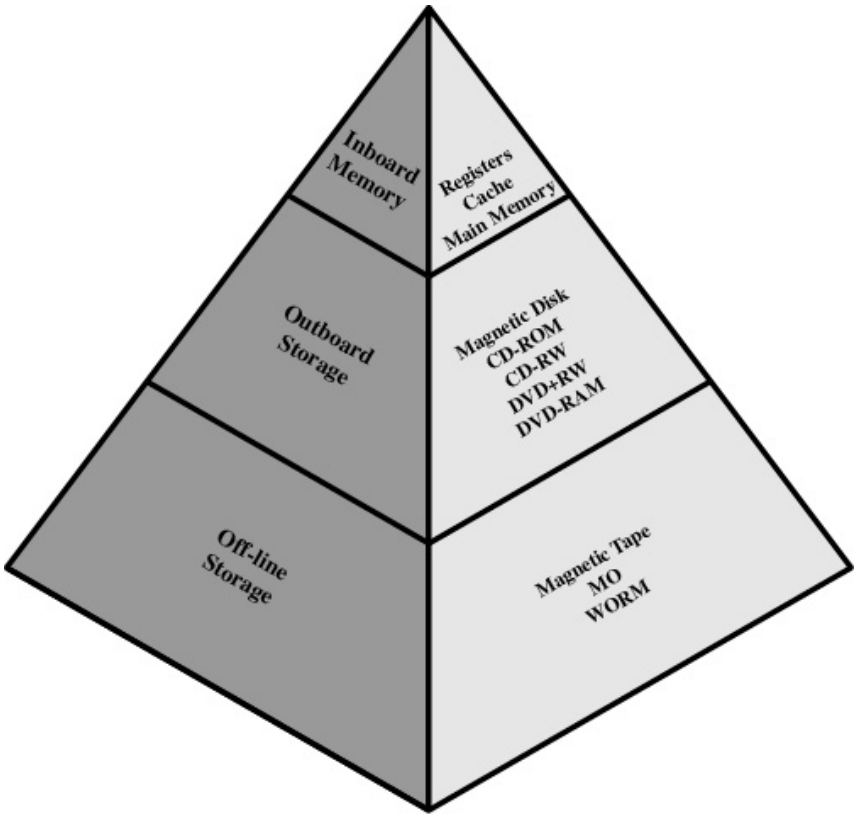
H. Çoklu programlama

İşlemci yürütmek için birden çok programa sahiptir. Ardışıl programlar onların öncelik durumuna ve G/Ç için bekleyip beklemeyeceği durumuna bağlı olarak yürütülür. Bir kesme programı tamamlandıktan sonra, kontrol kesme anında çalışan programa devredilmeyebilir (kesme programı öncelikleri değiştirebilir).

I. Hafıza hiyerarşisi

Hafıza hiyerarşisinin getirdiği avantajlar:

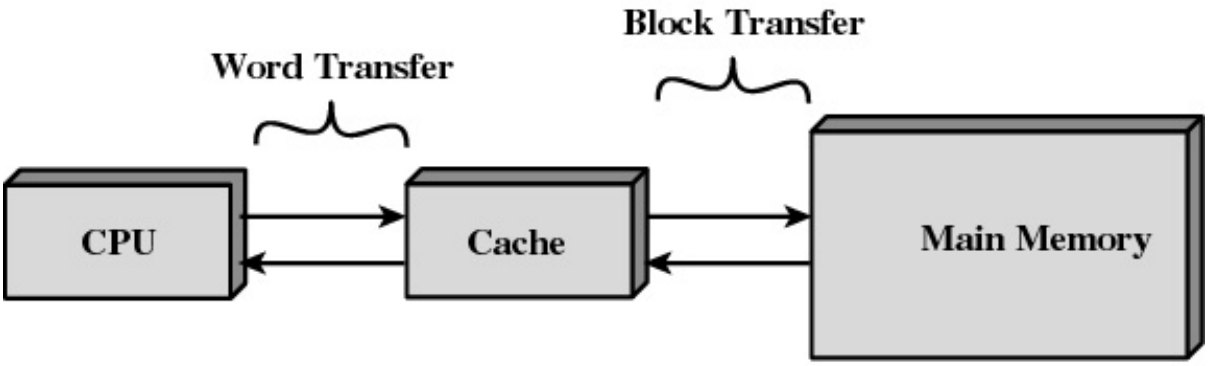
- Bit başına maliyet düşer.
- Kapasite artar.
- Zaman erişimi artar.
- Mikroişlemci tarafından hafızaya erişim sıklığı düşer.
  - Yerellikten söz edilebilir.



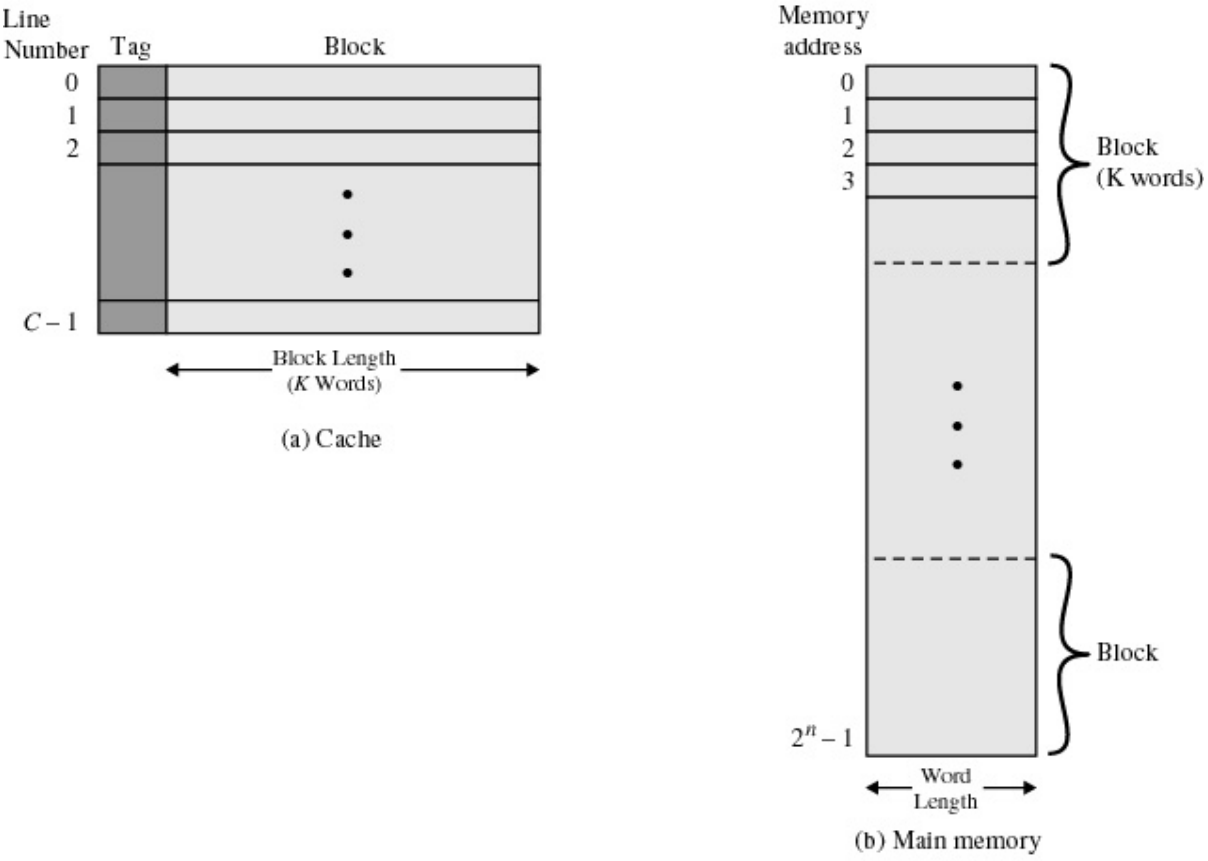
J. Ön-bellek (Cache)

Geçici olarak bilgilerin tutulmasına yönelik kullanılan ana hafızanın bir kısmıdır. Diskteki sık kullanılan kümelerin tutar. Veriye disk ten yavaş ulaşmak yerine yazılım ön belleğinden daha hızlı ulaşılabilir. Özellikleri:

- İşletim sistemine görünmez
- İşlemci hızı hafıza hızından çok daha fazladır. Önbellek kullanımı ile hafıza kullanım hızı artar.
- İşlemci ilk önce ön belleği kontrol eder; gerek duyulan bilginin hafıza bloğu ön-bellekte yoksa, ön-belleğe taşınır.
- Önbellek kapasite büyüklüğü, performans üzerinde önemli bir etkiye sahiptirler. Daha geniş blok büyüklüğü, aranılan yeni bilginin ön-bellekte bulunma ihtimalini artırır.



(a)

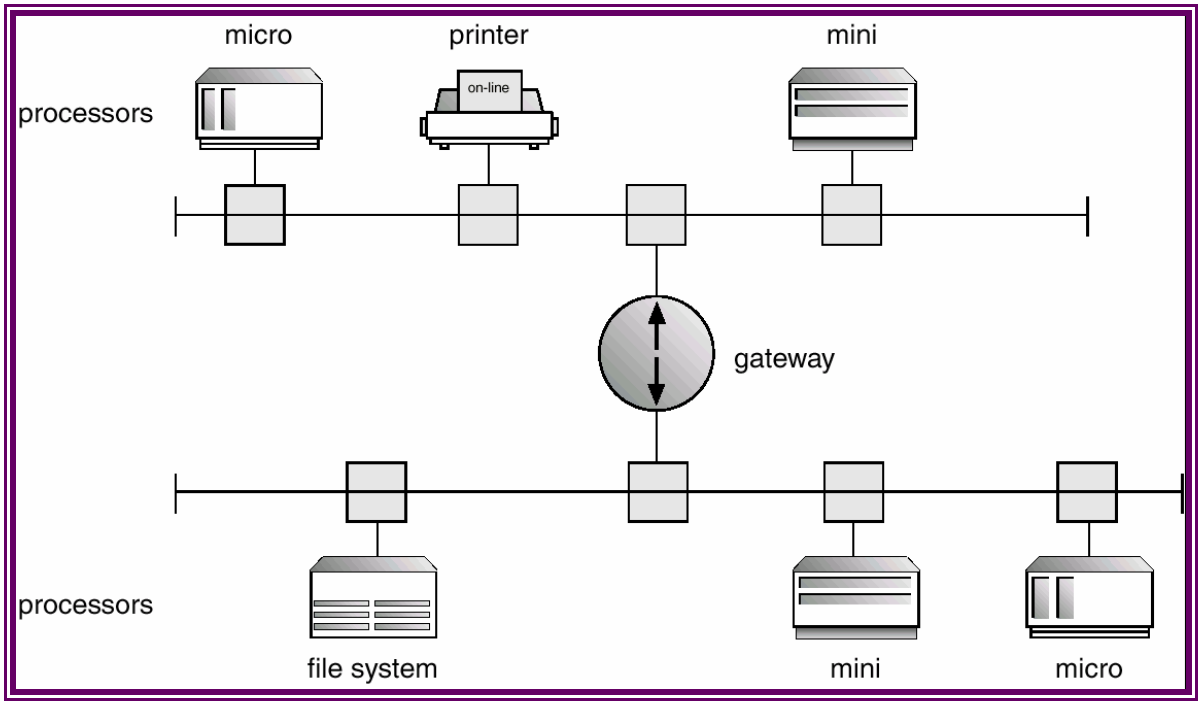


(b)

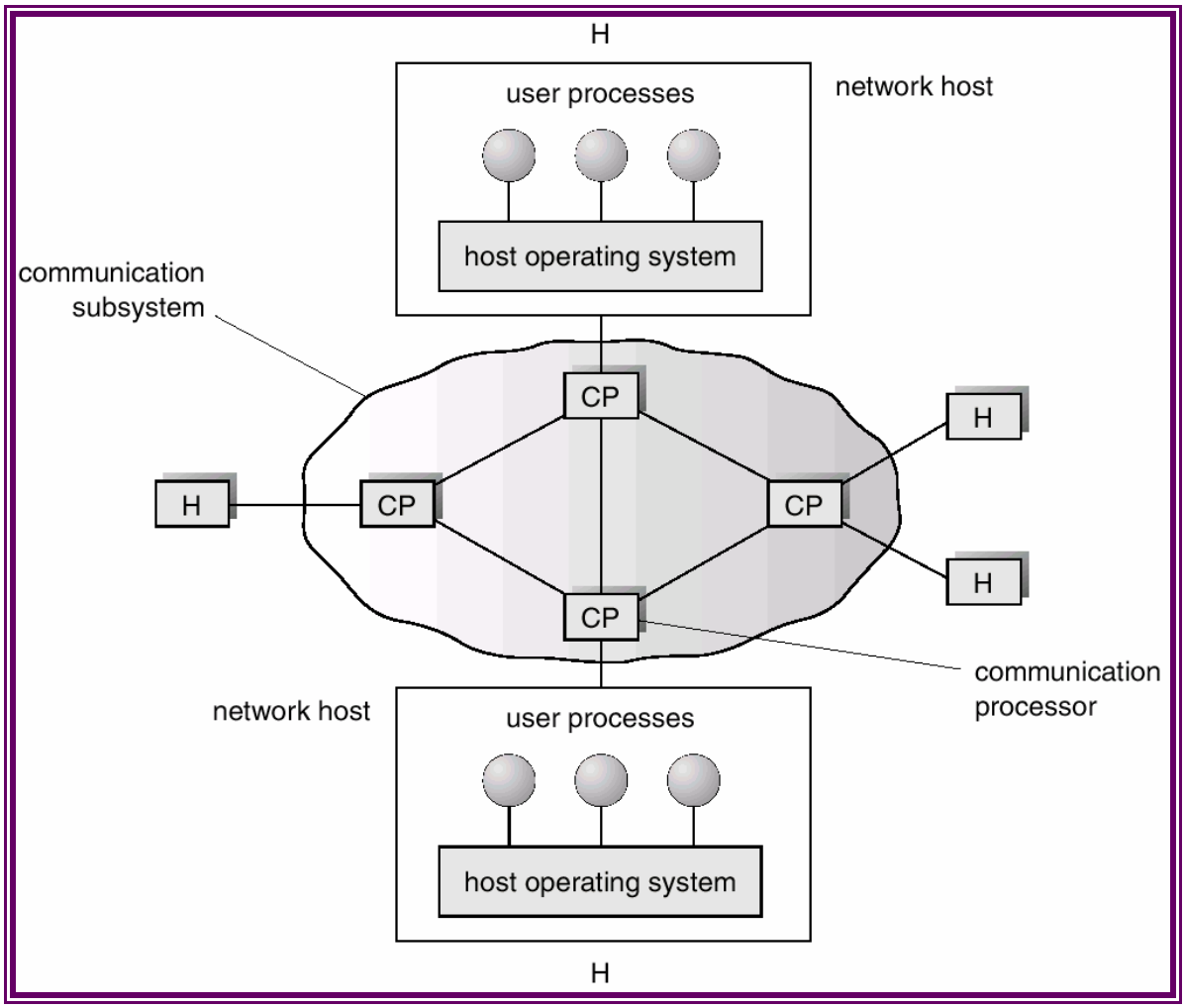
Şekil. Ön bellek kullanımı

1.3. Bilgisayar Sistemlerinde Ağ Yapısı

- Yerel alan ağları (LAN)
- Geniş alan ağları (WAN)



Şekil. Yerel alan ağ yapısı



Şekil. Geniş alan ağ yapısı

2. İŞLETİM SİSTEMLERİNE BAKIŞ

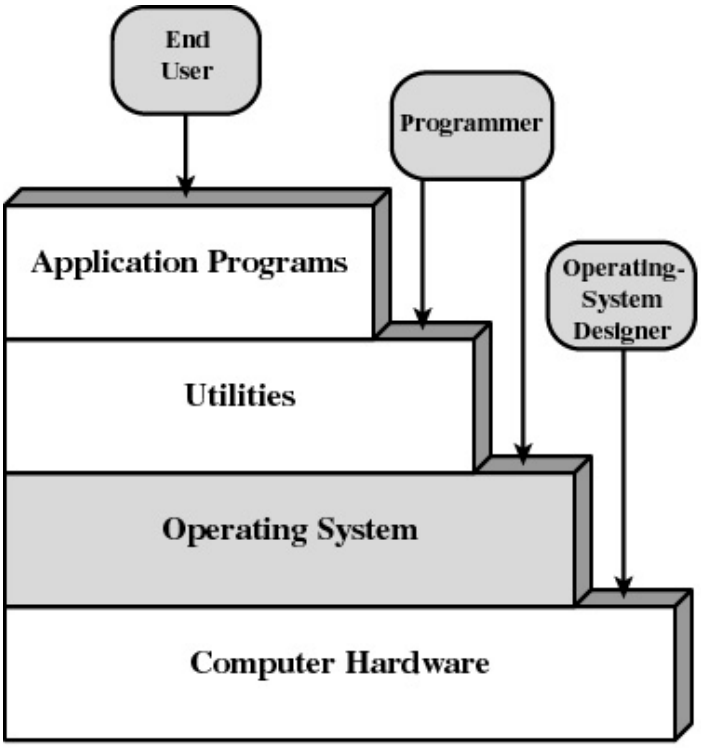
İşletim sistemi kullanıcıyla bilgisayar donanımı arasında iletişim sağlayan programdır. Programın amacı kullanıcı programlarını çalıştırmak için kullanıcıya ortam oluşturmaktır. İşletim sistemi bilgisayarı, yazılım ve donanımlarını kullanmak için elverişli duruma getirir ve donanımı etkili kullanır. Buna göre işletim sistemlerinin başlıca amaçları:

1. Kullanıcı programlarını çalıştırılması
2. Kullanıcı problemlerinin çözümünün kolaylaştırılması
3. Bilgisayar sisteminin kullanımını daha elverişli hale getirilmesi
4. Bilgisayar kaynaklarının verimli bir şekilde kullanılması (Çok kullanıcılı sistemlerde önem kazanır)

- İşletim sistemi, donanımla yazılım arasında bir yönetici arayüzü olarak görev yapar. Her bir donanım birimi, tüm kaynakları erişip yönetebilmektedir.

- Çalışma sırasında oluşan hataların ve çakışmaların önlenmesi işletim sisteminin görevidir.

- İşletim sistemleri bilgisayar kullanıcısı ile donanım arasında çalışan bir yazılımdır. Amacı kullanıcının programlarını çalıştırabilmesi için ortam sağlamaktır.



Şekil: İşletim sisteminin konumu

- İşletim sistemi faklı kullanıcıların uygulama programlarının donanım kaynaklarını kullanımını kontrol eder ve koordine eder.

- Bilgisayar kaynaklarının dağıtımı için şu problemlerin çözümü gerektirir:
  - Ana işlem biriminin zamana göre paylaşımı
  - Disk alanının yönetilmesi
  - Giriş/Çıkış aygıtlarının yönetimi
- İşletim Sistemi Tarafından Sağlanan Hizmetler:
  - Program geliştirme
  - Editörler ve Debuggerlar
  - Program yürütme
  - G/Ç sürücülerine erişim
  - Dosyalara erişimi kontrol etme
  - Hata denetimi ve cevaplama
    - Dahili ve harici donanım hataları
      - Hafıza hataları
      - Sürücü arızaları
    - Yazılım hataları
      - Aritmetik taşma
      - Yasaklanmış hafıza bölgelerine erişim
    - İşletim sisteminin, uygulama isteklerini yerine getirememesi
  - Muhasebe (hesaplama)
    - İstatistik toplama
    - Kullanıcı hesaplarını sağlama
- İşletim sistemleri, sıradan bilgisayar yazılımları gibi aynı tarz fonksiyonlara sahiptirler. Örn: Programları yürütür.

2.1. İşletim Sistemlerinin Sınıflandırılması

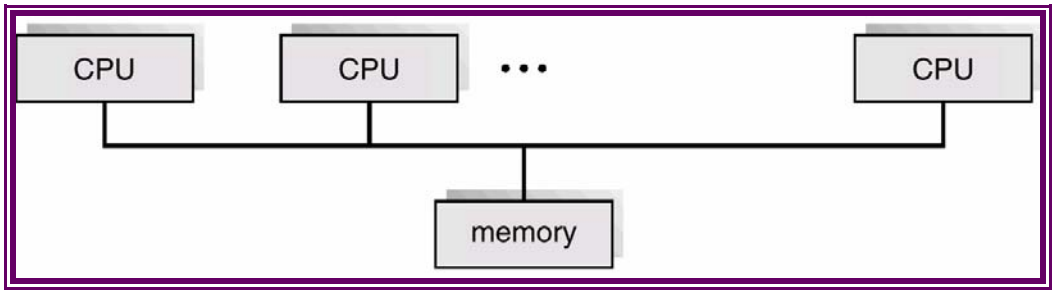
- **Büyük Bilgisayarlar için Sistemler**
  - Basit toplu işlem sistemleri
  - Benzer işlerin toplu işlenmesi sonucu işlem zamanının kısalması
  - Otomatik iş ardışıklığı: bir işten diğerine otomatik geçmek (İlk basit işletim sistemidir).
  - Kalıcı monitör

- Monitör'de başlangıç denetimi yapılır.
- Denetim, yapılacak işe devredilir
- İş bittikte denetim monitöre tekrar devredilir.

➢ **Masaüstü Sistemler**

- *Kişisel bilgisayarlar* – tek kullanıcıya ayrılmış bilgisayar sistemi.
- G/Ç aygıtları – klavye, fare, monitör, yazıcı.
- Kullanma rahatlığı sağlar.
- Birkaç farklı işletim sistemleri çalışabilir (Windows, MacOS, UNIX, Linux)

➢ **Çok İşlemcili Sistemler**



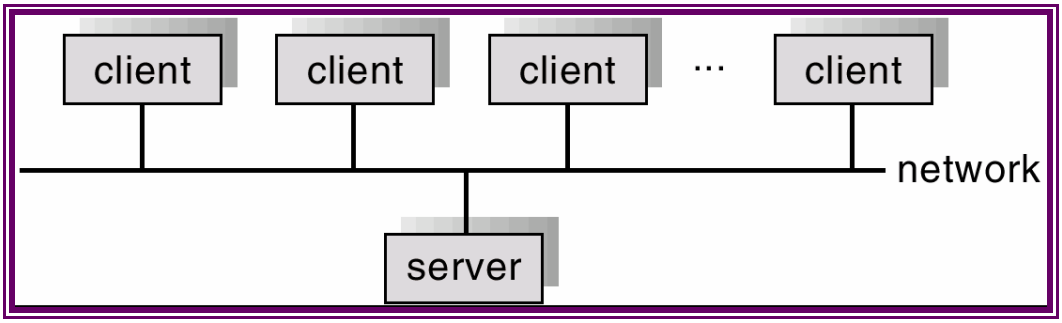
Şekil: Çok işlemcili mimari

- Çok işlemcili sistemler – birden fazla AİB'nin yakın iletişimde bulunduğu sistemlerdir.
- *Sıkıca birleştirilmiş sistem* – işlemciler belleği ve saati paylaşıyorlar; İletişim, genelde ortak bellek aracılığıyla gerçekleştiriliyor.
- Paralel sistemlerin üstünlükleri:
  - Yüksek işlem yeteneği
  - Yüksek güvenilirlik
- Simetrik çoklu işlem (Symmetric multiprocessing -SMP)
  - Her işlemci işletim sisteminin aynı kopyasını çalıştırır.
  - Başarım düşmeden, çoklu işlemci yapısı çalışabilir.
  - Pek çok işletim sistemi SMP'yi destekliyor
- Simetrik olmayan çoklu işlem (Asymmetric multiprocessing)
  - Her işlemci özel bir probleme tahsis edilir; ana işlemci işleri planlaştırır ve diğer işlemciler arasında dağıtır
  - Genelde, çok büyük sistemler için kullanılmaktadır.

➢ **Dağıtık Sistemler**

- İşlem, birkaç fiziki işlemci arasında dağıtılır.

- *Zayıf birleştirilmiş sistem* – her işlemcinin kendi yerel belleği bulunur; işlemciler birbirleriyle yüksek hızlı ana iletişim yolları üzerinden veya telefon hatları gibi çeşitli iletişim hatlarıyla iletişim kurarlar.
- Dağıtık sistemlerin üstünlükleri:
  - Kaynakların ortaklaşa kullanımı
  - İşlem hızının yükselmesi - yükün paylaşımı
  - Güvenilirlik
  - İletişim
- Ağ yapısı gerektirmektedir: Yerel alan ağları veya Geniş alan ağları
- Ağ için, ya istemci-sunucu, yada eşit bağlantı (peer-to-peer) modeli kullanılabilir.



Şekil: İstemci-Sunucu Modelinin Genel Yapısı

- **Kümeleşmiş Sistemler**
  - Kümeleşme, bellek alanını ortak kullanmaya izin veriyor.
  - Yüksek güvenilirliği sağlıyor.
  - *Asimetrik kümeleşme*: bir sunucu uygulamayı çalıştırdığı zaman, diğer sunucular yedek kalıyor.
  - *Simetrik kümeleşme*: tüm ana makineler uygulamayı çalıştırıyor.
- **Gerçek Zaman Sistemler**
  - Bilimsel denemelerde, fabrikalarda üretimin otomatik denetiminde, tıbbi görüntü sitemleri gibi uygulamalarda kontrol amacı ile sıkça kullanılmaktadır.
  - Önemli özelliği, iyi tanımlanmış belirli zaman kısıtlamalarının bulunmasıdır.
  - Gerçek zaman sistemleri ya sert (hard), yada hafif (soft) gerçek zamanlı olabilir:
    - Sert gerçek zaman sistemleri:
      - İkinci bellek sınırlıdır veya yoktur, veriler kısa süreli bellekte, veya sabit bellekte saklanır.
      - Zaman paylaşımlı çalışmalar genel amaçlı işletim sistemleri tarafından desteklenmiyor.

- Hafif gerçek zaman sistemleri:
  - Sanayi robotlarının denetiminde sınırlı kullanılmaktadır.
  - Gelişmiş işletim sistemlerinin özelliklerini gerektiren uygulamalarda yararlıdır.

- **"El" Sistemleri**
  - Kişisel rakamlı yardımcılar (Personal Digital Assistant (PDA))
  - Hücresel telefonlar (Cellular telephones)
  - Özellikleri:
    - Sınırlı bellek
    - Düşük hızlı işlemciler
    - Küçük ekran.
- **İşlem Ortamları**
  - Web tabanlı işlemler
  - Gömülmüş işlemler

2.2. İşletim Sisteminin Gelişimi

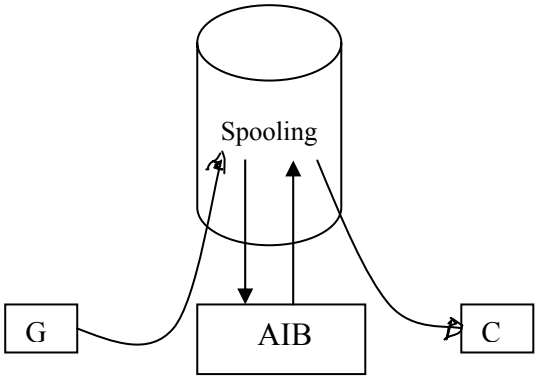
İşletim sistemlerinin gelişim süreci aşağıdaki aşamalardan geçmiştir:

- Basit toplu işletim sistemleri
- Zaman paylaşım tabanlı çoklu programlama
- Kişisel bilgisayarların işletim sistemleri
- Paralel sistemler
- Dağıtık sistemler (Paylaşılmış sistemler)
- Gerçek zaman sistemleri

2.2.1. Basit toplu işletim sistemleri (Batch Systems)

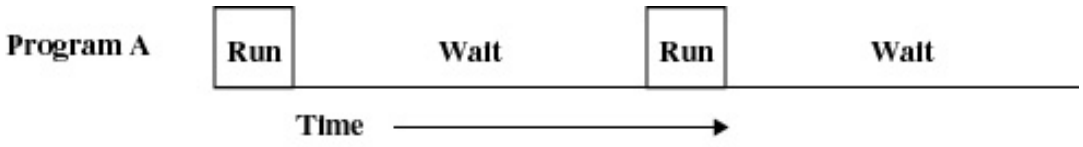
İlk bilgisayarların genelde işletim sistemleri yoktu. İşletim sistemleri bilgisayarların donanımları geliştikten sonra ortaya çıkmaya başlamıştır.

Bu tür sistemlerde işlemcinin zamanının büyük kısmı giriş çıkış işlemlerini beklemekle geçer. Çünkü, G/Ç aygıtlarının hızı işlemcinin hızından düşük olduğu için işlemci bu aygıtları beklemek zorundadır. Bunu önlemek için Spooling tekniği (Ana işlem birimini giriş çıkış biriminden ayırma işlemi) kullanılır. Böylece CPU gelen girişi ya doğrudan, yada işleyip çıkışa aktarır.

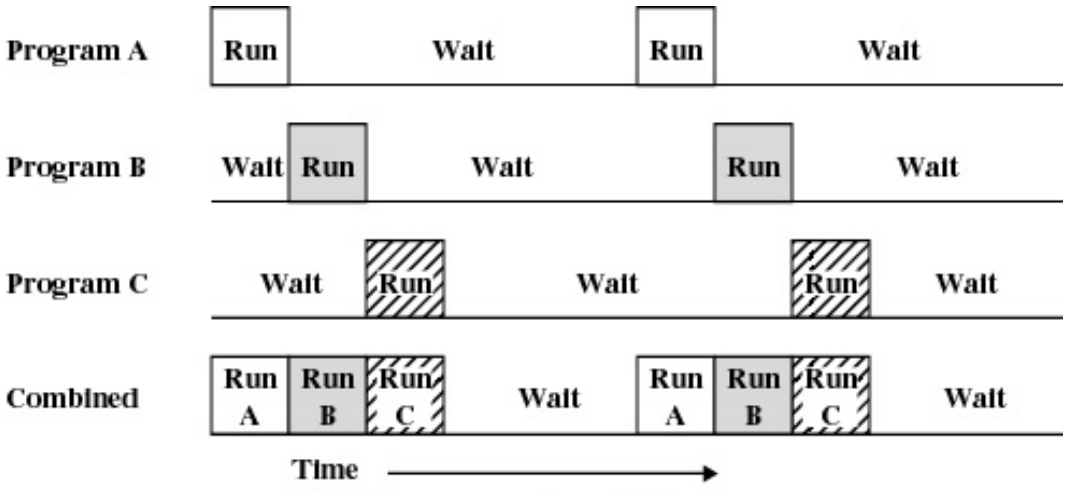


2.2.2.Zaman Paylaşım Tabanlı Çoklu Programlama

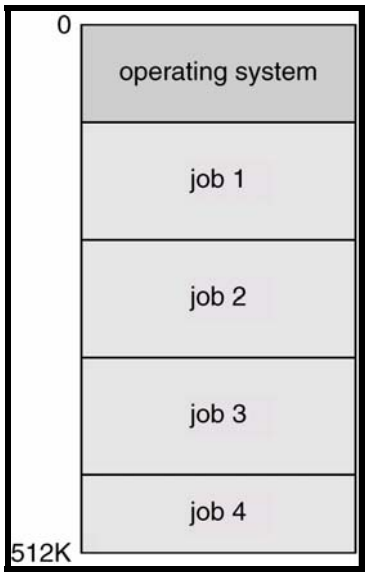
Aynı zaman diliminde bellekte birkaç iş bulunur. İşlerin hangisinin ve hangi ardışıklıkla çalıştırılacağının belirlenmesi için iş planlanması (job schduling) yapılır. Kaynaklar paylaştırılarak kullanılır. Bu sistemlerde AİB (CPU)'nun çalışma zamanı küçük zaman aralıklarına bölünür, ve bu zaman aralıkları ardışık olarak programlar tarafından kullanılır. Onun için ayrılan zaman aralığı bittiğinde bu iş kuyruğa geçer ve yeniden işlenmek için bekler.



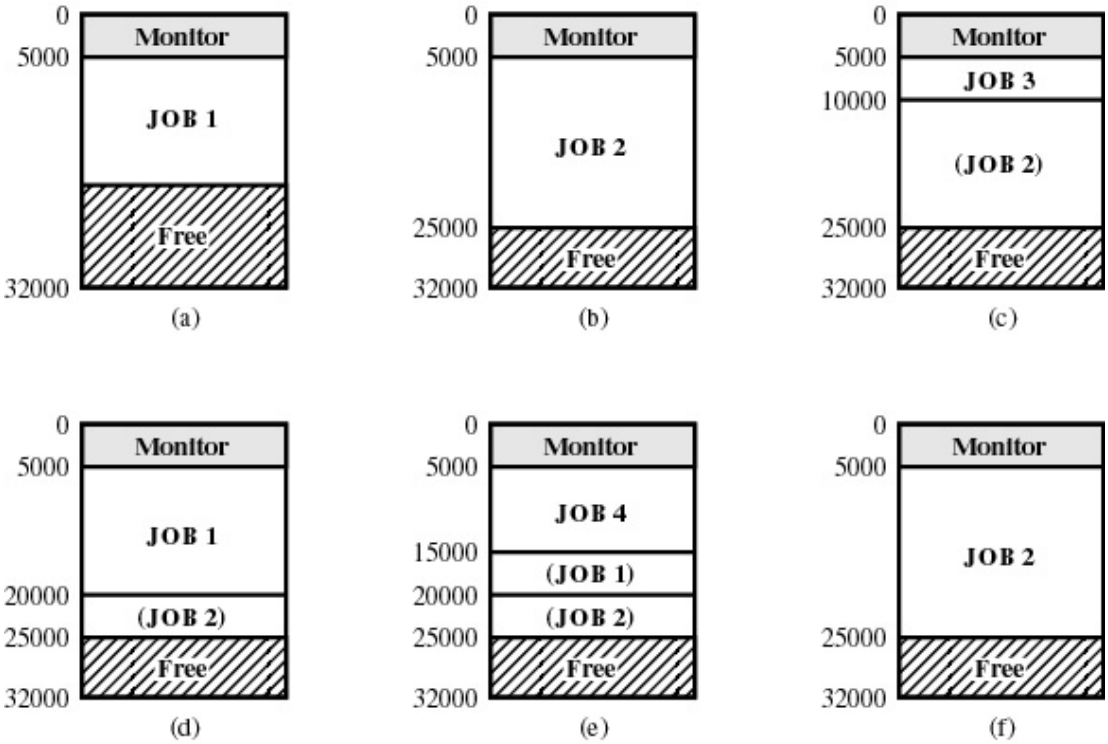
Şekil: Tek program; *İşlemci G/Ç biriminden gelebilecek olanı bekler*



Şekil: Çoklu programlama; *Bir iş G/Ç birimini beklediği zaman, işlemci başka bir işe yönelebilir.*



Şekil: Yapılacak işler (operation system = monitor)



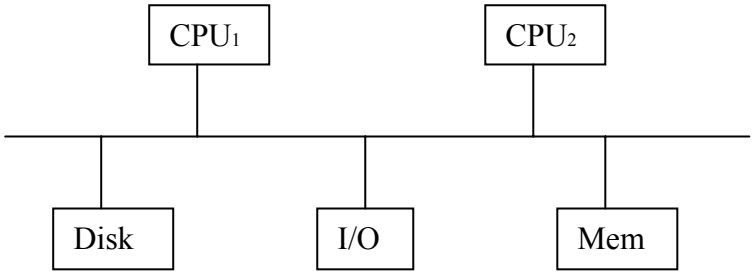
Şekil: Belleğin işler arasında paylaşımı; *işletim sistemi belleği birkaç iş arasında paylaştırabilmektedir.*

2.2.3.Kişisel bilgisayarların işletim sistemleri

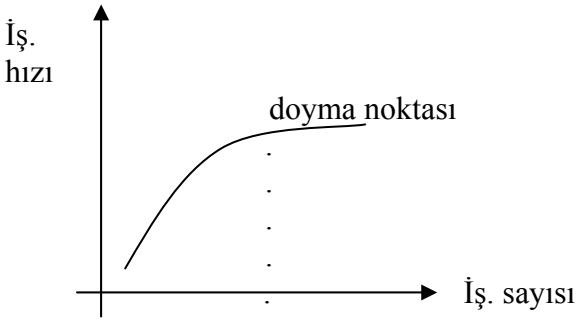
En önemli özelliği kullanıcı kolaylığını sağlamasıdır. Çoklu ortamda çalışması ve güvenilir olması önemlidir. Tek kullanıcılı sistem olduğu için kaynakların paylaşımı mekanizmasına gerek yoktur.

2.2.4.Paralel işletim sistemleri

Paralel sistemlerde birden fazla işlemci olur. Bu işlemciler bazı kaynakları (ana veri yolu, saat (cp), bazen de ana belleği, G/Ç kaynakları) ortak kullanırlar. Böyle sistemlere **güçlü bağlı sistem** denir.



Çok işlemcili sistemlerin oluşturulma nedeni işlem hızının ve güvenirliğinin arttırılmasıdır. İşlemci sayısı arttıkça işlem hızı artar ancak bir noktada sabitlenir.



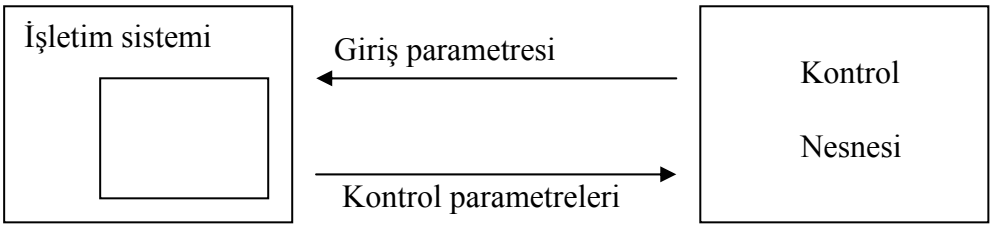
2.2.5.Dağıtık sistemler

Bu sistemlere **ağ sistemler** de denir. Burada herbir bilgisayar sisteminin kendi kaynakları, işlemcisi, I/O aygıtları ve fonksiyonları bulunur. Bunlara **zayıf bağlı sistemler** denir. Ağ ortamında çalışan bilgisayarlardan oluşur. Bilgisayarlar arasında iletişim, ağ ortamı ile gerçekleşir. Bu sistemlerin oluşturulma nedenleri;

- Kaynakların paylaşımı
- Işlem hızının yükseltilmesi
- Güvenilirlik
- Yeni iletişim olanaklarının sağlanması

2.2.6.Gerçek zaman sistemleri

Bu sistemlerde işlemlere uygulama alanlarına bağlı olarak zaman sınırlamaları getirilir. Böyle sistemler sanayi kontrol sistemlerinde, bilimsel araştırmalarda kullanılır. Bu tür sistemlerde çok gerekli olmamakla beraber disk belleği kullanılır. Daha fazla ROM (sabit bellek) kullanılır.



3. İŞLETİM SİSTEMLERİNİN BİLEŞENLERİ

Birbirinden farklı çeşitli işletim sistemleri bulunmasına rağmen, bunlar arasında ortak yapı özellikleri ve bileşenleri vardır. Fonksiyonel bakımdan işletim sistemlerinin bileşenleri:

- Görev yönetimi (Process manager)
- Ana bellek yönetimi (Memory manager)
- Kütük yönetimi (File manager)
- Disk (2. bellek) yönetimi (Second Storage)
- Ağ üzerinde çalışma (Ağ fonksiyonları yönetimi: Networking)
- Sistem koruması
- Komut derleyici sistemler

Bu bileşenler aşağıda kısaca açıklanmaktadır. Daha ileri ki bölümlerde, her biri ayrıntılı bir şekilde verilecektir.

3.1.Görev Yönetimi

Genel halde **görev** çalışır durumda olan programdır. Bu program pasif bir varlıktır. Görev ise aktiftir. Görevler, AİB zamanı, bellek, kütükler, G/Ç aygıt kaynakları gerektirir. Bu kaynaklar ilgili göreve onun oluştuğu anda aktarılır. Görev sonlandıktan veya kesildikten sonra, işletim sistemi bu kaynakları görevden alır ve bir diğer görevle arasında paylaştırır. Görevin çalışması ardışık işlemlerdir. Her bir zaman diliminde ana işlem birimi tarafından görevi bir komut çalıştırır. İki görev aynı programa ait olsa da ayrı görevler gibi bakılır ve çalıştırılır. Görev işletim sisteminde bir iş birimidir. İşletim sistemi, farklı görevlerin aynı zamanda çalışmasını ve kaynakların ortak kullanımını kontrol eder.

İşletim sistemi, görev yönetiminde aşağıdaki işleri icra eder:

- Kullanıcı ve sistem birimlerinin oluşturulması, silinmesi
- Görevlerin oluşturulması, durdurulması ve yeniden çalıştırılması
- Görevlerin zamana uyum sağlama mekanizmasının gerçekleştirilmesi
- Görevler arasında iletişim sağlanması
- Kilitlenmelerin yönetimi

İki görev aynı zamanda çalıştırıldığında aynı kaynakları kullanmak isteyebilirler. Bu durumda, zamana uyum sağlama mekanizması kullanılır. (Örneğin, araçların trafikte yeşil yanıncaya kadar bekleyip sonra geçmesi gibi)

Ortak kaynakların kullanımında her iki görev de bekleme durumuna geçerse (bu sonsuz döngü oluşturur) **kilitlenme** olur. Yani, biri diğerinin sonucunu beklerken, diğeri de ötekinin sonucunu bekler.

3.2.Ana Belleğin Yönetimi

Bellek her birisinin kendi adresi olan baytlar veya kelimelerden oluşan büyük dizidir. Bellek AİB ve G/Ç aygıtlarının paylaştığı ve hızlı erişilebilen bir veri ambarıdır. Ana Bellek geçici bellektir.

