Bilgisayarlarda işletim sistemi, donanımın doğrudan denetimi ve yönetiminden, temel sistem işlemlerinden ve uygulama programlarını çalıştırmaktan sorumlu olan sistem yazılımıdır.

En yaygın olarak kullanılan işletim sistemleri iki ana grupta toplanabilir: Microsoft Windows grubu ve UNIX benzeri işletim sistemlerini içeren grup (bu grup içinde pek çok Unix versiyonu, Linux ve Mac OS sayılabilir).

İşletim sistemi, bütün diğer yazılımların belleğe, girdi/çıktı aygıtlarına ve dosya sistemine erişimini sağlar. Birden çok program aynı anda çalışıyorsa, işletim sistemi her programa yeterli sistem kaynağını ayırmaktan ve birbirleri ile çakışmamalarını sağlamaktan da sorumludur.

1.1. Sistem Kaynakları

Sistem kaynakları, bilgisayar sistemi içerisinde kullanılan aygıtların (seri, paralel, usb port, fare v.s.), programların kontrol edilebilmesi, kullanıcılara hizmet edebilmesi için gerekli mekanizmaları anlatmak için kullanılan kelimelerdir. Sistem kaynakları, sistem içerisindeki donanım elemanlarının CPU ile haberleşebilmesi için paylaştırılır.

Sistem kaynakları iki veya daha fazla donanımın aynı zamanda haberleşmeye çalışmasını engeller. CPU'nun sistem aygıtlarını tanımlayabilmesini ve onlar ile haberleşebilmesini sağlar.

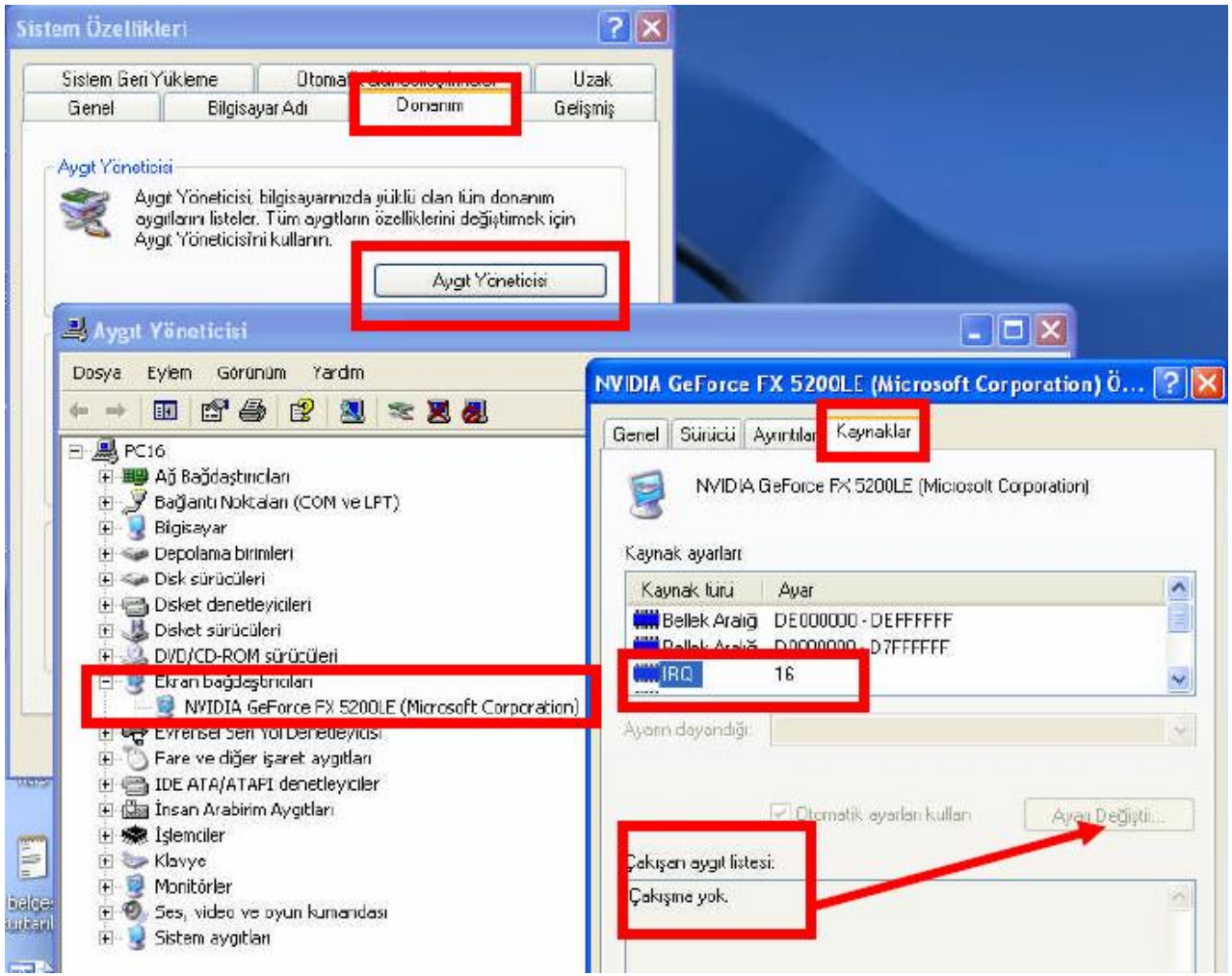
1.1.1. Kesme İstekleri (IRQ)

IRQ İngilizce karşılığı Interrupt Request, yani kesme isteği kelimelerinin kısaltmasıdır. IRQ ile donanımlar sistem işlemcisi ile iletişim kurarlar. Bir nevi her donanımın özel adresi denebilir.

Bilgisayarımızın merkezi işlem ünitesi olarak CPU çevre birimleri ile olan tüm iletişimleri başlatır, onların yönetimini elinde tutar. Peki herhangi bir çevre birimi CPU'nun kendisi ile ilgilenmesini nasıl sağlayacak, onun dikkatini nasıl çekecek. İşte bu noktada IRQ (Donanım kesmeleri) devreye girer. IRQ, çevre birimlerinin dikkat çekmek için kullandığı bir yöntemdir.

- **Kesmeler (IRQ) Nasıl Çalışır?**

Bilgisayarımızdaki kesmeler Intel 8259 öncelikli kesme denetleyicisi (PIC) tarafından sağlanır. Önceleri ayrı bir çip olarak bulunan bu kesme denetleyicisi, günümüz bilgisayarlarında anakartın çipsetinde yerleşik olarak bulunmaktadır. Bir kesme talebi geldiğinde 8259 CPU'yu elindeki işi geçici olarak durdurmaya ve hemen bu kesmeyi yönetmesine olanak sağlayan bir programı çalıştırmaya iter. CPU kesme hizmet programına dallanırken dönüş adresini yığın hafızada saklar ve işi bitince işleme yine kaldığı yerden devam eder. Birçok aygıt aynı anda kesme hizmeti isteyebilir. Sistem tarafından belirlenmiş öncelik sırasına göre talepler karşılanır. Genelde kesme hizmet programı yüksek önceliğe sahip bir işlem tarafından kesilebilir. Ama daha öncelikli veya eşit seviyedeki bir cihazdan kesme talebi gelirse o anki kesme programı bitene kadar bu istek saklanır.



Resim1.1: Bir donanım için IRQ ayarını görme

Eğer bilgisayarımızda bir donanım cihazımız doğru çalışmıyor ise aygıt yöneticisi penceresinden donanım elemanını seçerek **çift tıklarız** ve de açılan sekmede **kaynaklar** sekmesi ile boş olan bir irq seçmeliyiz. Ancak unutmamak gerekir ki, bu ayarlar için bilgi düzeyimiz yeterli değil ise müdahale etmememiz gerekir.

1.1.2. Doğrudan Bellek Erişimi (DMA)

DMA İngilizce karşılığı Direct Memory Access anlamına gelen direkt hafıza erişimi kelimelerinin kısaltmasıdır. Özellikle disk sürücüleri ve benzeri cihazlar için bu seçeneğin aktif halde olması belli bir performans artışı sağlamaktadır. Çünkü bu durum sayesinde cihaz gerek duyduğu bilgileri işlemciye uğramadan direk olarak sistem belleğinden elde edebilir.

Bu kanallar sistem belleğine bazı aygıtların (ses kartı, ethernet kartı gibi) erişimini hızlandırmak için kullanılırlar. Bir sabit disk disk denetleyicisi sabit diskten bazı verileri aldıktan sonra bunları RAM'e depolamak ister. Aynı şekilde yerel iletişim ağı (ethernet)

kartından da veri geldiğinde bunların RAM'e depolanması gerekebilir. Bunları I/O adresleri üzerinden CPU'ya oradan da RAM'e göndermek yerine bazı kartların kullanabildiği DMA (Direct Memory Access - direk bellek erişimi) kanalları vasıtasıyla daha hızlı ve CPU'yu da meşgul etmeden direk RAM'e ulaştırmak mümkün. Bu sayede CPU meşgul edilmemiş olacak ve de bizim isteğimiz daha hızlı bir şekilde yerine getirilmiş olacaktır.

Tekrar özetlersek DMA verileri bir çevre biriminden RAM'e veya RAM'den çevre birimine CPU'nun müdahalesine gerek kalmadan aktarabilmeyi sağlar. Çevre birimlerinin birbirine direk ulaşmasına imkan sağlayamaz. Sisteminize DMA kullanmak üzere kaç tane kart takılabileceği sınırlıdır.

Hafiza erişim bilgilerini de **kaynaklar** sekmesinden görebiliriz. Ancak erişim adres bilgileri çoğunlukla bizim değiştirebileceğimiz bilgiler değildir. Bütün donanım kartları ile ilgili erişim adresleri bilgilerini bilmemiz gerekir ki bu da çok düşük bir olasılıktır.

1.1.3. Giriş/Çıkış Adresleri (I/O)

Bilgisayarımızın patronu olan CPU'nun çevre aygıtlarıyla ve devre kartları (ses kartı, ethernet kartı vs.) ile iletişim kurmak ve bu aygıtları birbirinden ayırt edebilmek için kullandığı Giriş/Çıkış (Input/Output) adresleridir. Bu adresler "port adresleri" veya "donanım adresleri" olarak da bilinir. Zaten CPU'nun dış dünya ile iletişim kurmak için kullandığı iki yol vardır denilebilir. Bunlardan biri bilgisayarımızın ana belleğinin adresleri diğeri de bahsedildiği üzere I/O adresleridir.

- **I/O Adres Çakışmaları**

Her kartın mikroişlemci ile haberleşmesi için farklı bir I/O adresi vardır. Birden fazla kartın aynı adresi kullanması durumuna çakışma denir. İki kartın aynı adresi kullanması durumunda mikroişlemci tarafından gönderilen komutlar bu kartlar tarafından doğru algılanmaz. Bu durum kartların çalışmamasına ya da hatalı çalışmasına neden olur.

Çoğu çevre birimi ve kartlar tek bir I/O adres aralığını kullanır. En basit şekliyle klavyenizin kullandığı I/O adres aralığını başka bir kart kullanmaya kalkarsa, bu kart çalışmayacak, bununla birlikte klavyeniz de devre dışı kalacaktır. Zaten kart üretilirken klavyenin I/O adresini kullanacak bir kart tasarımı yapılmaz. Çünkü bu adres sabittir, klavye denetleyicisi tarafından kullanılmaktadır ve bir standart haline gelmiştir. Kartlar üretilirken bunlar göz önünde bulundurulan önemli kriterlerdir. "Peki o zaman I/O çakışmaları nasıl olabilir?" diye bir soru gelebilir aklınıza. Bazı I/O değerleri standart değildir, sorunları da zaten bu aralık değerlerini kullanan kartlarda görülmektedir. Şayet aynı adresi birden fazla kart için ayarlarsanız çakışmaya sebep olacağı için kartlar görevlerini yapamayacaktır.

Giriş-çıkış adresleri bilgilerini de **kaynaklar** sekmesinden görebiliriz. Dediğimiz gibi bu bilgiler çoğunlukla bizim değiştirebileceğimiz bilgiler değildir.

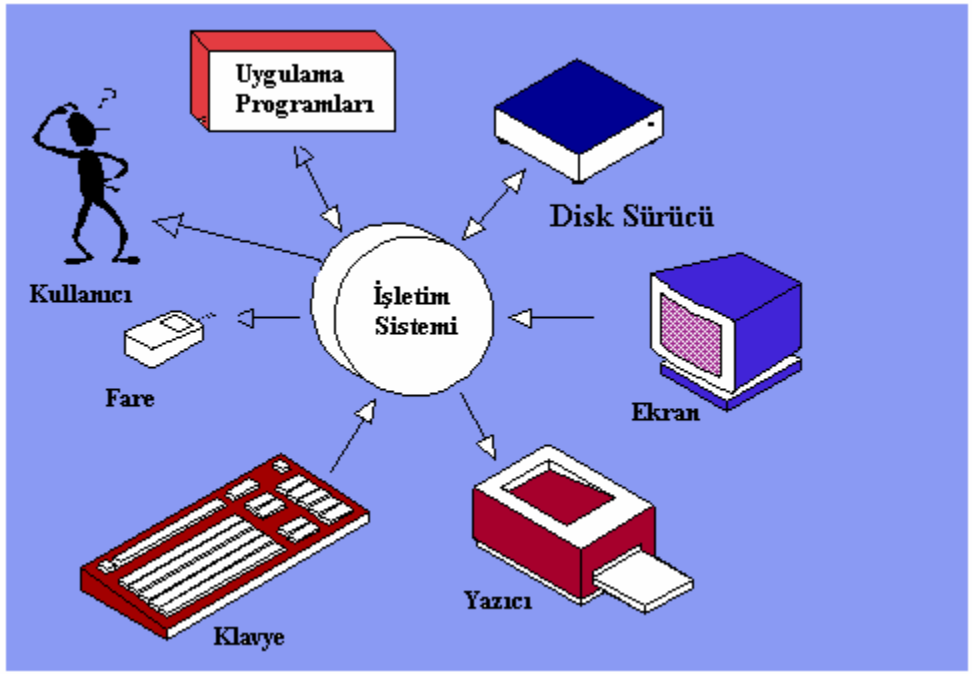
Bu bilgiler neden vardır öyleyse? Cevap basit: Programlama ile uğraşanlar için gerekli olabilir. Özel bir program geliştirildiğinde kullanacağı donanım birimi ile ilgili özel ayarlar gerekir ise bu bilgilerden faydalanarak ayarlamalarını yaparlar.

