İŞLETİM SİSTEMLERİ
DERS NOTLARI

BÖLÜM 1 – GİRİŞ

Yard. Doç. Dr. Deniz KILINÇ

CELAL BAYAR ÜNİVERSİTESİ, YAZILIM MÜHENDİSLİĞİ

2015-2016

1. DERS İÇERİĞİ VE KAYNAKLAR

İşletim sistemi (Operating System - OS) dersinde görülmesi muhtemel konu başlıkları bölüm 1.1'de verilmiştir.

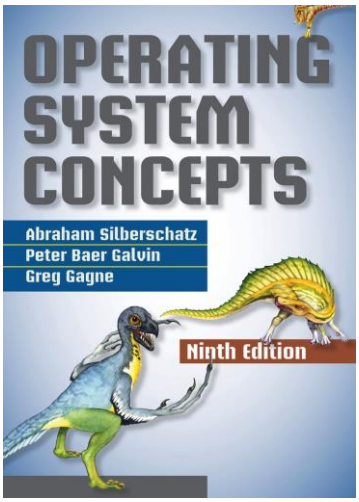
1.1 İşletim Sistemleri Dersi Muhtemel Konu Başlıkları

- İşletim Sistemlerine Giriş
- İşletim Sistemi Servisleri ve Tasarımı
- Proses (Process) Yönetimi
    - Proses States
    - Proses Scheduling
    - IPC (Inter-process communication)
    - Thread (İş parçacığı)
    - Proseslerin Senkronizasyonu
    - CPU Scheduling
    - Deadlocks
- Bellek Yönetimi
- Dosya ve Disk Yönetimi
- I/O ve Cihaz Yönetimi
- Güvenlik
- Sanallaştırma

1.2 Ders Kaynakları

Ders kitabı olarak [1] kullanılacak olup farklı kaynak kitap olarak aşağıdaki dokümanlardan faydalanılacaktır:

- Operating System Concepts, Ninth Edition, Abraham Silberschatz, Peter Bear Galvin, Greg Gagne [1]



- İşletim Sistemleri, Ali Saatçi
- Şirin Karadeniz, Ders Notları
- İbrahim Türkoğlu, Ders Notları

2. GİRİŞ

İşletim Sisteminin Kıymet Bilmez Yazılım Mühendisine İsyanı

Hava ılık…
Can sıkıntısı diz boyu…
Şu PC'yi açalım bari…
Bastık düğmeye, bekliyozzz…
Sonuçta, klavyesini de biz aldık faresini de, işlemcisi de bol çekirdekli…
i7 miş hem de, o ne demekse artık…
Açılırken bir ses geliyor bilgisayarın içinden, o neee?
Bir de ışık yanıp sönüyor...
Ne alaka şimdi ???
Yavaşladı mı bu bilgisayar yine?
Fareye iki tıklayalım. Gözü kapalı tarayıcımızın ikonuna basalım,
Nasıl olsa açar onun adı bilgisayar …
Hopppp bir ekran açılıyor yavaştan, default tabiki Feysbuk var,
Gir baba Feyse gir…
Ooooo şu fotoya bak…
O saniye fotoyu milyon kişi de görmüş, deli işi yapmışlar ama bana ne…
Uzat abi fareyi uzat nasıl olsa o ok gidiyor uzattığın yere…
Çak bir like…Ohhh misss….
Yine mutluyum…
Ben de olmasam bu bilgisayar bir işe yaramayacak, iyi ki varım ☺