Ana işlem biriminin (AİB) verimliliğini ve kullanıcı sorgularına yanıt verme hızını yükseltmek için, bellekte aynı zamanda birden fazla program saklanması gereklidir. Bu halde belleğin yönetimi için farklı yaklaşımlar ve algoritmalar kullanılmaktadır. Bu yaklaşımların seçimi, birçok farklı özellikte sistemin donanım tasarımına bağlıdır. Bellek yönetimiyle ilgili her bir algoritma özel donanım desteği gerektirir.

Bellek yönetimi ile ilgili, işletim sistem aşağıdakilerden sorumludur:

- Belleğin şimdiki durumda, hangi kısmının ve kim tarafından kullanıldığı hakkında bilginin elde edilmesi
- Bellek alanı boş olduğunda bu alana hangi görevlerin yüklenmesi hakkında, karar oluşturulması

3.3.Kütük Yönetimi

Kütük, oluşturucusu tarafından belirlenmiş ilişkili bilgiler topluluğudur. Çoğunlukla, kütükler kaynak ve nesne türünde programları ve verileri ifade ederler.

Görev ve bellek yönetimi kullanıcı için sanaldır. Kütük yönetimi ise kullanıcıyı görsel olarak ilgilendirir. Örneğin; diske yazılan kütükle, CD'ye yazılan kütük birbirinden farklıdır. Ancak İşletim Sistem bu ayrıntıları saklar ve aynıymış gibi görüntülenmesini sağlar. Yani, bilgi istenilen kütükte amacına uygun bir şekilde depolanır.

Bilgisayar sisteminin kullanımı kolaylaştırmak için İşletim Sistemi veri depolarını aynı biçimli mantıksal görünümde olmasını sağlar. Mantıksal depolama (kaydetme, saklama) birimini tanımlamak için uygun aygıtların fiziki nitelikleri boyutlanır (sektör, track,.vs.). Bu mantıksal depolama (kayıt) birimine **kütük** denir. Kütüğün oluşturulması, silinmesi ve adının değiştirilmesi kütük yönetimine aittir.

Genel halde kütükler programları ve verileri ifade eder. Kütüklerin yönetimini kolaylaştırmak için dizinler oluşturulur.

İşletim Sistem kütük yönetiminde aşağıdakilerden sorumludur.

- Kütüklerin oluşturulması ve silinmesi
- Dizinlerin oluşturulması ve silinmesi
- Kütük ve dizinlerin yönetimi
- Kütüklerin disk belleğine (2.belleğe) haritalanması (kaydedilmesi)
- Kütüklerin 2. bellekte yedeklenmesi (back up)

3.4.Giriş / Çıkış (I/O) Sistemlerinin Yönetimi

Giriş/Çıkış sisteminin yönetimi, içerdiği birimlerle aşağıdaki fonksiyonları gerçekleştirir:

- Bileşenleri oluşturur. (Tampon önbellekleme sistemi)
- Ön belleğe yazıp-okuma
- Spooling işlemlerinin gerçekleştirilmesi
- Aygıt-sürücü (device driver) arayüzlerinin yönetimi
- Belirli donanım aygıtları için sürücülerin yönetimi

3.5.Disk (2. Bellek) Yönetimi

Ana belleğin, geçici ve tüm veri-programların sürekli saklanılması için çok küçük olması nedeniyle bilgisayar sistemleri ikincil belleğe ihtiyaç duymaktadırlar.

İşletim Sistem ikincil belleğe bağlı aşağıdaki girişimlerden sorumludur:

- Boş disk alanı yönetimi
- Diskin paylaşımı
- Diskin planlanması

3.6.Ağ Fonksiyonları Yönetimi

Ağ ortamı veya dağıtık sistem, ortak giriş-çıkış aygıtları, saati ve ana belleği olmayan işlemciler (bilgisayarlar) topluluğundan oluşur. Her işlemcinin kendine ait belleği bulunur. Birbirlerine iletişim hatları aracılığıyla bağlanırlar ve iletişimi protokollerle gerçekleştirirler. Ağ kullanıcıları aynı ortak kaynaklara erişebildiği gibi, kullanıcı bazında erişimde mümkündür.Bu durum:

- Hesaplama (bilgi-işleme) hızını yükseltir.
- Verilerin kullanılabilirlik alanını genişletir
- Güvenilirliği yükseltir

Burada, İşletim Sisteminin en önemli görevi ortak kaynaklara erişimi sağlamaktır.

3.7.Koruma Sistemi

İşletim Sistemlerinde koruma; koruma ve güvenlik olarak iki çeşittir. Eğer bilgisayarın birden fazla kullanıcısı varsa ve çoklu görevin birden fazla çalışmasına izin verilirse bu görevler birbirlerin girişimlerinden korunmalıdır. Bu amaçla İşletim Sistemi; kütüklere, bellek kesimlerine (segment), işlemciye ve diğer kaynaklara görevlerin kontrollü erişimini sağlar. Örneğin; görev yalnız kendisi için ayrılmış adres alanında çalışabilir yada zamanlama görevin belirlenmiş zaman içerisinde ana işlemcinin de çalışmasını sağlamalıdır. Koruma mekanizması İşletim Sistemi bileşenleri arasındaki arayüzlerinde oluşabilecek hataları önlemekle güvenliği yükseltir.

İşletim Sisteminin koruma sistemi aşağıdaki işlemlerden sorumludur:

- Erişim Kontrolü
  - Sisteme kullanıcı erişimini düzenler; İzinli ve izinsiz kullanımları ayırt edebilme.
- Bilgi Erişim Kontrolü
  - Sistem içindeki veri akışını ve verinin kullanıcılara dağıtımını düzenler; Denetimleri belirleme.
- Sertifikasyon
  - Sertifikasyonlara göre erişim ve akış kontrol performansı sağlanır; İzinsiz erişimleri engelleyebilme.

3.8.Komut Derleyicisi

Sistemin kullanıcı ve İşletim Sistemi arasında arayüz oluşturan yorumlayıcı sistem programıdır. Gelişmiş sistemlerde kullanıcı arkadaşlığı bulunan, komut derleyicilerin tasarımına önem verilir. Böyle arkadaş arayüzler, pencerelerin ve menü sistemlerinin oluşmasını destekler, görevlerin oluşturulması ve yönetimi, giriş-çıkış yönetimi, kütük sistemlerine erişim, koruma ve ağ üzerinde çalışma, bu komutlarla gerçekleştirilir.

İşletim Sisteminin Komut Derleyicisi aşağıdaki işlemlerden sorumludur:

- Pek çok komut, işletim sistemine denetim fonksiyonları amacı ile yönlendirilir. Bu komutlar aşağıdaki işlemleri gerçekleştirir:
  - Görev oluşumu ve yönetimi
  - G/Ç yönetimi
  - Bellek yönetimi
  - Ana belleğin yönetimi

- Kütük sistemlerine erişim
- Koruma
- Ağ ortamında çalışma

- Denetim fonksiyonlarını okuyan ve derleyen programa
  - Komut satırının derleyicisi veya
  - Çekirdek (Kernel) (UNIX sisteminde) denir.

- Komut derleyicisinin işlevi, komut fonksiyonunu okumak ve yürütmektir.

4. İŞLETİM SİSTEMLERİNİN YAPISI VE SUNDUĞU HİZMETLER

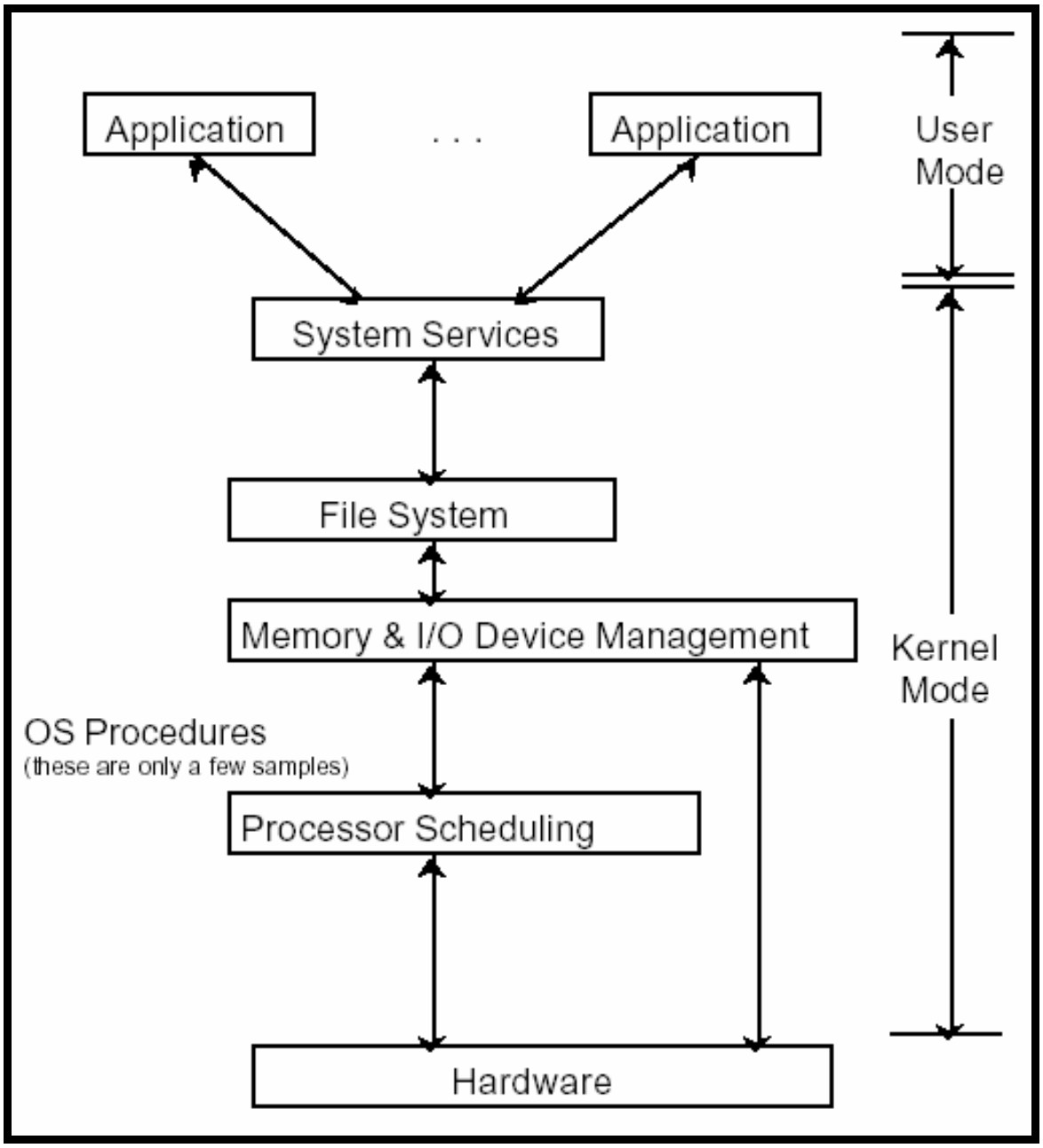
4.1. İşletim Sistemi Tasarım Hiyerarşisi

- Sistemin Tasarım Hedefleri:
  - Kullanıcıya yönelik hedefler; işletim sisteminin kullanımı ve öğrenilmesi kolay olmalı, sistem güvenilir ve hızlı çalışmalıdır.
  - Sisteme yönelik hedefler; işletim sistemi kolay tasarlanan, çalıştırılabilen, bakımı kolay, esnek, güvenilir olmalı, hatasız ve etkili çalışmalıdır.
- Geleneksel olarak assembly dilinde yazılmış olan işletim sistemleri yüksek seviyeli dillerde de yazılabilir. Yüksek dillerde:
  - Daha çabuk yazılabilir.
  - Daha az yer kaplar.
  - Anlaşılması ve çözümü kolaydır.
- İşletim sistemi yüksek seviyeli dilde yazıldığında kolaylıkla farklı bilgisayarlarda kullanılabilir.
- İşletim sistemleri her sınıf bilgisayarda çalıştırılabilecek şekilde tasarlanır.
- SYS - donanım sisteminin belirli biçimi ile ilgili bilgiyi içeren programdır.
- Booting – çekirdeğin yüklenmesi ile bilgisayarın çalışmaya başlamasıdır.
- Bootstrap program – çekirdeği belirlemek, belleğe yüklemek ve çalıştırmaya başlamak için ROM belleğinde saklanan koddur.

S.No	Katman Adı	Nesneler	Örnek İşlemler
13	Kabuk	Kullanıcı programlama ortamı	Kabuk dilindeki ifadeler
12	Kullanıcı Süreçleri	Kullanıcı Süreçleri	Çıkış, reddetmek, duraklatmak, Kaldığı yerden devam etmek
11	Dizinler	Dizinler	Oluşturma, Yok etme, bağlamak, ayırmak, araştırmak, listelemek
10	Sürücüler	Dış sürücüler; yazıcı, monitörler ve klavyeler	Açma,Kapama,Okuma,Yazma
9	Kütük sistemi	Kütükler (dosyalar)	Yok etme, açma, kapama okuma, yazma
8	Haberleşmeler	Bağlantılar	Açma, kapama, okuma, yazma
7	Sanal Hafıza	Segmentler, sayfalar	Okuma, yazma, bilgi çekme
6	Yerel ikincil bellek	Veri blokları, Sürücü kanalları	Okuma, yazma, yerleştirme, Serbest
5	Basit süreçler	Basit süreçler, hazır listeler	Erteleme, kaldığı yerden devam, bekleme, işaret etme
4	Kesme programları	Kesme programlayıcı	İstek, maskeli, maskesiz,
3	Prosedürler	Prosedürler, yığın, görüntüleme	Hedef yığın, çağırma, geri dönme
2	Komut seti	Yığın değerlendirme, Mikro-program yorumlayıcı, Tek ve dizisel veri	Yükleme, depolama, ekleme, çıkarma, dallanma
1	Elektronik devreler	Kayıtçılar, kapılar, BUS'lar	Silme, transfer, aktive etmek, tersleme

4.2. İşletim Sistemlerinin Mimarileri : Katmanlı

- Sistem çeşitli seviyelerden oluşur
- Her bir seviye ilişkisel alt fonksiyonları icra eder.
- Daha çok basit fonksiyon yürütmek için her bir seviye sonraki daha düşük seviyeye bağlıdır.
- Bu durum, bir problemin bir çok alt-problem olarak ayrıştırılmasını sağlar.

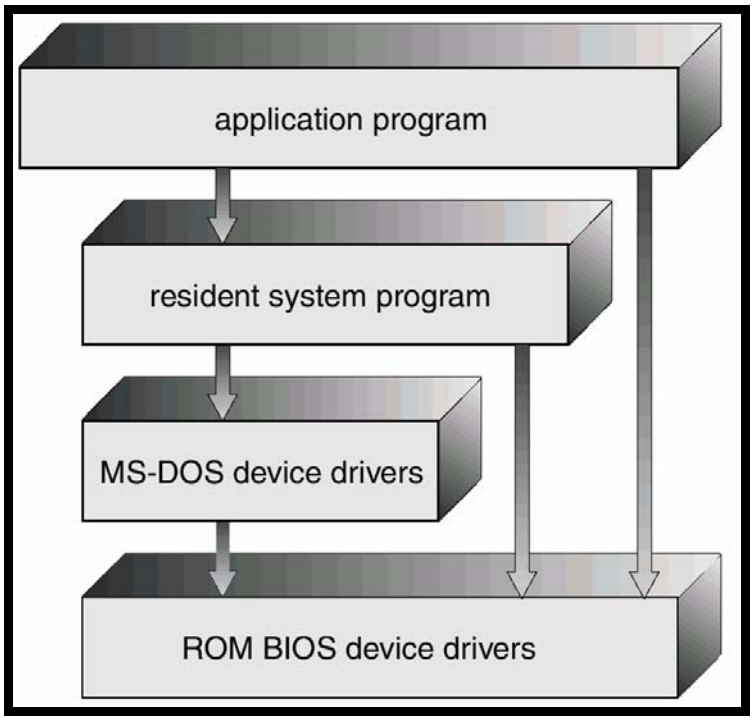


Şekil: Katman tabanlı işletim sistemlerinin yapısı

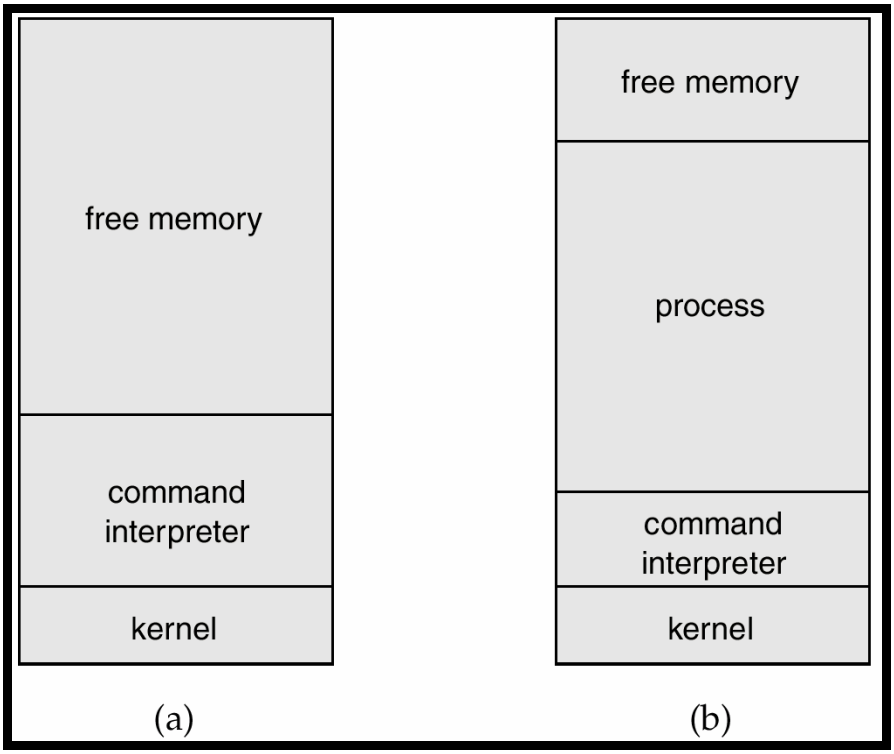
4.2.1. MS-DOS Sisteminin Yapısı

MS-DOS –küçük bellek alanında pek çok işlevin sağlanabilmesi için yazılmıştır:

- Modüllere bölünmez;
- Arayüzler ve işlev seviyeleri kesin ayrılmamıştır.

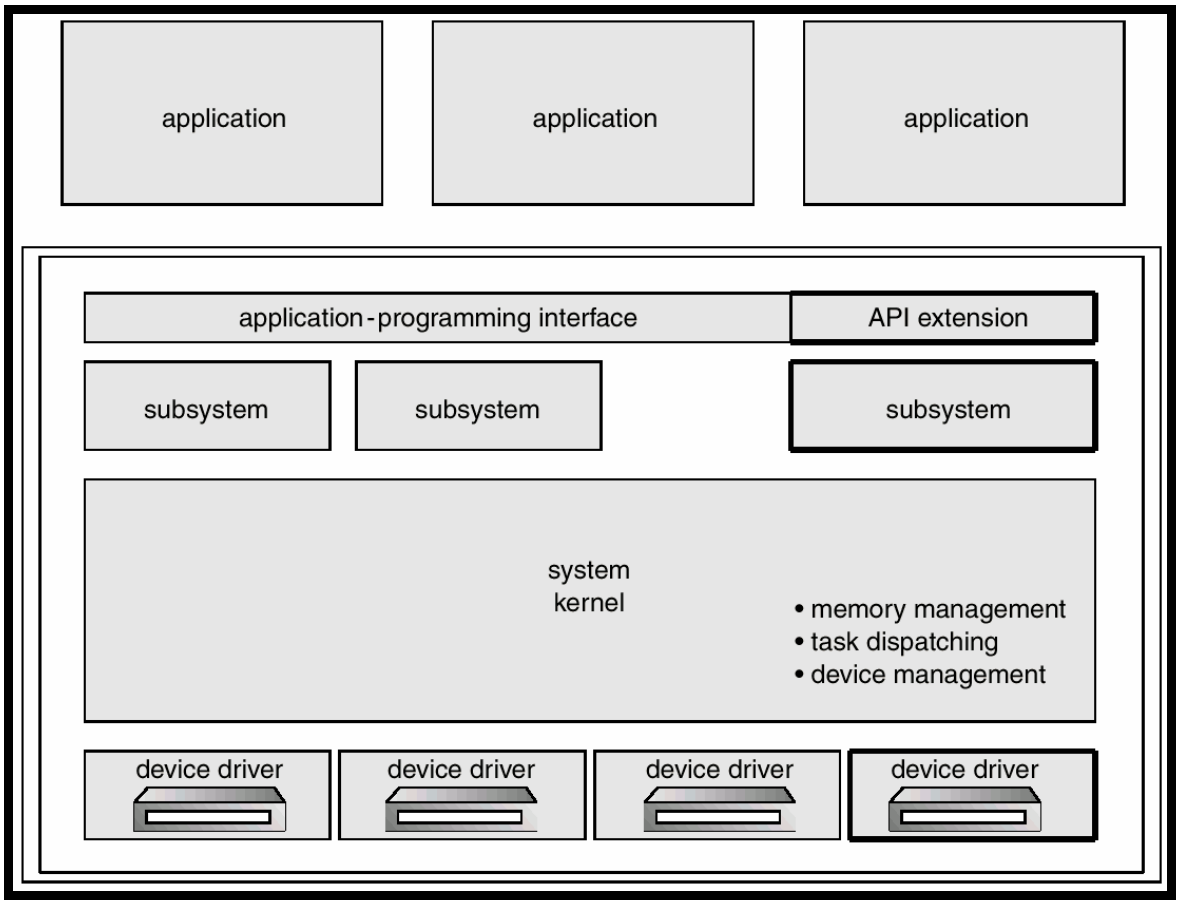


Şekil: MS-DOS Sisteminin yapısı



Şekil: MS-DOS Sisteminde; **a)** Sistemin başlangıç durumu, **b)** Programın çalışması durumu

4.2.2. OS/2 Sistemini Yapısı

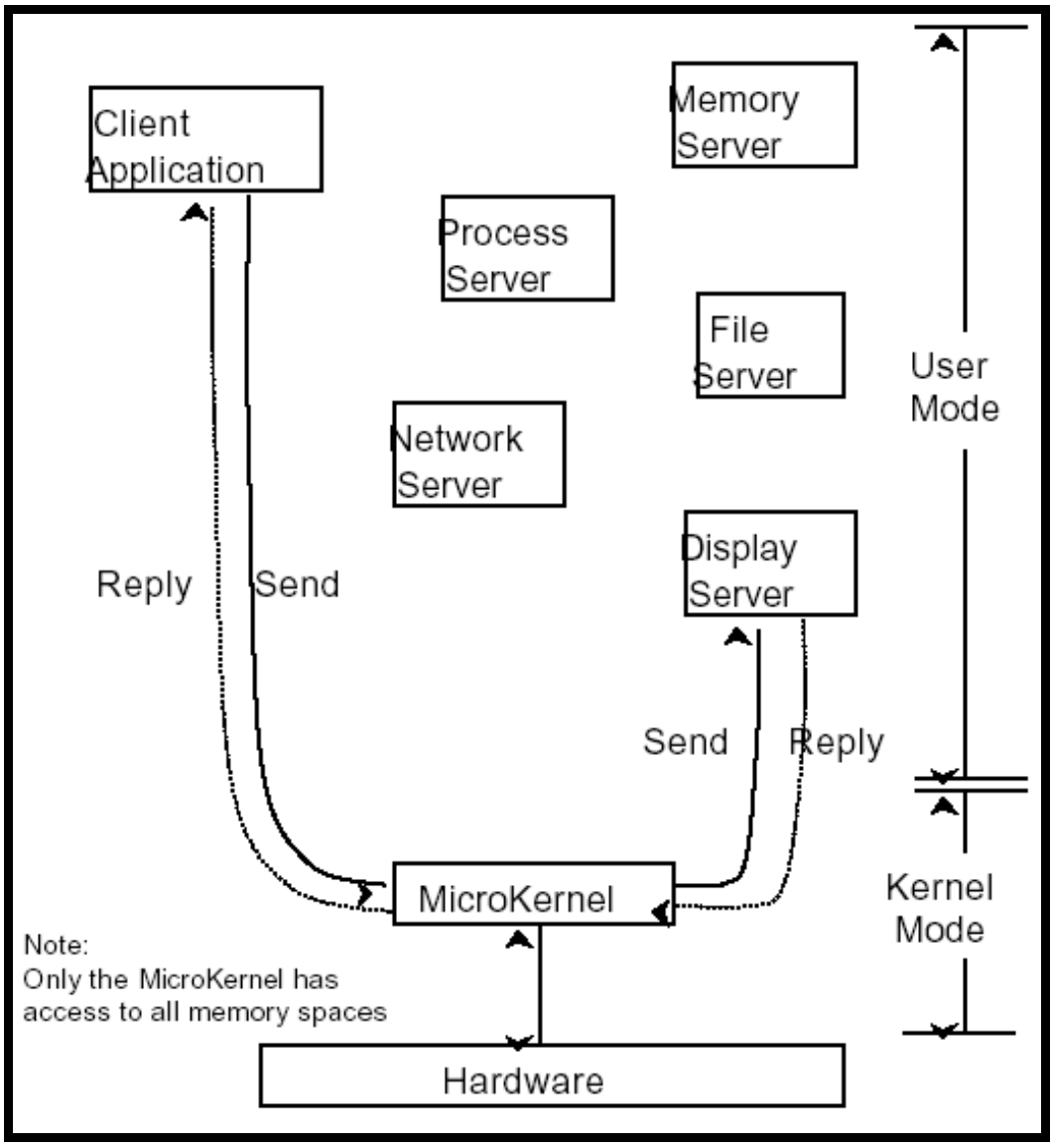


Şekil: OS/2 Sisteminin Yapısı.

4.3. Modern İşletim Sistemlerinin Mimarileri : Mikro-kernel

- *Mikro-kernel mimari:*
  - Kernele temel fonksiyonların görevlerini atar.
    - Adres uzayı
    - Süreçler arası iletişimi (IPC)
    - Temel programlama
  - Çekirdekten "kullanıcı" alanına doğru kayma sağlanır.
  - İletişim kullanıcı modülleri arasında haber göndermekle gerçekleştirilir.
  - Yararı:
    - Mikrokerneli genişletmek kolaydır
    - İşletim sistemini yeni mimarilere taşımak kolaydır
    - Daha güvenilirdir (daha az kod çekirdek modunda çalışmaktadır)

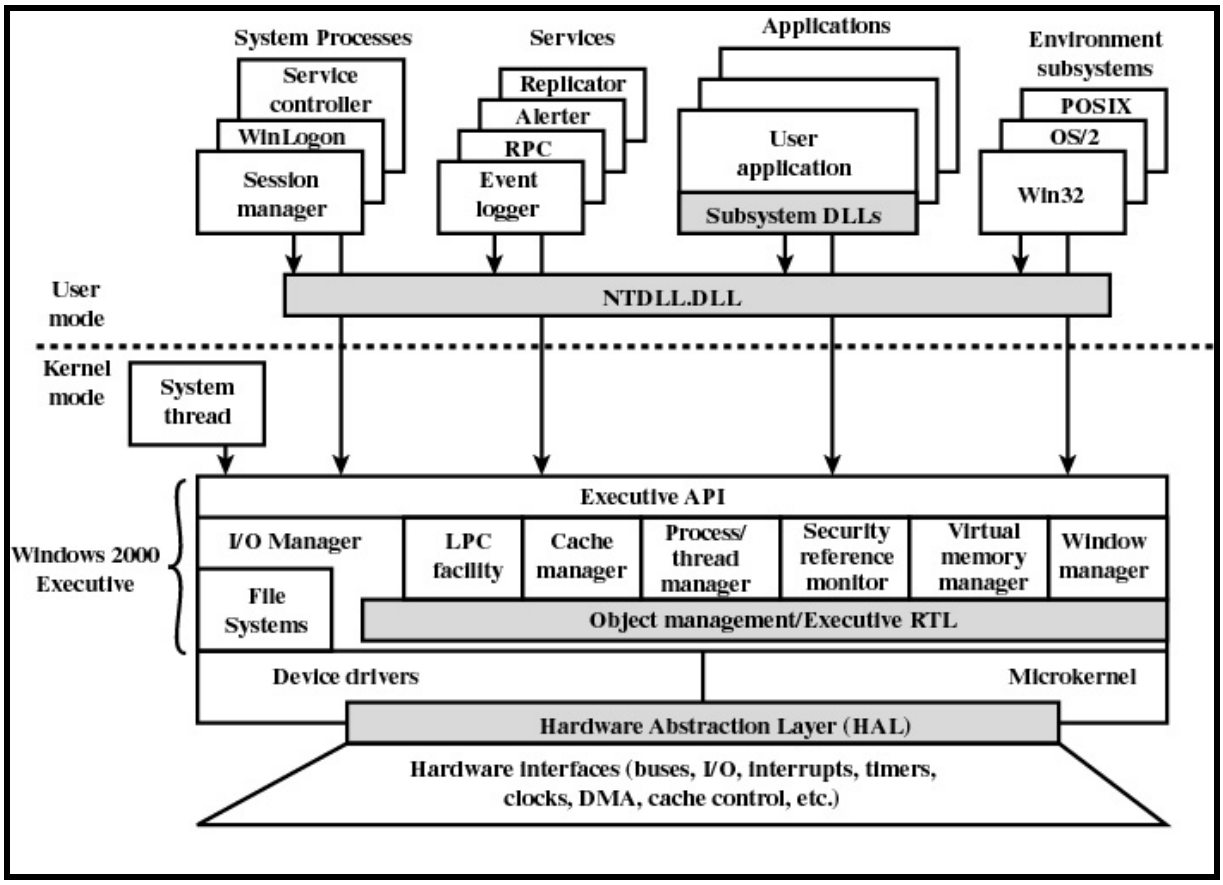
- *Modern İşletim Sistemlerinin Karakteristikleri:*
  - Çoklu iş parçacıkları (Multi-threading)
    - Süreçler, eşzamanlı çalışabilen iş parçacıklarına bölünür.
  - İş parçacığı (Thread)
    - Yürütülecek işin kernele gönderilmesi
    - Ardışıl yürütmeler kesilebilir.
  - Bir süreç, bir veya daha fazla iş parçacığının toplamıdır.
  - İş parçacığı; bir süreç içinde bulunan bir yürütme işlemi için kerneli programlayan bir program parçacığıdır.
  - Simetrik çoklu işleme
    - Çok – işlemcili yapı kullanılır.
    - Bu işlemciler aynı ana hafızayı ve G/Ç birimlerini paylaşırlar
    - Tüm işlemciler aynı fonksiyonları yürütebilirler.
  - Dağıtık işletim sistemleri
    - Bir ana hafızanın yansıması (hayali) ile ikincil hafıza uzayını elde eder.
    - Dağıtık dosya sistemlerinde kullanılır (Ağ sistemi)
  - Nesne tabanlı tasarım
    - Küçük bir kernele yardımcı modüller ilave edilerek kullanılır
    - Programcıların sistem bütünlüğünü bozmaksızın bir işletim sistemi uyarlamasını mümkün kılar.



Şekil : Mikrokernel tabanlı işletim sistemlerinin yapısı

4.3.1. Windows 2000 (W2K)

- Günümüzün güçlü 32 bitlik (iç yapısı) mikroişlemcileri kullanır
- Tek bir kullanıcı ortamında çok görevliliği sağlar
- İstemci/Sunucu (Client/Server) kullanım özelliğine sahiptir.
- Modülar yapısı sayesinde, esneklik sağlar.
- Farklı donanım platformları üzerinde çalışır
- Diğer işletim sistemleri için yazılan uygulamaları destekler



Şekil: Windows 2000 mimarisi

- Mikro-kernel mimarisindeki değişiklikler:
  - Tam anlamıyla mikrokernel değildir
  - Mikro kernel dışındaki bir çok sistem fonksiyonu kernel modunda çalıştırılabilir.
- Her hangi bir birim çıkarılabilir, güncellenebilir veya yenilebilir.
- Katman yapısı:

  Donanım soyutlama katmanı
  - Farklı donanım platformlarından işletim sistemini yalıtır.
  - Mikrokernel (mikro çekirdek)
  - İşletim sisteminin en çok kullanılan ve en temel bileşenidir.
  - Aygıt sürücüleri
  - Kullanıcının G/Ç fonksiyonlarını, bir aygıt G/Ç isteğine çevirir
- W2K Yürütümü:
  - G/Ç yöneticisi
  - Nesne yönetici

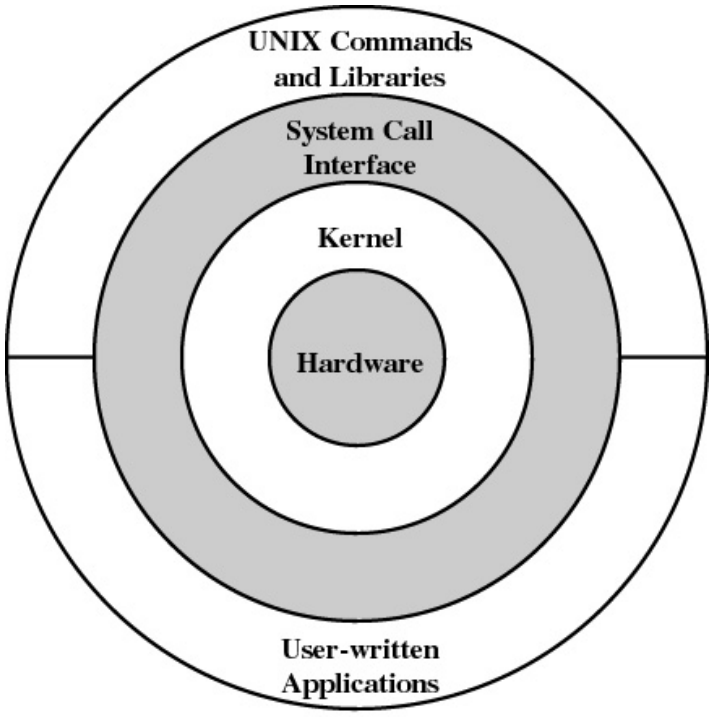
- Ekran tabanlı güvenlik
- Süreç/İş parçacığı yöneticisi
- Yerel yordam çağırma olanağı
- Sanal hafıza yöneticisi
- Ön bellek yöneticisi
- Windows/grafik birimleri

- Kullanıcı Süreçleri:
  - Süreçlere belirli sistem desteği

    Örn: Oturum açma süreci ve oturum yöneticisi
  - Sunucu süreçleri
  - Çevresel alt birimler
  - Kullanıcı uygulamaları
- İstemci/Sunucu Modeli:
  - Yürütümü basitleştirmek
  - Çeşitli API leri yürütmek
  - Güvenlik iyileştirmeleri
  - Her bir hizmet, hafızanın kendine ait olan kısmında ayrı bir süreç olarak çalışır
  - İstemciler donanıma doğrudan erişemezler
  - Yerel yordam çağrımı yolu ile iletişim kurmak için uygulamalara bir bağlantı sağlar.
  - Dağıtık hesaplama için taban oluşturur.
- İş parçacıkları:
  - Farklı yordamlar farklı işlemciler üzerinde eşzamanlı yürütülebilir.
  - Bir süreç içindeki bir çok iş parçacığının yürütümü farklı işlemciler üzerinde eşzamanlı yürütülebilir.
  - Sunucu süreçleri bir çok iş parçacığını kullanabilir.
  - Süreçler arasında veri ve kaynak paylaşma mümkün.

4.3.2. UNIX Sisteminin Yapısı

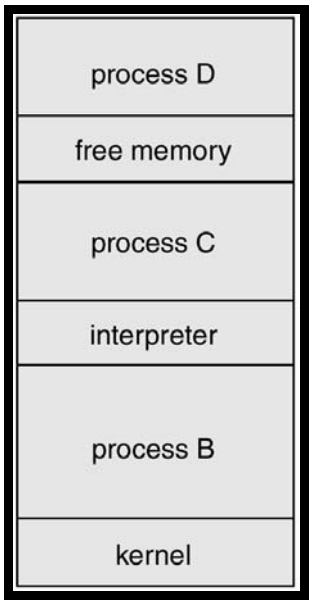
- Donanım işletim sistemi tarafından çevrelenmiştir.
- İşletim sistemi Kernel olarak isimlendirilir.
- Ardından arayüzler ve kullanıcı hizmetleri gelir.
  - kabuk
  - C derleyicisi



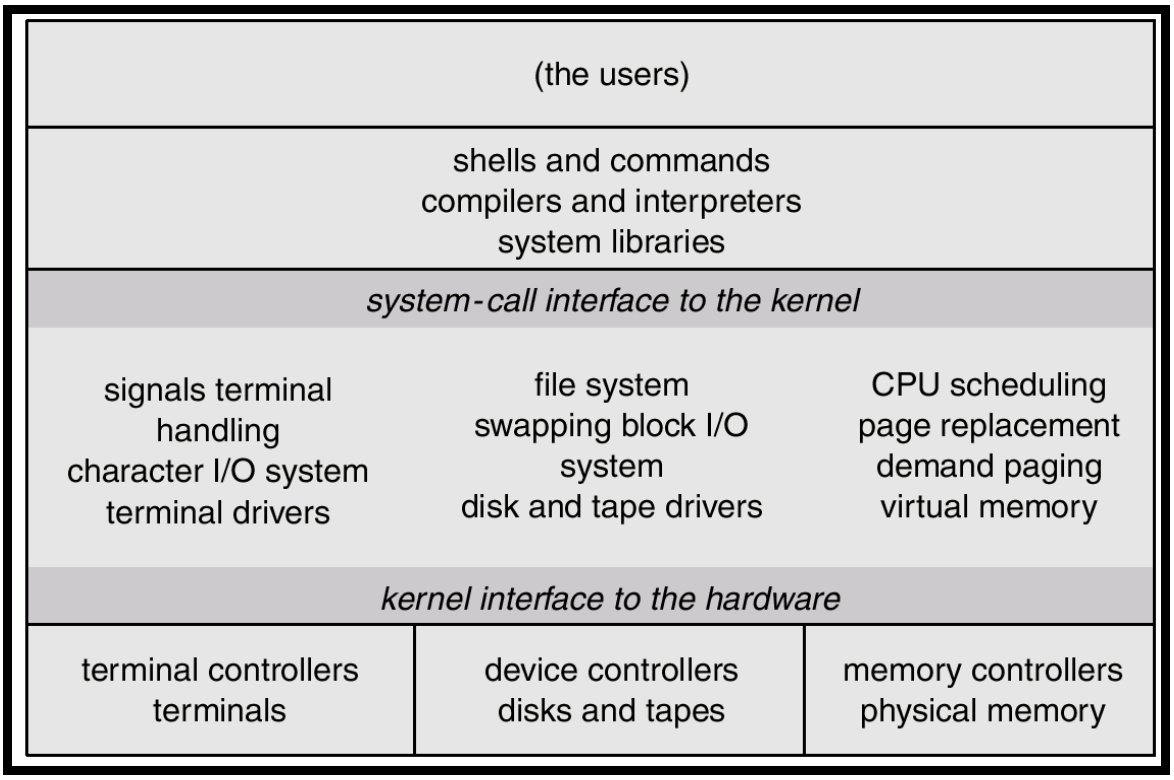
Şekil: UNIX Mimarisi.

UNIX sistemi iki ayrı kısımdan oluşmaktadır:

- Sistem programları
- Çekirdek
  - Sistem çağrıları, arayüzden fiziki donanıma kadar her şeyi içeriyor.
  - Kütük sistemlerini, AİB planlamasını, bellek yönetimini, diğer işletim sistemi işlevlerini sağlar.



Şekil: UNIX sisteminde çoklu program çalışması



Şekil: UNIX Sisteminin yapısı

4.4. Sunulan Hizmetler

İşletim sistemlerinin çeşitli türleri bulunmakla birlikte, bunlar arasında ortak özellikler vardır. İşletim sistemlerini bu özellikleri üzerine sınıflandırmak mümkündür. Bu sınıflardan birisi de kullanıcıya sunulan hizmetlerdir. İşletim Sisteminin kullanıcıya sağladığı hizmetler, bilgisayarla çalışmayı kolaylaştırdığı gibi, işletim sisteminin parçalarını kullanıcıdan saklar.

- *Program Çalıştırma :* İşletim Sistemi herhangi bir programı belleğe yükleyebilir ve çalıştırabilir. Programın çalışması, ancak normal sonlanma veya hata oluşması durumunda kesilebilir.
- *Giriş/Çıkış İşlemleri :* Çevresel donanımların kullanılması
- *Kütüklerin İşlemleri :* Kütük oluşturma, yazma, silme
- *İletişim :* Bir görevin diğer bir görev ile bilgi alışverişinde bulunması

  Görevler arası iletişim genelde iki yolla gerçekleşir;

  - Aynı yöntemle çalışılması
  - Ağ üzerindeki farklı bilgisayarlar arasında iletişim, ortak bellek alanı kullanmakla ve haber mesaj göndermekle sağlanır.

- *Hata Bulma :* İşletim Sistemi bilgisayar kaynaklarında (bellek, ana işlem birimi, disk, programlar, kütük, G/Ç aygıtları) oluşan hataları bulur, analiz eder ve mümkünse bu hataları önler.

- *Kaynakların Paylaşımı:* İşletim Sisteminin bazı fonksiyonları kullanıcıya yardım için değil sistemin verimli kullanılması için kullanılır. Örneği; kaynakların paylaşımı: aynı zamanda birden fazla program çalıştırılırsa sistemin kaynakları bu programlar arasında paylaştırılır. Her bir kaynağın yönetimi içi çeşitli yöntemler ve algoritmalar vardır. Örneğin; ana işlem biriminin daha iyi kullanımı için AIB planlama algoritması kullanılır. Bu algoritma için giriş parametreleri; işlemcinin hızı, yazmaçların sayısı, aynı zamanda çalışabilecek görevler sayısı vb.