İşletim Sistemi

Bilgisayar sisteminin en önemli kısımlarından birisi olan işletim sistemi,bilgisayar donanım aygıtları ile kullanıcılar arası ilişkiyi sağlayan bir yazılımdır. Ayrıca kullanıcı programlarının çalıştırılabileceği bir ortam sunar. İşletim sisteminin amacı, kullanıcılar için çalışabilecekleri, bilgisayar aygıtlarını kullanabilecekleri

1- Rahat ve elverişli
2-Verimli

bir ortam sunmaktır.



İsletim Sisteminin Amacı

İşletim sistemleri programların çalışabileceği bir ortam sağlar. Bu şekilde bir ortam oluşturabilmek için işletim sistemi, mantıksal olarak, birbirleriyle iletişimi (arayüzleri) çok iyi tanımlanmış küçük alt yapılara bölünür. Bu alt yapılara " işletim sistemi öğeleri " denir. İşletim sistemleri aynı yapıda olmamakla birlikte, çoğu modern işletim sistemlerinin temel ögeleri şunlardır :

1- İşlem Yönetimi

2- Ana bellek Yönetimi

3- İkincil Bellek Yönetimi

4- Giriş-Çikiş Yönetimi

5- Dosya Yönetimi

6- Güvenlik Yönetimi

7- Ağ Yönetimi

8- Komut Yönetimi

İşletim Sisteminin Tarihsel Gelişimi

Başlangıçta, sadece bilgisayar donanımı vardı. İlk bilgisayar sistemleri bir kontrol konsolünden yönetilen çok büyük makinelerdi. Programcı bu geniş makinelerin her bir aygıtının kullanımını ve yönetimini kendisi yapmaktaydı. Yani, programcı bir program yazar ve bunu yönetim konsolundan kontrol ederek çalıştırırdı. İlkönce program teyplerden ya da delikli kartlardan manuel olarak makinaya (belleğe) yüklenirdi. Sonra programın çalışması için gerekli butonlara kontrol panelinden basılırdı. Program doğru olarak çalışırsa kontrol paneli üzerindeki ışıklar yanardı. Programın çıktısı teyplere ya da delikli kartlara yine programcının yardımıyla kaydedilirdi. Eğer hata oluşmuşsa programcı halt işlemi yapar ve programı kendisi debug (hata ayıklama) ederdi.
İleri ki zamanlarda, makinanın aygıtlarını, kullanıcı programının isteği doğrultusunda kullanabilecek programlar yazıldı. Bunlar modern işletim sistemlerinin temellerini oluşturdu. Daha sonraları tek-kullanıcılı, çok-kullanıcılı, çok-kullanıcılı_çok-işlemcili işletim sistemleri geliştirildi.

Tamponlama(Buffering) :

Tampon (buffer) donanım aygıtlarının veya program işlemlerinin ortaklaşa kullandığı paylaştırılmış bir bellek alanıdır.
Tamponlama (buffering) bir işletim sistemi işlevidir. MİB işleyeceği bilgiyi, tampon olarak kullanılan bellek alnından alır ve işlemeye başlar. Aynı anda girdi aygıtı boşalan bu bellek alanına bir sonraki bilgiyi yerleştirir. Bu işlem girdi aygıtı ile MİB arasında gerçekleştiği gibi, MİB ile çıktı arasında da gerçekleşebilir. Yani MİB işlediği bilgiyi bir tampon alanına yerleştirir. Çıktı aygıtı da bilgiyi buradan alır ve gerekli işlemini gerçekleştirir. Bu işlevin amacı işlemci ile G/Ç aygıtlarının her zaman çalışır durumda olmasını sağlamaktır.

Kuyruklama(Spooling) :

Kuyruklama bir diğer işletim sistemi işlevidir. Kuyruklama da depolama aygıtında, tampon alandan çok daha büyük bir bellek alanı kullanılır. G/Ç ile MİB arasında gerçekleşek olan bilgi alış-verişi bu alanlar üzerinden gerçekleştirilir. Örneğin yapılan depolama aygıtında bir programın MİB'ne yüklenmesi ve koşturulması gerekmekte. Fakat program tampon bölgeye sığmayacak kadar büyük. Bunu içşn depolama aygıtında bir bellek alanı ayrılır ve program o alan üzerine yüklenir. MİB programı bu alandan alarak çalıştırır. Yapılan işlem sonucunda yazıcıdan bir çıktı almak istenebilir. Gönderilen bilgi yazıcının tampon bölgesine sığmayacaka kadar büyük. Bu durumda tampona sığmayan kısım yine depolama aygıtı üzerinde tutulu ve tampon boşaldıkça yeni bilgi tampona gönderilir.
Tamponlama ile kuyruklamanın yararı, sistem performansını arttırmaktır.

Çoklu-programlama :

Birden fazla programın aynı zamanda bir işlemci üzerinde çalıştırılmasıdır. Bu yapıda işletim sistemi belirli bir zaman aralığında bir program parçasını çalıştırır. Bir sonraki zaman aralığında, diğer bir programın bir parçasını çalıştırır. Böylece kullanıcı tüm programların aynı zamanda çalıştırıldığını görür.
Bu yapıda işletim sistemleri oldukça gelişmiştir. Bellekte aynı zaman diliminde birden fazla iş çalıştırılmaya hazır olarak beklemekte olduğundan iyi bir bellek yönetimini gerektirir. Bunun yanında hazır kuyruğunda birden fazla işin çalıştırlmak için beklemesi bir MİB çizelgelemesi gerektirir.

Zaman paylaşımı (Çoklu-işlem) :

Zaman paylaşımı(veya çoklu-işlem) çoklu-programlamanın mantıksal bir neticesidir. Bu sistemlerde işletilecek olan herbir işlem için MİB'de belli bir zaman ayrılır. Takip eden sürede bir diğer işlem gerçekleştirilir.
Zaman paylaşımlı sistemlerde bir bilgisayar birden fazl kullanı tarafında kullanılmak üzere paylaştırılabilir. Bu şekildeki bir sistemde , bilgisayar herbir kullanıcı isteği için belirli bir MİB süresi ayırır. Bu süre içerisinde bir kullanıcının işini yapar. Diğer MİB süresinde diğer kullanıcının işlemini gerçekleştirir. Bu tür işletim sistemlerinde de iyi bir bellek yönetimi, MİB çizelgelemesi ve ikincik bellek yönetimine ihtiyaç vardır.

Dağıtık Sistemler :

Bilgisayar sistemlerinde dağıtık işleme iki şekilde yapılmaktadır.

1. Sıkı sıkıya bağlı (tightly coupled)
2. Gevşekçe bağlı (loosely coupled)

"Sıkı sıkıya bağlı" sistemlerde herbir işlemci belleği ve saati paylaşır. Birbirleri arasındaki veri iletişimi genelde paylatırılmış bellek üzerinden gerçekleştirilirler.

"Gevşek bağlı" sistemlerde ise herbir işlemci kendi belleğine sahiptir. Haberleşme çeşitli iletişim hatları üzerinden gerçekleştirilir. Bu tür sistemlere "dağıtık sistemler" denir.

Gerçek Zamanlı Sistemler :

Bir diğer işletim sistemi çeşididir. Gerçek zamanlı sistemler belirli uygulamaların kontrol aygıtı olarak kullanılır. Bu tür sistemlerde, bilgisayara kontrol bilgisi sensörler(algılayıcı) yardımıyla ulaştırılır. Bilgisayar bu verileri analiz eder. Daha sonra kontrol bilgilerini yine algılayıcılar yardımıyla sisteme iletir.
Gerçek zamalı sistemler, bilimsel deneylerde, tıbbi görüntü sistemlerinde, endüstri kontrol sistemlerinde ve bazı görüntleme sistemlerinde kullanılırlar.

Tek Kullanıcılı Sistemler :

Kişisel bilgisayar sistemleridir. En yaygın işletim sistemi MS-DOS'tur.

3- BİLGİSAYAR SİSTEMİ

3.1 - Giriş

Programlanabilen bir aygıt olan bilgisayar, iki temel karakteristiğe sahiptir.

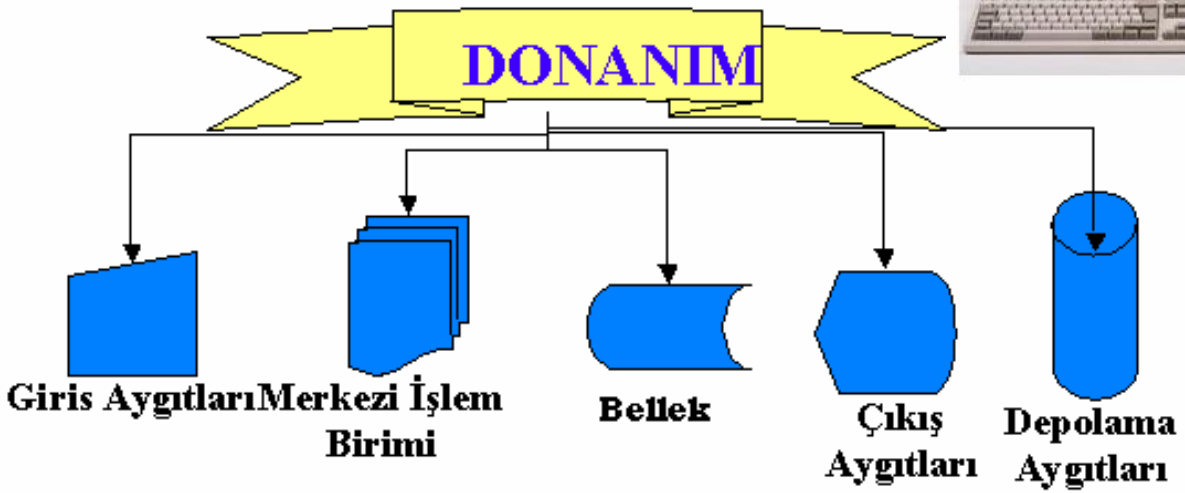
1-Daha önceden tanımlanmış komutlara cevap vermesi
2-Önceden kaydedilmiş komut satırlarını işletebilmesidir

Bilgisayar donanım ve yazılım olmak üzere iki ana bölümden oluşur. Genel bilgisayar yapısı aşağıdaki şekilde gösterilmiştir.

3.1.1 - Donanım

Ekran, klavye, fare, MİB, yazıcı gibi bilgisayarın somut donanım aygıtlarıdır. Donanım, beş ana bölümden oluşur.

- Giriş Aygıtları
- Merkezi İşlem Birimi
- Bellek
- Çıkış Aygıtları
- Depolama Aygıtlar



Donanım yapısı.

3.1.2 - Yazılım

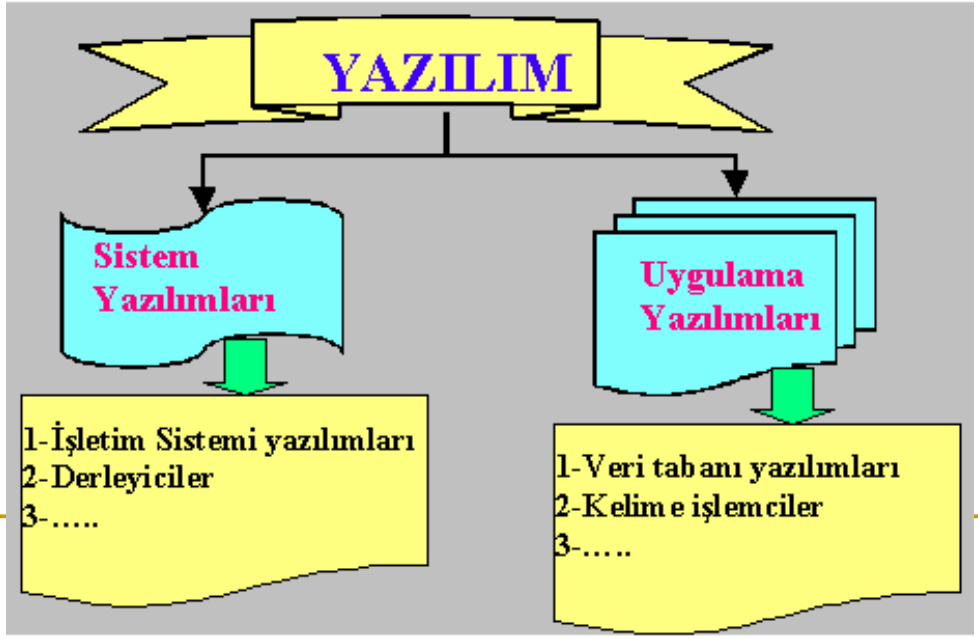
Son kullanıcılar için tasarlanmış uygulamalardır. Yazılım,

1- Sistem yazılımları
2- Uygulama yazılımları

olmak üzere iki ana bölümden oluşur.

Sistem yazılımları, bilgisayar ile doğrudan etkileşen işletim sistemleri ve derleyiciler gibi düşük seviyeli yazılımlardır.

Uygulama yazılımları, son kullanıcıların amaçları doğrultusunda kullanacakları veritabanı, kelime işlemci gibi programlardır. Uygulama yazılımlarının tamamı bir işletim sistemi üzerinde çalışırlar.



Yazılım yapısı.

3.2 - İki Modlu İşleme

Sistem kaynaklarının paylaştırıldığı bir sistem içinde, koşturulmakta olan işlemde meydana gelecek bir hata diğer işlemlerin koşturulumunu engellememelidir. İyi tasarlanmış bir işletim sisteminde, meydana gelecek hatalar diğer programların koşturulmasını etkilemez. Örneğin, koşturulan bir program yazıcıyı kullanmak istediğinde, donanımdan ya da yazılımdan dolayı meydana gelecek bir hata, yazıcı tarafından işletim sistemine bir sistem çağrısı olarak iletilir. Bu çağrıyı alan işletim sistemi, çalışan programın çalıştırılmasını durdurur. Oluşan hataya karşılık gelen uygun bir hata mesajını kullanıcıya gösterir. Bu durum Şekil 3.3'de gösterilmiştir. Bilgisayar sisteminin oluşacak olan hatalara karşı korunup, oluşan hataların diğer işlemlerin işleyişini etkilememesi için, sistemin iki farklı modda çalışmasına ihtiyaç vardır.