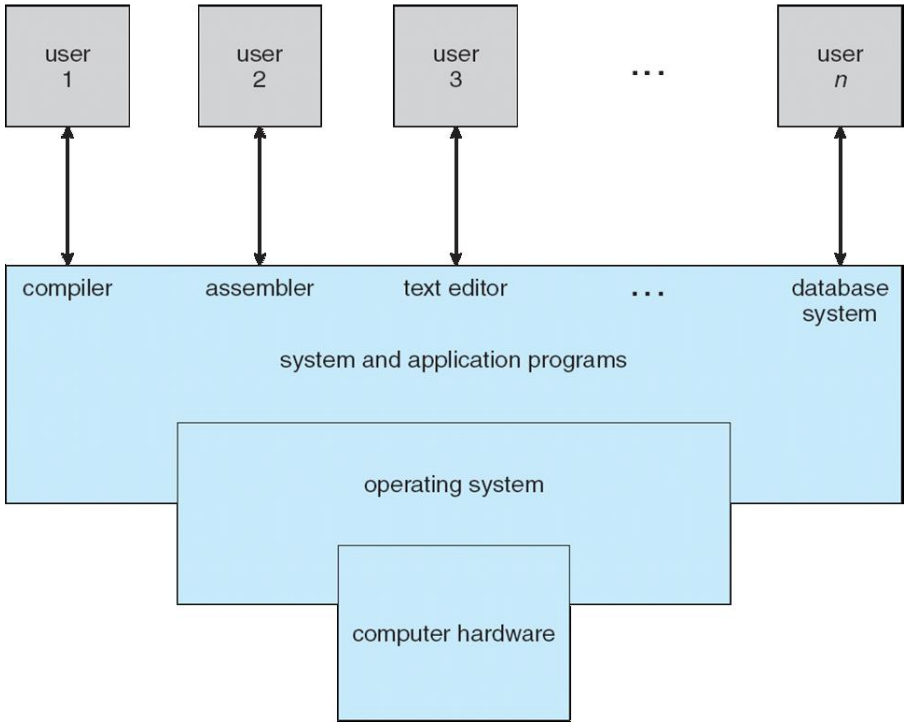
3. İŞLETİM SİSTEMLERİNE GİRİŞ

İşletim sistemi (OS – Operating System), bilgisayar donanımı (hardware) ile kullanıcılar arasında (users) arabulucu rolünde görev yapar. OS aslında bilgisayar donanımını ve kaynakları yönetmeyi sağlayan bir yazılımdır. OS'lerin temel amacı bir kullanıcının herhangi bir programı etkin (**efficient**) ve uygun (**convenient**) bir şekilde çalıştırması için gerekli ortamı sağlamaktır. Etkinlik ve uygunluğun **öncelik derecesi** (*OS için hangisinin ne derece önemli olduğu*) OS türünden türüne farklılık gösterebilir. Örneğin;

- **Mainframe** (veya sunucu yönetim amaçlı işletim sistemleri) OS'ların birinci amacı donanım kaynaklarını optimum şekilde kullanmaktır.
- **Kişisel bilgisayarlar** üzerinde çalışan (PC – Personel Computer) OS'lar ise karmaşık iş uygulamalarını, oyunları vb. uygulamaları "tek kullanıcı için" sorunsuz çalıştırmayı hedefler.
    - Birden fazla kullanıcı için donanım kaynaklarının nasıl paylaştıracağı önemlidir, ancak ikincil hedeftir.
- **Mobil bilgisayarlar üzerinde çalışan OS'lar** ise bir kullanıcının uygulamalarla **direk ve kolay** etkileşimde bulunabilmesi ve pil ömrünü etkin kullanabilmesi için gerekli ortamı sağlamayı hedefler.

OS büyük ve karmaşık olduğu için OS detaylarına girmeden önce, genel sistem yapısı, işleyişi ve bilgisayar organizasyonu ile ilgili bilgilendirmeler bu bölümde yapılacaktır.

3.1. İşletim Sisteminin Bilgisayar Sistemindeki Yeri



Şekil 1. Bilgisayar sistemindeki bileşenlerin soyut görünümü

Şekil 1'de de görüldüğü üzere bir bilgisayar sistemi 4 temel bileşenden oluşmaktadır. Bu bileşenler aşağıdaki gibidir:

- **Bilgisayar donanımı (Hardware):**
  - CPU (Central Processing Unit, "işlemci"), bellek (memory), I/O (Input/Output) cihazları ve disk gibi temel hesaplama (computing) kaynaklarını içerir.
- **Uygulama programları:**
  - Sistem kaynakları dahilinde, kullanıcıların iş, eğlence ve hesaplama gibi ihtiyaçlarını karşılayacak yazılım uygulamalarıdır.
    - Ofis uygulamaları (kelime işlemciler, excel hesap tablosu, e-posta uygulamaları.), web tarayıcıları, ticari yazılımlar, oyunlar, multimedya uygulamaları (video, müzik, resim oynatıcıları) vb.
- **İşletim sistemi (OS):**
  - Donanımı kontrol ederek, donanım ve kullanıcı uygulamaları arasındaki koordinasyonu sağlar (Nasıl?).
- **Kullanıcılar:**
  - İnsanlar, akıllı cihazlar veya başka bilgisayarlar kullanıcı olabilirler.