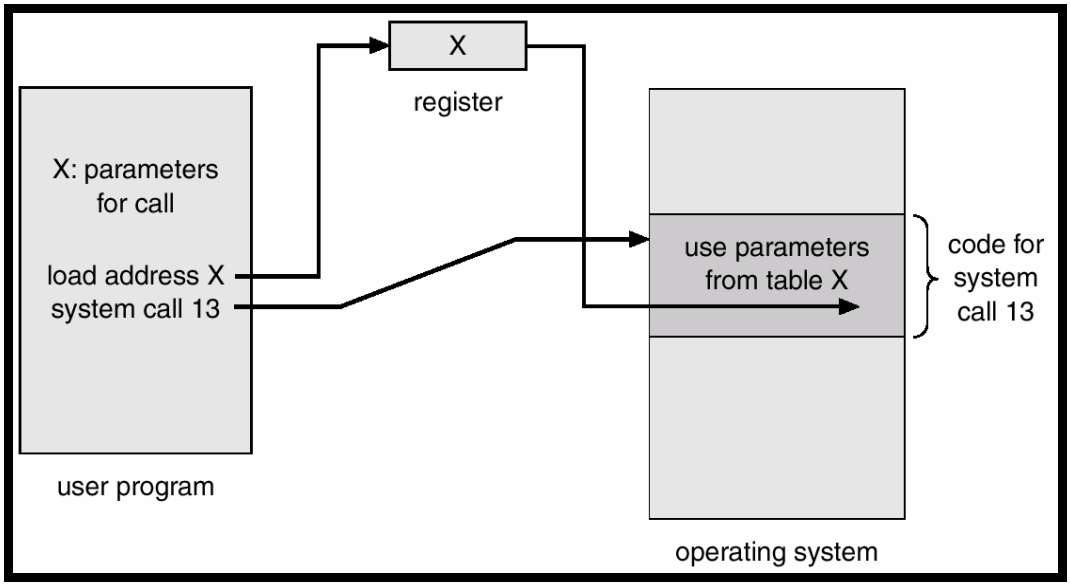
- *İstatistiksel Verinin Hesaplanması :* Bu tür bilgiler, hangi kullanıcıların hangi kaynakları, ne kadar kullanacağını, sistemde oluşan hata türlerini, kaynakların kullanım oranlarını içerir.

- *Koruma :* Tüm sistem kaynaklarına erişimin denetimini sağlar.

4.4.1. Sistem Çağrıları

- Sistem çağrıları çalışan program ile işletim sistemi arasındaki ara yüzü sağlar.
  - Genellikle assembly dili komutlarıyla erişilebilir.
  - Bazı yüksek seviyeli dillerde sistem çağrılarına doğrudan erişimi sağlar (Örn;C++)

- Çalışan program ile işletim sistemi arasında parametrelerin gönderilmesinin üç temel yöntemi vardır:
  - Parametrelerin yazmaçlarda gönderilmesi
  - Parametrelerin bellekte, tabloda saklanması ve tablonun adresinin parametre gibi yazmaca gönderilmesi
  - Parametrelerin programla yığına yazılması ve işletim sistemi tarafından yığından alınması.



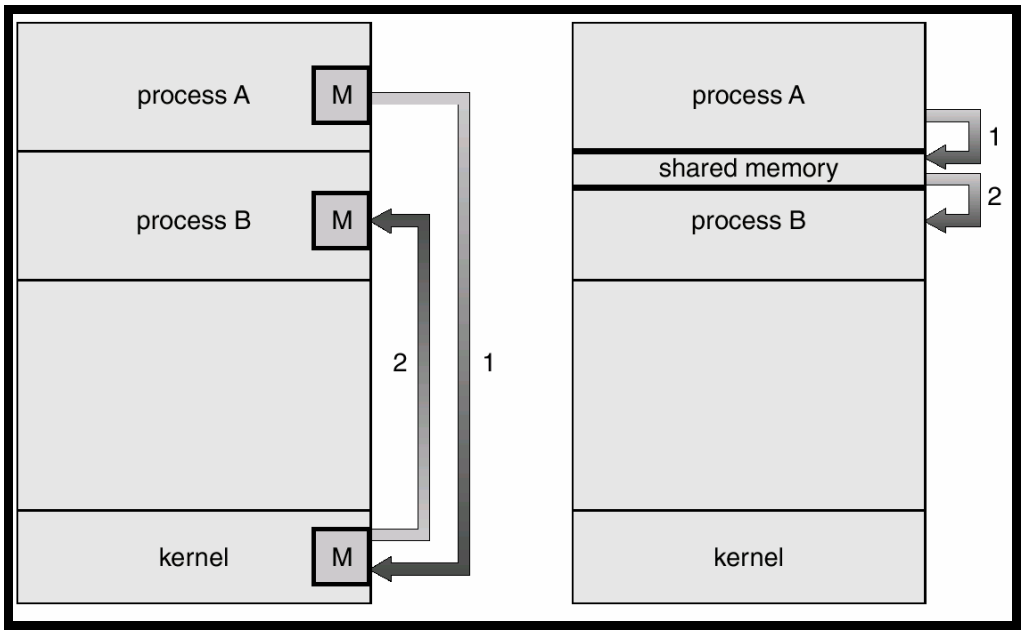
Şekil: Parametrelerin tablo ile gönderilmesi

4.4.2. Sistem Çağrılarının Türleri

- Görev denetimi
- Kütük denetimi
- Aygıt yönetimi
- Bilgi saklama
- İletişim

4.4.3. İletişim Modelleri

İletişim haber göndermekle, yada ortak belek aracılığıyla gerçekleştirilir:



Şekil: İletişim türleri; **a)** Haber gönderme, **b)** Ortak bellek alanı

4.4.4. Sistem Programları

Sistem programları program geliştirme ve yürütmek için elverişli ortam sağlar. İşletim sisteminin pek çok kullanıcı görünümlerini, sistem çağrıları değil, sistem programları belirler. Sistem programları şunlardır:

- Kütük işlemleri
- Durum bilgileri
- Kütük güncelleme
- Programlama dili desteği
- Program yükleme ve yürütme
- İletişim
- Uygulama programları

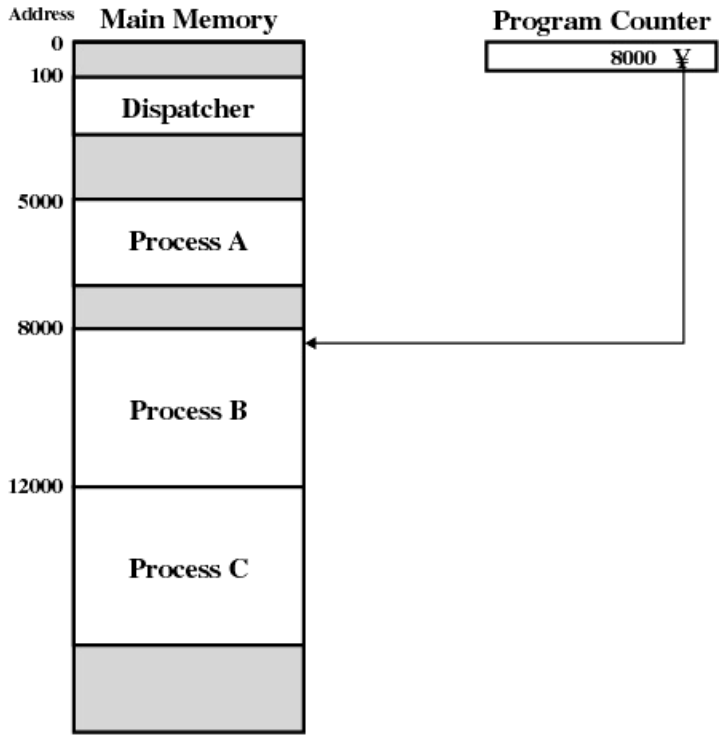
5. GÖREV YÖNETİMİ

Görev (process), işletim sistemi tarafından yürütülen ardışık işlemler sürecini içeren bir programdır. Aynı zamanda süreç olarak da adlandırılır. Görev için gereken kaynaklar, görev ortamını oluşturur.

Görev yönetiminin amaçları:

- Makul cevaplama süresi içerisinde, birkaç görevi aynı anda yürüterek işlemciden maksimum faydalanmayı sağlamak
- Görevleri kaynaklara dağıtmak
- Görevler arası haberleşmeyi sağlamak
- Kullanıcıların görev oluşturabilmesine ortam oluşturmak.

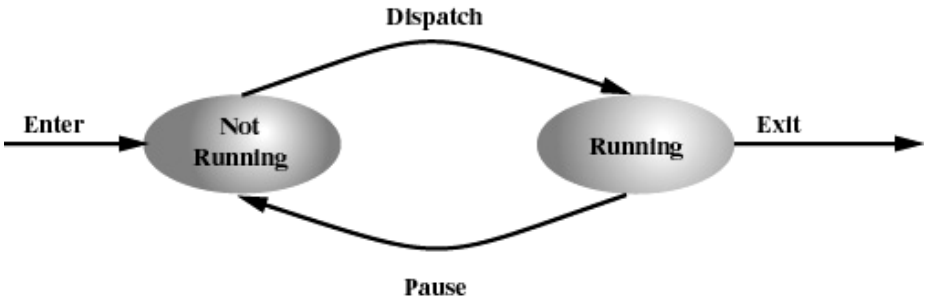
Görevin çalışması, ardışık işlemler süreci şeklinde olup, her bir zaman diliminde görevin yalnız bir komutu çalışabilir. Görev yalnız program kodunu değil, aynı zamanda *program sayacı*nın (PC) değeri ile ifade edilen girişimleri, işlemci kayıtçılarının içeriklerini, kesme verilerini (alt program parametreleri, geri dönüş adresleri, geçici değişkenleri) içeren *yığın*, genel değişkenleri içeren *veri bölümü*nden oluşur dahil edilir.



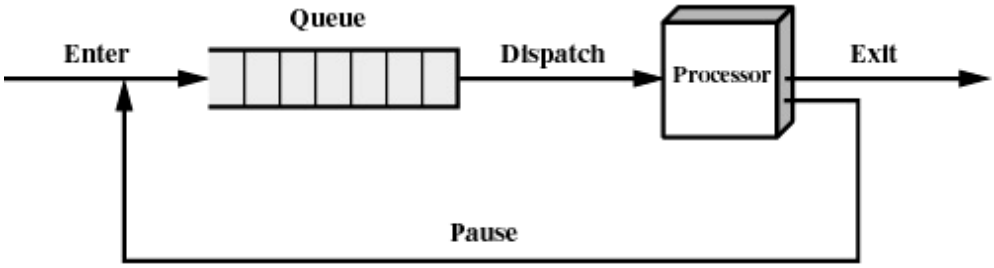
Şekil: Bir komut saykılının Yürütümü

İki görev aynı programda birleşebilir, ancak farklı çalışma süreçlerini içerirler.

Görev Durumu : Görev, yürütülmekte olduğundan durumu değişken yapı sergiler. Görev genel olarak, yaptığı işe girişimlere göre iki durumda birinde olabilir: Çalışma veya Çalışmama durumu.

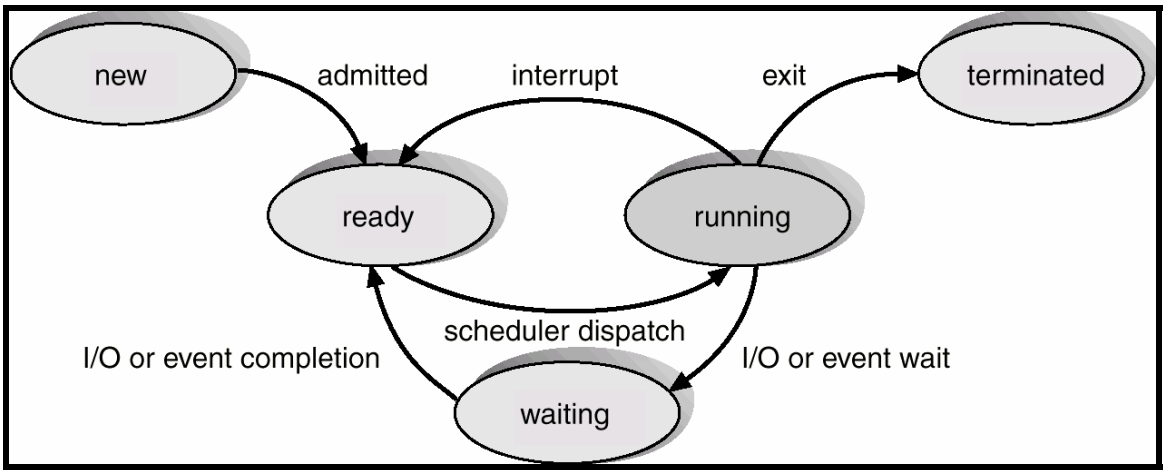


Şekil: Görev durum geçiş diyagramı



Şekil: Çalışma ve çalışmama durumu.

Görev her bir zaman diliminde aşağıdaki durumlardan birinde olabilir.

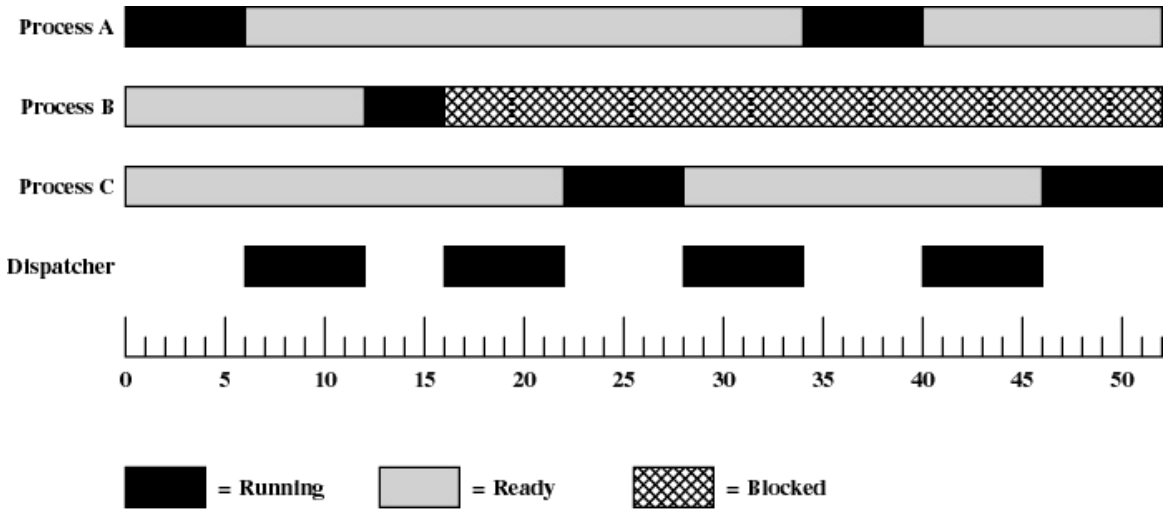


Şekil: Beş durumlu görev modeli.

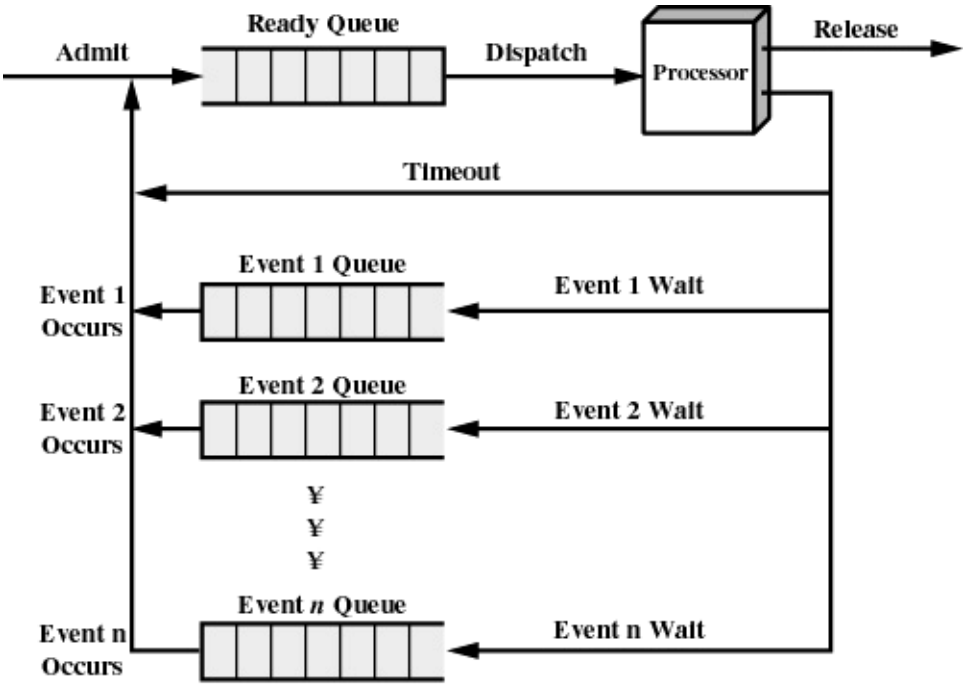
- *Sunuş (new):* Görev oluşmaktadır.
- *Çalışma Durumu (running):* Görevin bir komutu işlemcide gerçekleştirilmektedir.
- *Bekleme Durumu (waiting) :* Görev herhangi bir olayın oluşmasını beklemektedir. (Örneğin; G/Ç işleminin bitmesi, herhangi bir sinyalin gelmesi).

- *Hazır Olma Durumu (ready) :* Görev işlemciye aktarılmak için beklemektedir. Bu duruma çalışmama durumu veya yürütüme hazır durumda denilmektedir.
- *Bitiş veya Kesilme Durumu (terminated) :* Görev çalışmasını bitirdikten sonra bu duruma geçer. Görev sonlandırma sebepleri:
  - Uygulamadan çıkma
  - Normal tamamlanmış
  - Hafıza kullanıma uygun olmaması
  - Koruma hatası, Örn: Sadece okunabilen bir dosyaya yazma
  - Aritmetik hata
  - Çalışma zamanı aşımı: Bir görevin veya bir iş parçacığının belirlenen maksimum süreden daha uzun süre beklemesi
  - I/O başarısızlığı
  - Geçersiz komut: Bir verinin yürütüm için ele alınması
  - İmtiyazlı komut
  - Üst görevlerin alt görevleri sonlandırması

Herhangi bir anda sadece bir görev çalışma durumundadır, ama birden fazla görev hazır ve bekleme durumunda olabilir.



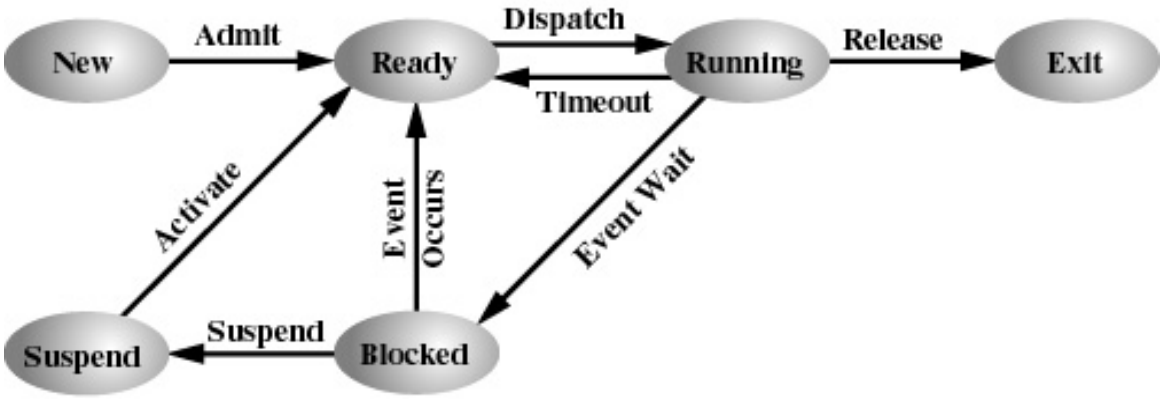
Şekil: Görevlerin durumları



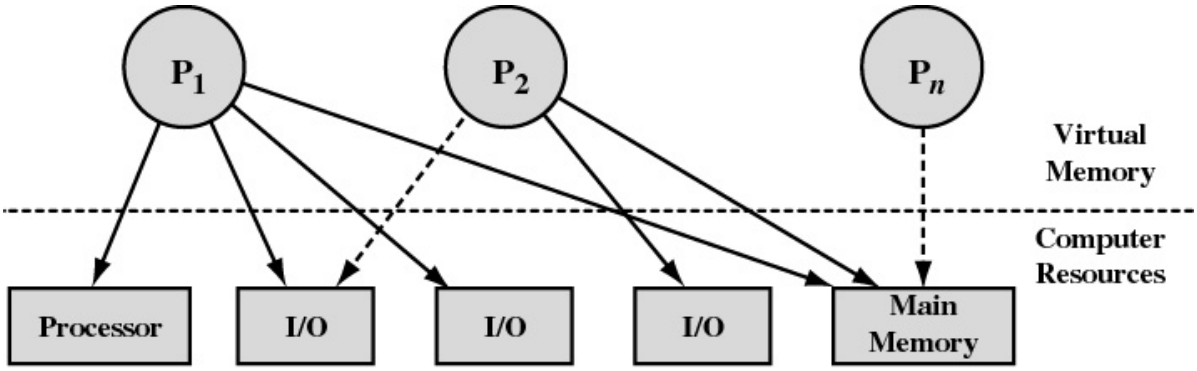
Şekil: Görev durumlarının değerlendirilmesi

Görevin Askıya Alınması: Görevin askıya alınma nedenleri;

- Mikroişlemci G/Ç birimlerinden çok daha hızlı olmasına rağmen, tüm görevler G/Ç işlemini beklerler.
- Swapping (Değiş tokuş): Bir işletim sistemi, bir görevi yürütebilmesi için yeterince ana hafızayı serbest bırakılmasına ihtiyacı duyarlar.
- Kullanıcı isteklerinin etkileşimi
- Zamanlama
- Üst görev istekleri.



Şekil: Görev askıya alınması durumu

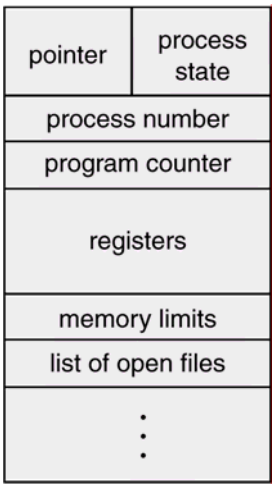


Şekil: Kaynakların (işlemci, I/O, hafıza) görevlere (P_1,P_2, P_n) dağılımı

5.1.GÖREV DENETİMİ

Her bir görevin çalışması için gereken bilgiler görev denetim bloğunda (PCB=Process Control Blok) bloğunda gösterilir. Görev denetim bloğundaki bilgiler, program sayacı, ana işlem birimi yazmaçları v.s bulunur. Her bir görevle ilgili bilgiler:

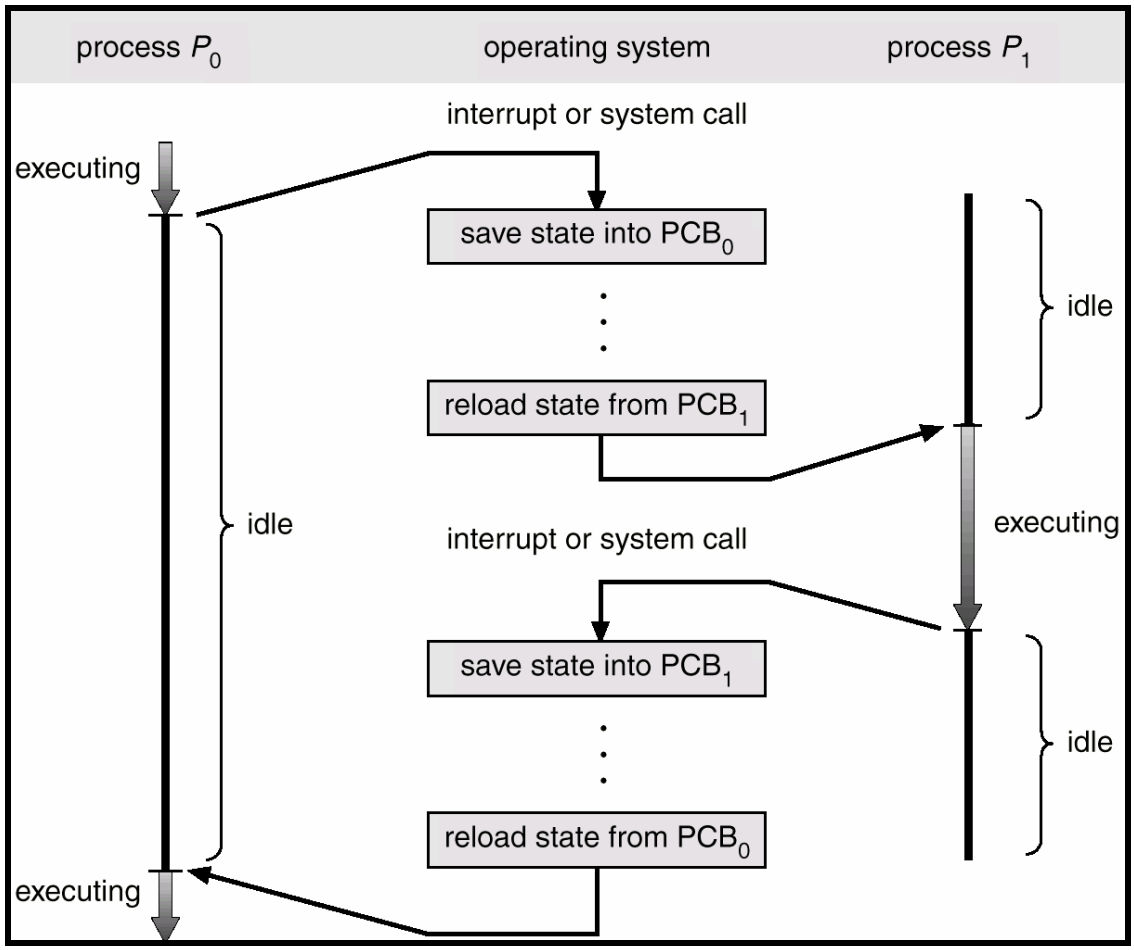
- Görev durumu
- Görev sayacı
- AİB yazmaçları
- AİB planlanması bilgileri
- Bellek yönetimi bilgileri
- İstatistiksel bilgiler
- G/Ç durumu bilgileri



Şekil: PCB

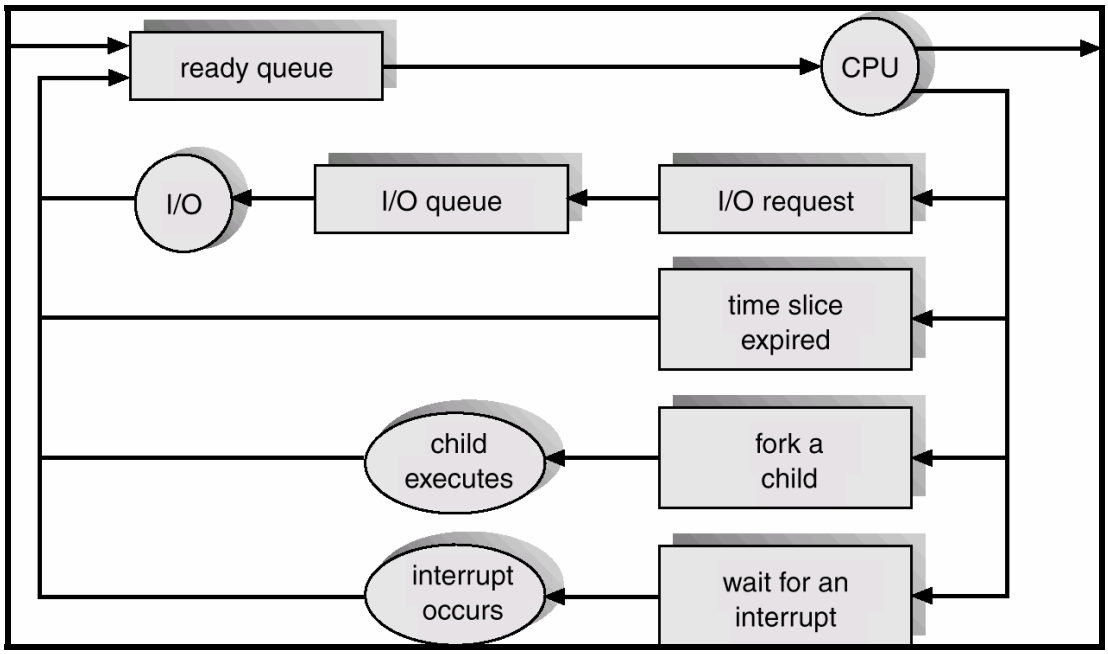
- *Program sayacı:* Görev için sonraki çalıştırılacak program kodunun adresi yazılır.
- *AİB yazmaçları:* Bilgisayarların mimarilerine göre farklılık gösterir. Bu yazmaçlarda görevin durumuna ait bilgiler saklanır. Bu bilgilerin saklanmasının önemi, görev kesildikten sonra yeniden çalışmaya başlaması için görev adresinin belli olmasıdır. Örnek; Program sayacı, Yığın.
- *AIB'nin planlanması bilgileri:* Görev önceliklerini, zamanlama parametrelerini içerir.
- *Bellek yönetimi bilgileri:* Taban adreslerini, sayfa ve kesilme adreslerini içerir.
- *İstatistiksel bilgiler:* Ana işlem biriminin kullanım süresi görevlerin çalışması için zaman sınırlamaları, görev numaraları gibi bilgiler içerir.

- *G/Ç durumu bilgileri:* Göreve ayrılan G/Ç aygıtlarının listesi, görev için açılmış kütüklerin listesi gibi G/Ç durumu bilgilerine içerir.



Şekil: AİB'nin görevler arasında paylaşımı

Görev sisteme sunulduktan sonra görev kuyruğuna yerleşir. Çalışmak için hazır olan ve çalışmayı bekleyen hazır görevler kuyrukta bulunurlar. Kuyruk genelde bağlantılı liste biçiminde oluşturulur. Görevlerin PCB göstergelerini içerir. Görev CPU'da belli bir sure içersinde çalıştırılır. Bu sure bitiminden sonra görev, kesme veya herhangi özel bir durumun oluşmasını bekler. I/O aygıtlarını bekleyen görevler, aygıtlar kuyruğunda oluşur. Her aygıtın kendi özel kuyruğu vardır (Örn: Yazıcı). Görevlerin kuyrukta seçilmesi görev planlayıcısının işidir.

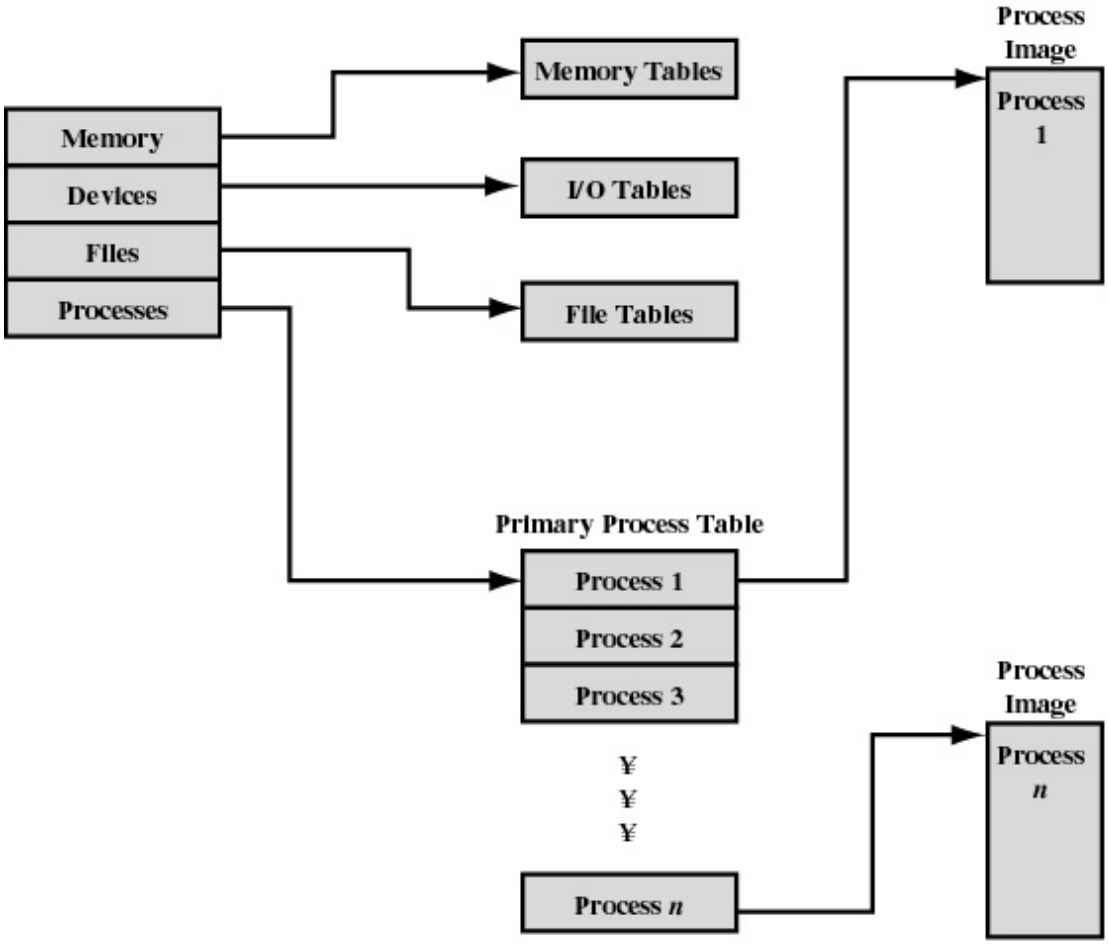


Şekil: Görev planlamasının akışı

Planlayıcılar :

- Uzun vadeli planlayıcılar (veya iş planlayıcısı) – hazır görevler kuyruğuna girecek görevleri seçer. Uzun vadeli planlayıcılara daha az sıklıkla gereksinim duyulmaktadır. Bu nedenle yavaş olabilir.
- Kısa vadeli planlayıcı (veya AİB planlayıcısı) – bir sonraki hangi görevin yürütüleceğini belirler ve AİB'ye yerleştirir. Kısa vadeli planlayıcılarına sıkça görev düşmektedir. Bu nedenle hızlı olmalıdır.
- Uzun vade planlayıcısı çok programlama seviyesini denetler. Görevler:
  - *G/Ç yönlü görev* – zaman hesaplamalardan ziyade G/Ç işlemlerine zaman harçlar, AİB kullanımı çok kısadır.
  - *AİB yönlü görev* – hesaplamalara daha çok zaman harçlar; AİB kullanımı uzundur.

İşletim Sisteminin Kontrol Tabloları : Herhangi bir görevin ve kaynağın mevcut durumu hakkında bilgi edinmek için kullanılır. Tablolar, işletim sistemi yönetiminin her bir bileşeni için oluşturulur.



Şekil: İşletim sisteminin kontrol tabloları.

Hafıza Tabloları

- Görevlerin ana hafızaya yerleştirilmesi
- Görevlerin ikincil hafızaya yerleştirilmesi
- Paylaştırılan hafıza bölgelerine erişim niteliğini koruma

G/Ç Tabloları

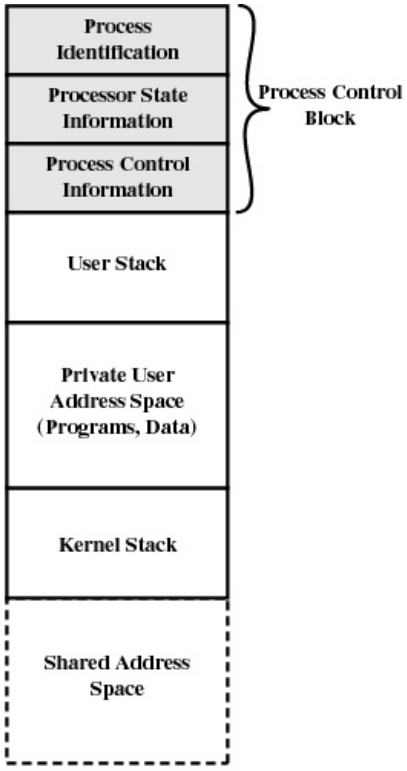
- G/Ç aygıtının kullanımı veya belirli bir görev için ayrılması
- G/Ç işleminin durumu
- G/Ç transferinin, tanımlayıcısı veya kaynağı olarak kullanılan ana hafızadaki yeri tutar.

Kütük Tabloları

- Varolan dosyalar
- İkincil hafızadaki yer
- Mevcut durum
- Nitelikler

Görev Tabloları

- Görevin konumun bulunduğu yer
- Görevin yönetimi için gerekli nitelikler



Şekil: Kullanıcı uzayı içinde işletim sisteminin görev yürütüm yapısı

– Görev Tanımlayıcısı

- Nümerik tanımlayıcılar, görev kontrol bloklarının içerisine depolanabilir.
- Görev tanımlayıcısı
- Görevin oluşturduğu görevin tanımlayıcısı
- Kullanıcı tanımlayıcısı

– Görev Durumu Bilgileri

- İşlemci durum bilgisi
  - Kullanıcıya görünen kayıtçılar
  - Kontrol ve Durum (PSW) kayıtçıları (Örn: Pentiumlarda EFLAGS)
  - Yığın İşaretçileri (Stack Pointers)

ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag
DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

Şekil: EFLAGS Kayıtçısı

– Görev Kontrol Bilgileri

- Programlama (Scheduling) ve durum bilgisi: Performansı artırmak için işletim sistemi tarafından zaman programlama fonksiyonlarına gereksinim duyulur. Bunun için tipik bilgi parçaları:
  - •*Görev durumu:* Görevin yürütüm için gerekli olan zamanlamasını tanımlar (Örn: çalışma, hazır, bekleme, durdurulma).
  - •*Öncelik:* Bir veya daha fazla görevin zaman önceliklerini tanımlamak için kullanılır. Bazı sistemlerde, birkaç değer (örn: varsayılan, yürürlükteki, en yüksek imtiyaz) gereklidir.
  - •*Bilgi Tabanlı Zamanlama:* Bu kullanılan zamanlaman algoritmasına bağlıdır. Görevin bekleme zamanının süresi, yürütüm zamanının süresi, çalıştığı en son zaman.
  - •*Olay :* Görev, çalışmayı devam ettirebilmek için olayın tanımlayıcı kimliğini bekler.
- Veri Yapısı: Bir görev, kuyrukta, halkada, veya diğer yapılarda bulunan diğer bir görev bağlanabilir. Örneğin;
  1. Bir kuyruktaki, belirli bir öncelik seviyesi nedeniyle bekleme konumunda bulunan tüm görevler birbirine bağlıdırlar.
  2. Görev, başka görevler ile üst-alt (imtiyazlılık açısından) ilişkisine sahiptir. Görev kontrol bloğu, bu yapıları desteklemek için diğer görevlere işaret edicileri içerebilir.

- Görevler arası iletişim: Bağımsız iki görev arasındaki iletişim birlikteliği çeşitli bayraklar, işaretler, ve mesajlar ile sağlanabilir. Bu bilgilerin tamamı veya bir kısmı, görev kontrol bloğunda bulunabilir.

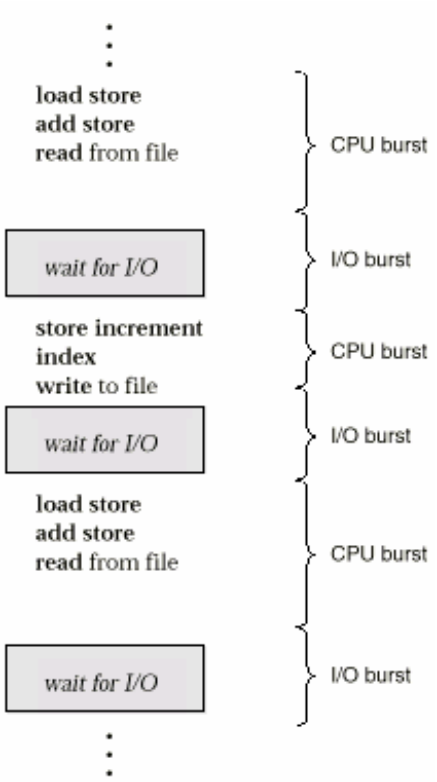
- Görev ayrıcalıkları: Görevler hafıza açısından kabul edilen ayrıcalıklara sahiptirler. Bunlar, erişim ve yürütüm için gerekli olan komut türleridir. Ayrıca, ayrıcalıklar sistem hizmetleri ve programlarının kullanımı içinde uygulanabilir.

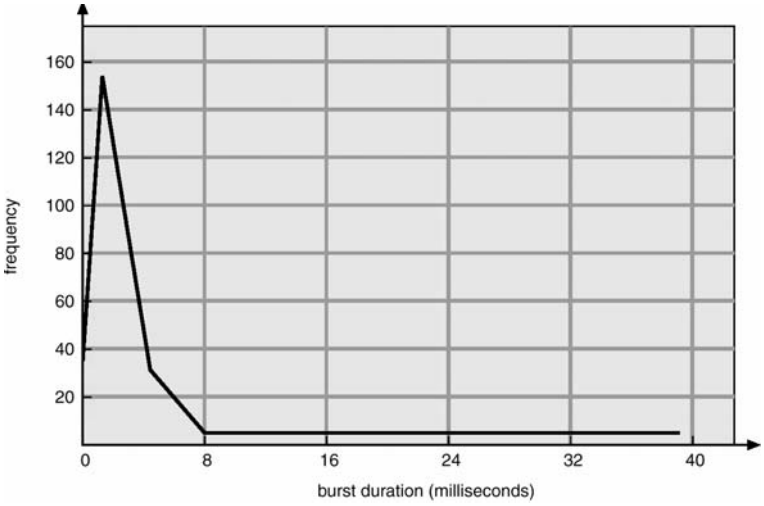
- Hafıza Yönetimi:Bu bölüm, bir görev için tahsis edilen sanal hafızaya işaret eden segment göstericisi veya sayfa tablolarını içerir.