1 - Kullanıcı Modu : Kullanıcıya yönelik işlemler gerçekleştirilir.
2 - İzleme Modu(sistem modu ya da denetçi modu olarakta bilinir) : İşletim sistemine yönelik işlemler yapılır.

Sistemin çalıştığı modu belirtmek için **mod biti** denilen bir bitlik bir bilgi kullanılır. Bu bit **0** ise sistem izleme modunda, **1** ise sistem kullanıcı modunda çalışıyor demektir.
Kullanıcıyla ilgili işlemler sistem kullanıcı modundayken gerçekleştirilir. Kullanıcı modunda işlemler koşturulurken bir sistem çağrısı gerçekleşirse, sistem kullanıcı modundan izleme moduna anahtarlanır. Böylece işletim sistemi bilgisayar sistemini daima kontrol altında tutar.

Çift modlu işleme, kullanıcı işlemlerinden ya da sistemden dolayı meydana gelen hatalardan, işletim sisteminin etkilenmesini engeller. İzleme moduna geçildiğinde öncelikli komutlar denilen ve sadece izleme modunda çalışan komutları kullanarak hatanın düzeltilmesi ya da hatanın şekline göre gerekli işlemin gerçekleştirilmesi sağlanır. Animasyon

İşletim Sistemi Yapısı

1 - İsletim Sistemi Hizmetleri :

İşletim sistemleri, kullanıcı ve programlar için çeşitli hizmetler sağlar. Bu hizmetler işletim sistemine göre değişiklik gösterebilir. Bu hizmetlerin belli başlıları şunlardır:

Tablo 1 : İşletim Sistemi Hizmetleri

1- Program koşturulması	**İşletim sistemi, programları ana belleğe yükleyebilmeli ve programları çalıştırabilmelidir. Örneğin bir kelime işlemci programının çalıştırılması.**

2- Girdi / Çıktı işlemleri	Çalışan bir program G/Ç (girdi/çıktı) işlemine gereksinim duyabilir. Bu G/Ç işlemi bir dosya ya da G/Ç aygıtı olabilir. İşletim sistemi bu tür ihtiyaçları karşılar. Örneğin bir kelime işlemci programından daha önceden yazdığımız ve sakladığımız bir dosyanın tekrar açılması gibi.
3- Dosya sistemi yönetimi	İşletim sistemi, programların ya da kullanıcıların bir dosyayı okuma, yazma, oluşturma ya da silme işlemlerini gerçekleştirir. Örneğin bir kelime işlemci programından daha önceden yazdığımız ve sakladığımız bir dosyanın tekrar açılarak üzerinde değişiklikler yapılması ve yeni haliyle bu dosyanın tekrar kaydedilmesi gibi.
4- İletişim	Bir işlem diğer bir işlemle bilgi alış-verişinde bulunmak isteyebilir. Bu iletişim sırasında işlemler aynı bilgisayarda olabileceği gibi; bir ağ ortamında bulunan iki farklı bilgisayarda da bulunabilirler. İşlem arasındaki bu tür iletişimler, ya bellek paylaşımı ya da mesajlaşma gibi çeşitli tekniklerle işletim sistemi tarafından gerçekleştirilir. Örneğin bir kelime işlemci programından daha önceden yazdığımız ve sakladığımız bir dosyanın açılarak yazıcıdan çıktısı alınmak istenebilir. Bu durumda bu aygıttan dosyanın yazdırıldığına dair bir mesajın ekrana gelmesi gibi.
5- Hata tespiti	İşletim sistemi bilgisayar sisteminde meydana gelen her türlü hataları algılayabilmelidir. Bu hatalar, MİB, bellek, G/Ç aygıtı ya da kullanıcı programında meydana gelmiş olabilir. Meydana gelen hataya uygun gerekli eylem ya da düzeltme işlemi işletim sistemi tarfından gerçekleştirilir. Örneğin bir kelime işlemci programından daha önceden yazdığımız ve sakladığımız bir dosyanın açılarak yazıcıdan çıktısı alınması istenebilir. Fakat yazıcı bu işlemi gerçekleştirirken, yazıcıda kağıt bittiğine dair mesaj ya da yazıcının açık olmadığına dair gelen mesaj gibi.

6- Kaynakların paylaştırılması	**Bir çok kullanıcının ya da bir çok programın aynı anda çalıştığı durumlarda elde bulunan tüm kaynakların paylaştırılması ve düzenli kullanılması gerekir. İşletim sistemi birçok farklı kaynağın düzenli bir şekilde kullanılmasını sağlar. Örneğin bir ağ ortamında yazıcının paylaştırılması ve birçok kullanıcının bu yazıcıyı kullanabilmesi gibi.**
7- Kullanıcı işlemleri	**Çok kullanıcılı ortamlarda hangi kullanıcının hangi haklara sahip olduğu ve sistem kaynaklarından ne kadar yararlanabileceği işletim sistemi tarafından yönetilir.**
8- Güvenlik işlemleri	**Çok kullanıcılı ortamlarda ya da ağ ortamlarında tüm bilgilerin güvenliği işletim sistemi tarafından sağlanır.**

2 - Sistem Çağrıları:

Sistem çağrıları, bilgisayarda çalışan program ile işletim sistemi arasında bir arayüz sağlar. Çalışan program ile işletim sistemi arasındaki parametre geçişinde üç temel metod vardır.

1. Kaydedici parametrelerin geçişi
2. Parametrelerin bellekte bir tabloda depolanması, ve tablonun adresinin bir parametre olarak registiriye geçmesi
3. Parametrelerin program tarafından bir stacka konulması, işletim sisteminin parametreleri buradan alması

Tablo 2: Sistem Çağrıları

İşlem Kontrol	*End, Abort, Load, Execute, Create & Terminate Proses, Get & Set Proses Attributes, Wait for Time, Wait & Signal Event, Allocate & Free Memory*
Dosya Yönetimi	*Create & Delete file, Open, Close, Read, Write, Reposition, Get & Set File Attributes*
Aygıt Yönetimi	*Request & Release Device, Read, Write, Reposition, Get & Set Device Attributes, Logically Attach or Detach Devices*
Bilgi Düzenlenmesi/Bakımı	*Get & Set Time or Date, Get & Set System Data, Get & Set Process,File or Device Attributes*
İletişim	*Create, Delete Communication connection, Send,recive Messages, Transfer status information, Attach or Detach Remove Devices*

3 - Sistem Programları:

Sistem programları, bir bilgisayar sisteminde kullanıcı programları ile işletim sistemi arasında yer alarak; program geliştirilmesinde ve çalıştırılmasında programlara daha güvenli bir ortam sunar. Sistem programları tablodaki gibi gruplandırılabilir:

Tablo 3 : Sistem Programları

Dosya Yönetimi	Dosya ve dizin oluşturan, silen, düzenleyen, kopyalayan ya da yeniden adlandıran programlardır.
Durum Bilgisi	Bazı programlar sisteme tarih, saat, bellek miktarı veya disk alanı gibi bilgiler sorarlar.İşte bu gibi bilgiler istenilen çıktı aygıtına, yine istenilen formatta gönderilir.
Dosya Düzenlenmesi	Çeşitli metin editörleri disk ya da teyp üzerinde dosya oluşturur ya da düzenler.
Programların Yüklenmesi ve Çalıştırılması	Programlar derlendikten sonra tekrar çalıştırılması ve ya debug edilebilmesi için gerekli programlardır.
İletişim	Işlemler, kullanıcılar ve değişik bilgisayar sistemlerinin iletişim gerekli sanal bağlantıyı sağlayan programlardır.
Uygulama Programları	Çeşitli amaç ve ihtiyaçlar doğrultusunda geliştirilen programlardır.

4 - Sistem Yapısı:

Geniş ve karmaşık modern bir işletim sistemi küçük alt parçalara bölünerek sorunsuz bir sistem oluşturulabilir. Bu şekilde küçük alt parçalara bölünmüş bir sistemde her bir alt sistemin girdi/çıktı ve yapacağı işlemler çok açık bir şekilde tanımlanabilir. Tüm işletim sistemleri belirli bir yapıda düzenlenmiş alt sistemlerden oluşur. Bu düzenlemede temel olarak iki yaklaşım mevcuttur.

1-Basit yapılı sistemler

2-Katman yapılı sistemler

a- Basit Yapılı Sistemler :

Bu yapılı sistemlerin temel örneği MS-DOS'dur. MS-DOS işletim sisteminin ilkel bir modüler yapısı vardır. Şekilde MS-DOS'un mevcut katman yapısı gösterilmiştir. Bu şekildeki yapılarda sınırlı donanım işlevi kullanımı, birçok işlemin tek bir katman üzerinde yapılması ve güvenliğin zayıf olması temel özelliklerdir.

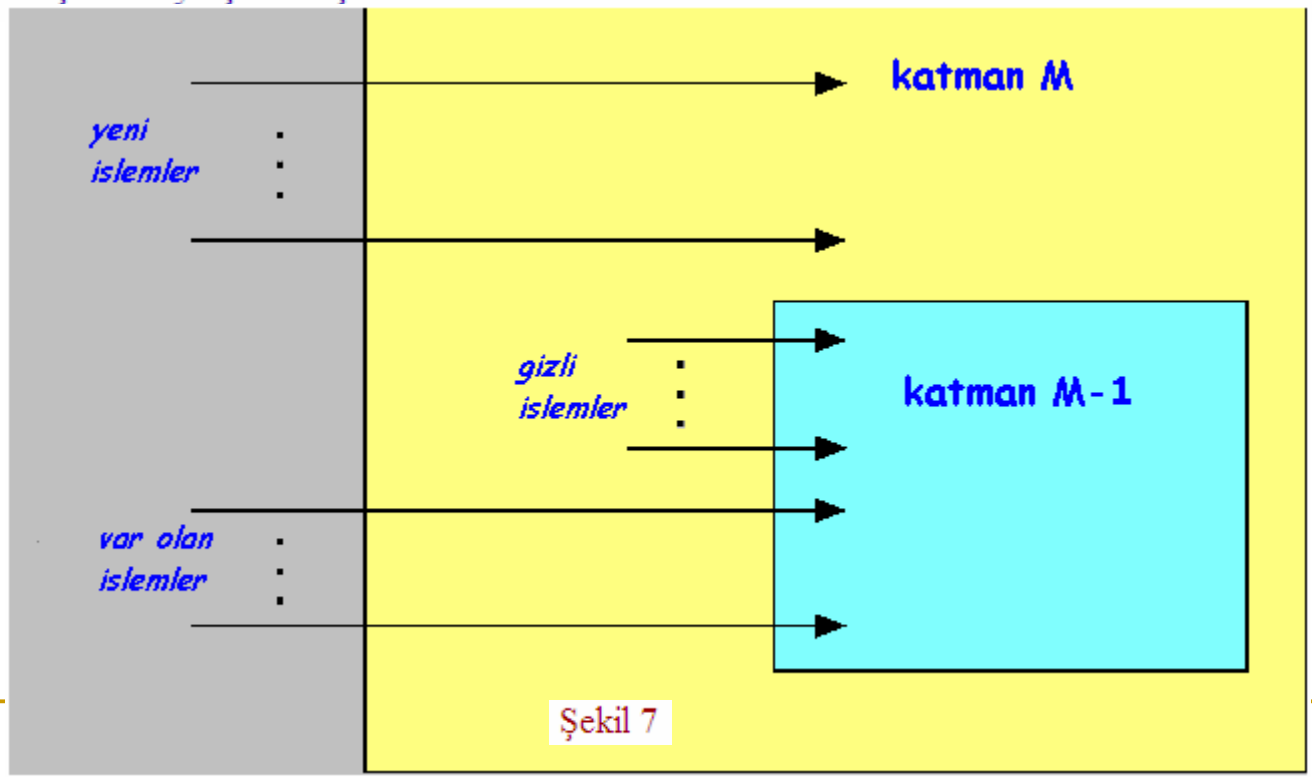
Bu yapıya diğer bir örnek Unix'in ilk sürümleridir. Unix'in ilk sürümleri sınırlı donanım işlevine sahiptir. Çekirdek ve sistem programları olmak üzere iki parçadan oluşur. Çekirdek kendi içinde çeşitli arayüz ve aygıt sürücü programlarından oluşur. Unix yapısını aşağıdaki şekilde inceleyebiliriz.

Şekil 6

Çekirdek, dosya sistemi, MİB tarifesi, bellek yönetimi ve diğer işletim sistemi hizmetlerini sağlar. Görüldüğü üzere birçok işlem tekbir seviyeye verilmiştir. Sistem programlarıda birçok işlemi çekirdek destekli sistem çağrıları ile gerçekleştirilebilir. Unix'in bu yapısı günümüze kadar değişik geliştiriciler tarafından değişikliklere uğratılarak geliştirilip kullanılmıştır.

b- Katman Yapılı Sistemler :

Bu yapılarda sistem daha küçük parçalara bölümlendirilmiş ve katmansal bir şekilde yerleştirilmiştir.Bilgisayar ve donanım daha etkili şekilde yönetilir olmuştur. Kullanıcılara daha rahat bir çalışma ortamı sunulmuştur. Bu yaklaşımda işletim sistemi belli sayıda katmandan oluşur.Bu şekildeki sistemlerde en alt katmanda (LEVEL 0) donanım, en üst katmanda (LEVEL n) kullanıcı arayüzleri bulunur. Bu şekildeki bir yapının ana avantajı modülaritesidir. Herbir katman kendisinin önünde bulunan katmanın servis ve fonksiyonlarını kullanır. Ancak, bu servis ve fonksiyonların nasıl gerçekleştiğini bilmez ve bunada gerek yoktur. Bu yapıyla kontrol ve debug işlemleri oldukça kolaylaştırılmış olur.