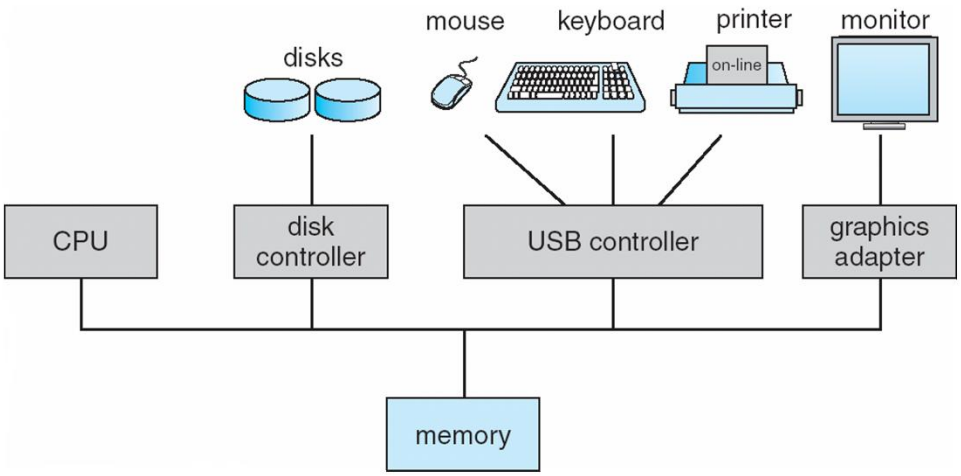
Şekil 1'den ve giriş bilgilerinden yola çıkarak aşağıdaki OS tanımları yapılabilir:

- Bir işletim sistemi, **orkestrayı yöneten** bir şeftir. Bir orkestrayı yöneten biri olmadığında, enstrümanlardan çıkan sesler birbiriyle uyumlu olmayacaktır. İşletim sistemi de bir bilgisayar sisteminin şefidir; bilgisayarın donanım elemanlarının birbiri ile haberleşmesini, birbirini tanımasını, kısacası birbiri ile uyumlu bir şekilde çalışmasını sağlar.
- İşletim sistemi aynı zamanda, kullanıcı ve donanım, yazılım ve donanım ve son olarak yazılım ve yazılım arasındaki ara bir kontrol yazılımıdır. Kullanıcı ve donanımın, donanım ve yazılımların ve birçok farklı yazılımın etkileşimini, birbirini anlamasını ve birbiri ile uyum içerisinde çalışabilmesini sağlar.

3.2. Bilgisayar Sistem Organizasyonu

Genel amaçlı modern bir bilgisayar:

- Bir veya daha fazla CPU (işlemciye) ve
- Paylaşımlı belleğe (shared memory) erişimi ve haberleşmeyi sağlayan bir **veri yoluna** (common bus) *bağlı* belirli sayıda cihaz kontrolöründen (**DC - Device Controller**) oluşur.
- Her DC farklı bir cihazdan sorumludur. Örneğin, USB bağlantılı cihazlar, disk drive, audio device, video display.
- CPU ve DC'ler **paralel çalışabilirler** ve hafızaya erişim (okuma, yazma veya okuma/yazma) için yarış içerisindedirler (**competing for memory cycle**).