- Kaynak sahipliliği ve kullanımı: Görev tarafından kontrol edilebilir kaynaklar gösterilebilir. Örneğin; açık dosyalar, işlemcinin veya diğer kaynakların kullanım geçmişi verilebilir. Bu bilgi programlayıcı için gereklidir.

5.1.1. Ana İşlem Biriminin Planlanması

Çoklu programlamanın hedefi; aynı zamanda birkaç görevi birlikte çalıştırmakla, ana işlem biriminin verimliliğini yükseltmektir. Görevlerin çalışmasını ilişkisel koordine etmek için planlama fonksiyonu (scheduling) kullanılır. Bu fonksiyon işletim sisteminin temel fonksiyonlarındandır. Tüm bilgisayar kaynaklarının görevi, çalışma öncesi planlanır. Esas kaynak olan AIB'nin planlanması işletim sisteminin tasarımında başlıca yeri tutmaktadır. Her bir görevin çalışması, AIB ve G/Ç işlemlerinden oluşur. AIB'nin çalışma süresi göreve ve bilgisayarlara göre farklılık sergiler. Genelde kısa süreli ana işlem birimi çalışmalarının sayısı uzun süreli çalışmalardan daha yoğundur.





Görevlerin planlanması için çeşitli algoritmalar kullanılmaktadır. Bu algoritmaları kıyaslamak için kriterler:

1. Ana işlem biriminin kullanım durumu: Gerçek sistemlerde AIB'nin kullanım oranı %40 ile %90 arasındadır.
2. Belirlenmiş zaman diliminde çalışmasını tamamlayan görev sayısı.
3. Görevi çalıştırmak için harcanan zaman: Görevin sisteme sunulmasından bitişine kadar geçen zamandır ve görevin bellekten okunması, kuyrukta beklemesi, işlemcide çalışması ve G/Ç işlemlerini yapması sürelerinin toplamından oluşur.
4. Görevin hazır görevler kuyruğunda bekleme zamanı
5. Cevaplama süresi: Değişken zaman paylaşımlı ortamlar için kullanılır ve istek sunulduktan sonra onun cevabının alınmasına kadar geçen zamandır.

5.1.2. Planlama Algoritmaları

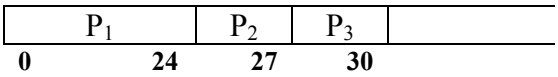
5.1.2.1. FCFS (First Come First Served – İlk Gelen Önce) Algoritması

Bu algoritmaya göre; AIB yi ilk talep eden görev, ilk olarak işlemciyi kullanır. FIFO kuyruğu ile çalıştırılabilir. Görev hazır görevler kuyruğuna sunulduktan sonra, onun görev denetim bloğu (PCB) kuyruğun sonuna ilave edilir. AIB boş olduğu zaman kuyruğun başındaki görev çalışması için AIB ye sunulur ve kuyruktan silinir. Bu algoritmada görevlerin bekleme süresi yüksek olur.

Örnek: P_1, P_2, P_3 görevlerinin sırasıyla kuyrukta yerleştiklerini kabul edelim:

Görev	Çalışma Zamanı (sn)
P_1	24
P_2	3
P_3	3

1.Görevler P_1, P_2, P_3 ardışıklığı ile sunulmuş olduğunu varsayalım. Buna göre planlama:



Burada;

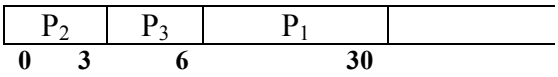
P_1 'in bekleme süresi → 0 msn.

P_2 'nin bekleme süresi → 24 msn.

P_3'ün bekleme süresi → 27 msn 'dir.

Ortalama bekleme süresi : (24+27+0) / 3 = 17 msn 'dir.

2.Eğer görevlerin gelme P_2, P_3, P_1 şeklinde sıralanırsa, planlama:



Burada ise;

P_2 'nin bekleme süresi → 0 msn.

P_3 'ün bekleme süresi → 3 msn.

P_1'in bekleme süresi → 6 msn 'dir.

Ortalama bekleme süresi : (3+6+0) / 3 = 3 msn'dir.

Görüleceği üzere, FCFS algoritmasında ortalama bekleme süresi daima minimum olmayabilir. Çünkü görevlerin geliş sırasına bağımlıdır.

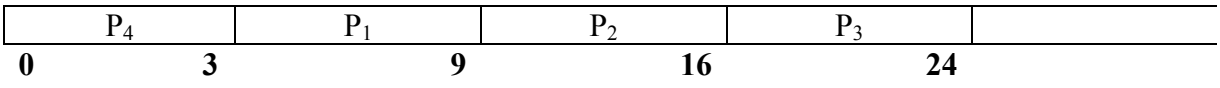
5.1.2.2.SJF (Shortest Job First – En Kısa İşletim Süresi Olan Önce) Algoritması

Bu algoritmada CPU boş olduğunda, kalan görevler içinde çalışma süresi en küçük olan görev, çalışması için işlemciye sunulur. Eğer iki görevin kalan süreleri aynı ise o zaman FCFS algoritması uygulanır. Bu algoritmada:

- Her görev, o görevin bir sonraki AİB işlem zamanı ile değerlendirilir. Bu, en kısa zamanlı işin bulunması için kullanılır.
- SJF Türleri:
    1. Kesilmesiz SJF : Eğer AİB bir göreve tahsis edilmişse, AİB işlem zamanı bitmeyince görev kesilemez.
    2. Kesilmeli SJF : Eğer AİB işlem zamanı, şu anda çalışan görevin kalan işlem zamanından küçük olan yeni bir görev sisteme sunulmuşsa, eski görev kesilecek. Bu yönteme, SRTF(Shortest Remaining Time First – En kısa işlem zamanı kalan, birinci) yöntem denir.
- SJF verilmiş görevler kümesi için en küçük ortalama bekleme zamanı oluşması için optimizasyon yapar.

Örnek: : P_1, P_2, P_3, P_4 görevleri aşağıdaki ardışıklık ile sunulmuş olduğunu varsayalım. Buna göre kesilmesiz SJF yöntemine göre ortalama bekleme süresini bulalım:

Görev	Çalışma Süresi
P_1	6
P_2	7
P_3	8
P_4	3



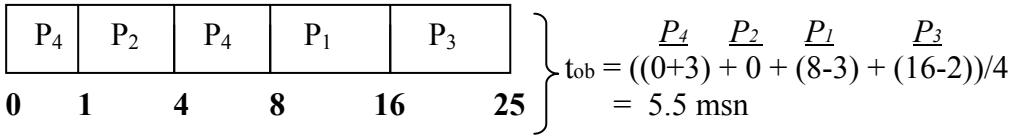
SJF : $t_{ob} = t_{P1} + t_{P2} + t_{P3} + t_{P4} = (3+9+16+0) / 4 = 7$ msn

FCFS : $t_{ob} = t_{P1} + t_{P2} + t_{P3} + t_{P4} = (0+6+13+21) / 4 = 10.75$ msn

olarak ortalama bekleme süresi (t_{ob}) bulunur.

Örnek: P_1, P_2, P_3, P_4 görevleri aşağıdaki ardışıklık ile sunulmuş olduğunu varsayalım. Buna göre kesilmeli SJF yöntemine göre ortalama bekleme süresini bulalım:

Görev	Sunuş Zamanı (sn)	Çalışma Süresi (sn)
P_1	3	8
P_2	1	3
P_3	2	9
P_4	0	5

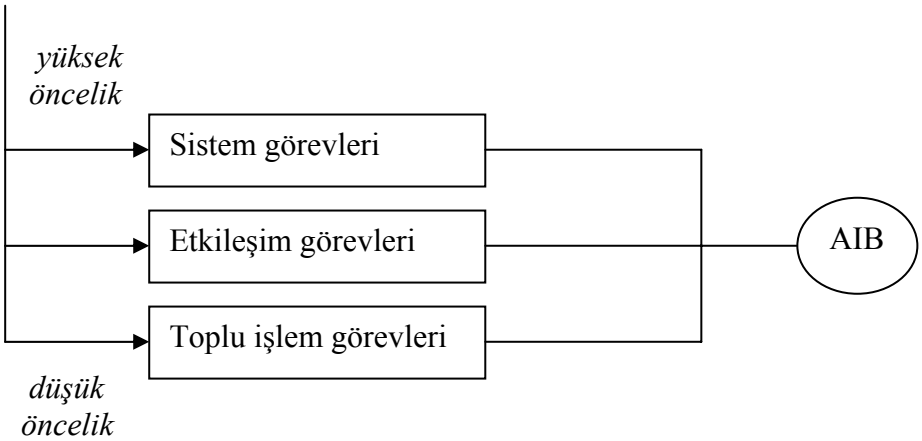


5.1.2.3.SRTF (Shortest Remaining Time First - En Kısa İşletim Süresi Kalan Önce)

İlk işlem yapılırken bu esnada sunulan görevin çalışma süresi daha kısa ise o yapılan görev kesilir, çalışma zamanı kısa olan görev işlenir.

5.1.2.4.Çok Kuyruklu Planlama Algoritması

Bu algoritmaya göre görevler belli sınıflara ayrılır ve her sınıf görev kendi kuyruğunu oluşturur. Başka deyişle hazır görevler çok seviyeli kuyruğa dönüştürülür. Görevin türüne önceliğine, bellek durumuna veya başka özelliklerine göre görevler belli kuyruğa yerleştirilirler. Her kuyruk için planlama algoritması farklı olabilir. Bununla birlikte görevlerin bir kuyruktan diğerine aktarılmasını sağlayan algoritmada oluşturulur.



Bu algoritmaya göre yüksek öncelikli kuyruktaki görevler önce işlenir. Eğer bu kaynak boş ise ondan aşağı seviyedeki görevler çalıştırılabilir.

5.1.2.5.Öncelikli Planlama Algoritması

Bu algoritmaya göre her bir göreve öncelik değeri atanır ve görevler öncelik sırasına göre işlemciyi kullanırlar. Aynı öncelikli görevler FCFS algoritması ile çalıştırılırlar.

Görev	Öncelik	Çalışma Süresi (msn)
P_1	3	10
P_2	1	1
P_3	3	2
P_4	4	1
P_5	2	5

P_2	P_5	P_1	P_3	P_4

0 1 6 16 18 19

$t_{ob} = (0 + 1 + 6 + 16 + 18) / 5 = 8.2$ msn

5.1.2.6.Döngülü Planlama (Round Robin - RR) Algoritması

Her görev küçük bir AİB zaman dilimini alır. Bu zaman bittiğinde, görev kesilir ve hazır görevler kuyruğunun sonuna eklenir.

Örnek: P_1, P_2, P_3, P_4 görevleri aşağıdaki ardışıklık ile sunulmuş olduğunu varsayalım. Zaman dilimi 20 msn ise buna göre:

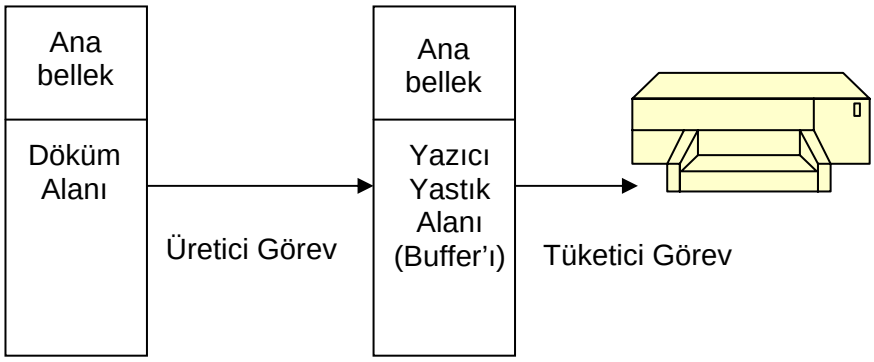
Görev	İşlem Zamanı (msn)
P_1	53
P_2	17
P_3	68
P_4	24

P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3

0 20 37 57 77 97 117 121 134 154 162

5.1.3. Birlikte Çalışan Görevler

İşletim sisteminde aynı zamanda çalışan görevler ya bağımsız ya da birlikte çalışan görevlerdir. Görevin çalışması diğer çalışır durumdaki görevleri etkilemezse, bu görev *bağımsız görev* dir. Bu tür görevlerin, herhangi bir değeri, geçici veya kalıcı olarak diğer görevlerle ortaklaşa kullanılmazlar. Buna karşın, birbirini çalışma süresince etkileyen görevler **birlikte çalışan görev**lerdir. Ortak verileri kullanırlar.



Şekil : Birlikte çalışan iki görevin gösterimi

Birlikte çalışmanın sebepleri

- *Bilgi paylaşımı:* Farklı kullanıcılar için aynı bilgiler gerekebilir, bu nedenle bu kaynaklara paralel erişim sağlanmalıdır.
- *Bilgi-İşlem hızının yükseltilmesi:* Problemin çözümünü gerçekleştirmek için problem alt problemlere bölünebilir ve bu problemler paralel çalışabilirler.
- *Modüllere parçalanma:* Sistemin modüllere parçalanması işlemi ve kontrolü kolaylaştırır.
- *Kullanıcı rahatlığı :* Her bir kullanıcı aynı zamanda birden fazla problemi çalıştırma isteğinde bulunabilir.
- *Görevlerin ortak çalışması :* Görevlerin, iletişim ve erişim işlemlerinde zaman uyumunu (senkronizasyon) sağlayan bir mekanizmanın olması gerekir.

Birlikte çalışan görevler problemine, üretici-tüketici problemi gibi bakılabilir. Üretici ürünleri tampona gönderir, tüketici ise alır, tampon dolu olduğunda üretici beklemeli, boş olduğunda ise tüketici beklemelidir.

```
saat()
{ vuru++;
  if (vuru == 100)
  { vuru = 0;
    saniye++;
    if(saniye == 60)
    {  saniye = 0;
       dakika++;
       if (dakika == 60)
       {  dakika = 0;
          saat++;
          if(saat == 24)
           saat = 0 ;
        }
    }
  }
}
```

Şekil: Saat örneği üzerinde birlikte çalışan görevler ve işletim bütünlüğü

Örnek: Sınırlı tampon için ortak bellek kullanımı :

```
#define BUFFER_SIZE 10
Typedef struct {
    . . .
} item;
item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

Ortak veri alanının kapasitesi BUFFER_SIZE-1 kadar elemandır. Burada, "*in*" değişkeni tablodaki ilk yahut da sonraki boş yeri, "*out*" ise son dolu yeri gösterir. Bu bir kuyruk oluşturur. "*in=out"* ise tampon boş, "*in+1=out"* ise bellek tamponu doludur.

5.2.GÖREVLER ARASI İLETİŞİM (IPC – Inter Proccess Communication)

Birlikte çalışan görevler ortak bellek alanı kullanarak iletişim sağlayabilirler. Ayrıca, görevler arasında iletişim sağlanmasının diğer bir yolu ise, görevler arası iletişim IPC aracının kullanılmasıdır. IPC araçları, en az iki işlem yapmaktadırlar: (1) Send (P message), (2) Receive (Q message) işlemleri. IPC nin haberleşme sistemi, görevlerin birbirleriyle ortak değişkenler kullanmadan iletişim oluşturabilmesidir.

Görevlerin gönderdiği haberler, sabit veya değişken olabilir. Yalnız sabit boyutlu haberler gönderilirse gereken fiziki yapı basitleşir, programlama daha zorlaşır. Değişken boyutlu haber gönderilirse, tam tersi fiziki yapı karmaşık olur.

Görevler arası iletişimi doğrudan yada dolaylı olarak gerçekleştirmek mümkündür:

1. **Doğrudan iletişim :** Haberi gönderenin ve alanın adları kesin gösterilmelidir. Doğrudan erişimde görevler arası iletişim yapısının özellikleri :
    1. İletişimde bulunan iki görev arasında otomatik gerçekleştirilir.
    2. Görevler yalnız iletişim oluşturdukları görevi tanıyabilir.
    3. Her çift görev arasında yalnız bir bağlantı oluşturulmalıdır.
    4. Bağlantılar genelde iki yönlü bazen ise tek yönlü olabilir.

Bu algoritmada kullanılan yöntem simetrik yöntemdir. Simetrik yöntemde haberi gönderen ve alan birbirlerinin adlarını bilmelidir. Eğer haber gönderen birim alıcının adını belirtirse ve haber alan ise haberin kimler tarafından gönderildiğini bilmezse bu yöntem asimetrik yöntemdir.

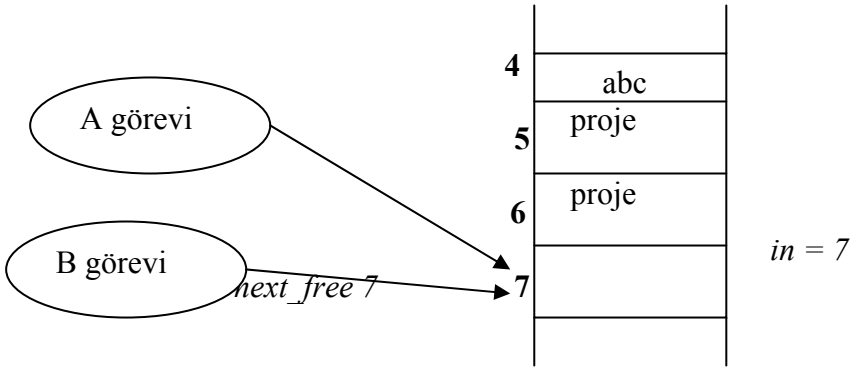
2. **Dolaylı İletişim :** Haberler posta kutusu ile alınıp verilir. İki görev yalnız ortak haberleşme kutuları bulunursa iletişim kurabilirler. Dolaylı iletişimin özellikleri :

- Bağlantı ikiden fazla görev arasında oluşturulabilir.
- Bağlantı yalnız ortak kutuları alan görevler arasında oluşturulabilir.
- İletişimi oluşturan her iki görev arasında birden fazla bağlantı olabilir. Bu bağlantıların her biri için bir bağlantı kutusu bulunur.
- Bağlantı bir yönlü yada iki yönlü olabilir. Haberleşme kutularını işletim sistemi kendisi oluşturduğu gibi, bir başka görevinde yeni haberleşme kutuları oluşturmasına, onlarla haber alıp göndermesine veya bunları silmene izin verir.
- İşletim Sisteminin oluşturduğu haberleşme kutuları bağımsızdır.

5.3. ZAMAN UYUMLAMA

Aynı zamanda birkaç görevin aynı kaynağa erişmesi durumunda, işletim bütünlüğünün sağlanması zaman uyumuna (Senkronizasyon) gereksinim vardır. Görevlerin ortak kaynağı (bellek yada kütük) farklı zamanlarda kullandıkları zaman görevler arası iletişimi sağlamak kolaylaşır. Ama görevler aynı zamanda ortak kaynağa erişmek isterlerse, çeşitli problemler çıkabilir.

Örnek: A ve B görevleri kütüğün yazıcıdan çıktısını almak istiyorlar. *in* ve *out* ortak değişkenleri olsun. o*ut;* çıktısı alınacak kütüün adresi, *in* ise kütük için ayrılan sonraki boş adres gösterilir.



A görevi "*in"*'in değerini okur ve bu değerin yerel değişkenine yazar. O zaman ana işlem birimi A'nın çalışma süresinin bittiğini karar alır ve B görevi de kendi kütüğünün adını 7. adrese yazar. A görevi yeniden çalışmaya başladığı zaman *next_free* değişkenine bakar, kendi kütüğün adresini 7 yazar.

B'nin yazmış olduğu değeri silmiş olur. Böylelikle iki görevin aynı zamanda çalışması hataya yol açabilir ve bu işlemlerin sonucu hangi görevin daha önce çalışmasına bağlıdır. Böyle duruma *yarışma koşulu* denir.

Görevlerin birlikte çalışmasında hedef yarışma koşulunu çıkartmaktır. Bunun en basit yolu birden fazla görevin ortak verilere erişimine izin verilmemesidir.

Belli bir zaman diliminde görevin yarışma koşuluyla ilgisi bulunmayan işlemlerle meşgul olur. Bazı bölümlerinde ise yarışma koşuluna bağlı işlemleri gerçekleştirir. Görevin bu bölümüne *kritik bölüm* denir. İki görevin aynı zamanda kendi kritik bölümlerinde çalışmasını engelleyebilecek durum oluşturulursa yarışma koşulu ertelenmiş olur.

Görevlerin kullandıkları sistem kaynakları, kullanım açısından iki gurupta toplanır:

1.Paylaşılabilen kaynaklar : Bir görev tarafından tümüyle kullanılmadan diğer görevler tarafından kullanılabilen kaynaklardır (CPU, RAM, HDD).

2.Paylaşılmayan kaynaklar : Bir görev tarafından tümüyle kullanımı bitince diğer görevler tarafından kullanılabilen kaynaklardır (YAZICI, CD-R). Bu kaynaklara *kritik kaynaklar* adı da verilmektedir.

Kritik kaynakların görevler tarafından paylaşılması önlem alınmadan yapılamaz. Kaynağa erişmeden önce kaynağın kullanım durumunun sınanması gereklidir ve diğer görevlerin o anda kaynak üzerinde işlem yapmadıkların emin olunması gerekir. Bir görevin kritik kaynak üzerinde işlem yaptığı anda diğer görevlerinde böyle bir isteği olursa, bu işlemin kaynak serbest olana dek ertelenmesi gerekir. Kritik kaynağa erişmeden önce kaynağın uyumunun sınanması ve erişimin ertelenmesi görevler arası *zamana uyumlama* olarak ifade edilir.

Zaman uyumlamanın aynı kritik kaynağa erişim yapmak isteyen görevlerden en çok birini kendi kritik bölümüne girmesiyle gerçekleştirmek mümkündür. O zaman bu görev diğer görevlerin kendi kritik bölümlerine girmelerini engellemelidir. Bu işleme *karşılıklı dışlama* denir.

Birlikte çalışan görevlerin işlem bütünlüğünün korunması için karşılıklı dışlanmanın yanı sıra, karşılıklı tıkanmanında engellenmesi gerekir. Görevlerin birbirlerinin işlerini karşılıklı olarak sürekli engellemelerine *karşılıklı tıkanma* denir.

Görevler arası zaman uyumlamada uyulması gereken önlemler:

1. Karşılıklı dışlanmanın sağlanması: Birlikte çalışan görevlerden en çok biri aynı anda kritik bölüme girmelidir.
2. Karşılıklı tıkanmanın engellenmesi: Aynı anda ortak kaynağa erişmek isteyen en az bir görev sonlu bir süre içinde kritik bölüme girmelidir.
3. Kritik bölümün dışındaki görevler ortak kaynağa erişmek isteyen görevleri engellemelidir.

5.3.1. Zaman Uyumlama Yöntemleri

Görevler Arası Zaman uyumlama yöntemleri şunlardır:

1. Özel donanım desteği gerektirmeyen yöntemler
2. Donanım desteği gerektiren alt-düzey araçlar
3. Üst-düzey zaman uyumlama araçları

5.3.1.1. Özel Donanım Desteği Gerektirmeyen Yöntemler

Kesme Yapısının Kullanımı: Karşılıklı dışlamayı sağlamak amacıyla bir görev kritik kaynak bölümüne girerken, sistemi kesmelere kapatabilir.

```
di;                         di;
saniye1 = saniye;           saat = 0;
dakika1 = dakika;           dakika = 0;
saat1 = saat;               saniye = 0;
ei;                         ei;
```

Burada; di → kesme devre dışı, ei → kesme kullanılabilir.

Algoritmik Yaklaşım

A) Sıra değişkeni ile zaman uyumlama: Aynı kritik kaynağa erişen görevler programlamada kullanılır. Bunun için kaynağın kullanım sırasını belirleyen bir gösterge (sıra göstergesi) öngörülür. Ortak kaynağa erişmek isteyen bir görev kritik bölüme girişte bu sıra göstergesini sınar. Eğer gösterge, sıranın sınayan görevde olduğunu gösteriyorsa kritik bölüme girilip ortak kaynağa erişilir. Bunun tersi olduğunda sıranın kendisine gelmesini bekler. Dezavantajı görevleri, ortak kaynağa değişmez sırayla erişmeye zorlar.

GÖREV 1 :

```
while (sira!=1); /*bekle*/
kritik-bölüm();
sira=2;
```

GÖREV 2 :

```
while (sira!=2); /*bekle*/
kritik-bölüm();
sira=1;
```

B) Dekker Algoritması : Dekker algoritmasına göre birlikte çalışan iki görev aşağıda verilmiştir. Bu algoritmaya göre birlikte çalışan iki görevin, kritik bölümlere eşzamanlı olarak girmeleri, *koşul1* veya *koşul2* nin ilgili görevin kritik bölümüne girmesinden önce mutlaka *false* olmasından dolayı olanaksızdır. Karşılıklı tıkanma, *sıra*' nın kritik bölüme girmeden önceki karar kısmında içerik değiştirmemesi sayesinde önlenmektedir.

5.3.1.2.Donanım Desteği Gerektiren Alt Düzey Algoritmalar

A) Test-and-set türü komutların kullanımı : İkili değişkenleri sınama ve gerektiğinde kaynağın kullanımda olduğunu belirten değişkenin güncellenmesi gibi işlemler tek bir komutla yapılmaktadır.

GÖREV 1 :

```
bekle1: cmp durum, 0
        jz bekle1
        mov durum, 0

        (kritik-kesim-1)

        mov durum, 1
         :
         :
```

GÖREV 2 :

```
bekle2: cmp durum, 0
        jz bekle2
        mov durum, 0

        (kritik-kesim-2)

        mov durum, 1
         :
         :
```

B) İlkel Semafor İşleçlerinin Kullanımı: Görevler arasında karşılıklı dışlamayı sağlamak için *semafor* denen değişkenler kullanılır. Semafor "0" meşgul, "1" serbest (boş) değerlerini alırsa, buna *ikili semafor* denir. *P* komutuyla kaynak diğer görevlerin kullanımına kapanır, *V* komutuyla açılır.

P(S) ve V(S) işleçlerinin görevler arası zaman uyumlama kullanımı, kritik kesimlerin bu işleçlerle ayıraç arasına alınması yoluyla olur:

```
:
P(S);
(Kritik kesim)
V(S);
:
```

Bir görev kritik kesime gitmeden önce P(S) işlecini çalıştırır. S'nin içerdiği değere göre ya kritik kesime girer yada bu işleç üzerinde bekler. Kritik kesim çıkışında mutlaka V(S) işlecinin çalıştırılması gerekir. Bu tanıma göre S' nin salt 0 ve 1 değerlerini alabildiği görülür. 0 meşgul ve 1 serbest anlamlarını taşımaktadır.

5.3.1.3.Üst Düzey Zaman Uyumlama Araçları

Programlama dillerinin, eşzamanlı uygulamalar hazırlamak için ek komutları vardır. Bu şekilde hazırlanan eş zamanlı uygulamalar, alt düzey zaman uyumlama algoritmaları (dekker gibi.) ile hazırlanan uygulamalara nazaran daha kolay tanımlanır. Fakat, çalışma hızları düşüktür.

A) İleti Aktarımı, send/receive Komutları: Görevler arasında aktarılan verilere ileti denir. *send* komutu ileti göndermek, *receive* komutu ise diğer bir görevden ileti beklemek ve almak için kullanılır. *Hedef* ve *Kaynak* adlı parametreler alan ve gönderen görevlerin kimlikleridir. İleti ise aktarılan veri öbeğini belirler.

send (hedef, ileti)

receive(kaynak,ileti)

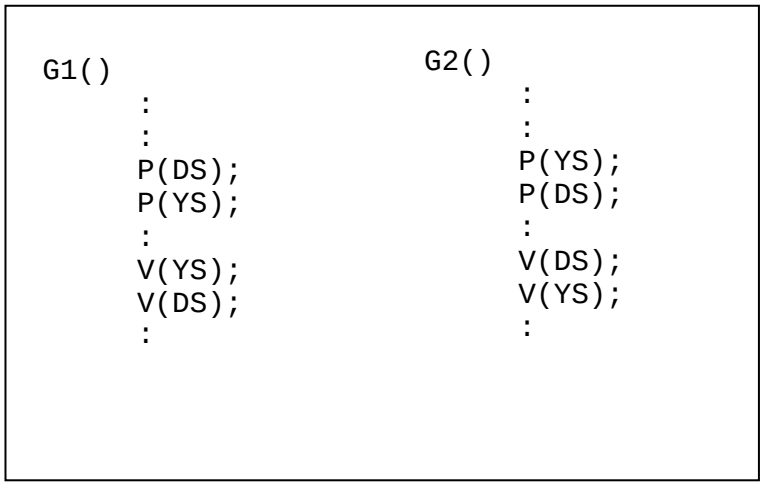
İletilerin kaynak kimlikli görevden hedef kimlikli göreve aktarımı, değişik yollarla gerçekleştirilebilir :

1. Kaynak görevin bellek alanından, doğrudan hedef görevin bellek alanına aktarılması
2. İşletim sistemine ilişkin tampon (*buffer*) alanlarının kullanılması
3. Posta kutularının (*pipe' ler*) kullanılması.

B) Monitor Kullanımı: Monitor, birlikte çalışan görevlerin , ortak kaynağa erişim yapan yordamlarının (kritik kesimlerinin) toplandığı kümeye verilen addır. Görevlerin ortak kaynağa erişimlerini denetim altında tutmanın bir yolu da bunların, kritik kaynaklara merkezi bir denetim altında erişmelerini sağlamaktan geçer.

5.3.2. Kilitlenme

Kilitlenme, bekleme durumundaki görevlerin, hazır görev durumuna geçebilmek için bir diğerinin sağlayacağı koşulu karşılıklı beklemelerine verilen addır. Görevlerin her birinin bekler olması ve çalışabilmek için bir diğerinin işletilmesinin gerekmesi durumu, görevlerin hiçbir zaman gerçekleşmeyecek bir koşulu beklemeleri sonucunu doğurur.



Şekil: Kilitlenen görevler

Görevler arasında tıkanma (kilitlenme) aşağıdaki koşullarda ortaya çıkabilir:

1. Karşılıklı Dışlama : Bir kaynağın aynı anda yalnız bir görev tarafından kullanılabilmesi.
2. İstem üzerine kaynak atama : Görevin gereksediği kaynakları, teker teker işletim aşamasında elde edilmesi.
3. Atanan kaynakların, görevler serbest bırakmadıkça geri alınamaması
4. Döngüsel Bekleme : Bir görevin elinde tuttuğu kaynaklardan bir yada daha çoğunun, bu görevle aynı döngüsel işletim zinciri içinde yer alan diğer görevlerce istenmesi.

Bu koşulların tamamı aynı anda bulunması tıkanmayı oluşturur. Tıkanmayı önlemek için bu durumlardan en az birini ortadan kaldırmak gerekir. Ortadan kaldırma yöntemlerinden birisi tıkanmaya neden olan görevlerin yok edilmesi veya sistemin yeniden başlatılmasıdır.

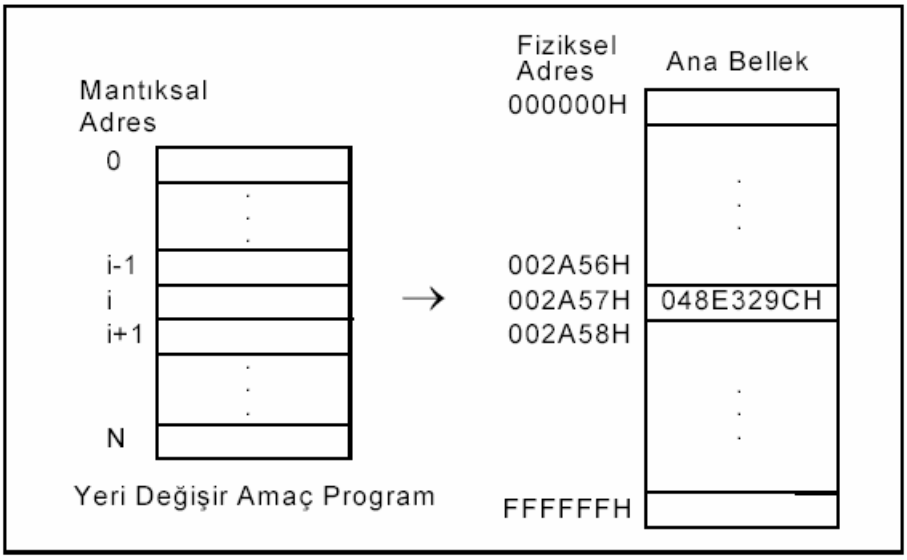
6. BELLEK YÖNETİMİ

Bilgisayar sistemini oluşturan temel birimlerden biride bellektir. Programların ve işlenen verilerin bellekte yer alacakları konumların belirlenmesi, düzenlenmesi, izlenmesi gereken alanların sağlanması, bu alanların dışına taşmaların denetlenmesi gibi işlevler bellek yönetimi kapsamındadır.

Bellek, Ana ve İkincil bellek olmak üzere ikiye ayrılır. Ana bellek; aynı zamanda aktif bellek, yarı iletken bellek, RAM bellek adlarıyla da anılır. Programların ve verilerin işlem aşamasında yer aldığı, ana işlem biriminin de doğrudan erişebildiği bellektir.

Ana bellek bir sözcük dizisi şeklindedir. Her sözcüğün bir adresi ve bir de içeriği vardır. Ana bellekte bir sözcüğün adresi, bu sözcüğe erişimde, adres yoluna yüklenen konum değeridir. Bu değer, ilgili sözcük içeriğinin ana bellekte bulunduğu fiziksel konumu gösterdiğinden fiziksel adres olarak nitelenir.

Ana bellek sözcükleri, komut kodları ile bu komutların işlenenlerini tutan öğelerdir. Yazılan kaynak programlar içerisinde işlenenler, kullanıcıların özgürce belirledikleri simgesel adlarla anılırlar.

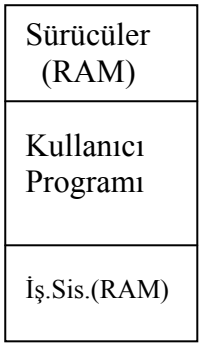


Şekil: Fiziksel ve Mantıksal adres uzayları

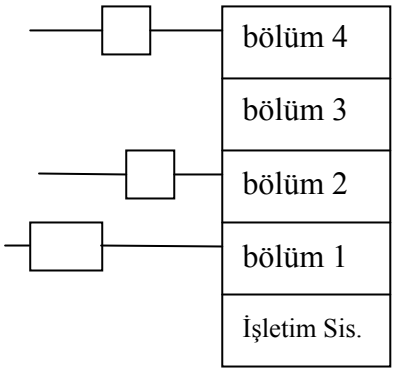
6.1. Tek Programlı ve Çok Programlı Sistemlerde Bellek Yönetimi

En basit bellek yönetim sistemi, her bir zaman dilimi içinde bellekte bir görev bulunması ve tüm bellek alanını bu görevin kullanmasıdır. Bu sistemlerde bellek, işletim sistemi ve tek görev arasında paylaşılır. Kullanıcının terminalden aldığı komut ile, işletim

sistemi gereken programı belleğe yükler. Görev çalışmasını bitirdikten sonra, sistemin gönderdiği mesajla kullanıcı yeni görevi belleğe yükler.



Fakat, bilgisayarın verimli çalışması için bellekte aynı zaman diliminde birden fazla görevin çalışması gerekir. Bunun en basit yolu belleğin "*n"* sayıda bölümlere ayrılmasıdır. Sistem çalıştırılmadan önce bölümler (bölümün sınırları) kaydedilir. Görevler geldiği zaman bölümün boyutuna uygun kuyruğa yerleştirilir.



Bu sistemler OS 360 işletim sisteminde uygulanmıştır ve bu mimariye **MFT** (Multi programming Disk Fixed Number of Task) denir.

6.2. Boş Bellek Alanlarının Aranması

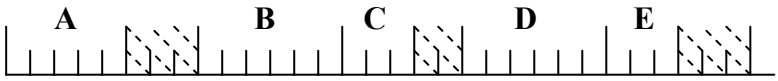
Boş bellek alanlarının aranması için genelde 3 yöntem kullanılır:

1.İkili haritalama (Bit-map)
2.Bağlaçlı Liste
3.Komşu Sistemler (Buddy System)

1. İkili Haritalama Yöntemi : Bu yöntemde bellek mantıksal olarak sabit boyutlu küçük birimlere bölünür. Bölümün boyutu birkaç sözcük uzunluğundan, birkaç KB'a kadar olabilir.

Bu bölümlerin boş yada dolu olması hakkında bilgiler ikili haritada gösterilir. Haritanın her biti bir bellek bölümünün durumunu gösterir. Bölüm boşsa bit = 0, doluysa bit = 1 değerini alır.

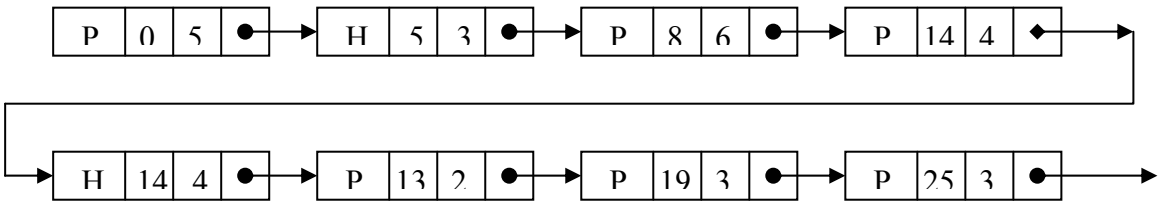
İkili haritalarda, gelen görev k boyutlu ise, ardışık k sayısıda 0 (boş) bellek yerinin aranması gerekir. Bu ise, ikili haritalamanın yavaş olmasına yol açtığından pratikte az kullanılmasına neden olur.



1 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1
1 0 0 1 1 1 1 1
1 1 1 1 0 0 0 1

Bellek bölümlerinin büyük boyutlandırılması, haritanın küçülmesine, bununla birlikte bellek alanının verimsiz kullanımına neden olur.Bölümler küçük boyutlu alınırsa, harita büyür ve arama işlemlerinin süresi uzar.

2. Bağlaçlı Liste Yöntemi : Boş ve dolu bellek alanları için yerleştikleri adreslerle birlikte *bağlaçlı liste* oluşturulur.



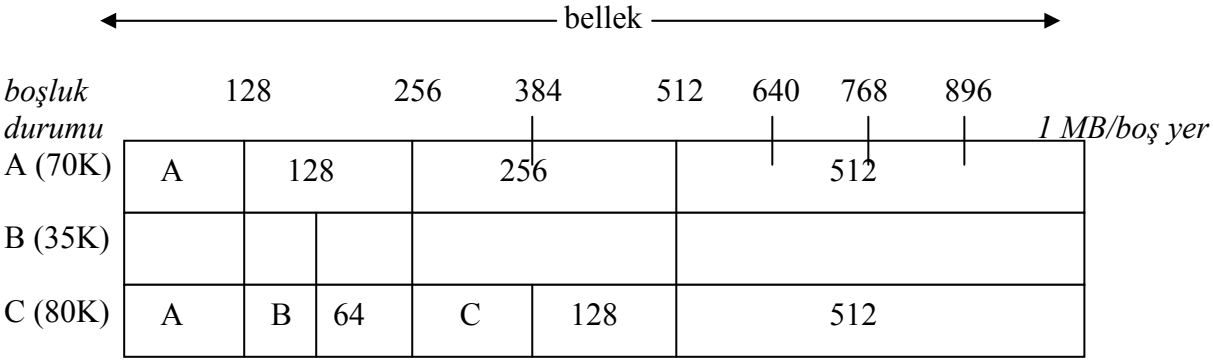
NOT : *P =Dolu, H = Boş bellek alanını göstermektedir.*

Bellek yöneticisi görev için gereken bellek alanının boyutunu bildiği takdirde, bu yöntemde sadece bu boyuta uygun bellek yeri bulmak amaçlanır. Uygun bellek yeri bulmak için birkaç algoritma vardır:

1. İlk uygun yer bulma (first fit) : Bellek yöneticisi listeyi tarar ve yeterli boyutta boş yer bulunan ilk yere görevi yerleştirir. Bu algoritma çok hızlıdır. Çünkü, arama en kısa yolla gerçekleştirilir. Ancak en uygun olan belleğe değil, ilk rast gelinen uygun bellek alanına yerleşim yapılır.
2. En uygun yer bulma (best fit) : Görevin boyutuna en yakın olan boş yer aranır. En verimli algoritmadır. Ancak yavaştır.

3. Hızlı uygun yer bulma (quick fit) : Bu algoritmada, çok sık kullanılan bellek alanı boyutları, ayrıca listelenir. Belleğe yerleştirilecek görev için uygun yer bu yeni listede aranır.

3. Komşu Sistemlerde Bellek Yönetimi: Komşu sistem yönetiminde bellek yöneticisi 1,2,4,.....,2^k boyutlu boşluklar listesi oluşturur. Örneğin 1 MB'lık bellek alanı için 21 liste oluşturulacaktır.



Başlangıç durumunda tüm bellek boştur ve yalnız 1MB'lık listede 1MB'lık boş yer bulunmaktadır.

70 K'lık A görevi belleğe aktarıldığında bellek yöneticisi bu görev için 128K'lık boş yer ayıracaktır. Öyle bir liste henüz oluşturulmadığından ardışık olarak 256K ve 512K'lık listeler aranacak ancak onlarda yoktur. Bu sebeple 1MB'lık alanın 512, 256 ve 128'lik alanlara bölecek ve A görevi ilk boş 128 K'lık alana yerleşecek.

B görevi içinde 64K'lık yer gerekir. 64K'lık listemiz olmadığından 128K'lık blok ikiye bölünür. İlk 64K lık yere B görevi yerleştirilir. Benzer şekilde C görevi yerleştirilir.

A geri gönd.	128	B	64	C	128	512	4
D (60)	128	B	D	C	128	512	3
B geri gönd.	128	B	D	C	128	512	4
D geri gönd.	256			C	128	512	3
C geri gönd.	1 MB						1

Görev bellekten çıktığında, kalan yerler birleşip 2^k alan oluşturuyorsa, bellek alanları birleştirilir. Yönetim avantajı, sadece 2^k 'lık listeler aranmasıdır. Buda hızlı çalışmasına neden

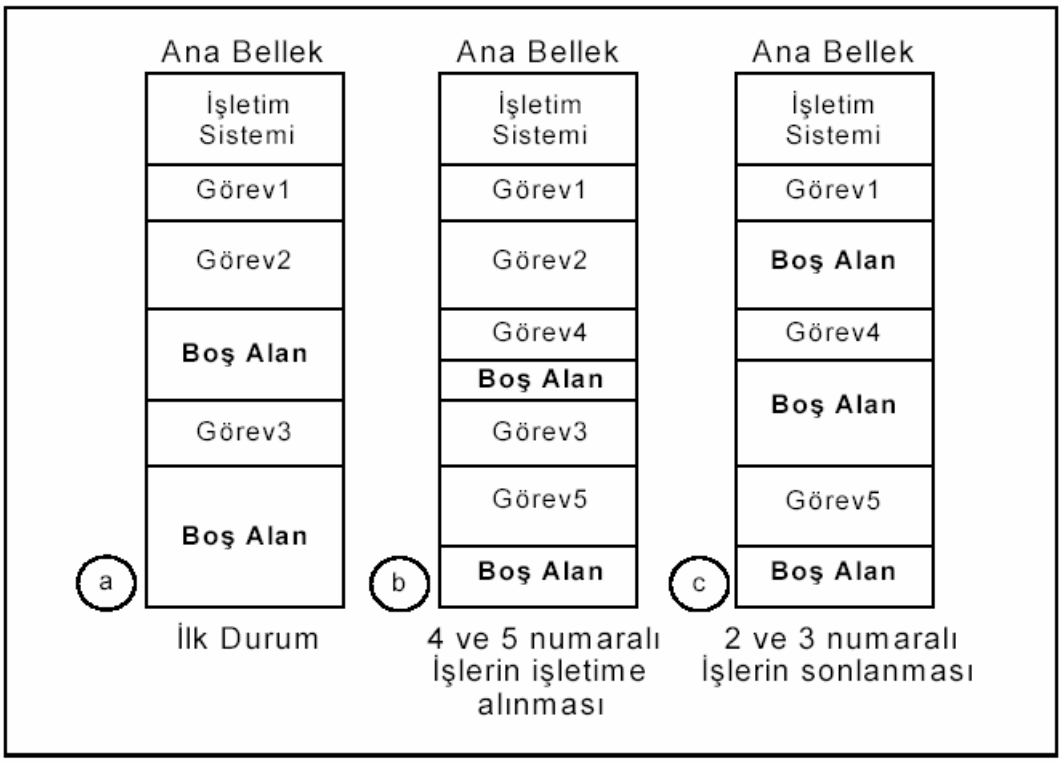
olur. Ancak bellek verimsiz kullanılır. Her görev için ayrılan alanlarda boş yerler oluşabilir. Buna *iç parçalanma (integral fragmentation)* denir.

6.3. Belleğin Parçalanma Sorunu Ve Bitiştirme İşlemi

Bellek parçalanması, bitişken alanların görevlere atanan bölümlerle, zaman içinde ufalanması olarak bilinir. Bu sorun, kullanılan bölümler arasına sıkışmış, işletim için bekleyen görevlerin gereksinimini karşılayamayan boş alanların varlığıyla ortaya çıkar. Bunun sonucu, bellekteki boş alanların toplamı, gereksenen sığaları karşılıyor olmasına karşın yeni görevlere yer sağlayamaz.

Görevlere gereksedikleri belleğin tümüyle ve bitişken bir bütün olarak, işletime sunuş aşamasında sağlanması ve sağlanan bu alanların konumlarının işletim sırasında değiştirilmesi parçalanma sorununun temel nedenleridir.

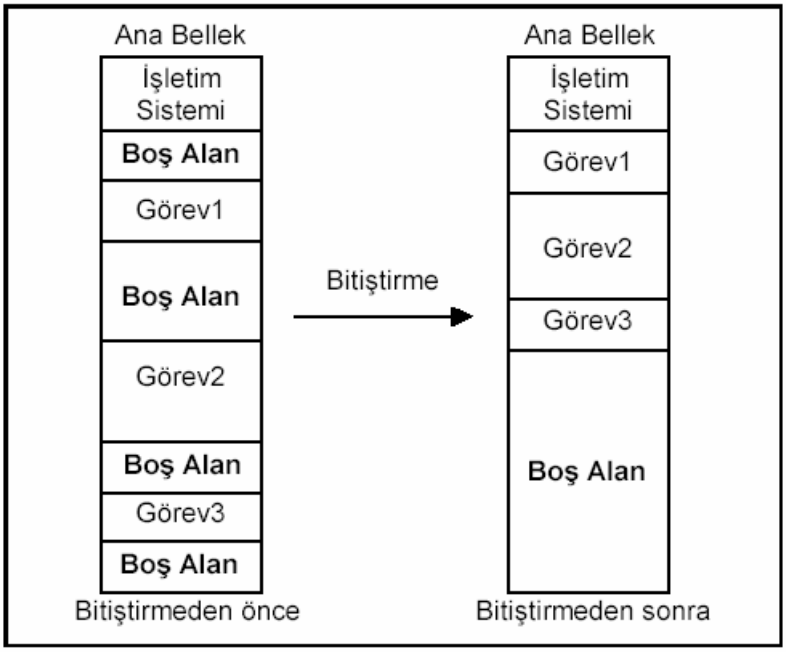
Bölüm içi yararlanılamayan boş alanlar iç parçalanma, bölümler arasında kalan boş alanlar ise dış parçalanma kapsamında düşünülür.



Şekil: Ana Belleğin Parçalanması

Bitiştirme; tüm bellek alanına dağılmış durumdaki bölümleri, yerlerini değiştirerek yan yana yerleştirme ve bu yolla bölümler arasında kalan boş alanları da yan yana getirerek tek bir bitişken boş alan oluşturma işlemine denir.

İlgili olduğu görevin işletimi sürerken bir bölümün yerinin değiştirilmesi kolayca ve hiçbir önlem alınmaksızın yapılabilen bir işlem değildir.



Şekil: Bitiştirme İşlemi

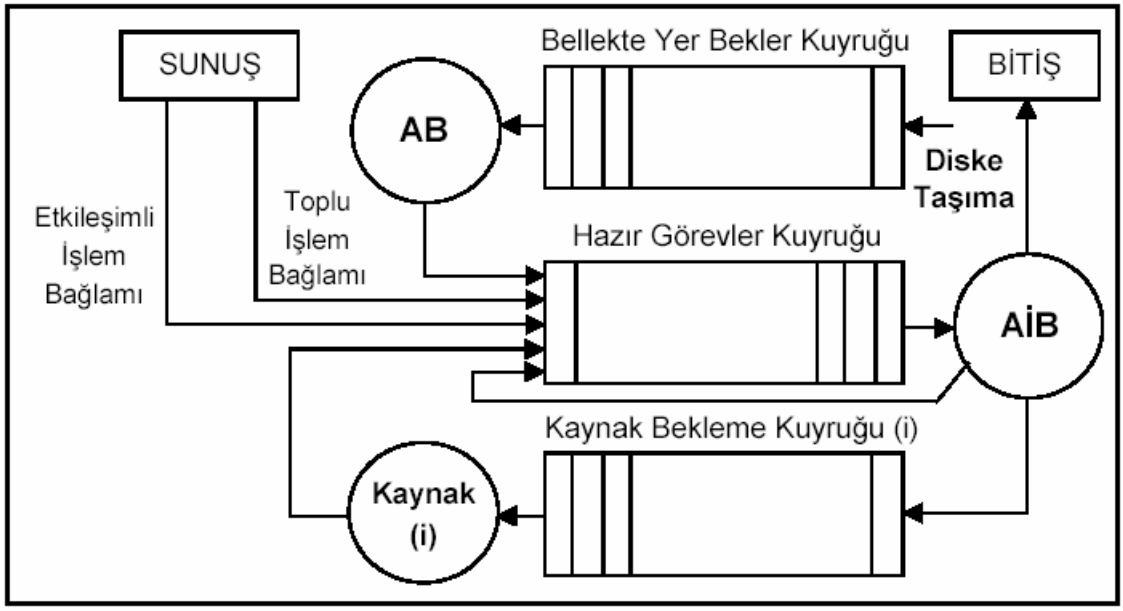
6.4. Diske Taşıma (Swapping - Takas)

Zaman paylaşımlı sistemlerde diskte yerleşen görevler dizisi gerektiğin ana belleğe çağrılır. Görevlerin disk ile ana bellek arasında değiş-tokuşuna *takas (swap)* denir.

Kimi durumlarda bitiştirme işlemleri de bellekte gerekli büyüklükte boş bitişken alan oluşturmak için yeterli olamaz. Sisteme sunulan iş, o an sistemde işletimde olan işlemden daha öncelikli ise, görev tanımlarının yapılarak hemen işletime alınması gerektirir.

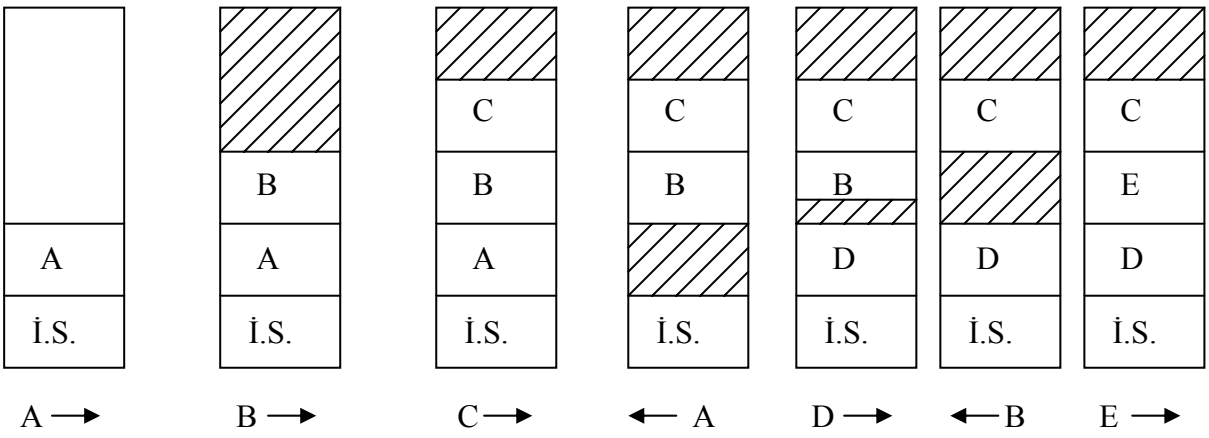
Kimi zaman öncelik görevlerin, işletimleri sonradan tamamlamak üzere geçici olarak diske taşınması boş bellek alanı yaratmak için başvurulan bir yol olabilir. Bu yolla açılan boş bellek alanları yeni görevlerin tanımlanabilmesine ve özellikli işlerin işletimlerinin bir an önce başlatılabilmesine olanak sağlar.

İşletimi tamamlanmamış bir görevin, daha öncelikli görevlere ana bellekte yer açmak üzere, geçici olarak diskte, bu amaçla öngörülen alanlara taşınmasına diske taşıma olayı olarak nitelendirilir.



Şekil: Diske taşıma ve bellekte yer bekler kuyruğu

Takas MFT sisteminde de uygulanabilir, ama onun değişken bölümlü çok programlama da (MVT) kullanımı daha verimlidir.



Değişken programlama (MVT) sisteminde, belleğe yazılan görevler arasında boş alanlar olabilir ve bellek sıkılaştırılması yöntemiyle bu boş alanlar birleştirilir. MVT sisteminde göreve yer ayrılırken bu görevin yeni verilerle genişleyebileceği ve büyüyebileceği göz önüne alınır.

6.5. Sayfalı Bellek Yönetimi

Sayfalı bellek yönetiminde görevlerin mantıksal adres evrenleri, birbirini izleyen, eşit uzunlukta parçalardan oluşur. Bu parçalar program sayfası olarak adlandırılır. Ana belleği oluşturduğu varsayılan sayfalar ise bellek sayfaları olarak adlandırılır

Sayfalı bellek yönetiminde, görevlerin her program sayfasına bir bellek sayfası atanır.

Mantıksal adres evreni içerisinde bitişken olarak yer alan program sayfalarının ana bellekteki karşılıklarının bitişken olma koşulu aranmaz.

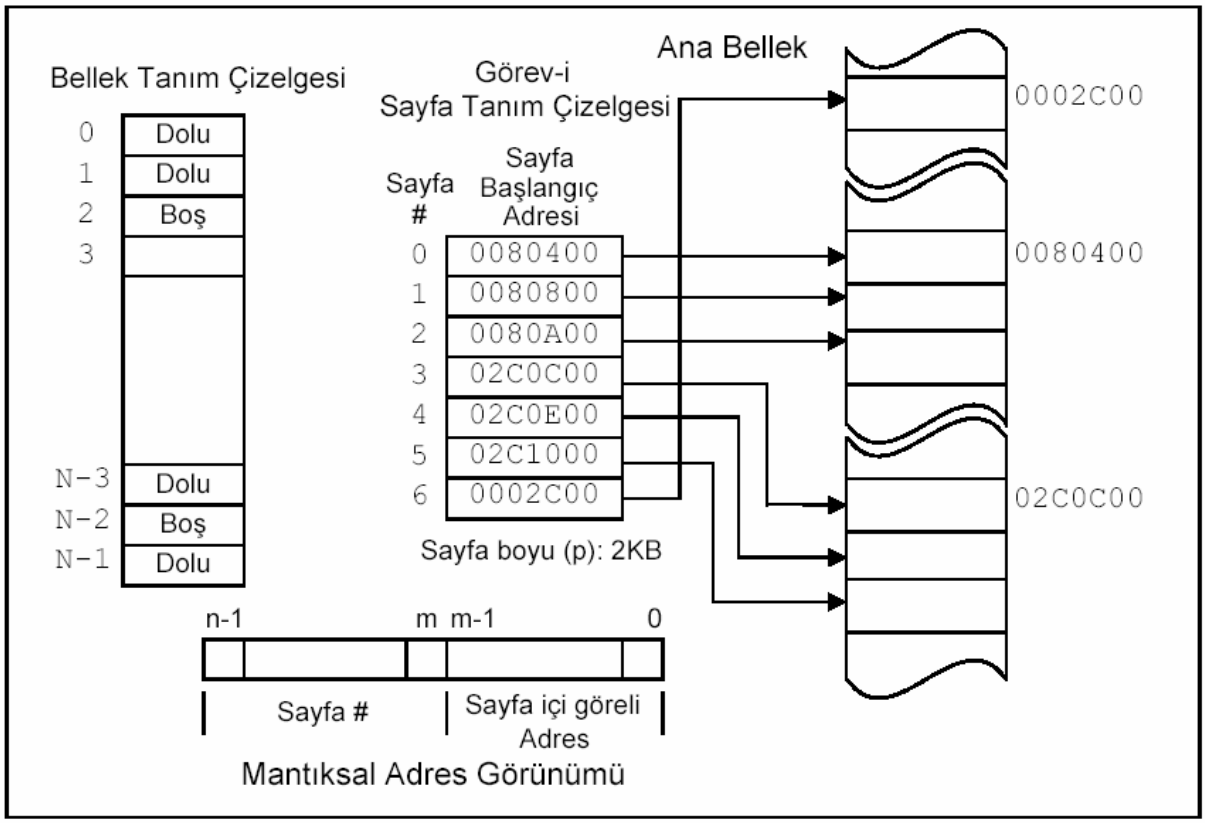
Hangi program sayfasının, hangi sayfasında bulunduğunu belirleyebilmek amacıyla, her görev için sayfa tanım çizelgesi tutulur.

Sayfa tanım çizelgeleri, ilgili görevin program sayfalarının yer aldığı bellek sayfası giriş adreslerini tutarlar.

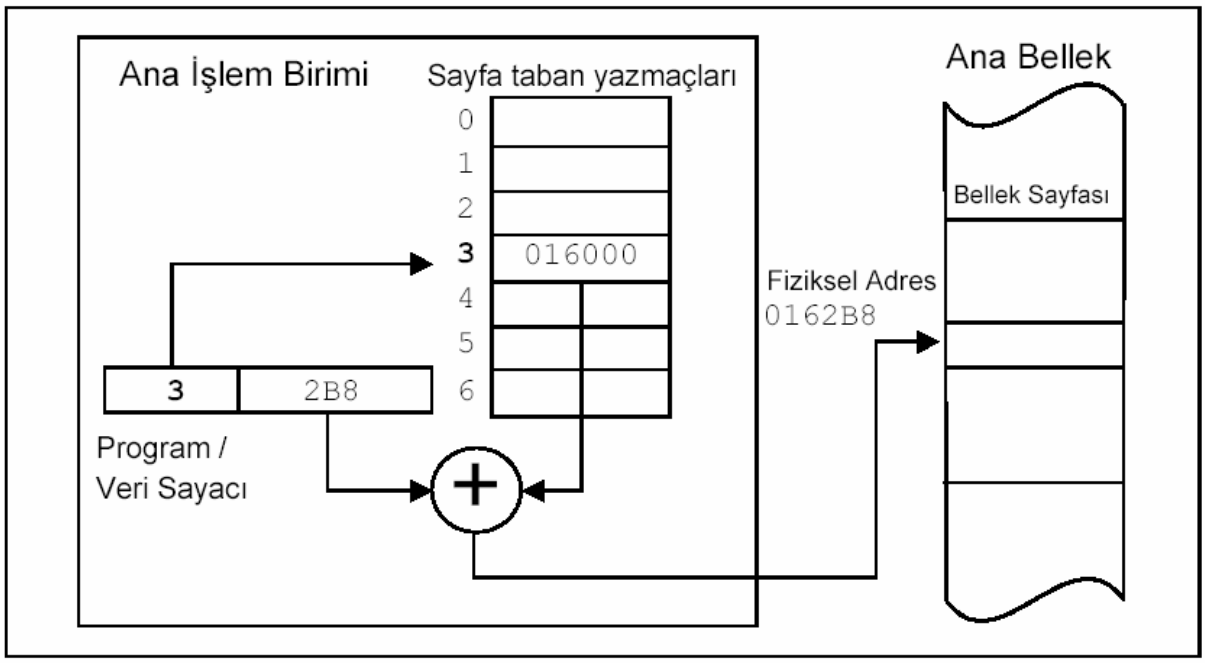
Her görev için ayrı ayrı tutulan bu sayfa tanım çizelgelerinin yanı sıra, bir de, tüm ana bellek için bellek tanım çizelgesi tutulur. Bu çizelge içinde hangi bellek sayfasının dolu, hangisinin boş olduğu bilgisi bulunur.

Görevler işletime alınacağı zaman önce bu çizelge taranır. Görevin gereksediği sayıda boş bellek sayfası belirlenir.

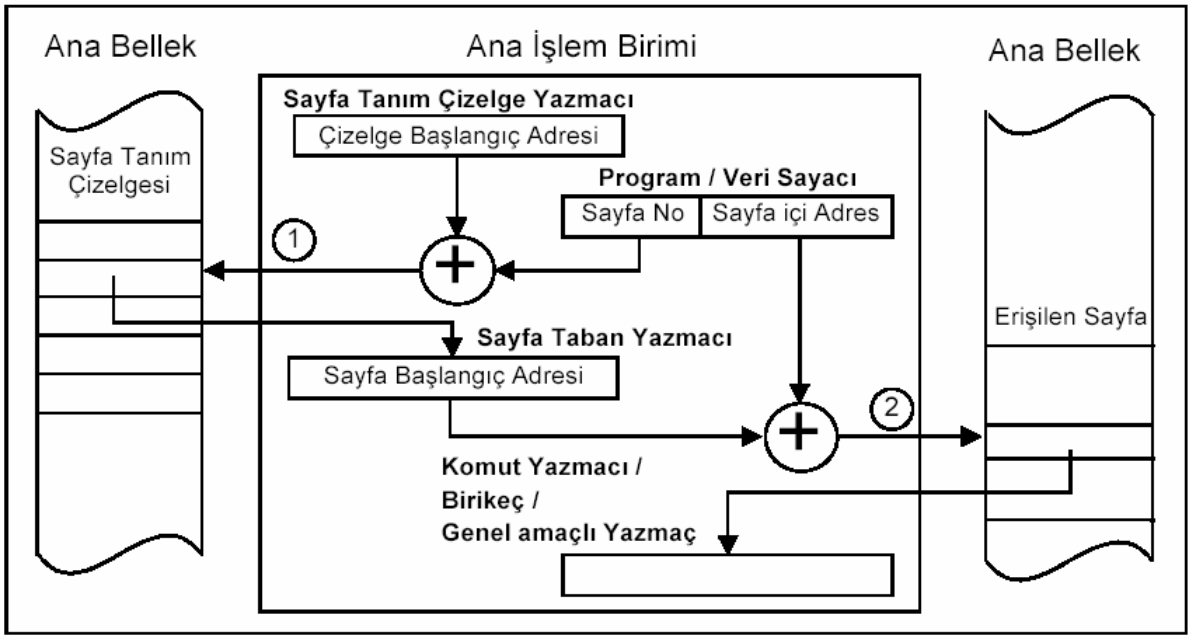
Görevin sayfa tanım çizelgesi oluşturularak içeriği, belirlenen bu boş sayfa giriş adresleriyle güncellenir. Bellek tanım çizelgesinde, kullanılan fiziksel sayfalar dolu olarak işaretlenir.



Şekil: Görev sayfa ve bellek tanım çizelgeleri

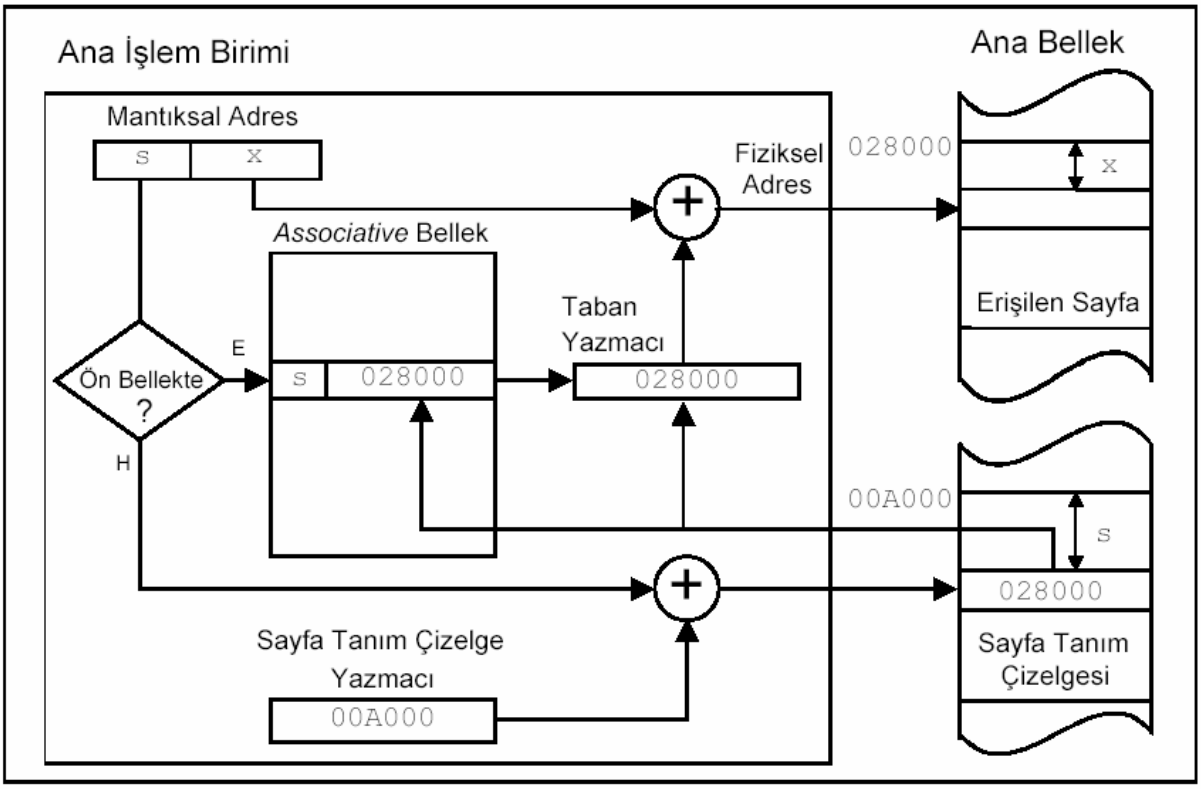


Şekil: Sayfalı bellek yönetimi adresleme mantığı



Şekil: Sayfa tanım çizelge yazmacının kullanımı

Çok hızlı erişime olanak veren bellek birimleri ön bellek (cache) olarak adlandırılır. Sayfa tanım çizelgesinin tümü, yine sıradan bellek kesiminde tutulurken ön belleğin, son kullanılan birkaç sayfa başlangıç adresini içermesiyle yetinilir. Ancak bu durumda adresleme sürecinin başında erişilen sayfa başlangıç adresinin ön bellekte bulunulup bulunmadığının sınanması; sayfa başlangıç adresi ön bellekte değilse, ön belleğe taşınmasının sağlanması gereklidir.



Şekil: Sayfalı bellek yönetimi ve ön bellek kullanımı

Bu hareketle, sayfalı bellek yönetiminin uygulandığı sistemlerde, belleğe erişimi bu sınamayla aynı anda gerçekleştirebilen associative (ilişkisel) tür bellekler kullanılır. Bu tür belleklerde, sözcüklere adresleri ile erişmek yerine doğrudan içerik değerleri ile erişilir.

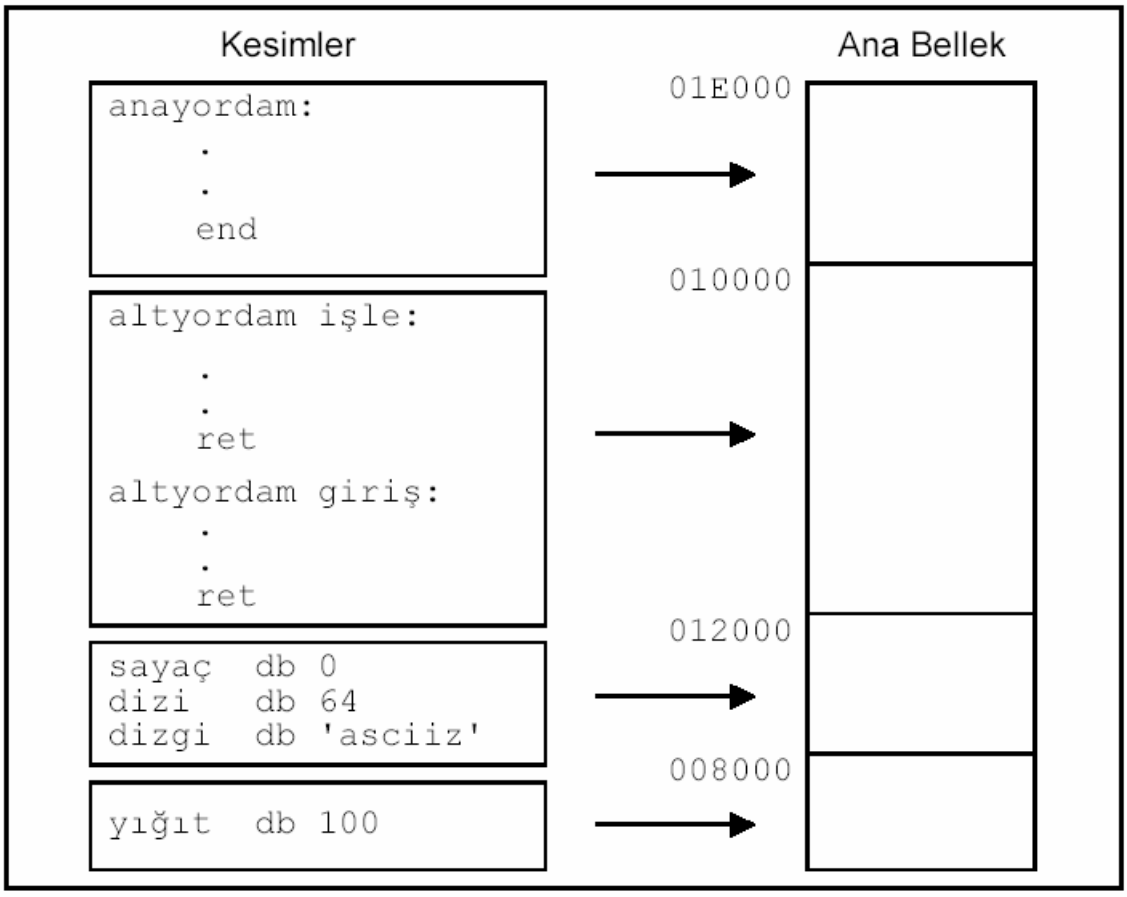
6.6. Kesimli Bellek Yönetimi

Görevlere atanacak bitişken bellek parçalarını küçültmenin bir yolu da program adres evrelerini kesimlere ayırmaktır.

Kesimler, program içinde, içerikleri yönünden mantıksal bütünlüğü bulunan parçalara verilen addır.

Programları oluşturan bağımsız kesimlerin birbirleriyle bitişken olarak düşünülmeleri de gerekmez.

Kesimlerden oluşan program içindeki adresler, sayfalama yöntemine benzer biçimde kesim kimliği ve kesim içi adres olarak iki bileşenden oluşur.



Şekil: Kesimler ve ana bellekte yer aldıkları konumlar

Kesimleme (Segmentation-Bölümleme): Sayfalama, belleğin verimli kullanılmasını sağlamakta idi. Belleğin verimli kullanılması bellek alanlarının çok kullanılması, boş alan bırakılmamasıdır. Adresleme yapılarak boş alanlar bulunabilir. Bunlar programın yapısına (içeriğine) bağlı olmayan (adreslemeler) işlemlerdir. Kesimleme ise, programın içeriğine bağlıdır. Yığınlar, kütükler, global değişkenler, yerel değişkenler gibi programda mantıksal bölümler mevcuttur. Programla verinin aynı anlamda kullanılması burada mantıksal açıdan iyi değildir. Ortak kullanılan veriler için kütükler oluşturulur. B programı A programının içine erişebilir (aynı kütüğü kullanmak için). Bu fazla hacim gerektirir, program büyür, verim düşer.

- Sayfalama, belleğin mantıksal yapısına bağlı olmadan parçalamadır.
- Kesimleme ise programın mantığına uygun belleğin parçalanmasıdır.

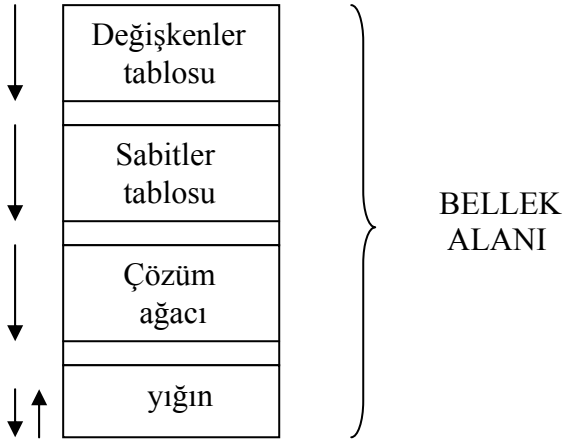
Bellek bir boyutlu bir dizidir. Kesimleme işleminde bellek iki boyutlu bir matris gibi kullanılır. Örneğin assemblerda kod, segment ve data segment vardır. Tüm segmentler için başlangıç adresleri 0 (sıfır) ‘ıncı adrestir. Bu mantıksaldır. Fiziksel adreste her bir kesimin taban adresi vardır.

Kesimlerde sayfalara bölünür.

Kesimin adresi + Sayfanın adresi + Sayfa içinde kaymalar

Çoğu programlar veya problemler bellekte birkaç ayrı ayrı alanların olmasını gerektirir. Örneğin derleme süresince aşağıdaki tablolar oluşturulabilir:

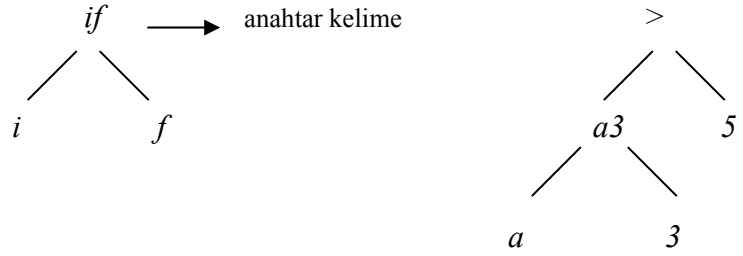
- Değişkenlerin adları ve özellikleri tablosu
- Sabitler tablosu
- Problem analizi için gereken çözümler aracı
- İç yordamların (prosedürlerin) çağrılması için yığın



Çözüm ağacı :

if a3 > 5 → (Burada kurallar vardır. *if* ' den sonra boş bırakılamaz, *then* , *if* 'den önce gelmez gibi)

Derleyici önce *i* harfini okur. Daha sonra *f* 'yi okur. i*f* adlı bir kelimenin bulunup bulunmadığına bakar. Sonra boşluk konulur. Bu şekilde cümleler çözümlenerek ağaçlar oluşturulur.



a3 gibi değişkenler bulundukça değişkenler tablosuna eklenir.

5 gibi sabitler bulundukça sabitler tablosuna eklenir.

Derleme süresince bu tablolar genişler, ağaç da büyür. Yığın, diğerlerinden farklı olarak dolar ve boşalır.

Her bölüm için ayrı bir yer ayrılır. Bu yer dolarsa tablolar birbiri içersine yerleştirilebilir (tablolar pointerlarla gösterilir). Buda karmaşıklığa neden olur. Bir boyutlu adres kullanmanın böyle bir dezavantajı vardır. bunu çözmek için bellek iki boyutlu adreslenmelidir.

Burada bellekte her bir tablo için farklı alanlar tahsis edilir. Her birinin başlangıç adresi vardır. Böylece gerektiğinde yalnız uygun kesimler kullanılır. Diğer kesimlerle birbirine karışma olmaz. Çünkü hepsinin başlangıcı 0 (sıfır) 'dır.

Baktığımız tablolar için bellek alanı yeterli olmazsa bu durumda bir tablo diğerinin üzerini örtmüş olacaktır. Problemin kesin çözüm yolu belleğin bağımsız adres alanları kesimlerine bölünmesidir. Her kesim 0. adresten başlayarak belli birimlere kadar ardışık ve doğrusal alanları içerir. Kesimin boyutu programın çalışması süresince değişebilir. Kesimler çok büyük olduğu için onların dolma veya taşma olasılığı çok düşüktür. Kesimlemeyi gerçekleştirmek için program iki boyutlu adresleme kullanır.

Mantıksal adres = kesim no + kesim içerisinde kayma

NOT : Mantıksal adresler aynı olabilir, 0 'dan başlar. Bunlar fiziki olarak farklı adreslere karşılık gelir.

Örnek : Plaka no verilirken

ADANA
1
2 100 Arabadan fazla olduğunda diğerlerini kaydırmak gerekir.
⋮
100
⋮
ADIYAMAN

0 'dan başlayıp ardışık verilse binlerce oluşur. Bunlar içinden aramak zordur. Fakat iki yararı vardır:

a. Daha kolay bulma
b. Birbirine taşmayı önleme

Her kesim mantıksal bir varlıktır. Kesim dahilinde prosedürler, diziler, yığınlar, değişkenler bulunabilir ama genelde kesim çeşitli verilerden oluşamaz.

Kesimlerin kullanımında bir üstünlük kesim dahilinde yordamların çalıştırılması kolaylaşır. Örneğin *n* kesimindeki yordam genellenirse bu değişme diğerlerini etkilemeyecek. Kesimleme ortak yordamlar kullanımını da kolaylaştırır. Buna örnek olarak kütüphaneleri göstermek mümkündür.

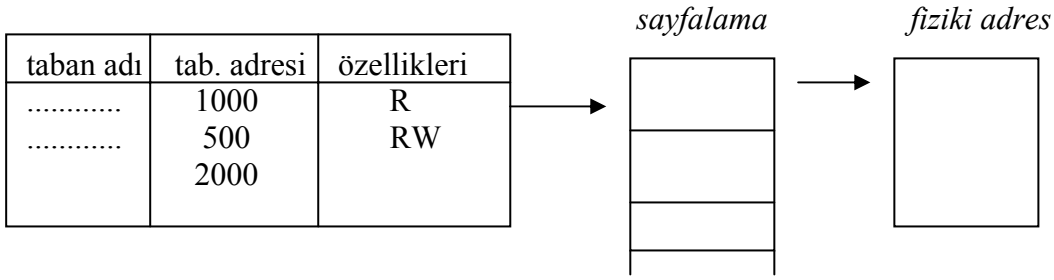
Örneğin ortak bir grafik kullanılır. Her bir kesim bu adresi bilir ve gerektiğinde oraya erişebilirler.

Kesimlemedeki veri türleri farklı olduğundan onlar için farklı olma mekanizmaları kullanılır. Örneğin yordam içeren kesim yalnız çalıştırılabilir, sayı değerleri içeren kesim okunup güncellenebilir ama çalıştırılamaz. Böyle mekanizmanın mevcutluğu program hatalarının bulunmasını (analizini) kolaylaştırır.