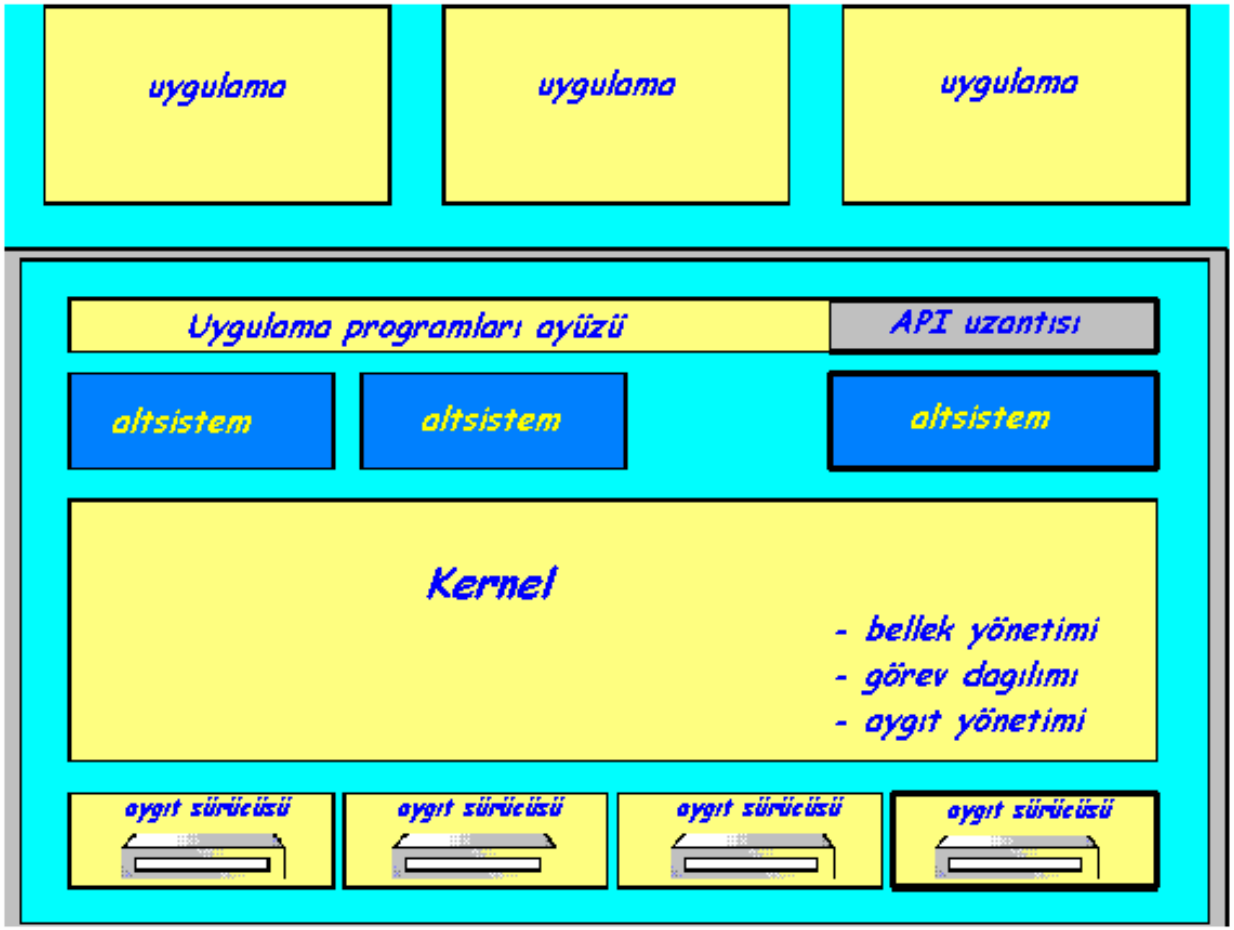
Şekil 7

Katman yapılı olarak gerçekleştirilen ilk işletim sistemi THE (Technische Hogeschool Eindhoven)'dır. THE sistemi altı katmandan oluşmuştur. En alt katman donanım, daha sonraki katman MİB tarifesi katmanıdır. Bu sistemin genel yapısı aşağıdaki şekilde gösterilmiştir.

Level 5:	**Kullanıc Programları**
Level 4:	**Giriş/Çıkış Aygıt Tamponu(Buffer)**
Level 3:	**Kullanıcı Konsolü Aygıt Sürücüsü**
Level 2:	**Bellek Yönetimi**
Level 1:	**MİB Kullanım Tarifesi**
Level 0:	**DONANIM**

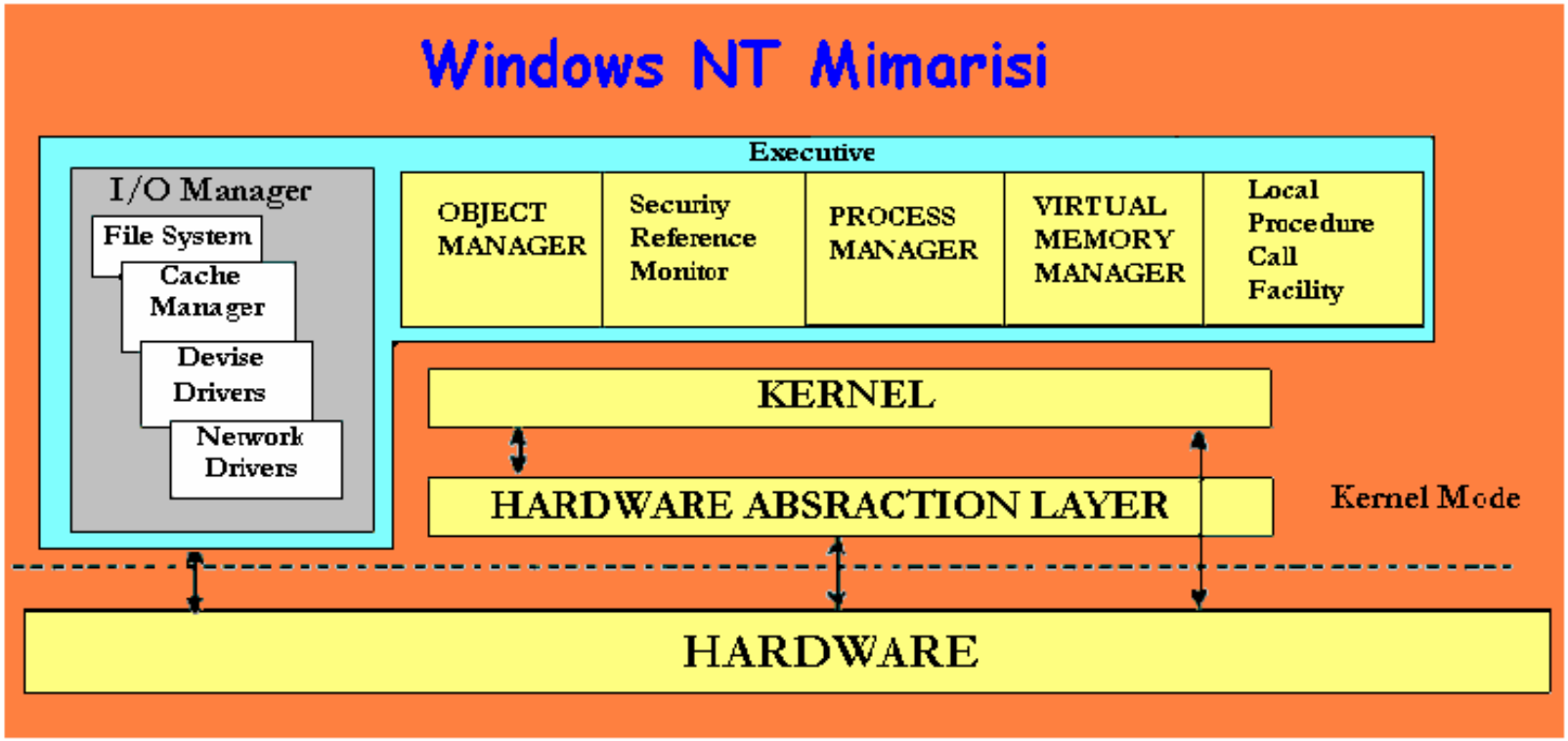
Şekil 8

Ayrıca, Unix sürümlerinin ileriki versiyonları da, katman yapılı ve daha ileri donanım desteğine uygun olarak dizayn edildi. Unix işletim sistemi, daha küçük parçalara bölünerek bilgisayar sistemi üzerindeki etkinliği arttırıldı. IBM tarafından MS-DOS işletim sisteminden sonra tasarlanan bir diğer işletim sistemi OS/2 'dir. OS/2, MS-DOS'un aksine çoklu işlemi destekleyen, çok güçlü donanım özelliklerine sahip ve katman yapılı bir işletim sistemidir. OS/2' de kullanıcıların direkt olarak alt seviyeli katmanlara girmesine izin verilmemiş, işletim sisteminin donanım üzerindeki kontrolü arttırılmıştır.


uygulama
uygulama
uygulama
Uygulama programları ayüzü
API uzantısı
altsistem
altsistem
altsistem
Kernel
- bellek yönetimi
- görev dagılımı
- aygıt yönetimi
aygıt sürücüsü
aygıt sürücüsü
aygıt sürücüsü
aygıt sürücüsü

Şekil 9

32 bitlik, boşaltmalı(preemptive) ve çok görevli modern işletim sistemlerinden biriside MicroSoft NT işletim sistemidir. MS NT işletim sistemide katmansal bir yapıya sahiptir. Güvenlik derecesi yüksek, çoklu işlemciyi destekleyen, MS-DOS ve Windows sürümleriyle tam uyuma sahip bir işletim sistemidir. MicroSoft NT işletim sisteminin mimari yapısı aşağıdaki şekilde gösterilmiştir.



Şekil 10

Sanal Makineler

Bilgisayar sistemi üzerinde, donanım ve işletim sistemi katmanlarının üstünde uygulama programlarının latında çalışan bir çeşit yazılım olarak tarif edilebilir. Bu yazılımlara en iyi örnek JAVA sanal makinesidir.
JAVA sanal makinası(JSM), derlenmiş JAVA programlarını(JAVA bytecode) çalıştıran sanal bir bilgisayardır. JSM donanım) ile işletim sistemi katmanları üzerinde geliştirilen bir yazılımdır. Tüm JAVA programları bir JSM üzerinde çalıştırılmak üzere derlenir. Bu nedenle, eğer derlenmiş bir JAVA programı bir sistem üzerinde çalıştırılmak istenirse, o sistemde JSM mutlaka bulunmalıdır.

JAVA dilinin geliştirilmesindeki ana amaçlardan biri, ortamdan bağımsız yazılımlar geliştirilmesiydi. JAVA sanal makinası bu amacı gerçekleştirmek için geliştirildi.

IŞLEM YÖNETİMİ

MİB 'nun temel görevi kullanıcı programlarının işletilmesi olmasına rağmen, MİB 'nun yapması gereken sistem işlemleri de mevcuttur (tamponlama, kuyruklama gibi). İşlem, MİB 'da gerçekleştirilen tüm eylemlerdir. MİB 'da koşturulan herbir program için belirli bilgisayar sistemi kaynaklarının kullanılmasına ihtiyaç vardır. Diger bir yaklaşımla, herbir işlem, MİB zamanını, belleği, dosya ya da G/Ç aygıtlarını kendi görevini tamamlamak için kullanır. Her bir işlem işletilmek için MİB'nun boş kalmasını, diğer bir ifadeyle bir önceki işlemin MİB 'da işlemini bitirmesini bekler.
İşlem yönetiminde işletim sistemi aşağıdaki eylemlerden sorumludur.

1- Sistem veya kullanıcı işlemlerinin başlatılmasını veya silinmesinden
2- İşlemin bekletilmesinden
3- İşlem senkranizasyonu için bir mekanizmanın sağlanmasından
4- İşlem iletişimi için bir mekanizmanın sağlanmasından
5- Kilitlenme için bir mekanizmanın sağlanmasından

a- Işlem Durumu

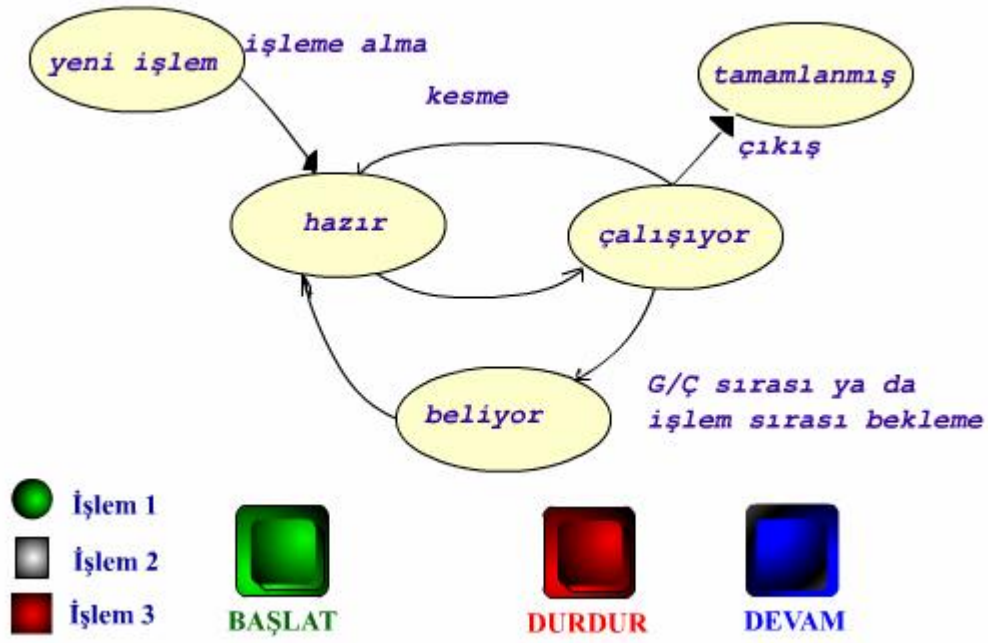
Bir işlemin herhangi bir andaki eylemi işlemin o andaki durumunu belirler. Bir işlem 3 durumda olabilir :

1- Hazır : İşlem işletilmek için hazır.

2- Çalışıyor :İşlem icra ediliyor.

3- Bekliyor :İşlem bir olayın olması için bekliyor.

Animasyon



Şekil 12

b) Işlem Kontrol Bloğu

Herbir işlem, işletim sisteminde kendi işlem kontrol bloğu ile tanınır. Bir işlem kontrol bloğu aşağıdaki bilgilerin bir araya gelmesiyle oluşur.