Şekil 2. Modern bilgisayar sistemi

Bilgisayarın güç düğmesine basılarak **açıldığında** veya **reboot** edildiğinde aşağıdaki adımlar gerçekleşir:

- Öncelikle basit kodlardan ve komutlardan oluşan bir programın çalışması gerekmektedir. **Firmware** olarak da adlandırılan bu program bilgisayarın ROM (Read-only memory – Sadece okunabilir hafıza) veya EEPROM (electrically erasable programmable read-only memory) hafızasında bulunmaktadır. **Not:** ROM tamamen okunabilir EEPROM ise ara ara yazılabilir bellek türüdür.
- **Firmware** programı, **POST (Power-on self test)** kontrolü yapar: CPU, RAM, ve *BIOS (Basic input-ouytput System)*'un çalışmasında hata olup olmadığını kontrol eder (klavye, fare, hard-disk dahil).
- POST testi başarılı değilse, bip sesi ve hata mesajı verilir.
- POST testi başarılı olursa, ROM'daki **firmware** yazılımı bilgisayar disk sürücülerini aktive etmeye başlar.
- Disk sürücüsü aktive olur olmaz, OS'un bir parçası olan **Bootstrap loader/program** devreye girer.
- Bootstrap loader, **OS kernel'ı (çekirdek)** *diskten okuduktan sonra* belleğe yükler.
- Kernel bir kere yüklendikten ve çalıştıktan sonra, kullanıcılara ve sisteme servis vermeye başlar.

- Kernel dışında, sistem programları tarafından boot esnasında belleğe yüklenen sistem prosesleri de bulunmaktadır, kernel'ın çalışma süresince onlar da çalışmaya devam ederler.
- UNIX tabanlı işletim sistemlerinde, **yüklenen ilk proses** **"init"** dir.
- Bu adım tamamlandıktan sonra sistem **tam olarak boot edilmiş olur** ve **eventlerin (olay)** oluşmasını **beklemeye başlar**.

Fully-boot edilmiş bir işletim sisteminde eventlerin (olayların) gerçekleşmesi:

- Bir event, yazılım (SW) ya da donanım (HW) tarafından gönderilen bir **interrupt (kesme)** sinyali ile gerçekleşir.
- HW, herhangi bir zamanda **sistem yolu** üzerinden (**system bus**) CPU'ya sinyal göndererek bir interrupt tetikler.
- SW ise, **sistem call** (**system call**) isimli özel işlemleri gerçekleştirerek bir interrupt tetikler.
- **Trap** veya **exception** da SW tarafından üretilen bir işlem veya hata sonucu oluşan bir **interrupt türüdür**.
- CPU'ya bir interrupt geldiğinde, CPU ne iş yapıyorsa durdurur. Execution işlemini, interrupt'ı gerçekleştiren servis rutinin başlangıç adresinin olduğu yere transfer eder. Servis rutini üzerinde execution işlemini gerçekleştirir ve yarıda kestiği hesaplama işlemine kaldığı yerden devam eder (**resume**).
- Interruptlar bilgisayar mimarisinin önemli bir parçası olup aynı zamanda işletim sistemleri de **interrupt-driven'dır**. Interrupt, asıl kontrolü uygun interrupt servis rutinine transfer etmelidir. Bu işlemler için içerisinde Device Number, Servis Rutin Adresi gibi bilgilerin olduğu **interrupt vektörü** kullanılır.

3.3. Depolama/Saklama Hiyerarşisi

CPU işlenecek komutları sadece bellekten **okuyabilir**. Dolayısı ile çalıştırılacak her program burada depolanmalıdır. **RAM (Random Access Memory)** olarak adlandırılan ana bellek (main memory) tekrar yazılabilir bir yapıda olup, buradaki bilgiler *geçicidir*. RAM dışında daha önceden kısaca değinilen sadece okunabilen ROM ve EEPROM bellek türleri de mevcuttur. EEPROM'un ROM'dan farkı ara ara yazılabilir bir yapıda olmasıdır. Örneğin, akıllı telefonların firmwareleri EEPROM üzerinde durmaktadır.