Sayfalamanın amacı, yeterli bellek alanı bulunmadığında büyük boyutlu program adresleri elde etmek içindir. Sayfaların boyutları eşittir. Kesimlemenin amacı program ve verilerin mantıksal bağımsız adresler alanlarına parçalanması ve onların ortaklaşa kullanımını sağlamaktır.

Modern işletim sistemlerin de sayfalı kesimleme yöntemi kullanılır. Yani her bir adres; kesim adresi, kesim dahilinde sayfa adresi ve sayfa içinde kaymanın toplamından oluşur (Sayfanın adresi; kesimin adresi, kesim no ve kaymadan adres oluşur). (Kesimleme 16 * K = $16*2^{10}$, $16*2^{10}$ kesim bulunur.) Her kesim 2^{32} (1 milyar) bitlik sözcük içerebilir. (Bu 386 'lardaydı. Pentiumlar da artık sözcük uzunluğu 2^{64} kullanılır).

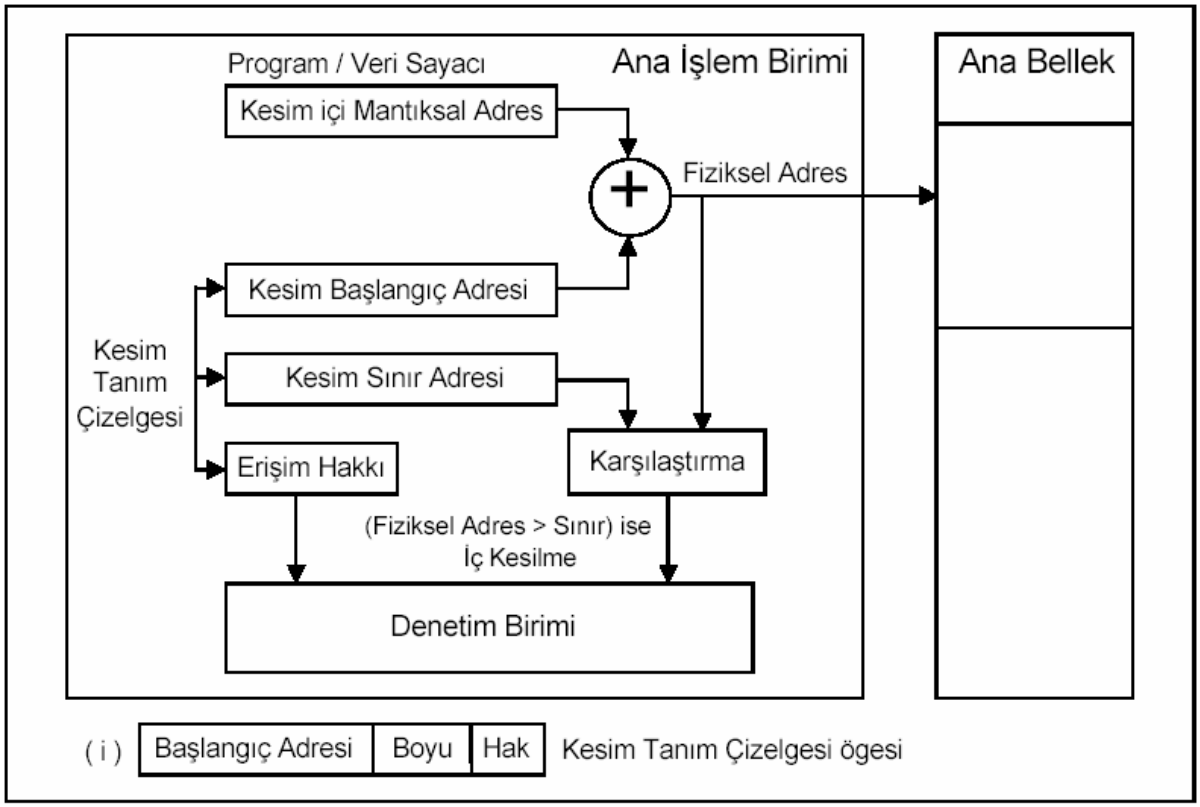
Kesim tablosu ;



- Kesimlemede ; her kesim fiziki adrese dönüştürülür.
- Sayfalı kesimlemede ; her kesim içinde sayfalama yapılabilir. Buradan fiziki adrese erişim yapılır.

Sayfalı ve Kesimli bellek yönetimleri arasındaki farklar:

- Programların mantıksal adres evrenleri; sayfalı bellek yönetiminde eşit uzunlukta sayfalara, kesimli bellek yönetiminde ise değişik uzunlukta, mantıksal bütünlüğü taban alan kesimlere ayrılır.
- Kesimli bellek yönetiminde, mantıksal adreslerden fiziksel adreslere geçiş, kullanılan adres dönüştürme yöntemi ne olursa olsun, kesim bellek konumu ilişkisini tutan kesim tanım çizelgeleri aracılığıyla gerçekleştirilir.
- Kesim tanım çizelgesinde, kesim başlangıç adreslerinin yanı sıra kesim boyu bilgisine de yer verilir. Bu bilgi, ilgili kesim işletilirken, hatalı çalışma sonucu oluşabilecek kesimden taşmaları denetlemek üzere kullanılır.

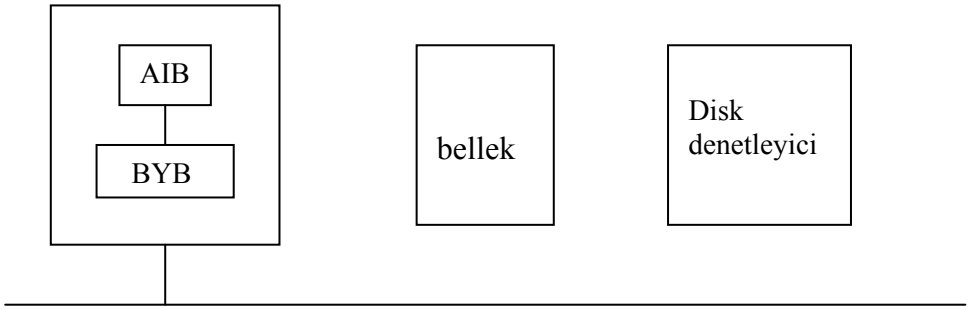


Şekil: Kesimli bellek yönetiminde bellek koruma yapısı

6.7. Sanal Bellek (Virtual Memory : Görüntülü Bellek) Yapısı

Sanal bellek, programların ve verilerin toplam boyutunun onların erişebileceği ve kullanabileceği fiziksel bellek alanından büyük olabilmesi düşüncesine dayanır. Sanal bellek çok programlı sistemlerde, verimli kullanılabilir. Ana bellek ile disk arasında yer değişim işlemi yapıldığında işletim sistemi başka görevleri çalıştırabilir. Bir çok sanal bellek sistemleri sayfalama yöntemi kullanır. Programların erişebildiği veya ürettiği adresler kümesi

sanal adresler uzayını oluşturur. Bellek yöneticisinin görevi bu sanal adresleri fiziki adreslere çevirmektir.



Not: *BYB: Bellek Yönetim Birimi*

Yukarıdaki bölümlerde incelenen bellek yöntemlerinin uygulandığı sistemlerde, görevlerin mantıksal adres evrenlerinin boyu ana belleğin fiziksel sığası ile sınırlı kalır.

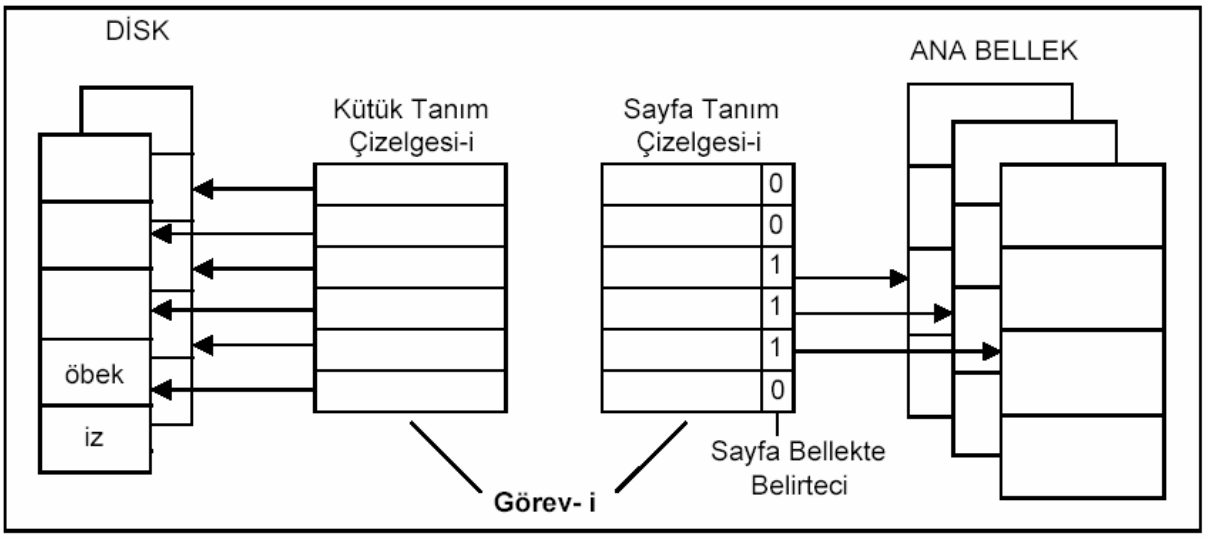
Sanal bellek yönetiminin uygulandığı sistemlerde, işletime alınan görevlere, tüm adres evrenlerini karşılayacak sığada ana bellek alanı yerine disk alanı atanır. Görev işletimi başlatılırken ilk kesimin yada ilk sayfaların ana bellekte bulunmasıyla yetinilir.

Sayfalı Sanal Bellek Yönetimi : Sayfalı bellek yönetiminde programlar, derleyiciler tarafından eşit uzunlukta sayfalar biçiminde hazırlanırlar.

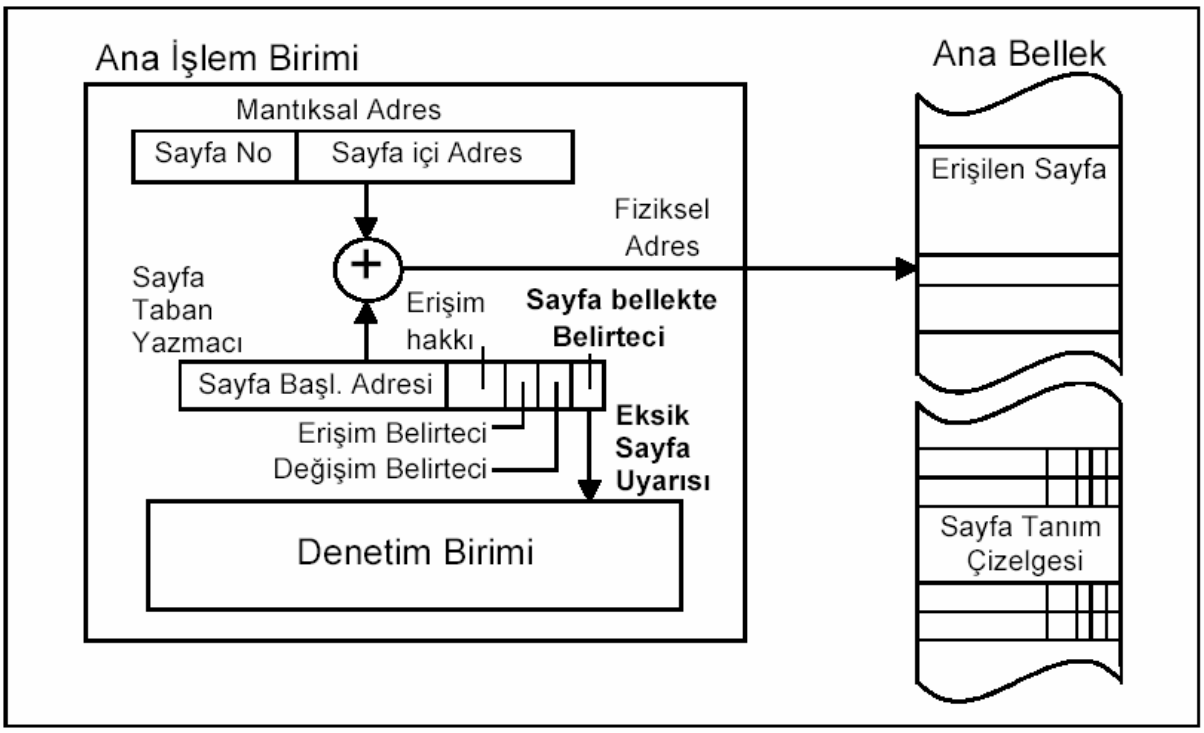
Sanal bellek düzeni çerçevesinde hazırlanan bu programların ana belleğe yüklenmeye hazır kopyaları, tümüyle diskte oluşturulur. Diskte yer alan sayfalardan bir kesimi ana belleğe yüklenerek işletim başlatılır.

Sanal bellek düzeninde program sayfaları diskte de saklandığından işletime alınan her görev için, sayfa tanım çizelgesi gibi, bir de kütük tanım çizelgesi tutulur.

Kütük tanım çizelgeleri program sayfalarının diskte saklandığı tutanak adreslerini içerir.



Şekil: Sayfalı sanal bellek yapısı ve kütük tanım çizelgesi



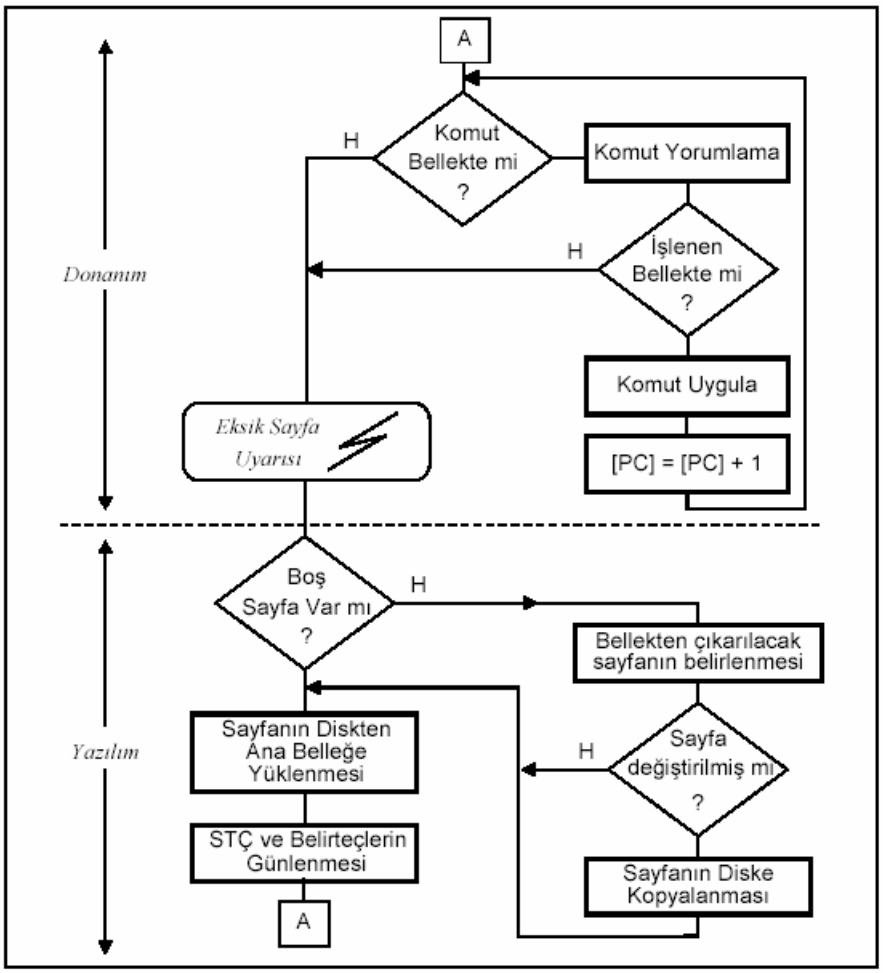
Şekil: Sayfalı sanal bellek yönetimi

Adres dönüştürme sürecinin ortasında, denetimin donanımdan yazılıma (bellek yöneticisine) aktarılmasından sonra gerçekleşmesi gereken işlemler şunlardır:

- Bellekte göreve atanacak boş bir sayfa aramak
- Bellekte kullanılabilir boş bir sayfa yoksa, yer açmak üzere ana bellekten çıkarılacak sayfanın seçimini yapmak
- Ana bellekten çıkarılacak sayfada günleme yapılmışsa(diskteki kopyasından farklı ise) sayfayı diskteki yerine yazmak.
- Erişilmek istenen sayfayı ana belleğe yüklemek.
- Yarıda kesilen komutun işletimini yeniden başlatmak.

Bellekte bulunmayan bir sayfaya erişim söz konusu olduğunda, eksik sayfa uyarısı ile bellek yöneticisi ana işlem birimine anahtarlanır.

İlgili göreve atanacak boş bir sayfa bulunup bulunmadığı sınanır. Boş bir sayfa bulunabilirse bu sayfanın başlangıç adresi, görevin sayfa tanım çizelgesinde, erişilen mantıksal sayfa satırına işlenir.



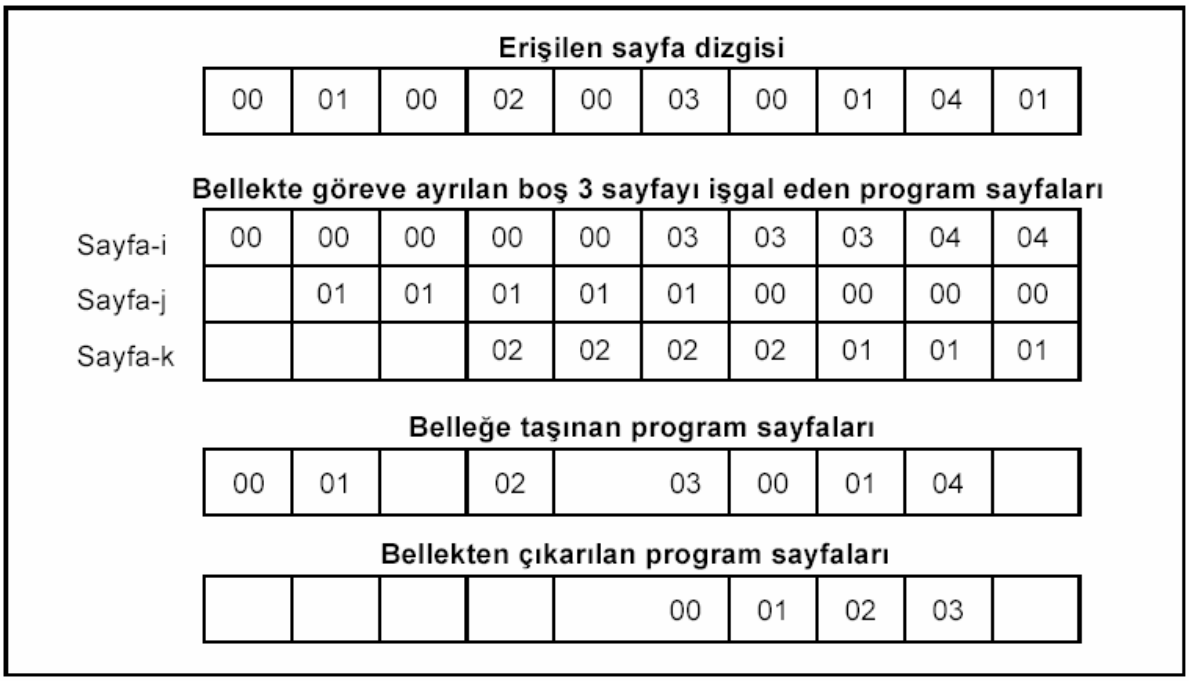
Şekil: Sayfalı sanal bellek yönetiminde adres dönüştürme süreci

Sayfa Çıkarma Algoritmaları: Eksik sayfa uyarısı sonrasında, erişilmek istenen sayfanın taşınacağı boş bir sayfa bulunamaması durumunda bellekten çıkarılacak sayfanın seçimi gündeme gelir. Bu seçim değişik algoritmalara dayalı olarak yürütülür. Bunlar;

- İlk giren sayfayı çıkarma (FIFO: First Input Fisrt Output)
- Yakın geçmişte kullanılmamış sayfayı çıkarma (NRU:Not Recently Used)
- İkinci adımda sayfa çıkarma
- Saat yöntemiyle sayfa yerleştirme
- En erken erişilmiş sayfayı çıkarma (Least Recently Used – LRU)
- En geç erişilecek sayfayı çıkarma (Optimal)

1.İlk Giren Sayfayı Çıkarma Algoritması : Bu algoritmanın temelinde belleğe birinci sunulmuş sayfanın gelecekte erişim olasılığının düşük olması varsayımına dayanır. Sayfalar FIFO kuyruğunu oluşturur. Yeni bir sayfanın belleğe sunulması gerektiğinde kuyruğun başındaki sayfa seçilir. Kuyruk her erişilen sayfa değiştiğinde güncellenir. Algoritması basit

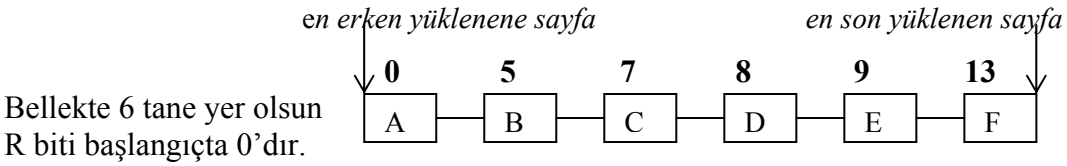
ve hızlıdır. Ancak belleğe ilk girmiş sayfanın gelecekte kullanım olasılığının düşük olacağı varsayımı doğru değildir. Öyle ki, görevin en sık kullanılan sayfaları da belleğe erken sunulmuş olabilir. Bu bakımdan sık sık erişilen sayfaların da bellekten çıkarılma olasılığı yükselir ve yer değişmelerin (takas:swapping) sayısı artar.



Şekil : İlk giren sayfayı çıkarma algoritmasına göre sayfa çıkarımı

2. Yakın Geçmişte Kullanılmamış Sayfayı Çıkarma Algoritması : Bu algoritmaya göre tüm sayfalara belirteçler (göstericiler) verilir ve bellek yöneticisi belirli bir aralıkla (Örn 10 sn'de bir) belirteçleri sıfırlıyor. Bellekten bir sayfa çıkarmak gerektiğinde ilk giren sayfa doğrudan çıkarılmıyor. Çıkarmadan önce sayfanın erişim belirteci kontrol edilir. Erişim belirteci *1* bulunan sayfa son sıfırlamadan sonra kullanılmış yada erişilmiş sayfa sayılır ve kullanılmakta olan bir sayfa gibi bellekte kalır. Algoritma listedeki bir sonraki sayfayı kontrol eder ve bu süreç belirteci *0* bulunan sayfa bulunana kadar devam ettirilir.

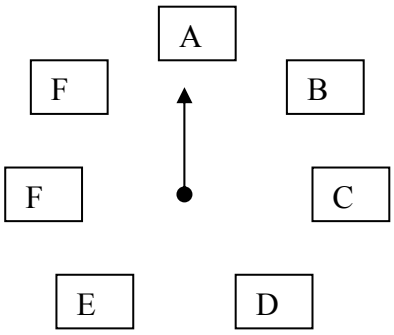
3. İkinci Adımda Sayfa Çıkarma Algoritması : NRU veya FIFO algoritmasının özel bir türüdür. Bu algoritmaya göre her sayfa için R ilave biti kullanılır. Eğer R=0 ise sayfa hem eskidir, hem de kullanılmayan bir sayfadır. Bu bakımdan söz konusu sayfa çıkarılabilir. R=1 ise uygun sayfa sayfalar listesinin sonuna gönderilir ve onun belleğe yüklenme zamanı kuyruğun sonuna gönderildiği zaman hesaplanır. Çıkarılacak sayfanın aranması bu şekilde devam ettirilir.



Sayfalar FIFO ardışıklığı ile sıralanır.

Örneğin, F sayfası kullanıldıktan sonra, 20.saniyede sayfa hatası oluştuğunu kabul edelim, bu durumda FIFO yönetimine göre ilk giren sayfa olan A sayfası çıkarılmalıdır. Eğer A 'nın R biti 0 ise A sayfası doğrudan bellekten çıkarılır. Eğer R=1 ise A sayfası listenin sonuna eklenir ve onun yükleme zamanı 20 saniye olarak gösterilir ve R=0 yapılır. Arama B sayfası için devam ettirilir. Eğer tüm sayfalar kontrol ettirilirse 2. adımda FIFO sırası uygulanır. Yani, tüm sayfaları erişim biti R=1 olduğunu kabul edelim. Bu halde sayfalar ardışık olarak listenin sonuna eklenecek ve 2. adımda en eski olan sayfa çıkarılacaktır.

4. Saat Yöntemiyle Sayfa Yerleştirme Algoritması : 2.adım algoritmasının yetersizliği sayfaların liste boyunca hareket etmesinden kaynaklanmaktadır. Bu ilave zaman kaybına neden olur. Daha iyi yaklaşım, dairesel döngülü liste şeklinde (saat biçiminde) yapılanmadır.



Saatin oku en eski sayfayı gösterir. Sayfa hatası oluştuğunda okun gösterdiği sayfaya bakılır. Eğer R=0 ise sayfa çıkarılır. Ok bir kademe ileri döner. R=1 ise okun gösterdiği kademe 0'lanır ve ok sonraki sayfayı gösterir. Bu işlem R=0 olan sayfa bulunana kadar devam ettirilir.

5. En Erken Erişilmiş Sayfayı Çıkarma Algoritması: Bu algoritmada temel olarak sayfaların belleğe yüklenme zamanı değil onlara erişilme zamanı önemlidir. Algoritmanın temelinde son zaman diliminde kullanılmamış bir sayfanın gelecekte de kullanılma olasılığının düşük olacağı varsayımına dayanır. Algoritmaya göre sayfa çıkarma gerektiğinde,

o ana kadar en erken erişilmiş sayfa çıkarılır. Algoritma ilk giren sayfayı çıkartma algoritmasına göre daha iyi sonuç verir ama daha karmaşıktır. En erken erişilmiş sayfayı belirlemek için bir sayaç kullanılır. Belleğe her erişimde sayaç bir arttırılır. Sayacın içeriği sayfa tablosunda saklanır. Bellekten sayfa çıkartmak gerektiğinde sayaç değeri en küçük olan sayfa seçilir. Her bellek erişiminde sayaçlarda güncelleme yapıldığından algoritmanın verimliliği hız açısından düşer. Algoritma özel donanımla gerçekleştirilir.

Algoritmanın çalışma prensibi şöyledir:

Erişilen sayfa dizgisi

00	01	00	02	00	03	00	01	04	01

Bellekte göreve ayrılan boş 3 sayfayı işgal eden program sayfaları

Sayfa-i	00	00	00	00	00	00	00	00	00	00
Sayfa-j		01	01	01	01	03	03	03	04	04
Sayfa-k				02	02	02	02	01	01	01

Belleğe taşınan program sayfaları

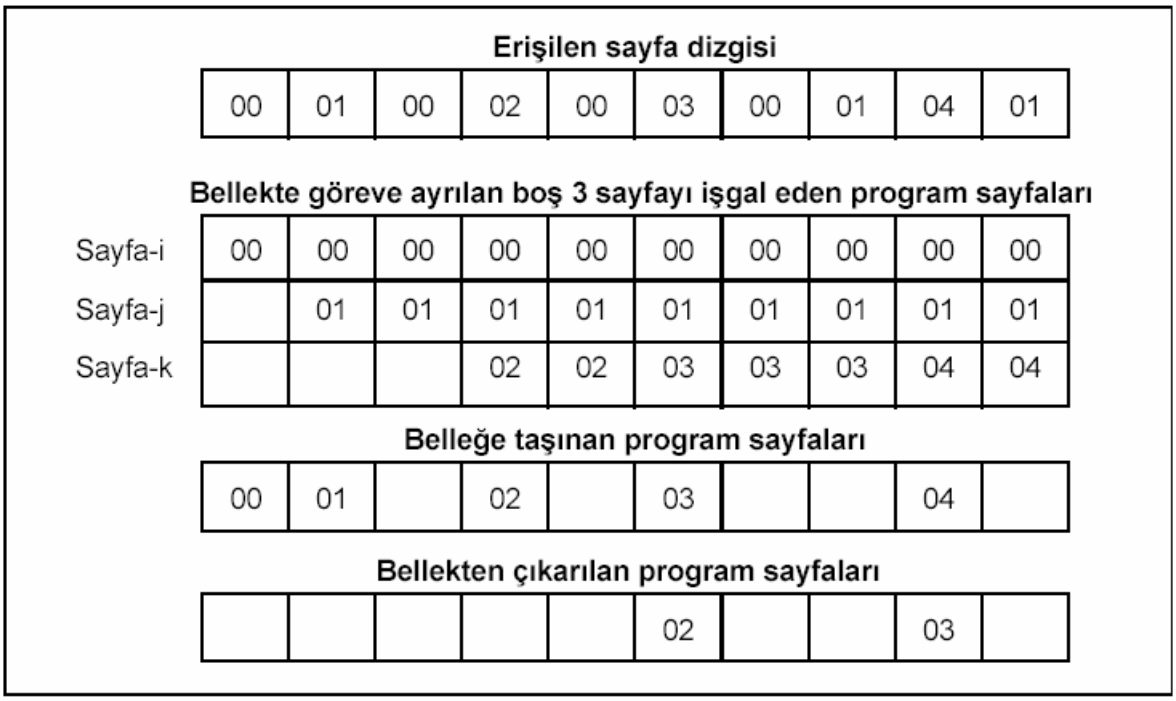
00	01		02		03		01	04	

Bellekten çıkarılan program sayfaları

					01		02	03	

Şekil: En erken erişilmiş sayfayı çıkarma algoritmasına göre sayfa çıkarma

6. En Geç Erişilecek Sayfayı Çıkarma Algoritması: Bu algoritmaya göre yakın gelecekte kullanılmayacak sayfalar sayfa tablosundan çıkarılır. Örn: Eğer bir sayfa 800 komuttan, diğeri ise 600 komuttan sonra erişilecekse sayfa çıkarmak gerektiğinde algoritmaya göre 1 sayfa (800 komuttan sonra çalışacak olan) çıkarılacaktır.



Şekil: En geç erişilecek sayfayı çıkarma algoritmasına göre bellekten sayfa çıkarma

Sanal Bellek Kullanımı: Fiziksel belleğimizin yetersiz kaldığı durumlarda sabit diskimizin belli bir parçasını fiziksel belleğimizin bir parçasıymış gibi düşünerek oraya adresleme yapabilmemize olanak sağlayan hafıza birimidir.

Fiziksel hafıza kullanımında program belleğe sığmazsa bunu parçalara ayırıp diske taşıma (swap) işlemini yapmamız gerekir, fakat bu programın çalışmasında bir dezavantajdır, çünkü programın neresinin sık kullanılacağı önceden bilinemez ve bu sık kullanmaktan kaynaklanan sürekli swap işlemleri meydana gelebilir, bu da zaman kaybı olur, bu zaman kaybını önlemek için sanal hafıza geliştirilmiştir.

Belleği parçalara ayırmak, çözüldüğünden daha çok probleme sebep olmaktadır. Görevlerin ihtiyaçlarına göre belleği ayarlamak yerine sürecin boş belleği nasıl kullanacağı sorusunu sorabiliriz. Herhangi bir konfigürasyon için uyumlu bir görev oluşturabilir miyiz? Mesela boş bellek alanı bitişik olmayan bölgelerde bulunsun, görevleri birbirinden ayrı kodlar gibi görüp bitişik olmayan yerlere yerleştirebilir miyiz?

Gerçekte yapabiliriz ve çok kullanıcılı sistemde yaygın olarak kullanılmaktadır, görev kodu "page-sayfa" adı verilen birbirinden ayrı bölümlere ayrılır, aynı şekilde belleği de "frame-çerçeve" adı verilen birimlere böleriz. Bir "çerçeve","sayfa" ile aynı büyüklükte belleğin bölümleridir. Görevi belleğe yerleştirmek için "sayfa" leri uygun çerçeve lere yerleştiririz.

Örnek: İşletim sistemi belleği 20 "çerçeve" ye bölmüş olsun ve aşağıdaki süreçler sisteme gelsin:

Görev ID	"Page" Sayısı
A	3
B	4
C	1
D	4
E	1
F	6

a) Bellek görüntüsü nasıl olur?

Eğer sistem görevleri ID lerine göre belleğe yerleştirirse aşağıdaki bellek görüntüsü ortaya çıkar:

A	Frame 1-3
B	Frame 4-7
C	Frame 8
D	Frame 9-12
E	Frame 13
F	Frame 14-19
Boş	Frame 20

Şekil 1. Görev A-F için bellek görünüşü

A	Frame 1-3
B	Frame 4-7
Boş	Frame 8
D	Frame 9-12
Boş	Frame 13
F	Frame 14-19
Boş	Frame 20

Şekil 2. A,B,D ve F için bellek görünümü (C ve E bitmiş)

b) Görevlerin işleyişi nasıl?

Şimdi şunların sıra ile olduğunu kabul edelim: C biter, E biter, G adında 3 "sayfa" lık yeni bir süreç sisteme girer.

Frame 1-3	Frame 4-7	Frame 8	Frame 9-12	Frame 13	Frame 14-19	Frame 20
A	B	G	D	G	F	G

Eğer işletim sistemi bellek idaresi için değişken bölmeler kullanıyorsa G işini belleğe yerleştiremez. Kullanılmayan 3 çerçeve olduğu için G kodunu 3 "sayfa" ya bölmek suretiyle bellekteki boş alanlara yerleştirebilir. Yukarıdaki şekilde sonuçlar görülmektedir.

c) Görevleri "sayfa" lara bölersek belleği nasıl koruyacağız?

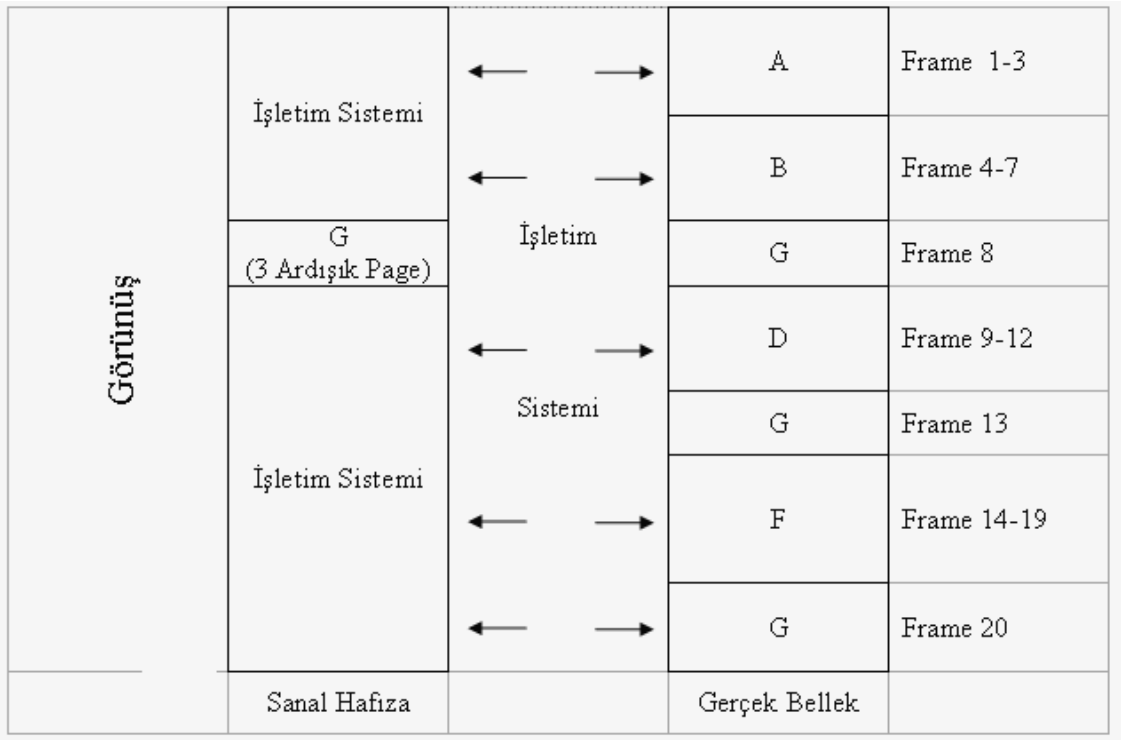
Bölmelerde, CPU bölmeyi sınırlayan bellek referanslarını registera yazıyor idi, bu görevler birbirini takip eden çerçeveler de iseler çalışırlar.

d) Bölünmüş süreç nasıl çalışır?

Bilindiği gibi, CPU komutları birbirini takip eden konumlardan alır, eğer böyle değilse CPU bu komutları nasıl bulur? Mesela G çalışıyor ve Çerçeve-13 deki son komuttan sonra CPU 14 deki ilk komuta mı devam eder? Dinamik zaman paylaşımlı ortamda verilen zaman diliminde hangi bellek çerçevenin kullanılabilir olduğunun belirlenmesine imkân yoktur. Görev belleğe girdiği zaman çalışmaya hazırlanır. Bu hazırlıklardan bir tanesi de görevin sonuçta bulunacağı yer ile bellek referansının uyumlu hale gelmesidir.

Kısaca, görev uygun bellek buluncaya kadar çalışmaya hazırlanamaz aynı zamanda sistemde görev çalışmaya hazır olmadan bellek ayırmaz.

Görevlerin bitişik olmayan bellek bölgelerinde yerleştirilmiş olmasına rağmen kullanıcı bu ayarlardan habersizdir. Donanım ve yazılım kullanıcıyı kandırır ve sanal hafızanın gerçekte var olduğuna inandırır. Bu durum aşağıdaki şekilde gösterilmiştir.



e) Çerçeve ve Sayfa ne kadar büyüklükte olmalıdır?

Çerçeve küçük olursa sayı artar takip zorlaşır ve tablodaki eleman sayısı çok olur, büyük olursa tablo küçülür ama israf artar. Pratikte çerçeve büyüklüğü 512 ile 4096 Byte arasındadır.

f) Adres Dönüşümü (Sanaldan Gerçeğe) Nasıl?

Gerçek ve sanal bellek dağınıklığını çözmenin bir yolu "sayfa tablo" dır. Bu tablo sanal adresi gerçek adrese çevirmek için kullanılır. Dinamik adres dönüşümü program çalışırken komut adresi alanının korunmasını tanımlar. Bunun komut çevrimi zamanında yapılması büyük avantaj sağlar. Kodu belleğin neresinde bulunursa bulunsun çalışacak şekilde oluşturur. Bu yazılımcılar için güçlü bir avantajdır. Programın düzgün çalışması için görevi özel bir yere yükleme endişesine gerek yoktur. Gerçekte görev sonunda yerleşeceği bellek konumundan bağımsız hale gelir. Hatırlanacağı gibi görev çalışmaya hazır hale gelince hangi konumun elverişli olacağı tahmine imkan yoktur. Diğer bir avantajı ise sisteme kodu değiştirmeden konumunu değiştirme esnekliği verir.

Şimdi sayfa tablosunu ve dinamik adres dönüştürme mekanizmasını inceleyelim. Tipik makine dili komut formatını hatırlayarak aşağıdaki şekilde direk adresleme örneği gösterilmiştir.

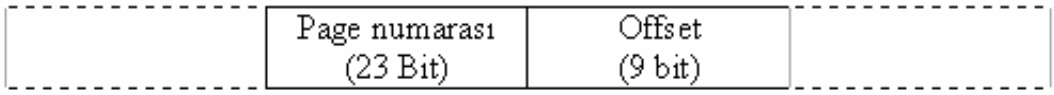
OP Code	Register	Bellek adresi

Bellek adresinin iki bileşeni vardır:

a – sayfa numarası

b – sayfa üzerindeki relatif konumu (offset)

Örneğin bellek adresi 32 bit uzunluğunda olsun ve çerçeve büyüklüğü de 512 byte uzunluğunda olsun (bu görev sayfa'sında 512 byte uzunluğunda olduğu anlamına gelir). Aşağıdaki şekilde bellek adresi gösterilmiştir.



Şekil : 32 bit için bellek adresi

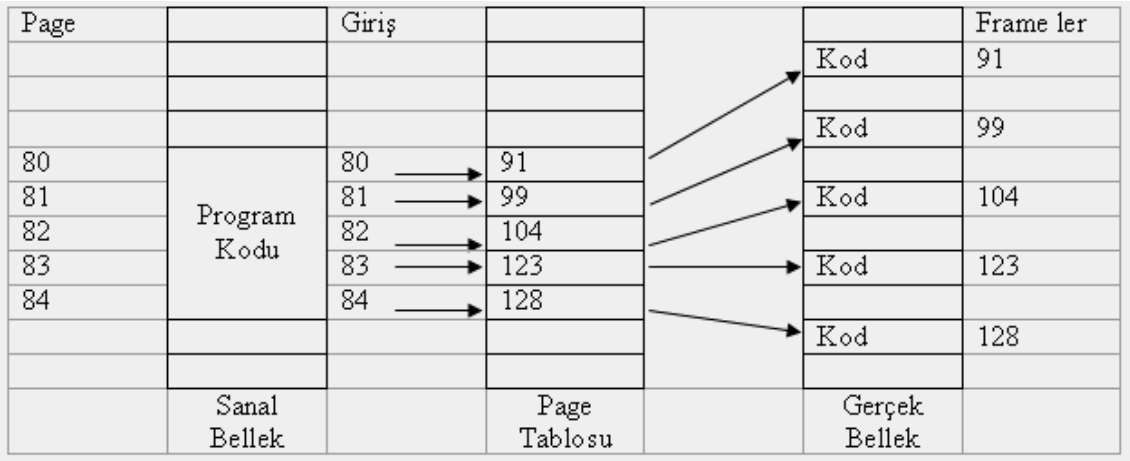
g) Sayfa Tablosu nasıl işler?

Örneğin;

Kullanıcı sanal sayfalarda 80-84 e kadar yerde bir görev oluşturdu. Diyelim ki görev CPU kontrolünü eline geçirdi, görev 91,99,104,123,128 çerçevelerine oturur. Görev CPU kontrolünü eline geçirmeden önce işletim sistemi sayfa tablosu içindeki 80 den 84 e kadar girişleri çerçeve numaralarına yönlendirir. Göreve göre komutlar ve veriler 80-84 sayfalar arasındadır.

Diyelim ki ilk komut 80. sayfada ve 0 offset de olsun görev çalışmaya başladığı zaman program sayıcı bu adresi içerir. Komutu alma işlemi sırasında 80 değeri high order adres bitinde olacaktır. Fakat 80 gerçekte sayfa tablosunda bir girişi referans gösterir. CPU bu sayfa tablosunda bulunan bu girişteki değeri ulaşılacak çerçeve'ye karar vermek için kullanır bu durumda 91. sayfadaki 0 offsetteki komutu alır (bu 80. girişin içeriğidir). Sonra komut çözülür ve çalıştırılır. Verilen adres sayfa 84. de offset 73 olsun, bu sanal adrestir CPU bunu 84. girişle değiştirir komut gerçekte sayfa 128. de offset 73. dedir.

Eğer işlem dallanmazsa bir sonraki komut sayfa 80 de ve offsete 1. dedir. Dinamik adresleme yoluyla CPU gerçek komutu sayfa 91. de offset 1. den alır. 80-81 de bulunan komutlar gerçekte 91 ve 99. çerçevelerde bulunmaktadır. Böylelikle dinamik adres dönüşümü takip eden referansta ne görünürse ora göre değişmektedir. Tabi ki bu sayfa tablo eşleşmesi bütün sistemlerde aynı değildir.



Şekil: Sanal, gerçek bellekler ve sayfa tablosu

6.8.İlişkisel Bellek (Associative Memory)

Sayfa tablolarının bellekte saklanması işlemcinin verimliliğinin aşağı düşmesine neden olur. Bunu aradan çıkararak, işlemcinin hızını yükseltmek için bellek yönetimine küçük bir donanım eklenir ve bu donanım sayfa tablosu kullanarak sanal adresi fiziki adrese dönüştürür. Bu bellek *ilişkisel bellek* tir. Bu belleğin prensibi, programların büyük kısmı az sayıdaki sayfalara daha çok erişebilsin.

İlişkisel Bellek Yapısı :

Sanal Sayfa	Güncelleme Biti	Koruma	Sayfa Çerçevesi
140	1	RW	31
20	0	RX	38
.	.	.	.
.	.	.	.
.	.	.	.
129	1	R	62

Her satır bir sayfa hakkındaki bilgileri içerir. Sayfaların içerisinde değişme güncelleme oluşmuşsa bu bit 1'dir olmamışsa 0'dır. Sanal adres bellek yönetim birimine sunulduğunda yönetici önce bu adresin ilişkisel bellekte olup olmadığını kontrol eder ve böyle bir adres varsa ve erişim koruması bozulmuyorsa o zaman fiziksel adres doğruca ilişkisel bellekten alınır.

RW → okuma-yazma

RX → okuma-çalıştırma

R → sadece okuma

X → Çalıştırma

Sanal sayfa ilişkisel bellekte yoksa, bellek yönetimi birimi ilişkisel bellekteki satırlardan birini çıkarır ve gereken adresi ana bellekteki sayfa tablosundan ilişkisel belleğe yerleştirir. Böylelikle yeniden gerektiğinde bu kez ilişkisel belleğe erişilerek sonuç daha hızlı elde edilir.

Satır ilişkisel bellekten silindikten sonra, güncelleme bitine uygun olarak satırın içeriği ana belleğe kopyalanır (güncelleme oluşmuşsa hiçbir şey yapılmaz). Böylece o satırın hakkındaki eski bilgiler ana bellekte bulunmaktadır. Belleğe erişimin ilişkisel bellek tarafından sağlanması oranına *başarı oranı* denir. Bu oranın yüksek olması verimliliğinde yüksek olmasını demektir.

Genelde işlemcinin verimliliği 3 kıstasa bağlıdır:

1. Sayfa tablosuna erişim süresi (hızı)
2. İlişkisel belleğe erişim süresi
3. Başarı oranı : küçük aralıklarla erişilen sayfa sayılarına ve ilişkisel belleğin boyutuna bağlıdır.

7. KÜTÜK SİSTEMLERİ

Kütük soyut bir kavramdır. Diske Lojik-0,1 ‘ler şeklinde yerleşir. Kullanımı kolaylaştırmak için kütük görünümleri kullanılır.

Kütük kullanımının amacı :

1. Program ve veriler üzerinde işlemleri kolaylaştırmak.
2. Büyük hacimde verilerin ana belleğe sığmaması durumunda kütüler kullanılır.