- Işlemin Durumu : Işlemin o anki durumunu (yeni işlem, hazır, çalışıyor, bekliyor veya bitirilmiş) gösterir.
- Program Sayacı : Işlem için çalıştırılacak bir sonraki komutun adresini gösterir.
- MIB Kaydecileri : Işlemin çalıştırılması sırasında MIB'nun register durumunu gösterir.
- MIB Tarife Bilgisi : Işlemle ilgili MIB tarife bilgilerini gösterir.
- Bellek Yönetimi Bilgisi : Register limitlerini veya sayfa tablolarını gösterir.
- Sayım Bilgisi : MIB kullanım süresini, zaman limitlerini, işlem sayıları vb. bilgileri gösterir.
- Girdi / Çıktı Bilgi Durumu : G/Ç aygıt kullanım bilgisi, açık dosyaların listesi veya kullanılan G/Ç listesini gösterir.

Aşağıdaki şekilde bir işlem kontrol bloğu gösterilmiştir.

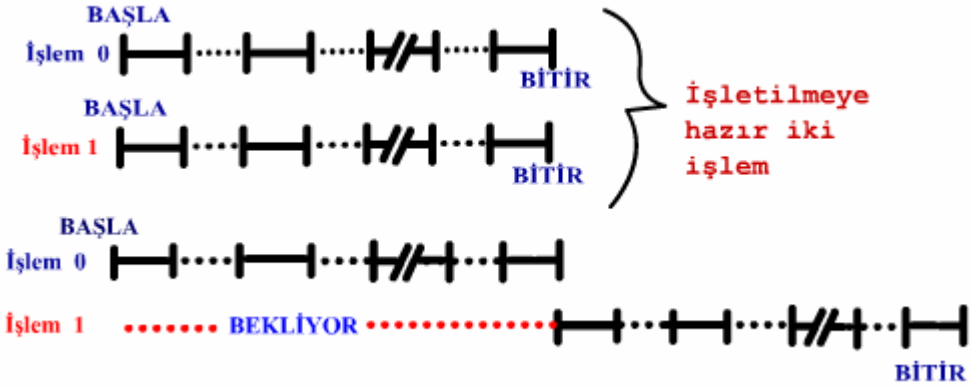
Şekil 13

5.2 Çizelgeleme

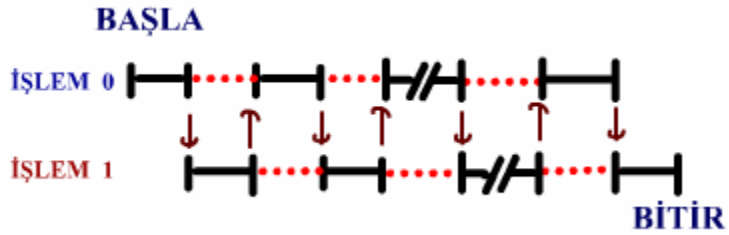
5.2.1 - Giriş

Çoklu program işlemenin amacı MİB'den yararlanmayı en üst seviyeye çıkartarak, MİB'de sürekli olarak çalışan bir işlemin olmasıdır. Fakat, MİB'de herhangi bir anda sadece bir işlem çalışır. Diğer işlemler MİB'nin boş kalmasını bekler. Şekil 5.3'de ilk anda işletilmeyi bekleyen iki işlem bulunmaktadır. Bunların uzunlukları aynı olup, düz çizgi ile gösterilen anlar işlemin çalıştığını, kesikli çizgiyle gösterilenler ise işlemin MİB'de bulunmadığı zamanı ifade eder. Bu iki işlemi çoklu programlama olmadan MİB'de koşturursak alınan toplam süre Şekil 5.3'de gösterilmiştir. Bu iki işlem çoklu program işlemeyle koşturulursa yaklaşık olarak %50 daha az bir zamanda işlemleri tamamlanmaktadır. Bu durum da Şekil 5.4'de gösterilmiştir.
İşte MİB'den yararlanmayı en üst düzeye çıkartmak için yapılan bu çalışmaya " çizelgeleme " denir.

Animasyon 1



Animasyon 2



Şekil 5.4 - Çoklu program ile işlemlerin koşturulması.

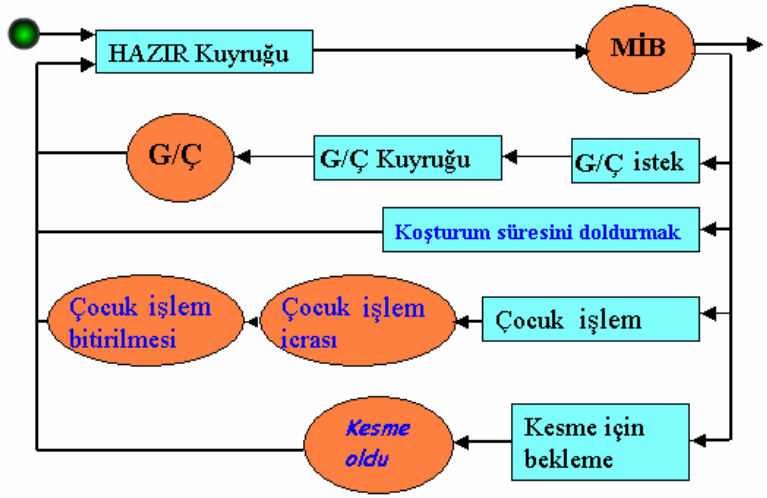
5.2 Çizelgeleme

5.2.2 - Çizelgeleme Kuyrukları

Işlemler bir sisteme girecekleri zaman belirli işlem sıralarına konurlar.

1- Eğer bir işlem ana belleğe yerleştirilmek için, sabit disk gibi ikincil saklama ortamında bekletiliyorsa, bu kuyruğa **İŞ KUYRUĞU** denir.
2- Eğer bir işlem ana bellekte işletilmeye hazır olarak bekletiliyorsa bu kuyruğa **HAZIR KUYRUĞU** denir. Genellikle birbirini takip eden işlemlerden oluşan bir listedir.
3- Eğer bir işlem bir G/Ç aygıtını kullanmak için bekletiliyorsa buna **AYGIT KUYRUĞU** denir. Herbir aygıt kendi aygıt kuyruğuna sahiptir.

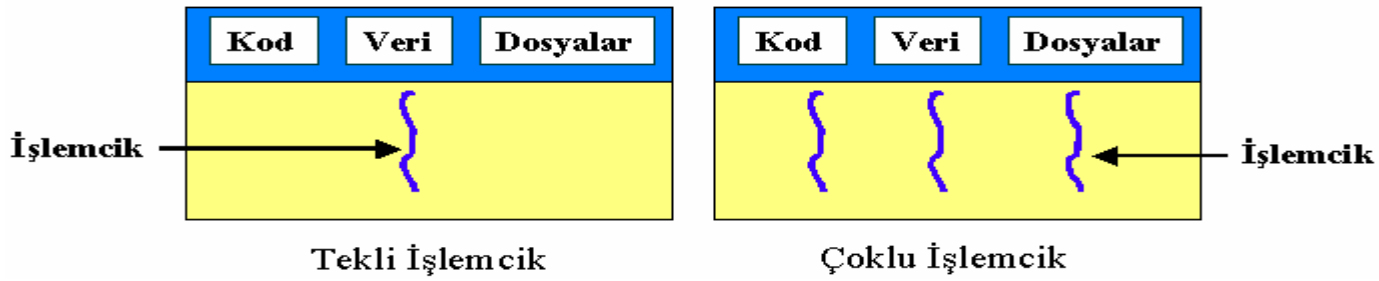
Animasyon



5.3 İşlemcik (Thread)

Çoklu program işleme tekniğinde, MİB'de, bir işlemden diğer bir işleme gerçerken, periyodik olarak konteks anahtarlama gerçekleşir. Gerçekleşen bu konteks anahtarlamadan dolayı belirli bir zaman kaybı olur. Konteks anahtarlama sırasında işlemler sürekli olarak ana bellek üzerine yazılır ya da ana bellek üzerinden okunurlar. Bu nedenle ana bellek üzerinde sürekli olarak okunup yazılma işlemi gerçekleştirilir. Bu iki durumda istenmeyen durumlardır. İyi bir işletim sisteminde konteks anahtarlama sayısının düşük olması istenir.

Modern işletim sistemlerinde işlemler geleneksel(yüksek ağırlıklı) işlemler ve hafif ağırlıklı (işlemcik -thread) işlemler olmak üzere ikiye ayrılır. Modern ve karmaşık yazılımlar tek bir işlem yerine birlikte çalışabilen işlemlerden oluşur. Bu işlemler arasında yukarıda anlatılan konteks anahtarlama sorununun yaşanmaması için, bazı işlemler için gerekli olan alt işlemler gruplandırılarak o işlemin bir parçası haline getirilmiştir. Bu tür işlemlere hafif ağırlıklı işlemler denir. Hafif ağırlıklı işlemler içerisinde birden fazla işlemciğe (çoklu işlemcik-multithread -) sahiptir. Ana işlemlere ise yüksek ağırlıklı işlemler denir. Yüksek ağırlıklı işlemlerin içerisinde bir tek işlemcik (tek işlemcik) bulunur. Çoklu işlemcik ile tek işlemcik arasındaki fark Şekil 5.7 'de görülmektedir. Örneğin, bir kelime işlemci programında, yazma işlemi gerçekleştirilirken, klavyeden bilginin alınması, ekrana basılması ayrı ayrı gerçekleşen iki işlemciktir. Bunun yanında bir diğer işlemcikte yazım hatalarını kontrol ediyor olabilir.



Tek işlemcikli ve çok işlemcikli işlemler.

İşlemcik(thread) için çeşitli tanımlamalar yapılabilir.

- İcra edilmekte olan bir işlemin, aynı kodunu ve adres boşluğunu kullanarak yeniden çalıştırılmasıdır. Modern işletim sistemlerin de bu işleme işlemcik(thread) adı verilir.
- Diğer bir ifadeyle işlemcik, bir işlemin adres boşluğu içinde yer alan yeni bir işlemdir. Bir işlem birden fazla işlemciğe sahip olabilir.
- Bir işlemin adres boşluğu içinde işlemlerini gerçekleştiren birbirinde bağımsız işlemlere "işlemcik" adı verilir.

Farklı tanımlamalar olmasına rağmen ortak noktalardan ortaya çıkan sonuç : İşlemcik,

- Belli bir işleme bağlıdır.
- Bağlı bulundukları işlemin adres boşluğunu kullanırlar.
- Kendi program sayacına sahiptirler.
- Gerçekleştirilen prosedürler için kendi adres ve yerel değişkenlere sahiptirler.
- İşletim sistemi kaynaklarını paylaşırlar.

İşlemcikler, işlemlere göre iki temel avantaja sahiptir.

- Aynı işlem içindeki işlemcikler arasında gerçekleşen konteks anahtarlama zamanı, işlemler arasında gerçekleşen konteks anahtarlama zamanından oldukça küçüktür. Aynı zamanda, işlemler ana bellek üzerine yazılıp-okunurken, işlemcikler kaydediciler (register) üzerine yazılıp-okunurlar.
- İşlemler arasında haberleşme mesaj iletimi ile sağlanırken, işlemcikler aynı adres boşluğunu kullandıkları için aralarında mesajlaşmaya gerek yoktur. Eğer arzu edilirse işlemcikler kendi aralarında mesajlaşa bildikleri gibi dışarıdan diğer bir işlem ile de mesajlaşabilirler.

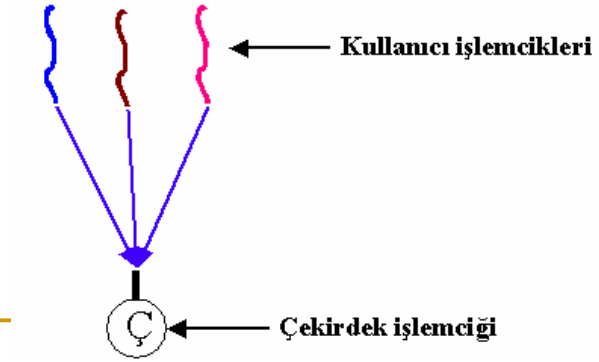
İşlemcikler kullanıcı işlemcikleri ve çekirdek işlemcikleri olmak üzere ikiye ayrılır. Kullanıcı işlemcikleri çekirdek katmanının üstünde olup kullanıcı işlemleri tarafından yaratılırlar. Bu işlemciklerin yaratılması, çizelgelemesi ve yönetilmesi çekirdekten bağımsız olarak gerçekleştirilir. Kullanıcı işlemciklerinin tamamı kullanıcı adres boşluğu içerisinde koşturulur.
Çekirdek işlemcikleri ise doğrudan doğruya işletim sistemi tarafında yaratılır. Çekirdek işlemciklerinin yaratılması, çizelgelemesi ve yönetimi işletim sistemi tarafında gerçekleştirilir.

5.3.1 Çoklu İşlemcik Modelleri

Modern işletim sistemlerinden çoğu hem kullanıcı hem de çekirdek işlemciklerini destekler. Bu nedenle modern işletim sistemlerinin kullandıkları çeşitli çok işlemcikli modeller vardır.

5.3.1.1 Çoktan-Bire Modeli

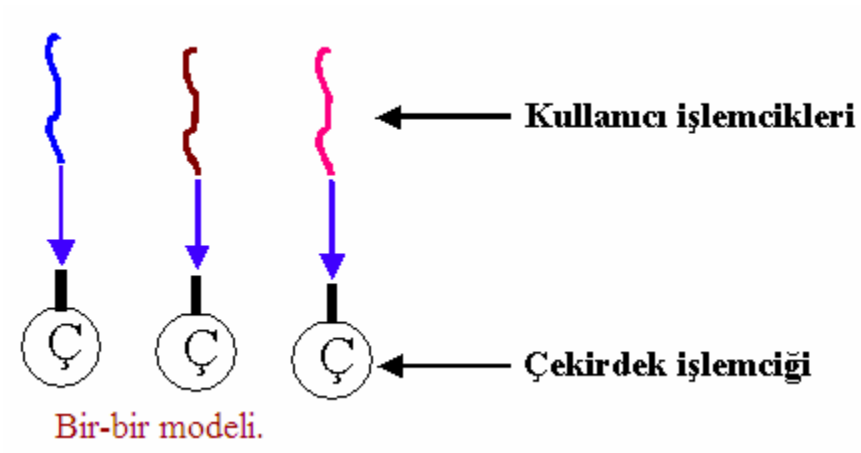
Şekil 5.8'de bu modelin şekli görülmektedir. Bu modelde birden fazla kullanıcı işlemciği ve tek bir çekirdek işlemciği bulunur. Bu modelde, aynı anda çekirdek katmanında sadece bir işlemcik bulunabildiği için işlemci üzerinde sadece bir işlemcik koşturulabilir.