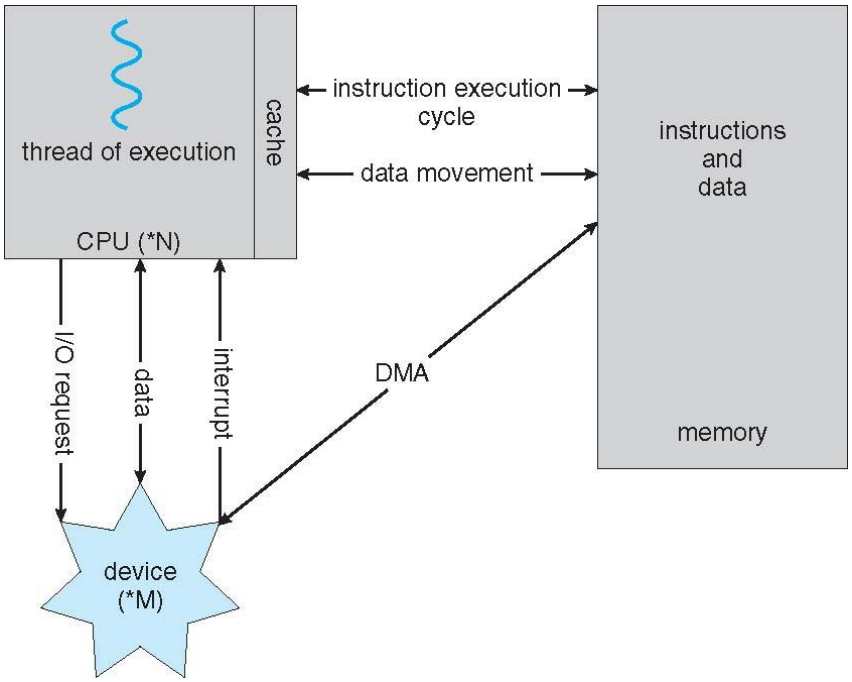
Tüm bellek formları, byte cinsiden diziler barındırır. Her byte kendi bellek adresine sahiptir. CPU-Memory (İşlemci←→Bellek) arasındaki etkileşim `load` ve `store` komutları aracılığı ile gerçekleşir.

- **Load komutu:** Bellekteki bir byte'ı veya word'u bellekten, işlemcideki dahili bir kayıtçıya (**internal register**) taşır (**move**).
- **Store komutu:** İşlemci kayıtçısındaki içeriği belleğe taşır.

Şekil 3'de Von Neumann mimarisine dayalı komut işleme çevrimi (instruction-execution cycle) gösterilmektedir. Von Neumann mimarisi tek bir veri yolu üzerinden komut ve verilerin iletişimini yapan işlemci, bellek, ve giriş/çıkış birimlerinden oluşur.

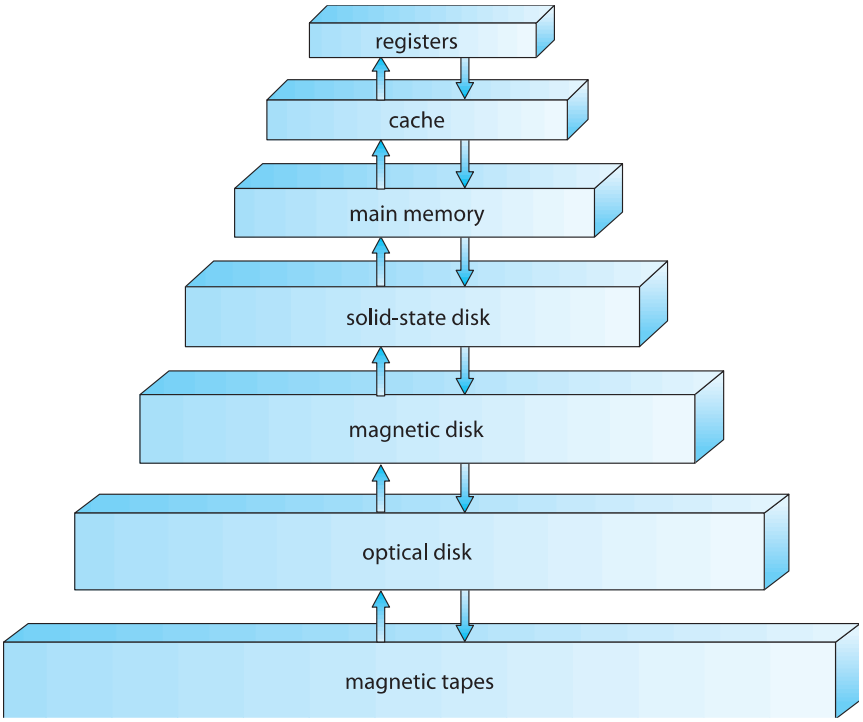
- İlk olarak bellekteki komut getirilerek (**fetch**), komut kayıtçısına (**instruction register**) saklanır.
- Daha sonra komut **decode edilerek**; bellekteki **gerekli operandlar** bellekten getirilir ve bazı dahili kayıtçılarda saklanır.