Dizinler bir ağaç yapısı oluşturur. Dizinlerde kütüğün bir türüdür. Tüm bilgisayar okumaları için bilginin kaydedilmesi ve okunması gerekiyor. Ama bu bilgilerin tümüyle ana bellekte tutulması ve saklanması çoğu zaman üç sebepten gerçekleştirilemez.

1. Bellek alanının yetersizliği
2. Görev kesildikten veya işini tamamladıktan sonra onunla ilgili bilgilerin de ana bellekten silinmesi kaybı.
3. Birden fazla görevin aynı zamanda aynı verilere erişiminin sağlanabilmesidir.

Bu üç problemi çözmek için bilgiler disk ortamına kaydediliyor. Buna *ikinci bellek* denir. Diske kaydedilmiş bu tür bilgi birimlerine *kütük* denir. Kütükteki bilgiler kalıcıdır ve görevin oluşturulması veya kesilmesine bağlı değildir. Kütükler İşletim Sistemi tarafından yönetilir. Onların yapılandırılması, adlandırılması, kullanımı, çalıştırılması İşletim Sisteminin başarımının başlıca konularıdır. İşletim Sisteminin kütüklerle ilgili kısmı kütük sistemini oluşturur. Kütükler soyut mantıksal varlık olduğundan onlara erişim, kütük adlarıyla gerçekleştirilir.

Çeşitli İşletim Sistemlerinde kütüklerin adlandırılması için farklı kısıtlamalar vardır. Çoğu kütükler iki kısımlı adı destekliyor. Birincisi kütüğün özel adı, ikinci kısım uzantısı.

- Linux’ta uzantıların sayısının sınır yoktur.
  Örn : prog_adi.tar.gz
- Windows da *.exe* uzantılı olamayan dosya çalıştırılamaz.
  Örn : prog_adi.exe
- Ms-DOS da büyük küçük harf ayrımı yoktur.
  Örn : prog.exe ile PROG.exe aynı

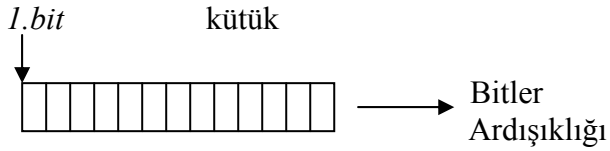
- Unix de büyük küçük harf ayrımı vardır.

Örn : prog.exe
PROG.exe } farklıdırlar.
Prog.exe

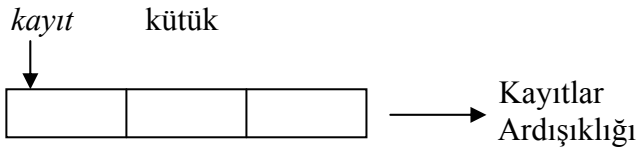
7.1. Kütük Yapılandırmaları

Kütükler çeşitli şekillerde yapılandırılabilir :

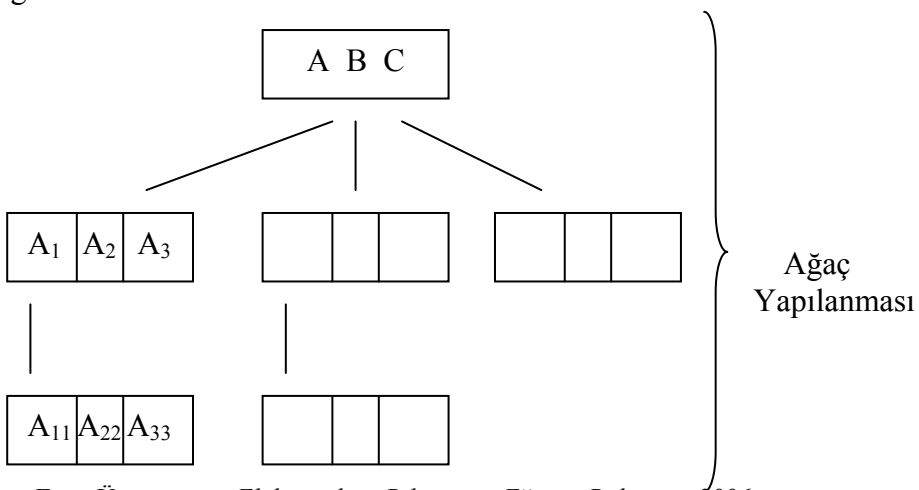
1. **Bitler ardışıklığı şeklinde yapılanma:** Erişim bit ardışıklığı ile yapılır. Bit sayısı çok olduğundan erişim kolay değildir. Eski sistemlerde kullanılır. Arama işlemleri zordur ve çok zaman gerektirir.



2. **Kayıtlar ardışıklığı şeklinde yapılanma:** Genelde kullanılan yapıdır. Kayıt ve alt seviyeleri bitlerden oluşur. Verilere erişim kayıtlar üzere yapılır. Kütük ise sabit uzunluklu kayıtlardan oluşur (80-128-132 karakter uzunluk).



3. **Ağaç şeklinde yapılanma:** Bu sistemler, DataBase machine türü sadece verilerle uğraşmak için yapılır. Kütük, kayıtlar ağaç biçiminde yapılanır. Kayıtların aynı olmayabilir, farklı olabilir. Her kayıtta kendi adı veya anahtarı ile erişim mümkündür. Bu tür yapılar özel işletim sistemlerinde (veri tabanına yönelik sistemlerde) kullanılır. Kayıtların yapılanmasında başlıca amaç, verilmiş anahtarla herhangi bir kayıta doğrudan erişimi sağlamaktır.



Kayıtların Yapısı : Sabit boyutlu ve değişken boyutlu kayıtlar vardır. Mantıksal kayıtlar değişken boyutludur. Her bir kaydın anahtarı vardır. Anahtar üzerinde sıralama yapılır. Erişim bu şekilde sağlanır. Kayıt adı ile erişilemez.

Kütük türleri : İşletim sistemleri, çeşitli kütük türlerini desteklemektedirler. Bunlar;

- Kullanıcı bilgilerini içeren kütükler
- Kütük sisteminin yapısını içeren kütükler
- Simge kütükleri (Ardışık giriş-çıkış aygıtlarında işlem yapmak için kullanılır. Yazıcı, ağlar, terminaller. Bunlar, geçici olarak oluşturulurlar ve işlem bitikten sonra silinirler).
- Blok kütükleri (Diskle işlem yapmak için kullanılır).

Kütüklere ardışık ve rast gele (doğrudan) erişim mümkündür. Ardışık erişimde gereken veriye ulaşmak için 1.'den başlayarak tüm veriler okunmalıdır. Şimdiki sistemlerde her ikisi de kullanılır. Genelde doğrudan erişim.

Her bir kütüğün adı ve içerdiği bilgilerin yanı sıra başka özellikleri de vardır. Sistemden sisteme bu özellikler değişebilir. Genel özellikler şunlardır.

- Koruma
- Parola
- Kütüğü oluşturanın adı, özellikleri, oluşturulma tarihi
- Bayraklar (Geçici yada kalıcı kütük, sistem veya kullanıcı kütüğü olduğu, erişim hakkında bilgiler bulunur).
- Kayıtın uzunluğu, boyutu
- Kütüğün son güncelleme tarihi
- Kütüğün şimdiki ve maksimum boyutu

Kütüğün okunması sırasında tüm kütük belleğe aktarılır (açma işlemi). İşlem bittikten sonra kapanır ve diske gönderilir.

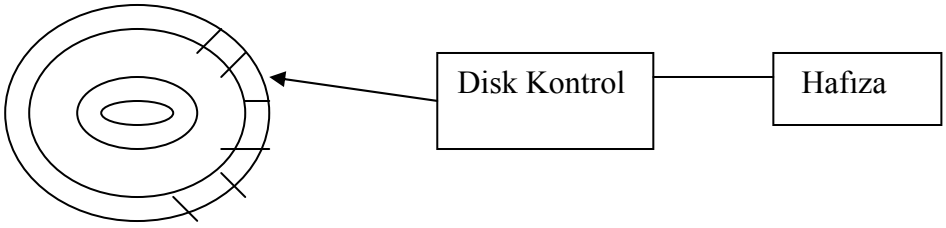
7.2.Kütükler Üzerinde İşlemler

Kütük diske yerleşmiş veri topluluğudur. Disk bloklarında yerleşir. Bloklara yerleşme çeşitli şekillerde olur. Bağlaçlı liste, vs. Kütükler için disk bloklarının kullanımı birkaç yöntemle gerçekleştirilebilir.

1. Ardışık yerleşim yöntemi : Bu yöntemle her bir kütük diskteki ardışık bloklarda saklanır. Örneğin 50K 'lık bir kütüğümüz varsa ve disk bloklarının boyutu 1K ise, 50 Blok gerekecektir.

Ardışık yerleşimin iki üstünlüğü vardır:

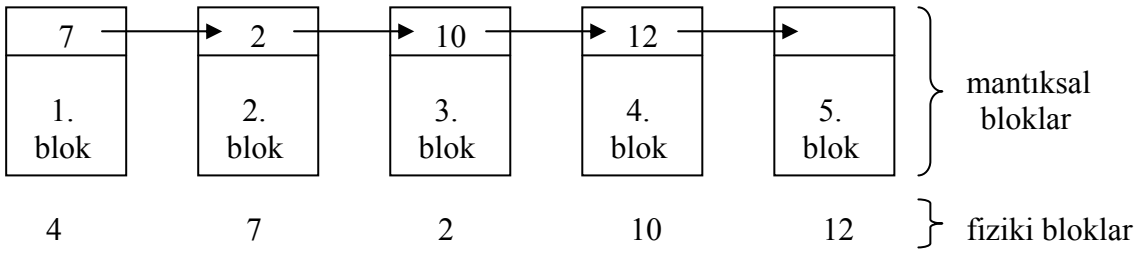
1. Çalıştırılması basittir ve sadedir; kütük bloklarının yerini belirlemek için yalnız 1. bloğun adresini bilmek yeterlidir.
2. Verimlilik yükseltir; bir işlemde tüm kütüğün okunması mümkündür.

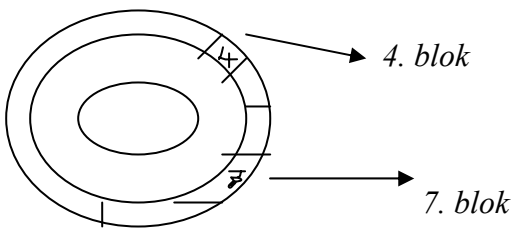


Ardışık yerleşimin 2 yetersiz yönü vardır:

1. Kütük oluşturulduğu zaman onun maksimum uzunluğu (boyutu) çoğu zaman belli olmuyor. Bu bakımdan kütükler için ne kadar yer ayrılacağı problem oluşturur.
2. Disk alanının verimsiz kullanımı ile ilgilidir. Bir kütük için ayrılmış fakat şu anda boş olan disk alanları diğer kütükler tarafından kullanılamaz.

2. Bağlaçlı Liste Yerleşimi : Bu yöntemde kütükler bağlaçlı liste oluşturan disk bloklarına yerleştirilirler. Her bloğun 1. kelimesi sonraki bloğun göstergesidir.

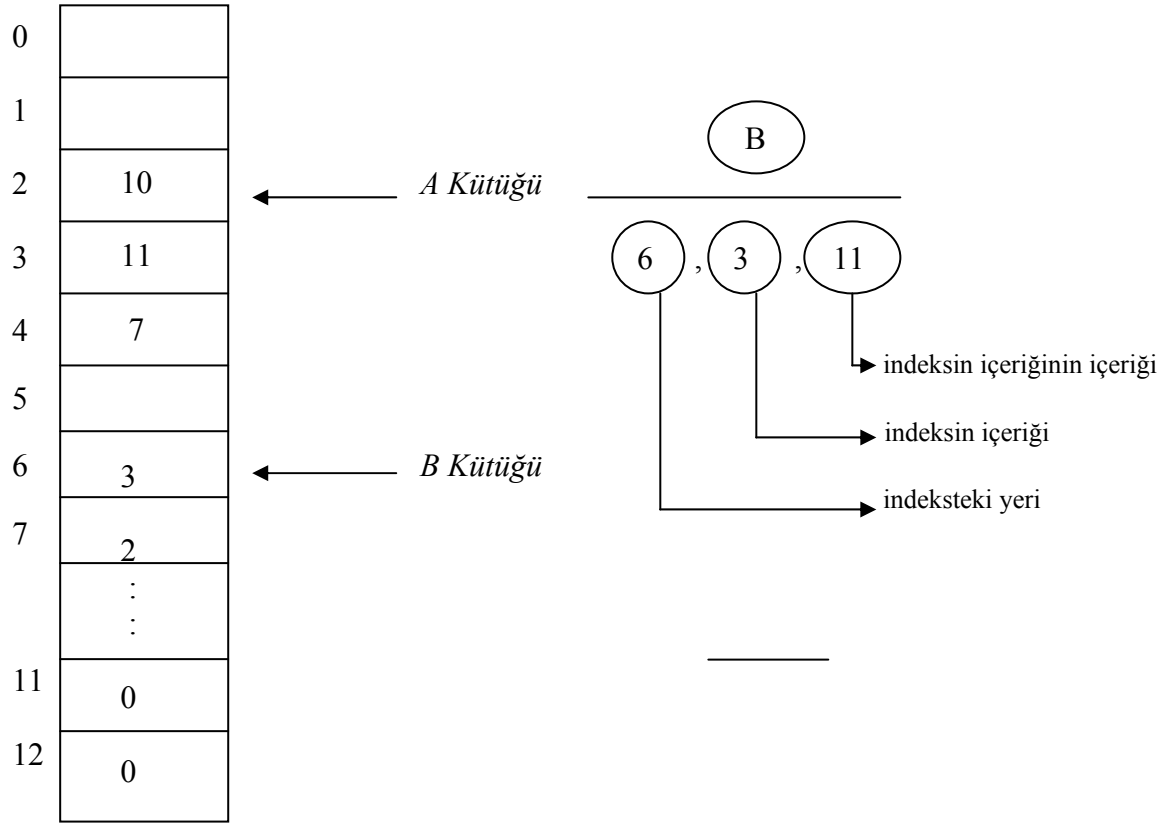




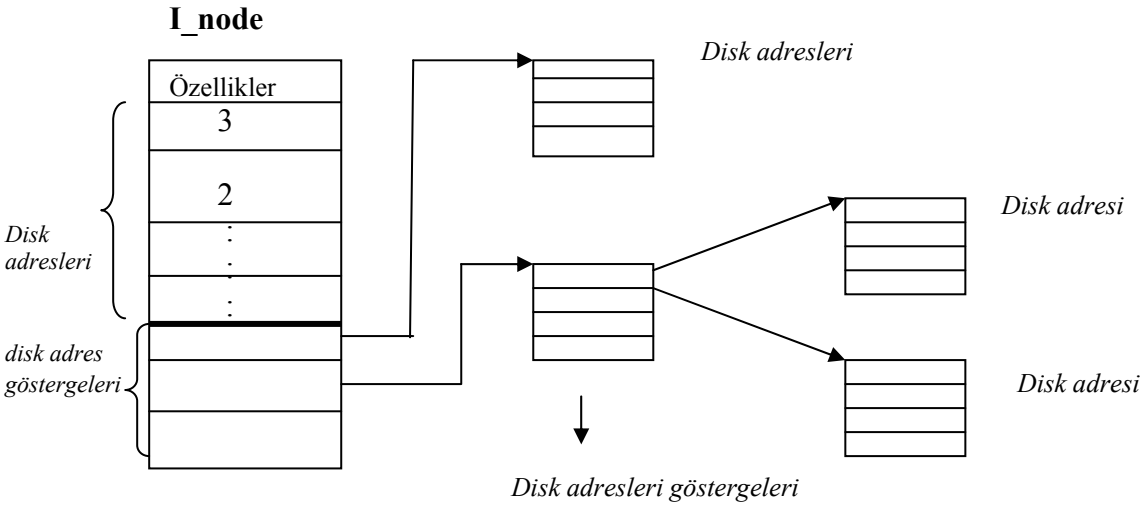
Bu yöntemde tüm disk blokları kullanılabilir. Kütüklerde, diskin kütükler arasında dağıtılmasından dolayı disk alanı kaybı olmuyor. Dizinlerde, 1. disk bloğunun adresi gösterilir. Doğrudan (rastgele) erişim, ardışıklığa oranla yavaştır. Burada, bir adresin bulunması daha kolaydır.

3. İndeksleri kullanan bağlaçlı liste yerleşimi : Bu yöntemde her bir disk bloğundaki göstergeler bellekte bir indeksler tablosu oluşturur. Bu yapıyı kullanmakla blokların içeriğine daha kolay erişilir. Rast gele (doğrudan) erişim hızı yüksektir. MS DOS 'da kullanılır.

Başlıca yetersizliği indeksler tablosunun her zaman (kütüğün çalışması süresince) ana belleğe yerleşmesidir.



4. İndeks düğümleri (I_node) yöntemi : Bu yöntemde kütük ve fiziksel bloklar arasındaki bağlantılar indeks düğümü denilen tabloda saklanır. Bu tablo kütük bloklarının özelliklerini ve disk adreslerini içerir. Kütük küçük ise, ona ait (onunla bağlantılı) bilgiler tek bir indeks düğümünde yerleşir. Kütük büyük olduğunda düğümün içeriklerinden birisi diğer bir düğümün adresini gösterir ve bu yapılanma kütükler için yeterli sayıda adres blokları oluşturulana kadar devam ettirilir. Unix 'te bu yöntem kullanılır.



7.3.Dizinlerin Çalıştırılması ve Yapısı

Kütüğün okunabilmesi için açılması gerekir. Kütük açıldıktan sonra, kütük yolu adını (erişene kadar gösterilen dizinler) kullanılmakla işletim sistemi dizin içeriklerine erişir. Dizin içeriğinde kütüğe uygun disk bloklarını bulmak için gereken bilgiler var. İşletim sistemine bağlı olarak bu bilgi tüm kütüğün disk adresi (ardışık yerleşimde), 1. bloğun numarası (bağlaçlı liste şemasında) veya indeks düğümünün (I_node) numarası olabilir. Tüm bu hallerde dizin sisteminin başlıca fonksiyonu kütüğün adını, verileri bulmak için gereken bilgiye dönüştürmek veya haritalamaktır.

Burada önemli problemlerden birisi kütüğün özelliklerinin saklanma yerinin belirlenmesidir. Çoğu sistemler de bu özellikler doğrudan dizin içeriğine yazılıyor. I_node ‘u destekleyen sistemlerde ise bu özellikler düğümlere kaydedilir.

MS-DOS’ ta disk içeriğinin yapısı :

Kütük adı		özellikleri	Tarih zaman	1. blok nosu	

Her bir dizin böyle bir yapı oluşturur.

UNIX ‘te daha basit bir yapı vardır :

I-node nosu	Kütük adı

UNIX ‘te dizin yapısı çok daha basittir. Her dizin içeriğinde yalnız kütük adı ve I-node numarası bulunur. Kütüğün tipi, boyutu, sahibi, tarihi ve disk blokları hakkındaki bilgiler I-node ‘da saklanır.

7.4. Disk Alanının Yönetimi

Kütüklerin diske yazılması için başlıca iki yöntem vardır.

1.Kütüğü saklamak için ardışık disk alanının kullanımı
2.Kütük birkaç bloğa bölünerek diske kaydedilmesi

Tüm kütük sistemleri kütükleri sabit boyutlu disk bloklarına kaydederler. Bloğun boyutu genelde 512 byte'dan az olamaz.

Disk alanının yönetiminde karşılaşılan problemler :

1. Kütüklerin yerleşimi
2. Boş disk bloklarının belirlenmesidir. Bunun için iki yöntem kullanılır.
    a. Boş disk alanlarını için bağlaçlı listesi kullanılır ve her blok da boş disk bloklarının adresleri (numaraları) gösterilir.
    b. İkili haritalama yöntemidir. Burada tüm disk alanı (blokları) bitler üzere haritalanır. Eğer blok meşgulse veya doluysa harita biti 1, boşsa 0, (veya tersi olabilir) gösterilir.

Not : Disk doldukça boş alanlar azalacağından bağlaçlı listenin kullanımı daha iyidir. İkili haritalama sabit boyutludur.

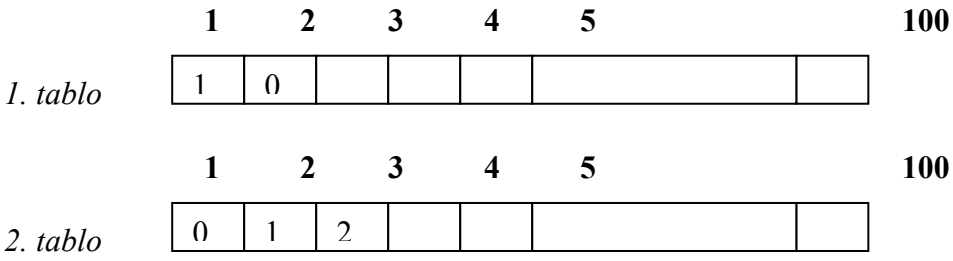
7.5.Kütük Sistemlerinin Güvenliği

Disklerdeki kötü blokların sağlam bloklarla değiştirilmesi için, donanım ve yazılım çözüm yolları bulunmaktadır. Donanım yönetiminde kötü bloklar listesi oluşturulur ve bu bloklar için bir disk bölümü (sektörü) ayrılır. Denetleyici ilk işe başladığında bu listeye bakar ve kötü bloklar, sağlam blokların adresleri ile değiştirir. Böylelikle kötü bloklara yöneltilmiş her türlü erişim sağlam bloklara yöneltilmiş olur.

Yazılım yönteminde kullanıcı veya işletim sistemi kötü bloklar içeren bir kütük oluşturur. Bu kütüğü boş bloklar listesinden çıkarır veya siler. Kötü blok kütüğüne erişim olmadığı için problem çözülmüş olur.

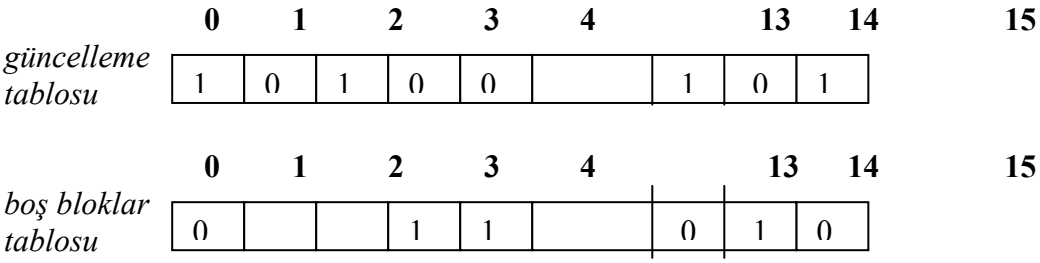
Güvenliğin yükseltilmesinin bir yolu da yedeklemedir. Her bir kütüğün arşivleme biti olur. Başlangıçta bu bit 0 'dır. Güncelleme veya erişim durumunda bu bit 1 olur. Bir yedeklemeden diğerine kadar güncelleme biti 1 olmuşsa yedekleme de güncellenir ve bit 0 'lanır.

Güvenliğin yükseltilmesine bağlı bir problem de kütüklerin kararlılığının veya bütünlüğünün korunmasıdır. Kütüğün güncellenmesi zamanı hata oluşmuşsa, kütük karasız duruma geçebilir. Mesela kopyalama sırasında elektrik kesilmesi durumu buna sebep olabilir. Kararsız duruma geçme bazı blokların güncellenmesi bazılarının ise güncellenmemesi anlamına gelir. Kararlılığı belirlemek için yardımcı programlar vardır. Bu programların iş prensibi birbirine benzerdir. Kararlığı kontrol etmek için kütüğün her bir bloğu kontrol edilir. Bu amaçla her bir blok için iki tablo oluşturulur. Tabloların içeriği sayaç fonksiyonunu yerine getirir. Başlangıçta her iki sayacın değeri sıfırlanır. 1. sayaç, bloğun kaç kez güncellendiğini, 2. sayaç ise, bloğun boş bloklar listesinde olmasını gösterir. Bloklara erişim zamanı (bloklar okunduğu zaman) 1. sayacın değeri arttırılır.



Bloğun güncellenmesi 1 ise boş listede 0 olur. Blok güncellenmiş ise boş listede olamaz. Boş listede varsa güncelleme 0 olmalıdır. Her ikisi de 0 ise bu blok kaybolmuştur. Hata olmuştur.

İkinci tabloda 2 değerinin olması bu blokların bağlaçlı liste şeklinde bağlandığını gösterir. Bir blok birkaç kütükle kullanılmış olur.



Bu durum **kararlı durum**dur. Yani eğer blok kullanımda ise, boş listede yer almamalı, boş bloklar listesinde yer alıyorsa kullanımda olmamalıdır.

Blok kullanımda olduğunda denetleyici tarafından boş listeden çıkarılmalıdır. Eğer kullanıma alınırken listeden çıkarılmazsa, bir hata oluşursa **blok kaybı** olur.

Varsayalım ki 2. blokta her iki tablo için değer 0'dır. Bu bloğun kaybı anlamına gelir. Sistemi kararlı duruma geçirmek için blok boş bloklar listesine aktarılır.

Başka bir halde aynı blok boş bloklar listesinde iki kez gösterilebilir. Diyelim ki 2. tabloda 3. blok için değer 2 olsun. Bu 3. bloğun boş bloklar listesinde iki kez gösterildiği anlamına gelir. Bu hal boş blokların bağlaçlı liste biçiminde gösterildiği zaman oluşabilir. Çözüm yolu boş bloklar tablosunun yeniden yapılandırılmasıdır.

Bir başka durumda, aynı veri bloğu birkaç kütük tarafından kullanılabilir. Bunun bir çözüm yolu bloğun içeriğini kütüklerden birine kopyalamak ve tabloyu yeniden oluşturmaktır. Bu durumda, kütüğün içeriğinin doğru olması garantilenemiyor. Fakat, sistem karasız durumdan kararlı duruma geçmiş olur.

7.6.Diskin Planlanması

Disk başlık konumunun ileri geri hareketini planlaştırmak için birkaç algoritma vardır. bu algoritmaların amacı diske erişim hızını yükseltmek ve arama işlemlerinin zamanını küçültmektir.

Diske erişim süresi başlıca 2 bileşenden oluşur.

1. **Arama süresi :** Disk konumunun başlığını arama bölümü içeren silindirin üzerine getirilme süresidir.

2. **Dönme süresi :** Aranan bölümün disk başlığına doğru dönme süresidir.

Disk başlığının ileri-geri hareketini sağlayan algoritmalar vardır. Bunlardan bazıları ;

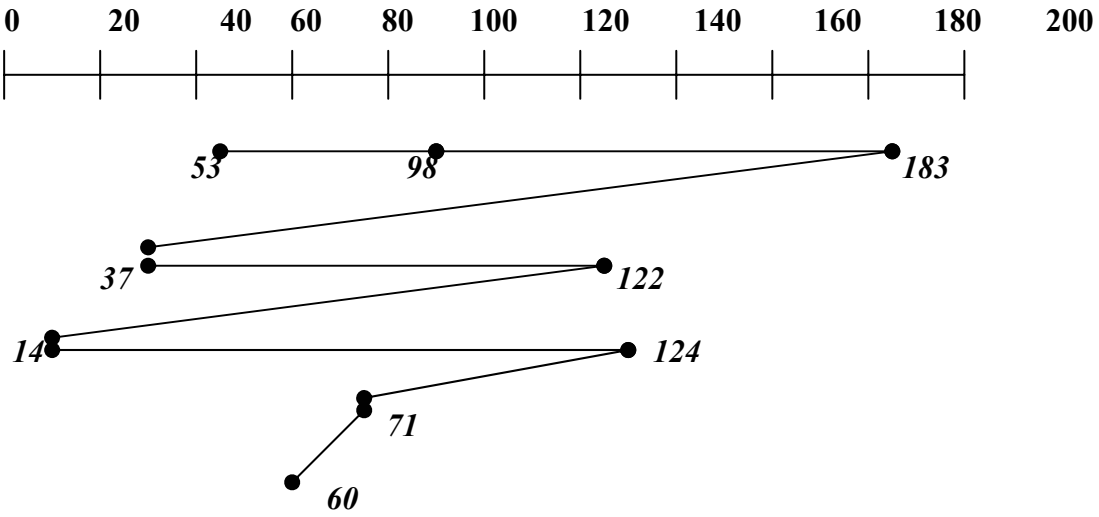
1. FCFS (First Come First Served) Algoritması :

Varsayalım ki disk blokları aşağıdaki gibi kuyruk oluşturur.

98, 183, 37, 122, 14, 124, 71, 60

Kütük için bu bloklar okunacaktır. Sırası önemli değil. Çünkü bir yerde yığılacaklar.

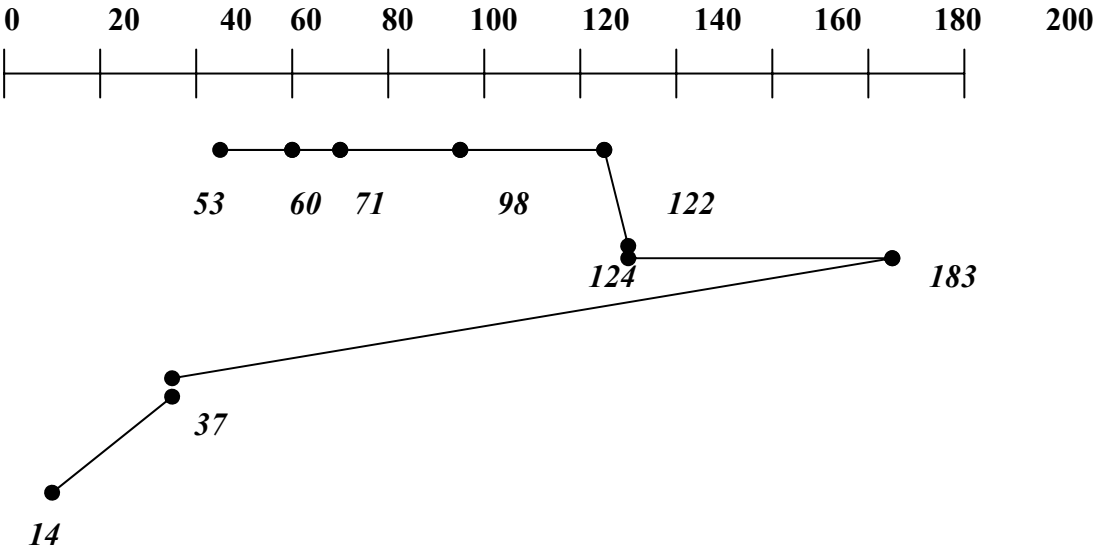
Diskin 53. silindir üzerinde olduğu varsayılsın.



Burada tüm bloklara erişmek için 640 silindir okunacaktır.

2. SSTF (En kısa arama süresi olan önce) Algoritması :

Bu algoritmada başlığın şimdiki durumuna göre en kısa arama süresi olan işlem seçilir. Arama süresi başlığın izleyeceği (tarayacağı) silindirler sayısı ile doğru orantılı olduğundan SSTF algoritması disk başlığının ilk durumuna en yakın olan silindiri ilk işlem için seçecektir. FCFS algoritmasındaki örneği uygularsak:

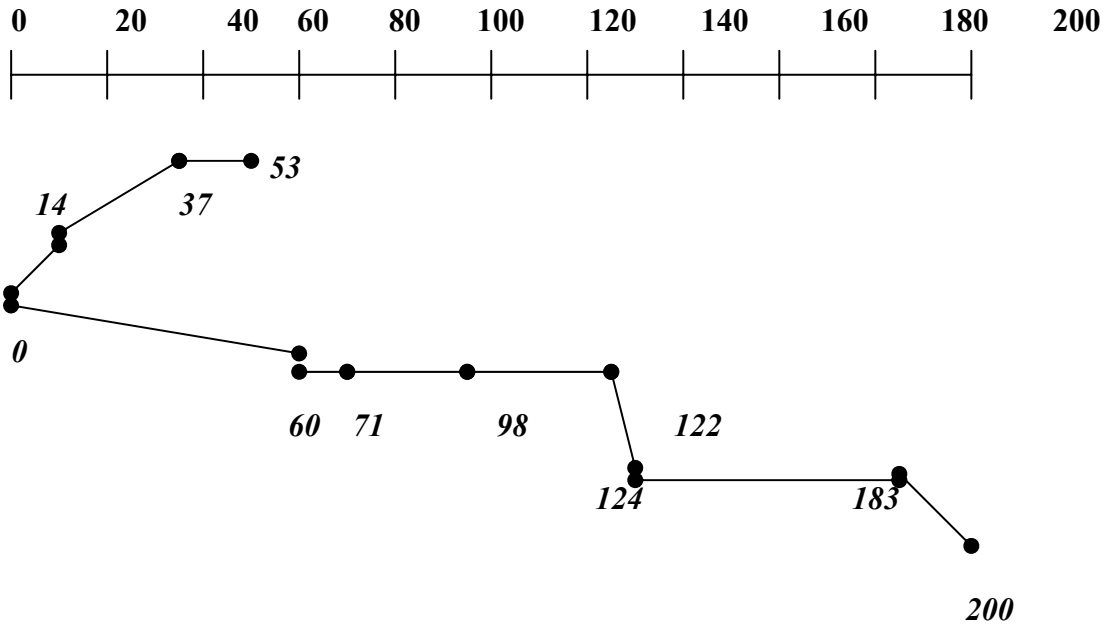


Bu algoritmada toplam izlenen silindir sayısı 299 ‘dur.

Algoritmanın yetersiz yönlerinden birisi her zaman kuyruğa disk başlığının o anki durumuna daha yakın olan silindir sunulabilir. Bunun sonucunda disk başlığından daha uzakta olan silindirlerin işlenmesi ertelenebilir.

3. SCAN Algoritması :

Bu algoritmada disk başlık kolu diskin bir ucundan başlayarak diğer ucuna (kenarına) gidene kadar hareket eder ve karşılaştığı silindirler üzerinde işlem yapar. Sonra diğer kenardan ters yönde hareket eder ve aynı işlemleri yapar. Böylelikle kuyruktaki tüm silindirlerin taranması için diskin önce ileri sonra geri hareketi gerekiyor. Burada önemli olan, disk başlığının hareket yönü belirlenmelidir. Yani disk başlık kolu 0 yönüne (kenarına) doğru hareket edecekse o zaman 53 'den sonra 37 işlenecek, 200 kenarına doğru hareket edecekse 53 'den sonra 60 işlenecektir.

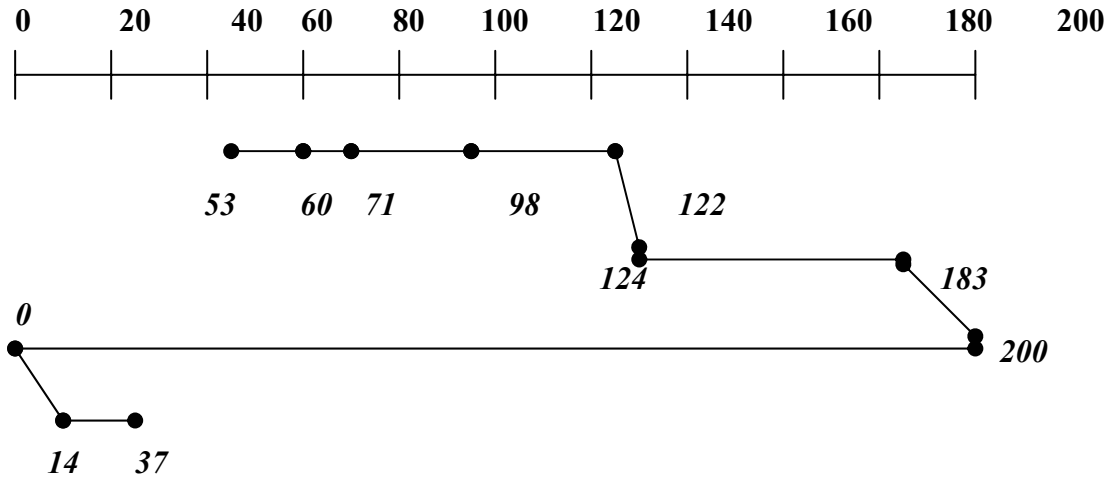


Başlığın hareketi yönünde yeni bir silindir sunulursa o derhal hizmete konulacaktır. Buna asansör algoritması da denir. Örneğin silindir başlığı 98 deyken 100 gelirse, önce 100 işlenir ondan sonra 122 'i işlenir.

Bu algoritmanın bir versiyonu da C-SCAN algoritmasıdır.

4. C-SCAN (Döngülü SCAN) Algoritması :

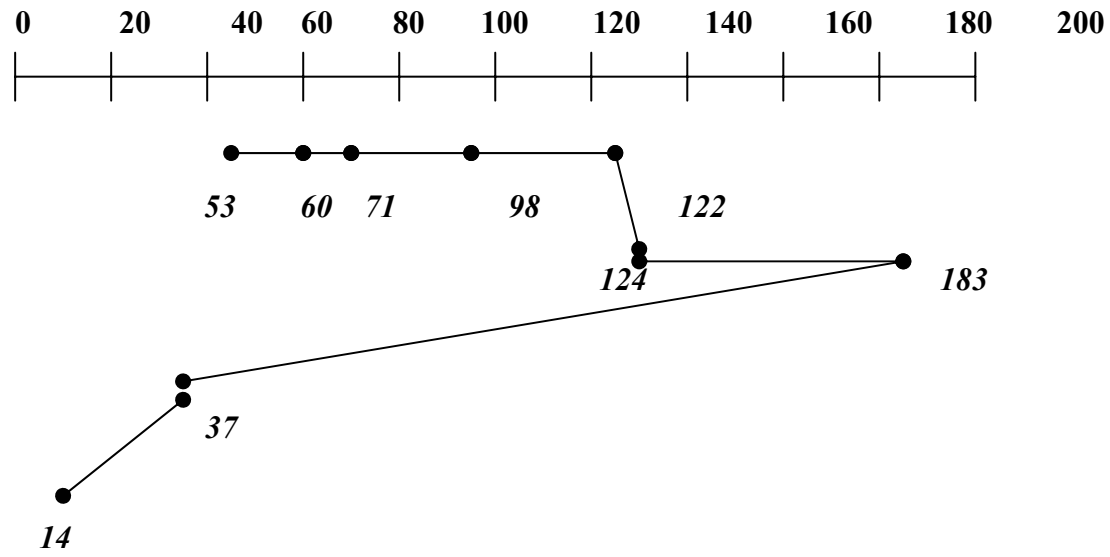
Bu algoritma prensip de SCAN algoritması gibidir. Ancak disk sonuca ulaştığı zaman derhal en başa döner ve taramayı devam ettirir.



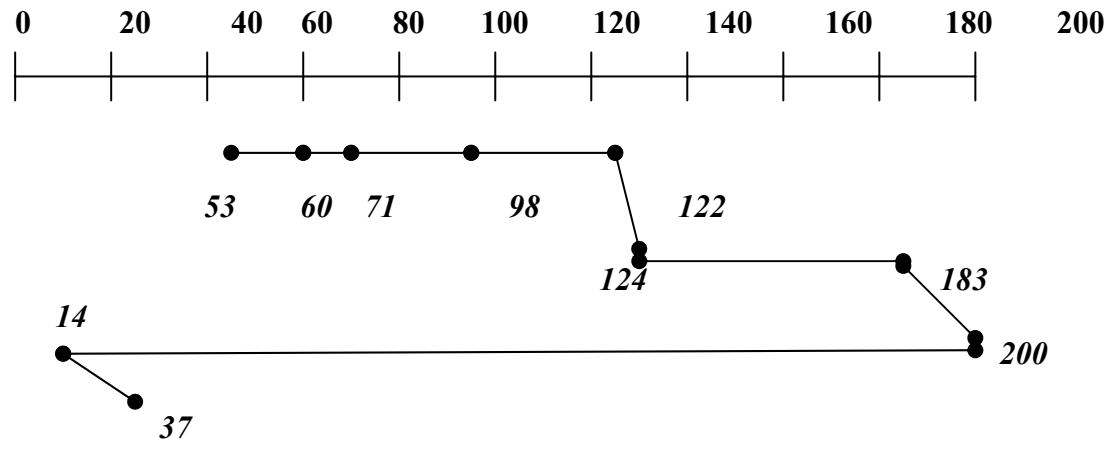
5. LOOK Algoritması :

SCAN algoritmasından farklı olarak LOOK ve C-LOOK algoritmalarında disk başlığı diskin sonuna kadar değil erişilmesi gereken son silindire kadar hareket eder.

LOOK



C-LOOK



8. GİRİŞ-ÇIKIŞ İŞLEMLERİ

İşletim sisteminin önemli fonksiyonlarından birisi G/Ç aygıtlarının yönetimidir. Bu yönetim aygıtlara gönderilen komutlarla, kesilmelerle, hataların kontrolü ve yönetimiyle ifade edilebilir. G/Ç yönetim sistemi aynı zamanda aygıtlarla sistemin diğer bileşenleri arasında ara yüzü oluşturur ve bu ara yüzü kullanıcı için basit bir biçime dönüştürür.

G/Ç aygıtlarını genelde 2 kategoriye ayırmak mümkündür.

- Blok aygıtları
- Karakter aygıtlar

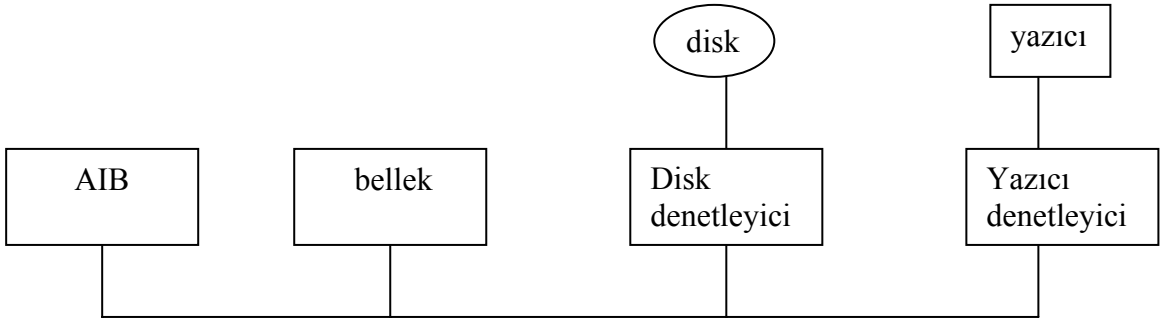
Blok aygıtlarının her biri sabit boyutta adreslerde saklanıyor ve her bir bloğun kendi adresi mevcuttur. Bu tür aygıtlar da blokları diğer bloklardan bağımsız olarak okuyup yazmak veya erişmek mümkündür. Disk blok aygıtıdır.

Karakter aygıtlarında ise, bilgilerin okunup yazılması (bilgi alışverişi) karakter yardımıyla gerçekleştirilir. Bunlar adreslenebilen değildir. Bu bakımdan bu tür aygıtlar da arama yapılamaz. Terminaller, bilgisayar ağ arayüzleri, yazıcılar bu tür aygıtlardandır.

Birçok aygıtlar bu 2 kategoriye ait değildir. Örneğin saat; saat ne adreslenebilendir ne de karakterlerle işlenir.

8.1. Aygıt Denetleyiciler

G/Ç aygıtları mekanik ve elektronik bileşenlerden oluşur. Elektronik bileşene **aygıt denetleyicisi** veya **adaptör** denir. Mekanik bileşen ise aygıtın kendisidir. Denetleyici ve aygıt arasında standart arayüzleri (ANSI, IEEE, ISO) vardır. İşletim sistemi aygıtlarla değil denetleyicilerle ilgilidir. Denetleyici genelde iki veya daha fazla aynı türlü aygıtı denetleyebilirler.



İşlemci ile denetleyici arasında kesin bir fark yoktur. Örneğin disk denetleyici disk işlemlerini yapan bir işlemci gibidir. Her denetleyicinin birkaç yazmacı vardır. Bu yazmaçlar AİB ile ilişki oluşturmak içindir. Bellekte her bir denetleyicinin kendi adresi bulunur.

8.2.Giriş Çıkış Yazılımlarının İlkeleri

Giriş-çıkış yazılımı çok seviyelidir. Her seviye donanımın aygıtlarını bir üst seviyeden saklar. En üst seviye kullanıcıya anlaşılır bir arayüz sunmuş olur.

Yazılımların oluşturulmasında başlıca 4 ilke vardır:

- Aygıttan bağımsızlık
- Aynı biçimli adlandırma
- Hataların yönetimi
- Aygıtların ortak kullanılabilirliği

1. ***Aygıttan bağımsızlık*** **:** G/Ç yazılımların tasarımında başlıca hedef yazılımların aygıttan bağımsızlığını sağlamaktır. Örneğin programın diskteki veya disketteki kütüklere erişimi arasında fark olmamalıdır.

2. ***Aynı biçimli adlandırma*** **:** Birinci ilkeyle sıkı bağlıdır. Kütük adlarıyla aygıt adlarının yapısında hiçbir fark olmamalıdır. Tüm kütüklere aynı yöntemle yol adlarıyla erişilir.

3. ***Hataların yönetimi*** **:** Genelde hatalar mümkün oldukça yönetim seviyesinde yönetilir ve oradan götürülür. Eğer disk denetleyici okuma hatası bulmazsa ilk önce bu hatayı kendisi düzeltmeye çalışır. Eğer bunu başaramazsa aygıt sürücüsü bu işi yapmalıdır. Büyük olasılıkla okuma hatası giderilmek için uygun blok yeniden okunmaya çalışılacaktır. Eğer alt seviyeler hatayı çözemezse yukarı seviyeler bu problemi çözmelidir.

4. ***Aygıt paylaşımı*** **:** Bazı giriş çıkış aygıtları (örneğin disk) birçok kullanıcı tarafından aynı zamanda kullanılabilir. Bu zaman G/Ç alt sisteminin görevlerinden birisi aygıtın kullanıcılar arsında düzgün ve verimli paylaşımını sağlamaktır. Bazı aygıtlar ise paylaşılamaz aygıtlardır. Aynı zamanda bu tür aygıtlara istek veya sorgu gelirse G/Ç sistemi bu taleplerin hangi ardışıklıkla ve yöntemlerle gerçekleşeceğini de belirlemelidir.

İşletim sistemi bu fonksiyonlarını gerçekleştirmek için dört katmandan oluşur.

- Kesilme yöneticileri
- Aygıt sürücüleri
- Aygıttan bağımsız işletim sistemi yazılımı
- Kullanıcı seviyeli yazılım

G/Ç işlemlerinin mümkün olduğu kadar AİB 'den uzaklaştırılması denetleyiciler tarafından yapılması önemlidir. AİB yapılacak işlemleri disk denetleyicisine sinyalle gönderir, ayrıca bellekte hangi adrese yerleştireceğini de belirler. Bundan sonra AİB başka işlemleri gerçekleştirir. Denetleyici verilen işlemi gerçekleştirir ve işini bitirince AİB 'ye sinyal gönderir.