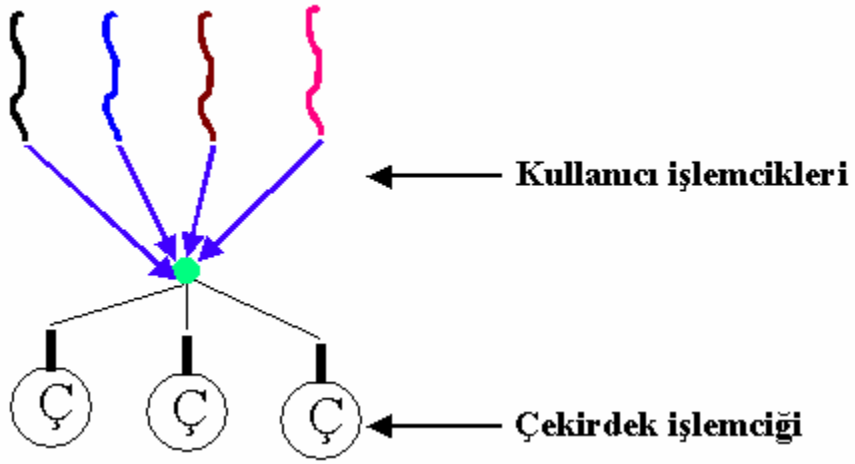
Çoktan-bire modeli.

5.3.1.2 Bire Bir Modeli

Şekil 5.9'da bu modelin şekli görülmektedir. Bu modelde her kullanıcı işlemciği için çekirdek katmanında bir çekirdek işlemciği koşturulur. Bu nedenle işlemci üzerinde birden fazla işlemcik aynı anda koşturulabilir.



Bir-bir modeli.



Çoktan-çoğa modeli.

5.3.1.3 Çoktan Çoğa Modeli

Şekil 5.10'da bu modelin şekli görülmektedir. Bu modelde bir çok kullanıcı işlemciğine karşıklık ya aynı sayıda ya da daha az sayıda çekirdek işlemciği bulunur. Bu modelde de işlemci üzerinde birden fazla işlemcik aynı anda koşturulabilir.

5.4 MİB Çizelgelemeleri

Çizelgelemeler işletim sisteminin temel işlevlerinden biridir. Hemen hemen her bilgisayar aygıtı bu kavramla ilişkilidir. MİB 'de, en önemli bilgisayar aygıtıdır ve belli çizelgelemelere sahiptir. Bu çizelgelemelerin amacı, MİB'den faydalanmayı, MİB'nin çalışmasını en üst düzeye çıkartarak; cevaplama zamanı ile bekleme süresini en alt düzeye indirmektir. Diğer bir yaklaşımla MİB'ni mümkün olduğu kadar meşgul tutmak ve MİB'nin boş beklemesini engellemektir. MİB için geliştirilen çizelgeleme algoritmalarından bazıları şunlardır :

1- İlk Gelen İlk Hizmet Alır (FCFS) Çizelgeleme Algoritması
2- En Kısa İşlem İlk (SJF) Çizelgeleme Algoritması
3- Öncelik (Priority) Çizelgeleme Algoritması
4- Round Robin Çizelgeleme Algoritması

5.4.1 İlk Gelen İlk Hizmet Çizelgeleme Algoritması

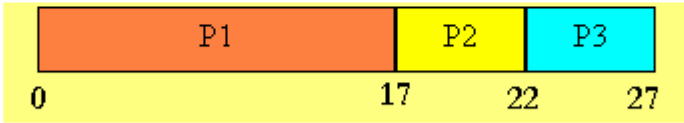
En basit MİB çizelgeleme algoritmasıdır. İşlemler hazır kuyruğundaki geliş sıralarına göre MİB' ne gönderilirler. MİB'de koşturulan işlem tamamlandığında, hazır kuyruğunun başındaki işlem MİB'ne çalıştırılmak için alınır.

Tablo 5.1 Gelen İşlemler ve süreleri

İşlem	İşleme Süresi
P1	17
P2	5
P3	5

Örnek 5.1:

İşlemlerin Tablo 5.1'deki gibi hazır kuyruğunda beklediği kabul edilirse; işlemlerin MİB'ne gidiş sırası İlk Gelen İlk Hizmet çizelgeleme algoritmasına göre P1,P2 ve P3 şeklinde olur. Bu durum Şekil 5.7 'deki gantt şemasında gösterilmiştir.



Şekil 5.11 - İşlemlerin işlenişinin gant şeması ile gösterilimi.

Bu örnekte;
P1 işlemi için bekleme süresi 0 msn'dir.
P2 işlemi için bekleme süresi 17 msn'dir.
P3 işlemi için bekleme süresi 22 msn'dir.
Böylece ortalama bekleme süresi (0+17+22)/3=13 msn'dir.

Aynı işlemler P3,P2,P1 sırasıyla MİB'ne gelseydi;

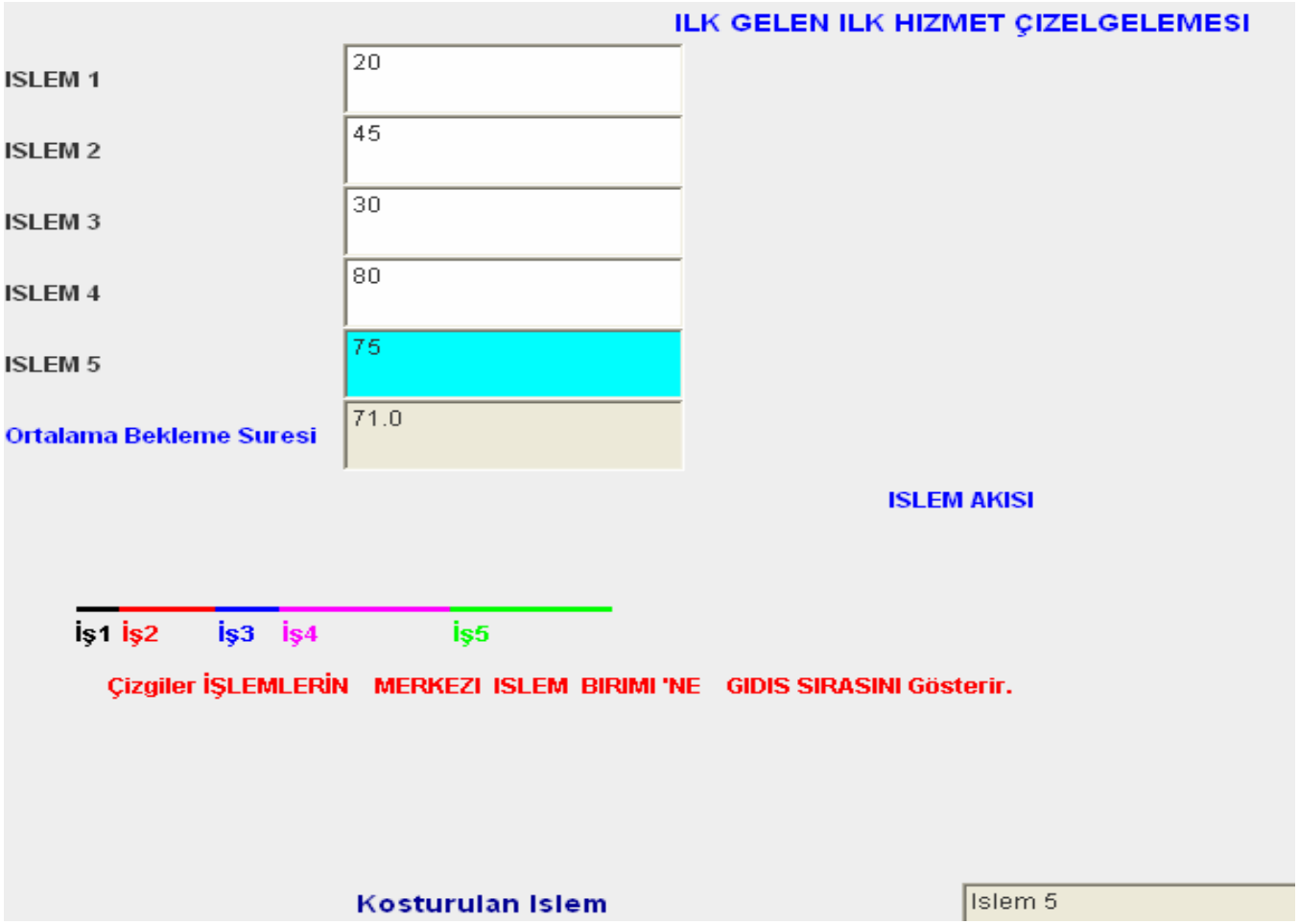
Şekil 5.12 - Geliş sırası değiştirilmiş işlemlerin, işlenişinin gant şeması ile gösterilimi.

Şekil 5.8'de görüldüğü gibi,
P1 işlemi için bekleme süresi 10 msn,
P2 işlemi için bekleme süresi 5 msn,
P3 işlemi için bekleme süresi 0 msn olacaktı.
Ortalama bekleme süresi ise (10+5+0)/3=5 msn.
Örnekten de anlaşılacağı üzere bu çizelgeleme algoritmasında kısa işlemler çalıştırılmak için uzun süre beklemek zorunda kalır.

Boşaltmasız (Nonpreemptive) : MİB'ne işletilmek üzere bir işlem verildiğinde, başka bir işlem MİB'ni kullanmak için ele geçiremez. Diğer bir ifadeyle, MİB'de koşturulan işlemin işlem süresi bittikten sonra, diğer işlem MİB'de koşturulmaya başlanır. Buna " Boşaltmasız (Nonpreemptive) " denir.

Boşaltmalı (Preemptive) : Örneğin, MİB bir işlemi işlerken eğer MİB'ne işleme süresi, o anki işlemin işleme süresinden daha az bir işlem gelirse koşulmakta olan işlem MİB'ni terk eder. MİB bu yeni gelen işlemi işlemeye başlar. Buna " Boşaltmalı (Preemptive) " denir.

İlk gelen ilk hizmet alır çizelgeleme algoritması boşaltmasız (nonpreemptive) bir algoritmadır. Bir işlem, MİB'ne çalıştırılmak üzere alındığında başka bir işlem MİB'ni ele geçiremez. Bu nedenle bu algoritma zaman paylaşımlı sistemlerde kullanılamaz.
Simülatör 1'de bu işlemi gösteren bir simülasyon olayı gerçekleştirilmiştir. Verilen değerler işlemlerin mikroişlemcideki işleme süresini göstermektedir. Çizilenler ise işlemlerin MİB'ne gönderilişini temsil eder. Simülatörün alt kesiminde bulunan "İşletilen Işlem" kutusunda ise o anda işlemekte olan işlemin hangisi olduğunu göstermektedir.



İlk gelen ilk hizmet çizelgeleme algoritması simülatörü.

5.4.2 En Kısa İşlem İlk (SJF) Çizelgeleme Algoritması

Boşaltmasız (nonpreemptive)

En kısa işlem ilk çizelgeleme (boşaltmasız) algoritmasına göre işleme süresi küçük olan işlem MİB'de ilk olarak işlenir. MİB'i hazır kuyruğundan bir sonraki çalıştırılacak işlemi seçerken, işlemler içinden işleme süresi en küçük olan işlemi seçer.

Tablo 5.2 Gelen İşlemler ve süreleri

İşlem	İşleme Süresi
P1	5
P2	9
P3	8
P4	10

Örnek 5.2:

İşlemlerin Tablo 5.2'deki sırada ve aynı zaman diliminde hazır kuyruğunda bekledikleri kabul edilirse En Kısa İşlem Ilk Çizelgeleme algoritmasına göre işlemlerin MİB'ne gidiş sıraları Şekil ***5.2a***'daki gantt şemasında gösterilmiştir.

5.2a

En kısa işlem ilk (boşaltmasız) çizelgeleme algoritması

5.4.3 En Kısa İşlem İlk (SJF) Çizelgeleme Algoritması

Boşaltmalı (preemptive)

Boşaltmalı(preemptive) En Kısa İşlem İlk (SJF) çizelgelemesinde işlemlerin hazır kuyruğuna varış zamanları önemlidir. MİB'de bir işlem koşturulurken, hazır kuyruğuna yeni bir işlem ulaşırsa,şu işlemler gerçekleşir :

- Ulaşan işlem o an MİB'de koşturulan işlemin kalan işleme süresinden daha kısa bir işleme zamanına sahipse, koşulan işlem durdurulup ana bellek üzerindeki hazır kuyruğuna alınır. Yeni gelen işlem MİB'de koşturulmaya başlanır.
- Eğer ulaşan işlem o an MİB'de koşturulan işlemin, kalan işleme süresinden, daha kısa bir işleme zamanına sahip değil, fakat hazır kuyruğunda bekleyen işlemlerden daha az işleme süresine sahip ise ulaşan bu islem MİB bos kaldığında, MİB'ne gönderilecek ilk islem olarak hazır kuyruğunda tutulur.

Örnek 5.3:

Tablo 5.3 Gelen İşlemler ve süreleri

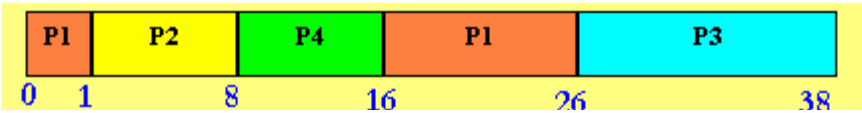
İşlem	Varış Zamanı	İşleme Süresi
P1	0	11
P2	1	7
P3	2	12
P4	3	8

P1 işleminin varış zamanı 0. Yani hazır kuyruşunda hiç bir işlem yok. Bu nedenle P1 işlemi koşturulmak için hemen MİB alınır. Koşturulma işleminin 1'inci msn'de hazır kuyruğuna P2 işlemi geliyor. Bu işlemin işleme süresi 7 olup, P1 işleminin o anki kalan işleme süresinden daha küçüktür. Bu nedenle P1 işlemi ana bellek üzerindeki hazır kuyruğuna gönderilip, MİB'ne koşturulmak üzere P2 işlemi alınır.

Koşturulma işleminin 2 msn'de P3 işlemi hazır kuyruğuna geliyor. Bu işlemin işleme zamanı P2 ve P1 işlemlerinin işleme zamanından büyük olduğu için hazır kuyruğunda bekletiliyor.