- Operandlar üzerindeki komutlar çalıştırıldıktan sonra, işlem sonucu **tekrar** *belleğe yazılır*.
- Ana bellek birimi sadece bellek adres bilgilerini (stream) görür. Onların nasıl ve ne için oluştuğunu bilmez ve ilgilenmez.



Şekil 3. Von Neumann Mimarisi

İdealde tüm programların **ana bellekte** kalıcı olarak saklanmasını ve çalışmasını isteriz. Ancak ana bellek geçici (volatile) yapıda olduğu ve kapasite olarak çok küçük olduğu için ana belleğe tüm bilgileri depolamak mümkün değildir. Dolayısı ile daha fazla ve kalıcı bilgi saklamak amacıyla ikincil bir depolama cihazına (**secondary storage**) ihtiyaç bulunmaktadır.



Şekil 4. Depolama cihaz hiyerarşisi

Genelde **magnetic disk** olan ikincil depolama cihazları, günümüzde *maliyetlerin azalması* ile birlikte **SSD (solid state disk) cihazlarına** dönüşmeye başlamıştır. Şu ana kadar gördüklerimizin dışında da farklı depolama cihazları bulunmaktadır. Birbirine benzer fonksiyonlara sahip olan bu cihazları birbirinde ayıran parametreler aşağıdaki gibidir:

- Hız
- Maliyet
- Kapasite
- Kalıcılık

Şekil 4'de hız ve maliyete göre sıralanmış depolama cihazlarının hiyerarşisi görülmektedir. Buna göre:

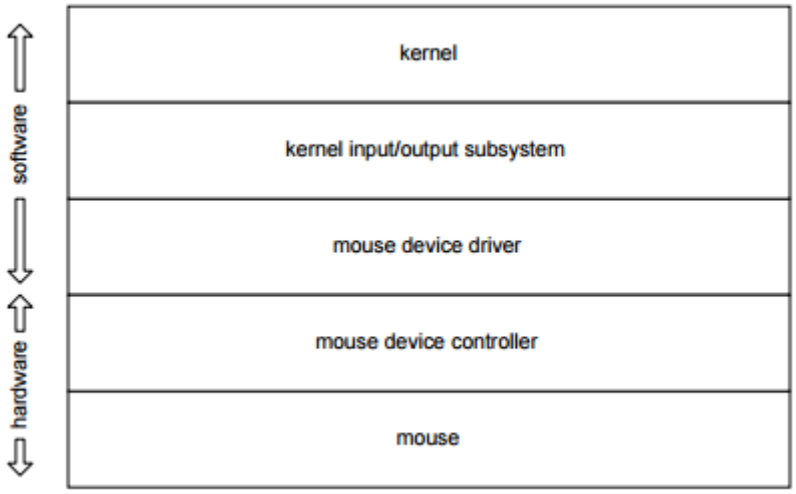
- Üst seviyedekiler pahalı ama hızlıdır.
- Hiyerarşide aşağı doğru indikçe maliyet azalır ancak cihaz erişim hızları artar (yani erişim yavaşlar).

3.4. Giriş/Çıkış (I/O) Yapısı

Depolama cihazı farklı birçok I/O cihazlarından sadece bir tanesidir. Bir bilgisayarda bunun dışında çok sayıda cihaza sahip olup, her biri ile **DC'ler (Device Controller**) ilgilenir (kontrol eder). DC türüne bağlı olarak bir DC'ye birden fazla cihaz bağlanabilir. Örneğin, 7 veya daha fazla sayıda cihaz bir **SCSI (Small Computer-System Interface)** controller'a bağlanabilir.