1. ***Kesilme yöneticileri*** **:** Kesilmeler genelde işletim sisteminin alt seviyesinde saklanır ve işletim sisteminin küçük bir kısmının bu kesilmeler hakkında ilgisi bulunur. G/Ç işlemleri başladıktan bitene kadar görev kesilmeli, ve uygun sinyal geldikten sonra görev yeniden işe başlayabilmelidir. Bu fonksiyonları kesilme yöneticisi gerçekleştirir. Kesilme yöneticisinin bir fonksiyonu da G/Ç aygıtlarıyla ilgili hataların analizidir.

2. ***Aygıt sürücüleri*** **:** Aygıt sürücüleri aynı sınıfa ait aygıtları yönetmek içindir. Aygıta bağımlı kodlar aygıt sürücülerinde bulunur. Denetleyicilerin kodlarında bulunan yazmaçlar aygıt sürücüleri tarafından alınır ve sürücüler bu komutların doğru çalışmasını kontrol eder. Sürücülerde denetleyicilerin kaç yazmacı olduğu ve bu yazmaçların hangi amaçlarla kullanıldığı hakkında bilgiler bulunur. Örneğin yalnız disk sürücüsünün diskin bölümleri, yol ve silindirleri, disk başlığı, motor sürücüleri hakkında bilgi bulunur.

3. ***Aygıttan bağımsız G/Ç yazılımları*** **:** Sürücüler ve aygıttan bağımsız yazılımlar arasında kesin sınır işletim sistemine bağlıdır. Öyle ki aygıttan bağımsız yazılımların gerçekleştirdiği bazı fonksiyonları verimliliği yükseltmek amacıyla aygıt sürücüleri ile yapmak mümkündür. Aygıttan bağımsız yazılımların başlıca fonksiyonu tüm aygıtlar için ortak olan G/Ç fonksiyonlarını gerçekleştirmek ve kullanıcı seviyesiyle tek biçimli ara yüzü oluşturmaktır. G/Ç aygıtları için ortak fonksiyonlar okuma yazmadır. Kullanıcının diske, diskete ve yazıcıya yazmak arasında hiçbir fark olmamalıdır.

4. ***Kullanıcı seviyesi G/Ç arayüzleri*** **:** G/Ç yazılımlarının büyük bir kısmı işletim sistemine aittir. Küçük bir kısmı ise kullanıcı programlarıyla birlikte bir bağlantı oluşturan (link) kütüphanelerdir. Bu programlar (kütüphaneler) işletim sisteminin dışında çalışan programlardır. G/Ç işlemlerini içeren (gerçekleştiren) sistem çağrıları kütüphane yordamları tarafından oluşturulur. Bütün aygıtlara aynı biçimli erişim sağlanmalıdır.

9. DAĞITILMIŞ İŞLETİM SİSTEMLERİ (D.İ.S)

Dağıtılmış sistemlerin en büyük özelliği bu sistemlerde yazılım mimarisinin (yapısının) merkezleşmiş sistemlerden tamamen farklı olmasıdır. (Not: D.İ.S. henüz gelişim aşamasındadır. Bu sistemlerde henüz kabul edilmiş standartlar yoktur.)

D.İ.S 'nin merkezleşmiş İ.S.'nden aşağıdaki farkları vardır.

- Düşük maliyet
- Yüksek işlem hızı ve verimlilik
- Dağıtılmış uygulamalara yönelik olması
- Daha yüksek güvenirlik

Bilgisayarların ağ ortamında kullanılması verilerin ve aygıtların ortak kullanımı ve iletişim kolaylığını sağlar. (bu bağımsız bilgisayarlardan farklıdır.)

D.İ.S'nin problemleri aşağıdaki gibidir.

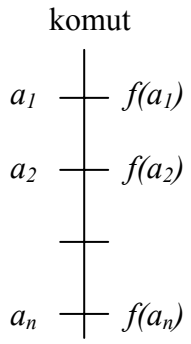
- Ağ sistemlerinin yazılımı yavaş yapılanmıştır, gelişim aşamasındadır.
- Ağda çalışma ile ilave problemler oluşmaktadır.
- Güvenlikle ilgili problemlerin çözülmesi gerekir.

Dağıtık sistemlerin 4 genel mimarisi mevcuttur.

1. SISD
2. SIMD
3. MISD
4. MIMD

1. **SISD (Single Instraction Single Data)(Tek Komut Tek Veri) :** Her komut üzerinde bir veri işlenir. İşlemciye bir komut gönderildiğinde o komut o anda o veri üzerinde işlenir. Tek işlemcili sistemler bu mimariye aittir.

2. **SIMD (Single Instraction Multi Data) (Tek Komut Çok Veri) :** bir tek komut birçok veri üzerinde etkilidir. Çok işlemcili bilgisayarlar (paralel bilgisayarlar) bu mimariye sahiptir.

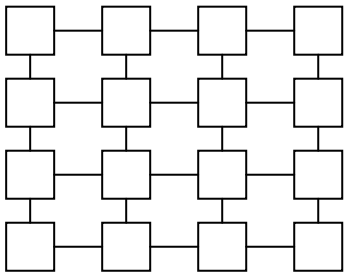


3. **MISD (Multi Instraction Single Data)(Çok Komut Tek Veri) :** Bu mimari teori olarak var ancak gerçekte yoktur. Aynı zamanda veri üzerinde farlı işlemler yapılmasını ifade

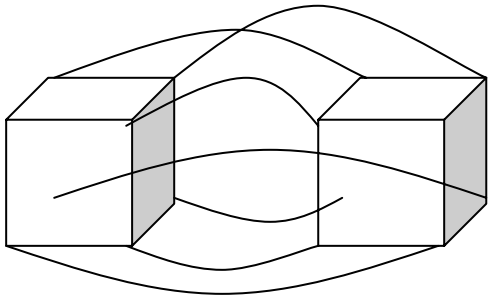
eder. Örneğin k'nın ünlü ünsüz olduğu belirlenirken aynı zamanda alfabenin kaçıncı harfi olduğu belirlenmesin.

4. **MIDM (Multi Instraction Multi Data)(Çok Komut Çok Veri) :** Özellikle ağ sistemlerinde bu mimari kullanılır.

   Süper bilgisayarlarda SIMD mimarisi kullanılır.



Her biri ayrı bilgisayar. Her birinin kendi diski, belleği,...vs. var.
Bu ağ sistemi değil çok işlemcili sistemdir.
Yönetimi kolaylaştırmak için yapısallık oluşturulur.



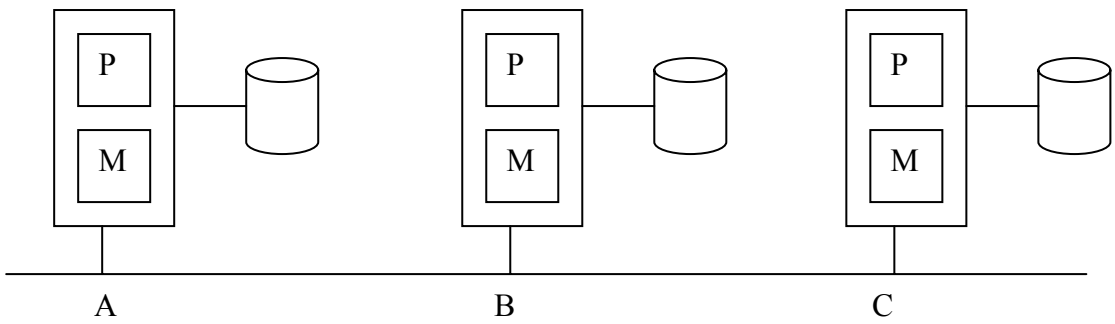
İki küp var. Bunlar birbirine bağlıdır.
Küp sayısı arttırılabilir.
Paralel olarak çalışan bir sistem

1. Çok işlemcili (bilgisayarlı) sistemler
2. Ağ sistemleri

Ağ sistemleri zayıf bağlı sistemlerdir.

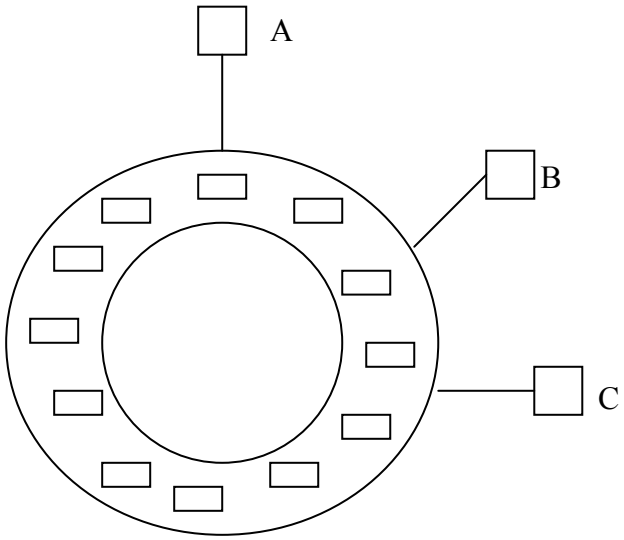
- Bilgisayarlar arasında veri alışverişi yapılabilir.
- Kabukların ortak kullanımı } *yerel sistemlerde*
- Aynı yazılımlara erişim } *tek kanallı*

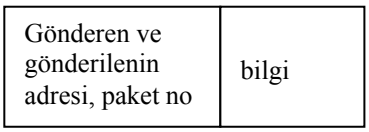
Çok işlemcili sistemler sıkı bağlı sistemlerdir. Bir ana bellekten işlemler gerçekleşir. İşlemcilerin kendi yerel bellekleri olabilir.



Tek kanaldan aynı anda birden fazla veri gönderilmesi bilgi kaybına yol açabilir. Bunun için çeşitli çözümler kullanılır. Örneğin her bilgisayara bir adaptör konulur. Adaptör ağı dinler. Bir bilgiyi göndermeden önce kanal hakkında bilgi edinir. Kanalda bir veri yoksa bilgiyi gönderir.

Bir başka yol da jetonlu halka yöntemidir.

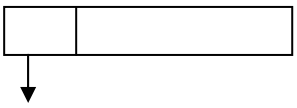




A ‘dan C ‘ye bir veri gönderilmek isteniyor. Veri vagondaki paketlere aktarılır. Paket ilerledikçe her bilgisayar kendine ait olup olmadığına bakar. Eğer kendine ait ise veriyi alır, değilse paket ilerlemeye devam eder.

En çok kullanılan yöntem budur.

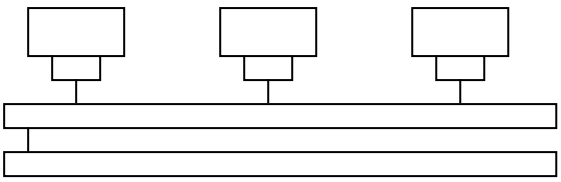
Veri belli büyüklükteki paketlere ayrılarak gönderilir. Veri uzunsa sabit paketlere ayrılır. Örneğin;



128 K’lık bir paket

Alıcı bilgisayar tüm gerekli paketleri aldıktan sonra işlem yapar. Aynı anda birden fazla veri gönderilince hata oluşabilir. Hatalara karşı güvenliği sağlamak için çeşitli yollar vardır. iki kanal kullanılabilir.

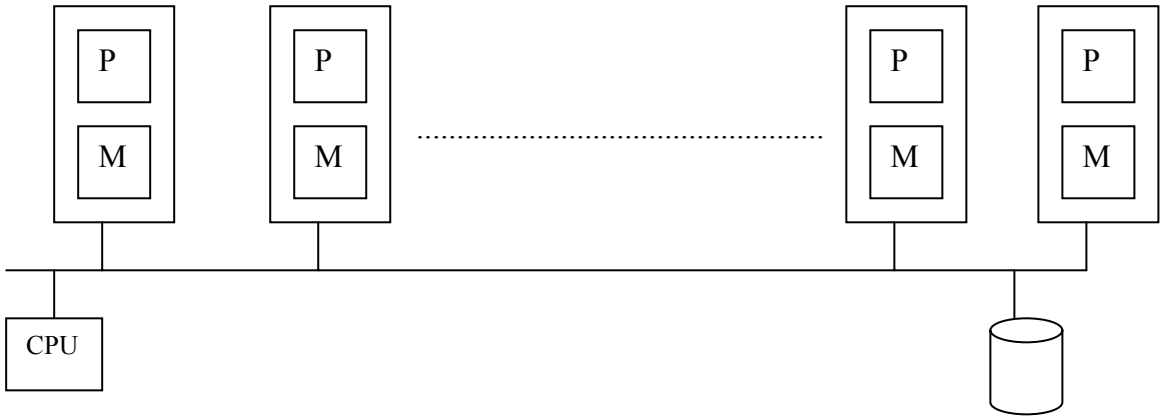
Soğuk Yedekleme : Bir kanalın bozulması durumunda diğeri kullanılıyorsa.
Sıcak Yedekleme : İki kanal aynı anda kullanılıyorsa.



Adaptör kanal ile bilgisayar arasında bağlantıyı sağlar.

Güçlü Bağlantı :

Ortak bir ana bellek vardır. ortak bir disk kullanılabilir. Ortak bir CPU 'yu kullanılabilir.(Sıkı bağlantılı sistemlerde)



Her birinin yerel belleği olabilir. Bunlarda kullanılır. (Ortak disk kullanarak veri alış verişinde bulunulabilir.) Bilgisayar bilgiyi cashe'de bulamazsa ana bellekte arar. Bilgisayarların hepsi daha hızlı çalışmak için cashe bellek kullanılır. Sık kullanılan veriler cashe bellekte saklanır.

9.1. GÖREVLERİN YERLEŞİMİ

Tek işlemcili sistemlerde görevlerin yerleşimiyle ilgili problem yok. Görev mutlaka bulunduğu bilgisayarda çalışır.

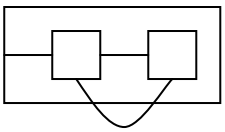
Dağıtılmış sistemlerin hangi mimari üzerinde kurulduğuna bakmaksızın her bir sistemde hangi görevlerin hangi sistemde çalışacağına karar verecek bir algoritma geçerlidir. Görevlerin işlemciler üzerinde yerleşmesi algoritmalarını iki sınıf modele ayırmak mümkündür.

1. **Sınıf model** de Görev birinci kez oluşturulduğu zaman onun nereye yerleşeceğine karar verilir. Herhangi bir bilgisayara veya işlemciye aktarılmış görev kesildiği (silindiği) ana kadar o bilgisayarda kalmış olur. Bu model de sistemde olan bilgisayarların boş durması veya fazla yüklenmesi hiçbir önem taşımaz. Bu model **(nonmigratory) hareketsiz görev modeli** olarak adlandırılır.

2. **Sınıf model** de (görevlerin hareketli olduğu modelde) görevler bir sistemden diğerine hareket edebilir. Hatta görevler bir işlemcide çalışır durumda ise o bilgisayarda kesilip diğer bilgisayarda işine devam edebiliyor. Birinci modele oranla hareketli görev modelinde bilgisayarlar daha dengeli yüklenmiş olurlar ama sistemin karmaşıklığı artar. Bu ve diğer modellerin seçimi minimumlaştırma kıstaslarına bağlıdır. minimumlaştırma kıstası ana işlem biriminin maksimum kullanımı olarak alınabilir veya sorulara cevap süresi olarak alınabilir.

Varsayalım ki iki işlemci ve iki görev bulunur.

Pr 1 Pr 2



P_1 = 10 MIPS
P_2 = 100 MIPS

T_1 = 5 sn (bekleme süresi)

A görevi komut sayısı 100 milyon
B görevi komut sayısı 300 milyon

1. P_1 'de A ve P_2 'de B görevi varsa

    a. A_{cs} = 100/10 =10 m/sn
    b. B_{cs} = 300/100 + 5 = 8 m/sn

2. P_1 'de A ve P_2 'de B görevi varsa

    a. B_{cs} = 300/10 = 30 m/sn
    b. A_{cs} = 100/100 + 5 = 6 m/sn

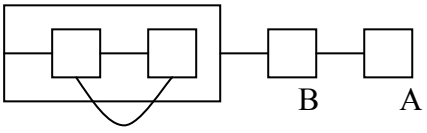
Ortalama çalışma süresi :

İlk durumda (10+8)/2 = 9 m/sn ikinci durumda (30+6)/2 = 18 m/sn

Bu arada iki kıstas önemlidir.

1. işlemcilerin dengeli kullanımı
2. paralellik

2 görevde 2. işlemciye sunulursa



B için : 300/100 = 30

A için : 30 + 100/10 = 40

t = 5 sn

9.2.DAĞITILMIŞ SİSTEMLERDE PLANLAMA (Scheduling)

Normalde her bir işlemcinin kendi yerel planlayıcısı ve planlanma algoritması mevcuttur ve bu planlama diğer işlemcilerde yapılan işlemlerden bağımsızdır. Görevler birbiriyle ilişki olduğu durumlarda bağımsız planlama bazı problemler doğurur. Bunu bir örnek üzerinde gösterelim.

Varsayıl ki A ve B görevleri 1. işlemcide, C ve D 2. işlemcide çalışırlar. İşlemciler zaman paylaşımı algoritmasıyla görevlere hizmet ediyor. Bu zaman diliminin 100 m/sn olduğunu

varsayalım. Kabul edelim ki A ve B çift sayılı dilimlerde C ve D ise tek sayılı dilimlerde çalışırlar.

Zaman Dilimleri	işlemciler I	II
0	A	C
1	B	D
2	A	C
3	B	D
4	A	C
5	B	D
6	A	C
7	B	D

Varsayalım ki A görevi D görevi ile sıkı ilişkidedir, ondan bilgiler alır. 0. zaman diliminde A görevi ise başladığında D 'ye uzaktan erişim süreci çalıştırır. Bu zaman D görevi bekleme durumunda olduğu için A 'nın mesajını alamıyor. C bekler duruma geçtik ten sonra D görevi A 'nın mesajını kabul ediyor ve gereken bilgileri A 'ya gönderir. Ama bu zaman A görevi bekleme durumundadır ve cevap 2. zaman dilimine kadar bekletilir. (200. sn'ye kadar.) Böylelikle D görevinin "0" anında çağrıldığını varsayarsak gereken cevap 200 m/sn 'den sonra alınmış olacaktır.

Birbiriyle sık ilişkili olan görevler aynı zaman diliminde çalıştırılırsa verimlilik sağlanır. Bu tür problemlerin önlenmesi için birbiri ile ilişkili görevlerin aynı zaman diliminde ama farklı işlemcilerde gerçekleştirilmesi gerekir.

9.3. DAĞITILMIŞ KÜTÜK SİSTEMLERİNİN TASARIMI

Dağıtılmış kütük sistemlerinin genelde iki farklı bileşeni vardır.

- Kütük hizmeti (File service)
- Dizin hizmeti (Directory service)

KÜTÜK HİZMETLERİ

Kütükler üzerinde işlemleri (okuma, yazma, güncelleme) gerçekleştirir. 2 modelde gerçekleştirmek mümkündür.

1. Yükleme modeli (Upload down modeli)
2. Uzaktan erişim modeli (Remote modeli)

Yükleme Modeli : Bu modelde kütük işlemi iki farklı işlemi sağlıyor.

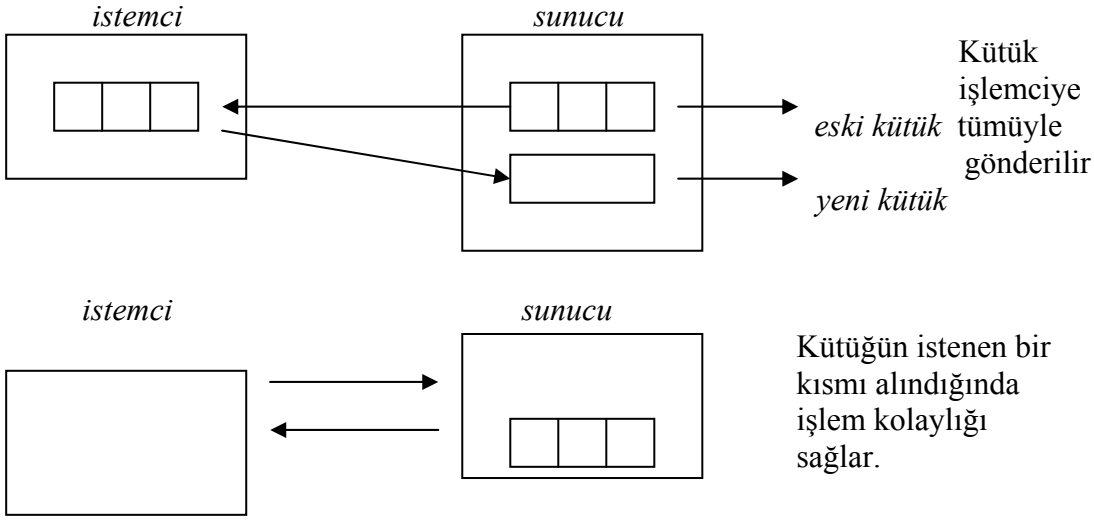
- Kütüğün okunması
- Kütüğün yazılması

Okuma işleminde kütük sunucusundan işlemciye gönderilir.
Yazma işleminde ise kütük işlemciden sunucuya gönderilir.

Bu model basittir. Uygulama programları gerektiğinde kütüğü tümüyle sunucudan alıp yerel diskine veya belleğine burada onlar üzerinde işlemler yapabilirler. Güncellenmiş veya yeni oluşturulmuş kütük geriye sunucuya gönderilir.

Bu modelin yetersizliği tüm işlemciler için büyük hacimde bellek alanı gerekmektedir.

Uzaktan Erişim Modeli : Bu modelde kütükler ve onların ayrı ayrı parçaları üzerinde işlemler yapılabilir. Kütük sistemleri yalnız sunucuda çalıştırılır. Eğer işlemci bilgilerin yalnız bir kısmını talep ediyorsa tüm kütüğün gönderilmesine gerek kalmıyor.



DİZİN İŞLEMLERİ

Dizinlerin oluşturulması ve silinmesi, kütük ve dizinlerin adlandırılması veya yeniden adlandırılması, kütüklerin bir dizinden diğerine taşınması işlemlerini içerir. Dağıtılmış sistemlerde dizinler genelde hiyerarşi yapı oluşturur. Bu yapı genelde ağaç veya graf biçiminde olabilir. Ağaç biçiminde her bir alt dizin yalnız bir üst dizinle bağlantı oluşturur. Graf veya ağ yapısında ise bir alt dizine birden fazla üst dizinden erişim sağlanabiliyor.

Dağıtılmış dizin sistemlerinin önemli problemlerinden birisi şeffaflıktır. Şeffaflığın gerçekleştirilmesi yönlerinden birisi sunucunun değiştirilmesine bakmaksızın kütük yolu adının sabit kalmasının sağlanmasıdır.

/server1/dir1/dir2/x
/server2/dir1/dir2/x

Verinin hangi sistemde saklandığının önem taşımaması, bir bilgisayardaysa başka türlü, başkasındaysa başka türlü erişim olamamalı. Bu dağıtık sistemlerde veri şeffaflığıdır.

Soldakinde (Hiyerarşi sis.) sadece alt dizinlere erişilir. Sağdakinde (Graf sis.) üst dizinlere de erişilebilir.

10. GÜVENLİK VE KORUMA

GÜVENLİK

- Koruma- iç sorundur; bilgisayar sistemindeki dosya ve programlara denetimli erişimin nasıl sağlana bileceğini ifade ediyor
- Güvenlik- iç koruma ile yanı sıra, sistemin çalıştığı dış ortamı da dikkate alıyor.
- Güvenliğin bozulması- kasıtlı,tesadüfi

  kasıtlı: verileri yetkisiz okuma; yetkisiz değişme, yetkisiz silme

YETKİLENDİRME

- Üç temel kavram:
  - kullanıcı sahipliği (anahtar,kart)
  - Kullanıcı bilgileri(kullanıcı kimliği,parola)
  - Kullanıcı özniteliği (parmak izi, imza, göz örüntüsü)
- Parolanın gizliliğinin kaybı:
  - Sıradan bilgilerin kullanımı (doğum tarihi, soyadı…)
  - Tüm mümkün kombinasyonların denenmesi
  - (her kombinasyon için 1 ms hesabıyla 4 rakamlı parola için 5 sn)
- Parolanın şifrelendirilmesi
  - Geçici parola

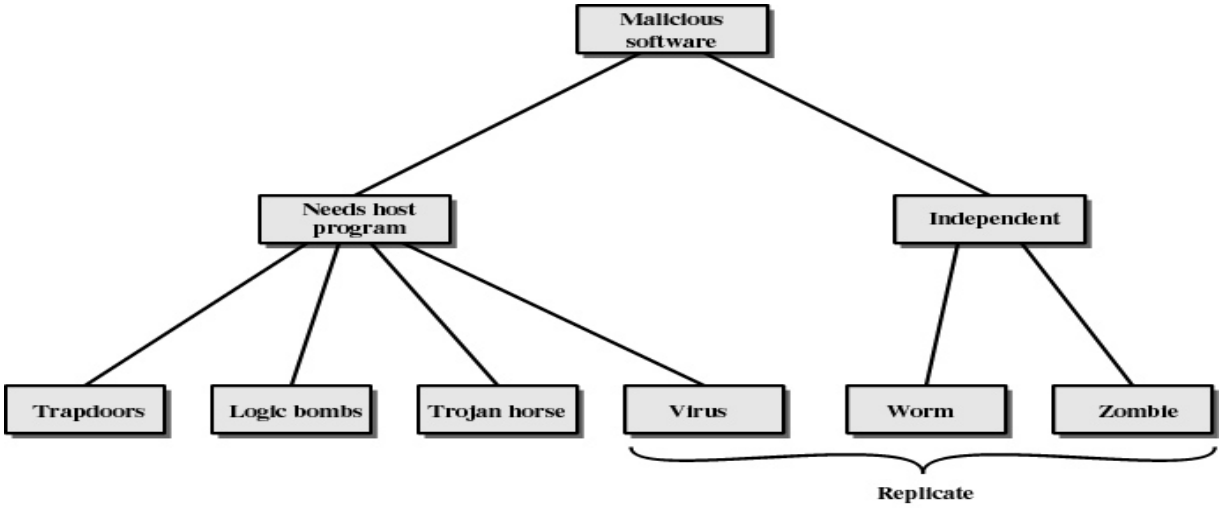
TEHDİTLER

1. Truva atı-Trojan Horse:
   - Çalıştığı ortamda kötüye kullanılan kod kesimi
   - Birileri tarafından yazılmış programların diğerleri tarafından çalıştırılmasına izin verilmesi.

2. Solucan-Worms: ağ bağlantılarını kullanarak mektuplar, uzak erişimler, v.s. yolu ile sistemden sisteme yayılma

3. Virüsler : yasal programın içine sokulmuş kod parçası

Kötü Amaçlı Yazılımlar



Tehditlerin Belirlenmesi

1.Tehdit yapanın davranışlarının yasal kullanıcınınkinden farklı olduğunu varsayalım

2. İstatistiksel sapmaların belirlenmesi

a-Belirli bir zaman zarfında yasal kullanıcının davranışları ile bağlı verilerin toplanılması

b-İstatistiksel denemeler davranışın yasal olmadığını belirlemek için kullanılıyor

3.Kurala dayalı belirleme

a-Önceki kullanım örnekleri esasında sapmaları belirlemek için kuralların geliştirilmesi

b-Yasal olmayan kullanıcıların davranışlarını belirlemek için uzman sistemler

4.Kontrol kayıtları

a-Tüm işletim sistemleri,kullanıcı girişimleri hakkında bilgiler toplayan yazılımlar içermektedir

Tehdit Türleri

1. Kesme-Interruption

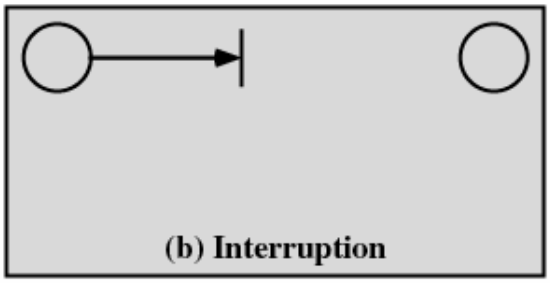
a- Sistemin nitelikleri dağıtılır ve erişilemez veya kullanılamaz duruma getiriliyor.

b- Edinebilirliğe engel

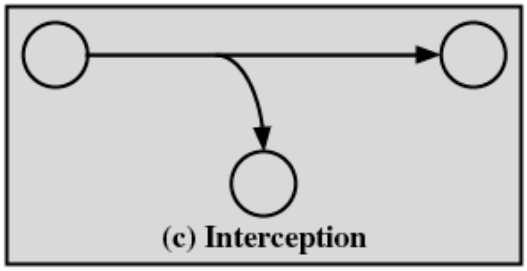
c- Donanımın tahribatı

d- İletişim hattını kırma

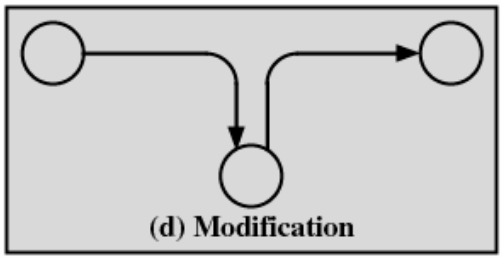
e- Dosya yönetimi sistemini çalışmaz duruma getirme



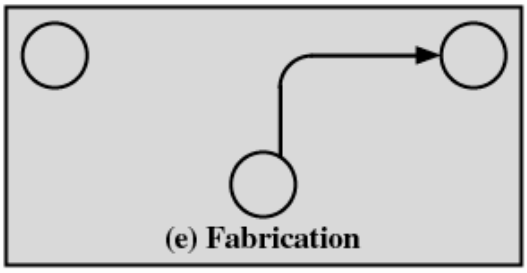
2. Engel olma-Interception
    a- Yetkisiz kişilerin sistemin niteliklerine erişmesi
    b- Gizliliğe müdahale
    c- Ağ üzerinde verileri zaptetme
    d- Dosya ve programların yasak olarak kopyalanması



3. Değiştirme - Modification
    a- Yetkisiz kişiler tarafından sistemin nitelikleri edinilir ve değiştirilir
    b- Bütünlüğü tahdit
    c- Veri Dosyalarında değerlerin değiştirilmesi
    d- Programı, farklı çalışacak biçimde değiştirme
    e- Ağ üzerinde aktarılan haberlerin içeriğinin değiştirilmesi



4. Sahtecilik- Fabrication
    a- Yetkisiz kişiler tarafından sisteme sahte nesneler sokulması
    b- Yetkilendirmenin tahdidi
    c- Ağda yanlış haberler gönderilmesi
    d- Dosyaya kayıtlar ilave edilmesi

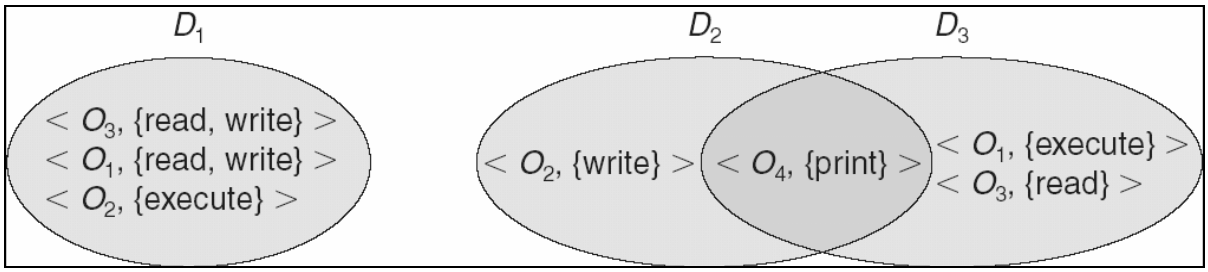


KORUMANIN AMAÇLARI

- İşletim sistemi nesnelerden- donanım ve yazılımlardan oluşuyor
- Her bir nesnenin diğerlerinkinden farklı adı var ve her bir nesneye iyi tanımlanmış işlemler kümesiyle erişile bilir.
- Koruma- her bir nesneye doğru ve yalnız bu nesne için izin verilen işlemlerle erişimin sağlanmasıdır.

Alan Yapısı

1. Alan = erişim hakları kümesidir
2. Erişim hakkı = <nesne-adı, haklar -kümesi>
   haklar-kümesi-nesne üzerinde yapıla bilecek tüm geçerli işlemlerin altkümesidir



Erişim Matrisi

1. Koruma mekanizması matrisle (erişim matrisi) ifade ediliyor
2. Satırlar alanlara,sütunlar nesnelere uygundur
3. *Erişim(i, j)* – Alan (i) ‘de çalışan görevin Nesne (i) üzerinde yapabileceği işlemler kümesini ifade ediyor

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Erişim matrisinin kullanımı

1. Eğer *Di alanındaki görev Oj* nesnesi üzerinde bir işlem yapmaya çalışıyorsa bu işlem erişim matrisinde bulunmalıdır

2. Dinamik koruma oluşturula bilir
   a- Ekleme,silme hakları.
   b- Özel erişim hakları:
   - *Oi'nin sahibi*
   - *Oi nesnesinden Oj nesnesine kopyalama*
   - *Kontrol (denetim) – Di , Dj alanının erişim haklarını değiştirebilir*
   - *Transfer (aktarım) – Di alanından Dj alanına geçiş*

3. Erişim matrisi tasarımı mekanizmayı (yöntemi) stratejiden ayırıyor
   a- Mekanizma
   - İşletim sistemi erişim matrisi +hakları sağlar
   - Mekanizma,matrisi yalnız yetkili kişilerin kullanmasını ve yalnız kesinleşmiş kurallarla erişimi sağlar

   b- Strateji
   - Stratejiyi kullanıcı belirler.
   - Kim hangi nesneye ve hangi kipte erişebilir

Erişim matrisinin çalıştırılması

1. Her sütun = bir nesne üzerinde erişim denetimi listesi. Kim hangi işlemi yapabilir.

   Domain 1 = Read, Write
   Domain 2 = Read
   Domain 3 = Read
   ⋮

2. Her satır = Yetenek listesi (anahtara benzer)
   Her bir alan için hangi işlemlerin hangi nesneler üzerinde yapıla bileceğini belirler.

   Object 1 – Read
   Object 4 – Read, Write, Execute
   Object 5 – Read, Write, Delete, Copy

Erişim matrisi - Alanlar nesneler

Switch(i,j) - i alanında çalışan görev j alanına aktarıla bilir:

Örnek: D2 alanında çalışan görev D3 veD4 alanlarına, D1'deki görev D2'ye aktarılabilir.

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Kopyalama hakkı

Kopyalama hakkı işlemin üzerinde * ile gösteriliyor.

Örnek: D2 alanında çalışan görevler uygun sütundaki (F2 dosyası) nesnenin içeriğini diğer nesnelere de okuma hakkını verebilir. Alt tabloda D3 alanındaki görevlere de bu hak tanınmıştır.

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Sahiplik hakkı

Sahiplik hakkı –yeni hakların ilave edilmesi veya alınmasını kontrol eder.

Eğer tablonun (i,j) içeriğinde sahiplik hakkı bulunuyorsa, Di alanında çalışan görev j sütunundaki her bir içeriğe istediği hakkı ilave ede veya silebilir.

Örnek: D2 alanındaki görev F2'nin sahibidir ve D3 alanındaki görevlere de F2'ye yazma hakkını vermiştir.

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

Değiştirilmiş erişim matrisi
Access(i,j) = Kontrol ise, Di alanında çalışan her bir görev, j satırındaki her bir hakkı silebilir (ilave edebilir).

Örnek: access(2,4)'de Kontrol olması, D2 alanında çalışan her bir görevin D4 alanındaki hakları değişebileceğini ifade ediyor.

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

KAYNAKLAR

1. Ali Saatçi – Bilgisayar İşletim Sistemleri, Bıçaklar Yayınevi, Ankara-2003
2. M.Ali Salahlı – İşletim Sistemleri Ders Notları, Çanakkale 18 Mart Üniversitesi, 2002.
3. Silberschatz Galvin, Peter Baer - Operating System Concepts, Addison Wesley, 1998.
4. Andrew S. Tanenbaum – Modern Operating Systems, Prentice Hall, 1996