Koşturulma işleminin 3 msn'de P4 işlemi hazır kuyruğuna geliyor. Bu işlemin işleme zamanı o an MİB'de koşturulmakta olan P2 işleminin kalan işleme süresinden büyük olduğu için MİB P2 işleminin koşturulmasına devam ediyor.

P2 işleminin bitiminden sonra hazır kuyruğunda bulunan işlemlerden en az işleme süresi olan P4 işlemi koşturulmak üzere işlemciye alınıyor. Daha sonra sırayla P1 ve P3 işlemlerinin koşturulumu yapılıyor.



İşlem	Varış Zamanı	İşleme Süresi
P1	0	11
P2	1	7
P3	2	12
P4	3	8

5.4.4 Öncelik (SJF) Çizelgeleme Algoritması

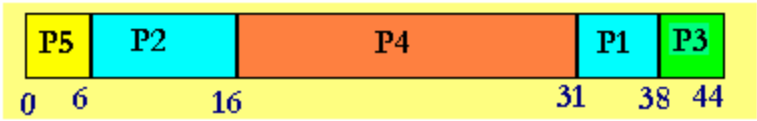
Öncelik çizelgeleme algoritmasında, MİB'ne gidecek olan her işlem bir öncelik değerine sahiptir. MİB işleyeceği yeni işlemi hazır kuyruğundan bu öncelik değerine bakarak seçer. Önceliği yüksek olan işlem MİB'de ilk olarak işlenir.

Örnek 5.4:

Tablo 5.4 Gelen İşlemler ve süreleri

İşlem	İşlem Süresi	Öncelik
P1	7	2
P2	10	1
P3	6	3
P4	15	2
P5	6	0

Tablo 5.4'deki işlemlerin hazır kuyruğunda beklediklerini kabul edersek Öncelik Tarife algoritmasına göre işlemlerin MİB'ne gidiş sıraları Şekil 5.11'daki gantt şemasında gösterilmiştir.



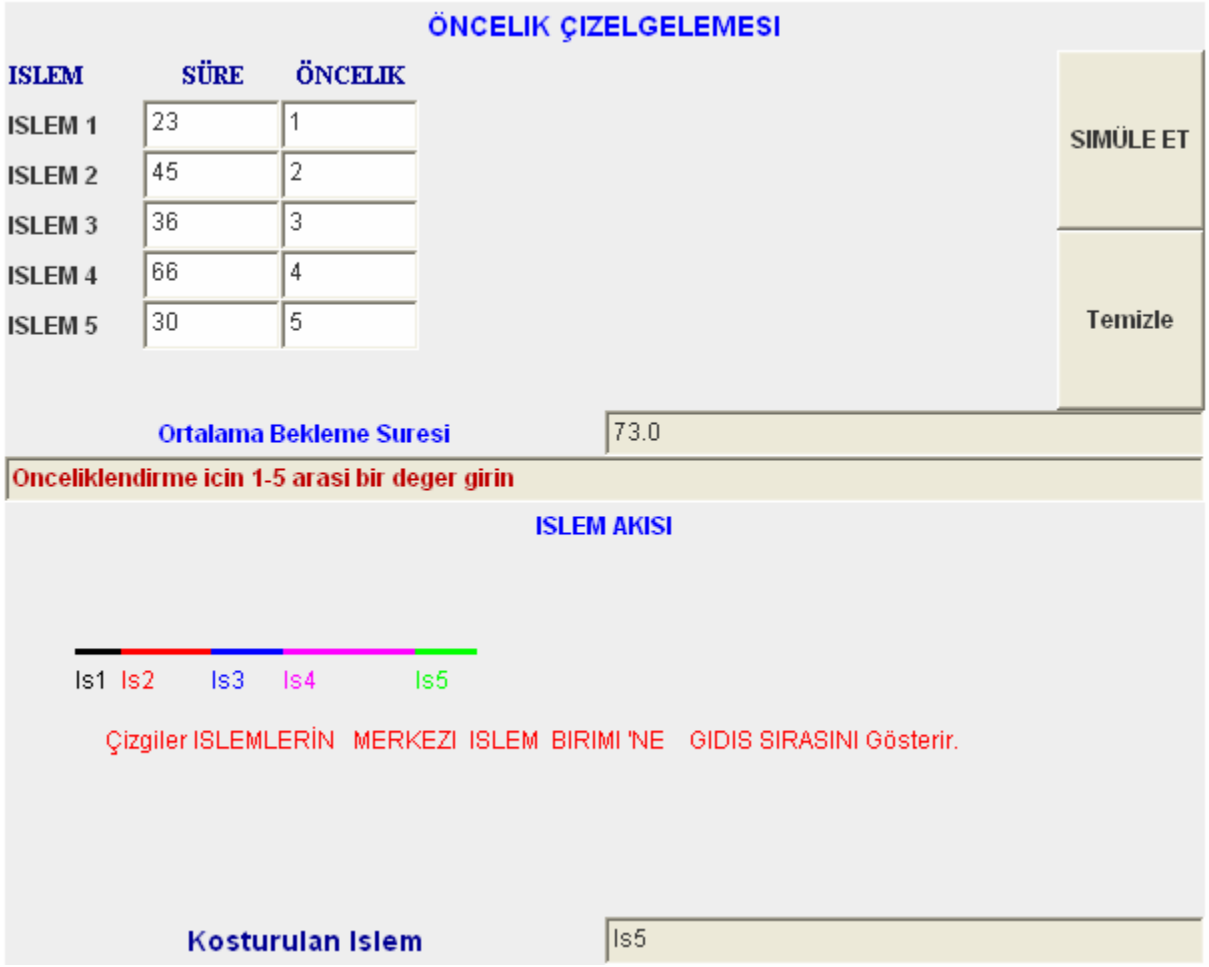
İşlemlerin işlenişinin gant şeması ile gösterilimi.

Bu örneğimizde ;
P1 işlemi için bekleme süresi 31 msn,
P2 işlemi için bekleme süresi 6 msn,
P3 işlemi için bekleme süresi 38 msn,
P4 işlemi için bekleme süresi 16 msn,
P5 işlemi için bekleme süresi 0 msn'dir.
Böylece ortalama bekleme süresi (31+6+38+16+0)/5=18.2 msn'dir.

Bu algoritmanın "sonsuz bloklama" ya da "açlıktan ölme" denilen bir sorunu vardır. Bu sorun şu şekilde açıklanabilir :
Öncelik değerleri 0 ile 127 arasında değişen sayısal değerlerdir. Öncelik değeri çok yüksek değerli bir işlem işletilmek üzere hazır kuyruğunda bekliyor. Fakat öncelik değeri bundan çok düşük değerli olan işlemler hazır kuyruğuna sürekli olarak geliyor. Bu nedenle de bu işlem hiç bir zaman koşturulamıyor. Bu işlem herhangi bir metod uygulanmazsa koşturulmak için sonsuza kadar bekleyebilir.

Bu sorun "yaşlanma" tekniğiyle çözümlenir. Bu tekniğe göre hazır kuyruğunda işletilmek üzere bekleyen işlemler, eğer 15 dakika içinde işlenmezlerse öncelik değerleri 1 derece azaltılır. Böylece çok büyük öncelik değerine sahip bir işleme hazır kuyruğuna gelse dahi maksimum 32 saat içinde işlenerek sistemi terk edebilecektir.

Simülatör 4'de Öncelik çizelgeleme algoritması simülasyonu gösterilmiştir. Verilen değerler işlemlerin işleme süreleri ve öncelik değerleridir. 0 en yüksek öncelik; 5'de en düşük öncelik değeridir. Diğer bir ifadeyle 0 öncelikli bir işlem, işlenilmesi gereken en acil işlemdir. Öncelik (Priority) çizelgeleme algoritması hem boşaltmalı (preemptive) hem de boşaltmasız (nonpreemptive) olabilir.


ÖNCELIK ÇIZELGELEMESI
ISLEM
SÜRE
ÖNCELIK
ISLEM 1
23
1
ISLEM 2
45
2
ISLEM 3
36
3
ISLEM 4
66
4
ISLEM 5
30
5
SIMÜLE ET
Temizle
Ortalama Bekleme Suresi
73.0
Onceliklendirme icin 1-5 arasi bir deger girin
ISLEM AKISI
Is1 Is2 Is3 Is4 Is5
Çizgiler ISLEMLERİN MERKEZI ISLEM BIRIMI 'NE GIDIS SIRASINI Gösterir.
Kosturulan Islem
Is5

5.4.5 Round Robin Çizelgeleme Algoritması

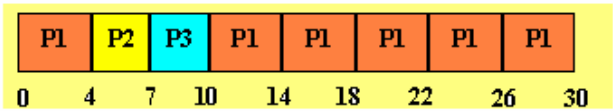
Round Robin (kısaca RR) çizelgeleme algoritması özellikle zaman paylaşımlı sistemler için tasarlanmıştır. MİB'de koşturulacak olan her işlem için *kuantum süresi* denilen belirli bir işlem süresi ayrılır. Bu süre genellikle 10 ile 100 msn arasındadır. MİB'ne gelen her işlem ayrılan süre kadar koşturulur. Koşturulma bu zaman içerisinde biterse işlem MİB'ni terk eder. Koşturulum daha tamamlanmamışsa hazır kuyruğunun sonuna geri gönderilir.

Örnek 5.5:

Tablo 5.5 Gelen İşlemler ve süreleri

İşlem	İşlem Süresi
P1	24
P2	3
P3	3

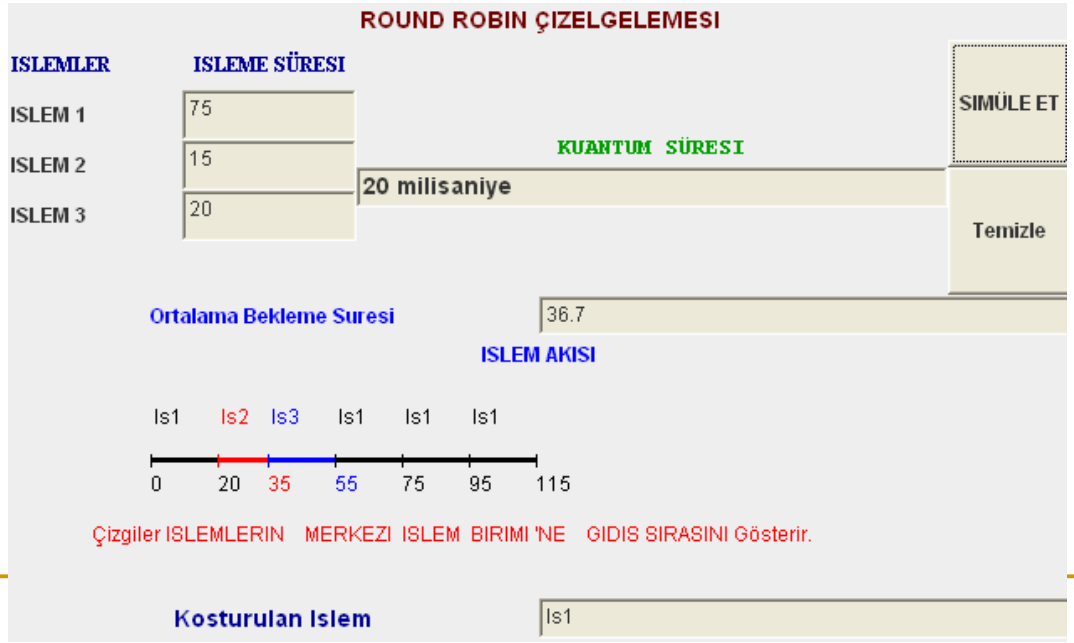
Tablodaki işlemlerin hazır kuyruğunda beklediklerini kabul edersek RR Çizelgeleme algoritmasına göre işlemlerin MİB'ne gidiş sıralarını gösteren gantt şeması Şekil 5.12'deki gibi olur. Kuantum zamanı, 4 msn olarak alınmıştır.



Şekil 5.16 - İşlemlerin işlenişinin gant şeması ile gösterilimi.

Bu örnekte;
P1 işlemi için bekleme süresi (10-4=6) msn,
P2 işlemi için bekleme süresi 4 msn,
P3 işlemi için bekleme süresi 7 msn,
Böylece ortalama bekleme süresi (6+4+7)/3=5,66 msn'dir.

RR algoritması boşaltmalı (preemptive)'dır. Bu algoritmanın performansı kuantum zamanının büyüklüğüne bağlıdır. Bu zaman çok büyük olursa, RR algoritması ile İlk Gelen İlk Hizmet Çizelgeleme algoritması aynı performansı gösterirler. Bu zaman çok küçük olursa, bu seferde konteks anahtarlama sayısı artacağı için performans düşer.
Simülatör 5.5'de Round Robin Çizelgeleme Algoritması'nın simülasyonu gösterilmiştir.