- Her DC'nin kendine ait lokal bir **buffer storage'ı** ve **özel amaçlı kayıtçıları** vardır.
- **DC'nin görevi** kontrol ettiği cihaz ile DC'nin buffer storage'ı arasında datayı taşımaktır.
- İşletim sistemi, her DC için bir **cihaz sürücüsüne (DD – Device Driver)** sahiptir.
- DD, DC ile anlaşarak, işletim sistemi fonksiyonlarının yerine getirilmesini genel bir interface aracılığı ile sağlar (Provides uniform interface between controller and kernel).

Aşağıdaki şekilde bir farenin, donanım ve yazılım seviyesinde yönetilmesini sağlayan bileşenleri görülmektedir.



Şekil 5. Farenin çalışmasını sağlayan bileşenler

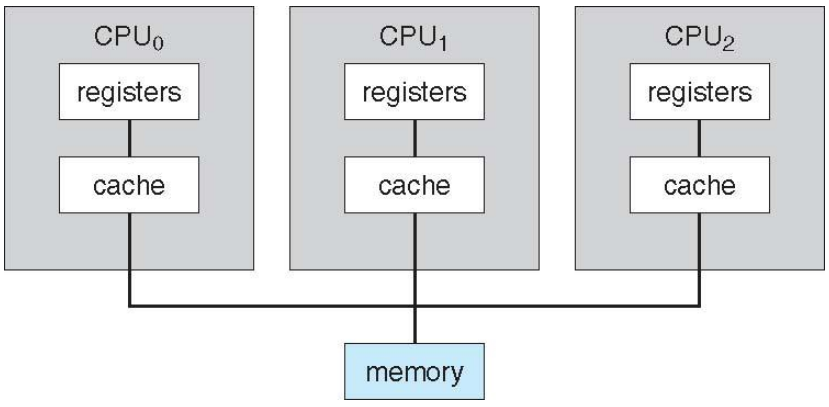
3.5. İşlemci Kullanımına Göre Bilgisayar Sistem Mimarisi

Sistem mimarileri işlemci kullanımına göre tek işlemcili (**single processor**) ve çok işlemcili (**multi processor**) olmak üzere *ikiye ayrılırlar*. Son yıllarda çok işlemcili sistemler daha yaygın kullanılır hale gelmiştir. Çok işlemcili sistemler, **paralel (parallel)** veya çok **çekirdekli (multi core)** sistemler olarak da kullanılmaktadır. İki veya daha fazla işlemciye sahip olan bu sistemlerde, bilgisayar kaynakları ortak olarak kullanılmaktadır (bellek, cihazlar, veri yolları). Çok işlemcili sistemler 3 tane önemli **avantaja** sahiptir:

- **Throughput (iş hacmi) Artışı:** İşlemci sayısını arttırdıkça, daha az zamanda daha fazla işin yapılacağı kesindir. Ancak işlemci sayısı ile doğru orantılı olarak iş sayısının artmasını beklememek gerekir. Çünkü çok işlemcinin getirdiği birlikte çalışma ve ortak kaynak kullanım yükleri olacaktır.
- **Maliyet Azalması:** Çok işlemcili sistemler, dengi olan birden fazla tek işlemcili sisteme göre çok daha az maliyetlidir çünkü her türlü donanım kaynağı paylaşılmaktadır.
- **Güvenilirlik Artışı:** Eğer fonksiyonlar birden fazla işlemciye düzgün çalışacak şekilde dağıtılırsa, bir işlemcinin hata vermesi tüm sistemin durmasına neden olmayacaktır, sadece sistemi biraz yavaşlayabilir. Güvenilirlik, günümüzdeki çoğu uygulama için oldukça önemlidir. Hata olsa da bir sistemin çalışmaya devam edebilmesi gerekmektedir. **Fault tolerant** (**hata dayanıklı**) olarak geçen bu sistemlerde bir hata oluştuğunda, bir hata tespit edilebilmeli ve mümkünse çözülebilmelidir.