6 - ANA BELLEK YÖNETİMİ

6.1 - Giriş

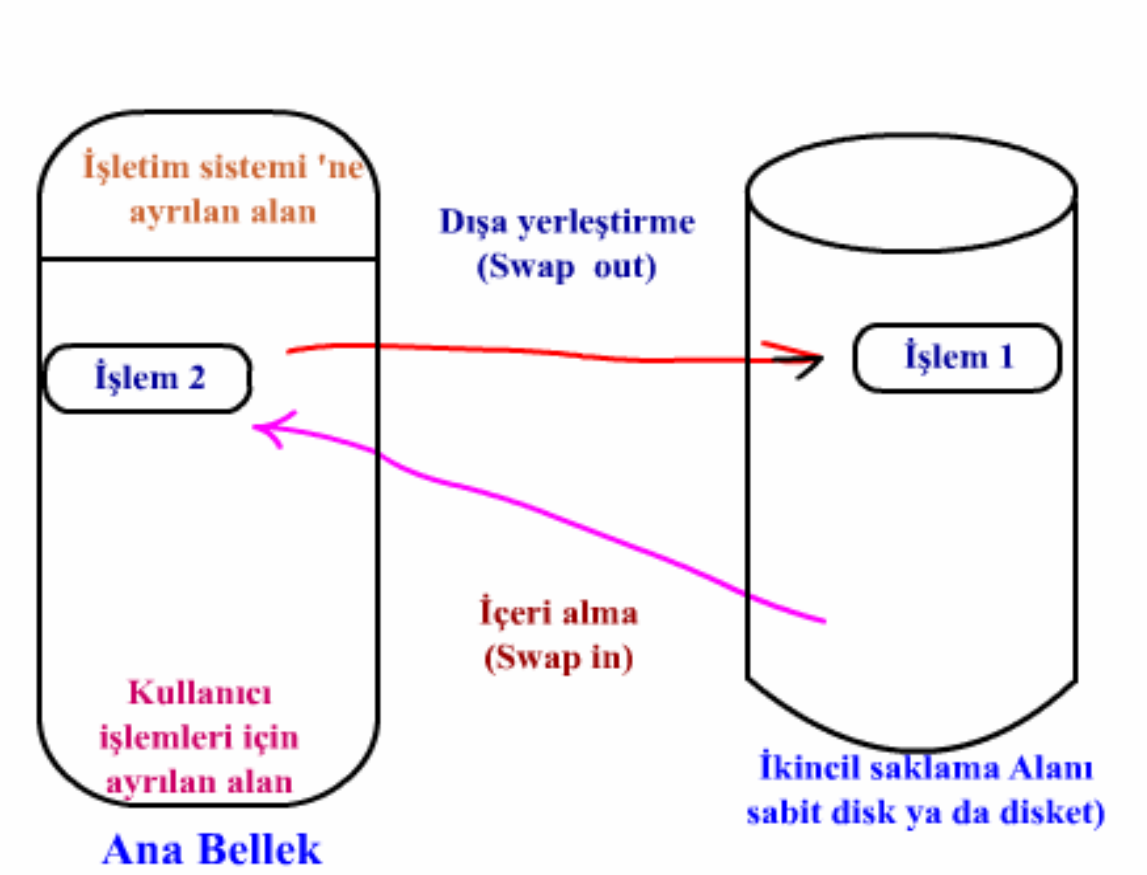
Modern işletim sistemlerinde, gerçekleştirilen işlemlerin geçici olarak saklandığı merkez ana bellektir. Ana bellek, byte ve kelimelerden oluşmuş çok geniş bir dizidir. Diğer bir ifade ile ana bellek, G/Ç aygıtlarının kolaylıkla ulaşabildiği bir bilgi deposudur. MİB, işlemleri ana bellekten alır ya da ana belleğe koyar. İşletim sistemi, ana bellek yönetimi ile ilgili aşağıdaki işlemlerden sorumludur :

1- Hangi programın kimin tarafından kullanıldığının izlenmesi,
2- MİB boşalınca, MİB'ne hangi işlemin gideceği,
3- Gerekli boş bellek alanının oluşturulması ya da kullanılması.

6.1.1 - Yerdeğiştirme

Bir işlemin MİB'nde koşturulabilmesi için mutlaka ana bellekte olması gerekir. Bazen, koşturulan işlem geçici olarak başka bir yerde (ikincil bellek alanında) saklanabilir. Koşturulan işlemin ana bellekten alınıp ikincil bellek alanına getirilmesi ve ikincil bellek alanındaki işlenmek istenen işlemin ana belleğe alınması işlemine *"Yerdeğiştirme"* denir.
Şekil 6.1'de görüldüğü gibi, MİB herhangi bir işlemi koştururken, "İşlem 2" nin koşturulması gerektiği düşünülür ve "İşlem 2" ninde ana bellekte olmadığı; aynı zamanda da ana bellekte boş bellek alanının kalmadığı kabul edilirse; sistem "İşlem 2" yi ana belleğe getirebilmek için sırayla şu adımları gerçekleştirecektir:

- Ana bellekten o an için kullanılmayan bir işlem (şekle göre İşlem 1) seçilerek ikincil bellek alanına yerleştirilir, ana bellekte boş bir bellek alanı oluşturulur. Bu işleme "Dışarı Yerleştirme (swap out)" denir.
- Daha sonra, işletilecek olan (şekle göre İşlem 2) işlem, ikincil bellek alanından alınarak, boş ana bellek alanına yerleştirilir. Bu işleme de "İçeri Alma (swap in)" denir.



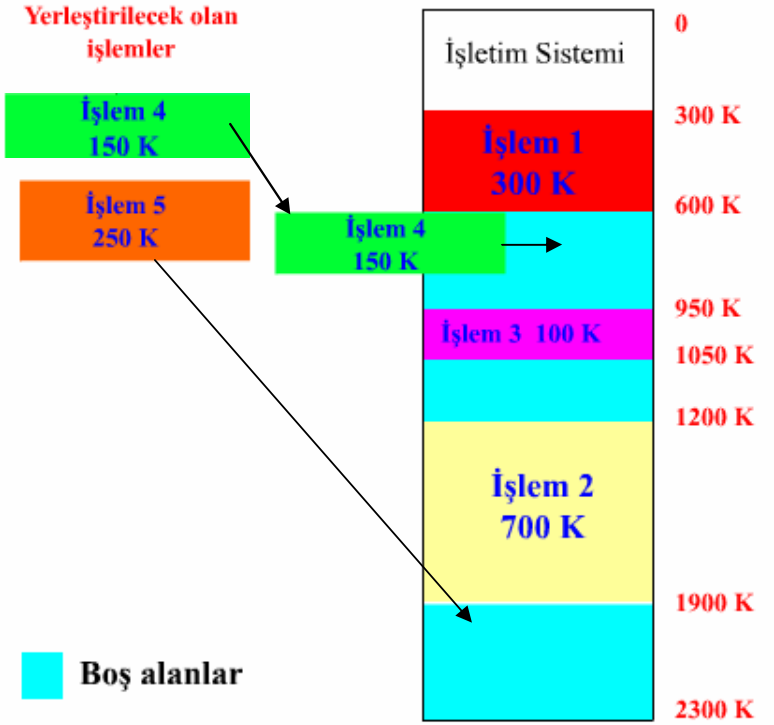
Animasyon

Şekil 6.1 Yerdeğiştirme

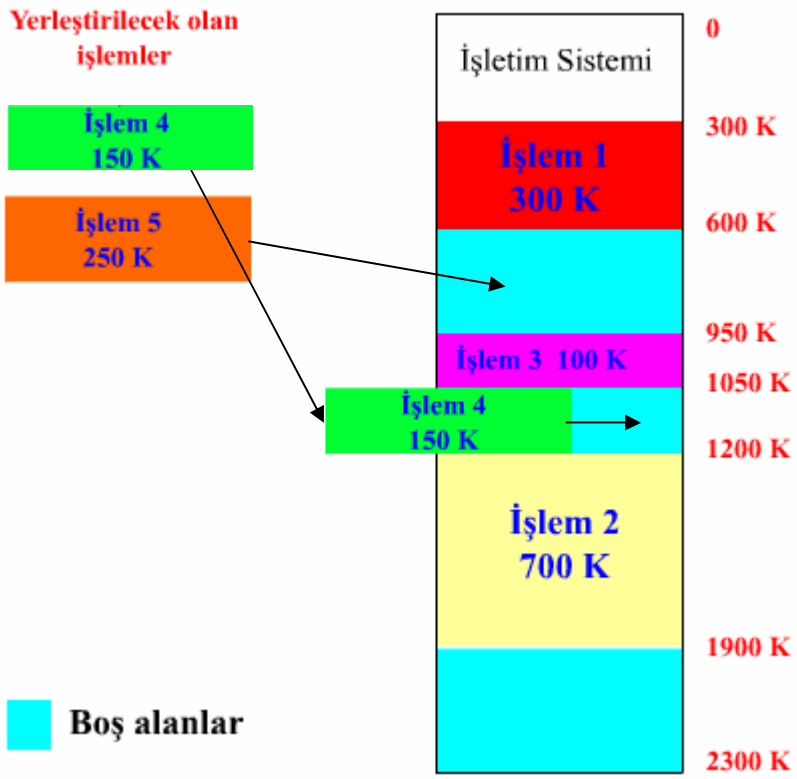
6.2 Ana Bellek Yerleştirme

İşlemlerin çalıştırılmak üzere disk üzerinden alınıp ana belleğe getirilmesine ' UZUN SÜRELİ ÇİZELGELEME ' denir. Bu çizelgeleme ile ana belleğe getirilen işlem üç tür teknik kullanılarak ana belleğe yerleştirilir :

1 - İLK BOŞLUK : İşlem, ana bellek üzerinde yerleşebileceği kadar büyüklükteki ilk boş bellek bölgesine yerleştirilir (First-fit).



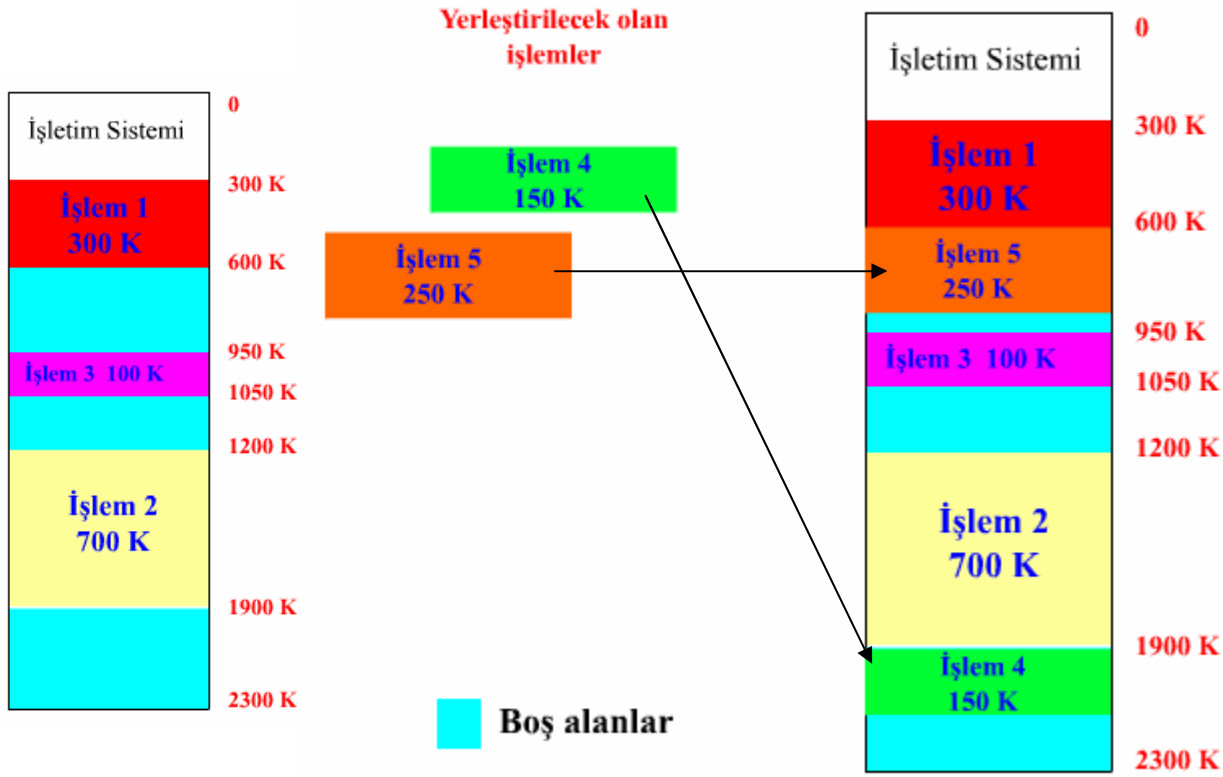
2 - EN UYGUN BOŞLUK : Ana bellek üzerinde işlemin yerleşebileceği en küçük boş bellek alanı bulunarak işlem o bölgeye yerleştirilir (Best-fit). Yerleştirme teknikleri içinde en iyisi bu yöntemdir.



Animasyon

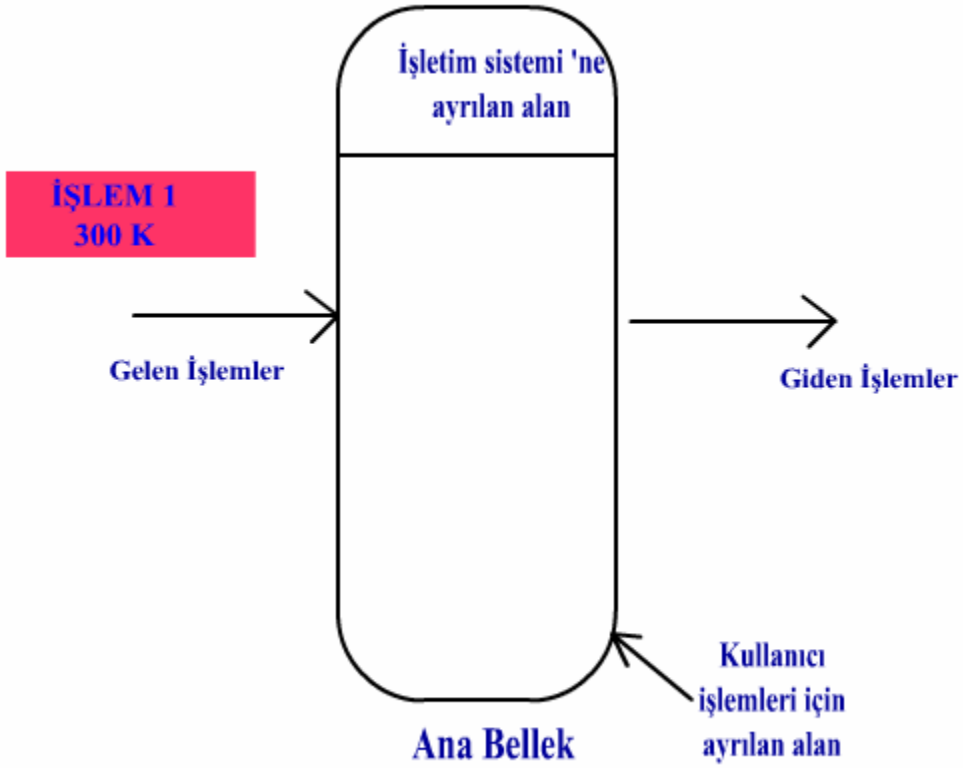
Şekil 6.3 En uygun boşluk yerleştirme.

3 - EN KÖTÜ BOŞLUK : Ana bellek üzerinde işlemin yerleşebileceği en büyük boş bellek alanı bulunarak işlem o bölgeye yerleştirilir (Worst-fit).



Şekil 6.4 En kötü boşluk yerleştirme.

Animasyon



Şekil 6.5 Ana bellek yerleştirme

Animasyon