İki tip çok işlemcili sistem türü kullanılmaktadır:

- **Asimetrik çok işlemcili (AMP):** Her işlemciye spesifik bir görevi atanır. Patron işlemci sistemi kontrol eder, diğer işlemciler patron işlemciden komut beklerler veya önceden tanımlanmış görevlerini gerçekleştirirler.
- **Simetrik çok işlemcili (SMP):** Tüm işlemciler işletim sistemindeki tüm görevleri yerine getirirler. Tüm işlemciler eşittir, patron-işçi işlemci gibi bir ilişki yoktur. Şekil 6'da da görüldüğü üzere her işlemcinin kendi kayıtçıları (**register**) ve ön belleği (**cache**) vardır. Günümüz yaygın kullanılan sistemler SMP'dir.

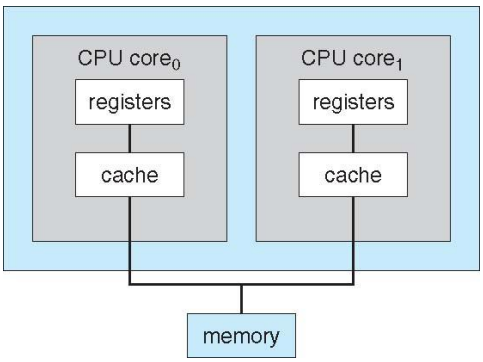


Şekil 6. SMP mimarisi

CPU tasarımında son eğilim, tek chip üzerine birden fazla işlem yapan çekirdek (core) eklemektir. Bu tarz çok işlemcili sistemler **çok çekirdekli (multi-core)** olarak

tanımlanmaktadırlar. Birer çekirdeğe sahip çok işlemcili sistemlerden daha avantajlıdırlar çünkü:

- Tek chip üzerindeki iletişim, chipler arası iletişimden **daha hızlıdır**.
- Ayrıca çok daha **az güç tüketirler.**



Şekil 7. Tek chip üzerinde çift çekirdekli işlemci

Şekil 7'de de görüldüğü üzere her çekirdeğin kendi kayıtçıları (register) ve ön belleği (cache) vardır.

3.6. İşletim Sisteminde Multiprogramming (Çoklu programlama)

Tek program, CPU'yu ve I/O cihazları sürekli meşgul tutamaz. Kullanıcılar genelde birden fazla programa sahiptirler. **Çoklu programlama**, işletim sistemi düzeyindeki işleri (**job** – *code and data*) organize ederek, CPU'nun *her zaman bir işle meşgul olmasını sağlar* ve CPU kullanımını attırır. Diğer bir değişle, CPU'nun **idle** duruma **düşmemesi** sağlanır.

Zaman paylaşımlı sistemler (**time-sharing system**), çoklu programlamanın mantıksal bir uzantısı / parçasıdır. Bu sistemlerde, CPU birden fazla işi, işleri kendi aralarında yer değiştirerek çalıştırır. Bu yer değişimleri, kullanıcıların programlar ile olan etkileşimleri (**interaction**) sonucu tetiklenir. Kullanıcı, bir program aracılığıyla veya klavye, fare, dokunmatik ekran (*input device*) aracılığıyla işletim sistemine komutlar gönderir ve hızlıca sonuç görmeyi bekler (*output device*).

Her kullanıcı belleğe yüklenen en az bir tane programa sahiptir. Belleğe *yüklenen* ve *çalıştırılan (executed)* program **proses** (**process**) olarak adlandırılır. Zaman paylaşımı ve çoklu programlama, birden fazla işin (*job*) aynı anda bellekte tutulmasını gerektirir. Öncelikle tüm işler disk üzerindeki iş havuzunda (**job pool**) yer alır ve sonra buradan belleğe aktarılırlar. Eğer birden fazla iş, **diskten → belleğe** alınmak için **hazırsa** ve hepsi için yeterli sayıda **room** yoksa işletim sistemi bu işler arasından birisini seçmelidir. Seçme işlemine **job scheduling** (**iş planlayıcısı / sıralayıcısı / dağıtıcısı**) adı verilir. Belleğe yüklenen birden fazla iş, aynı anda çalışmak için **hazırsa**, **CPU scheduling** ile hangi işin **execute** edileceğinin belirlenmesi gerekmektedir (neye göre?).

3.7. İşletim Sistemi Operasyon Mode'ları (Kip)

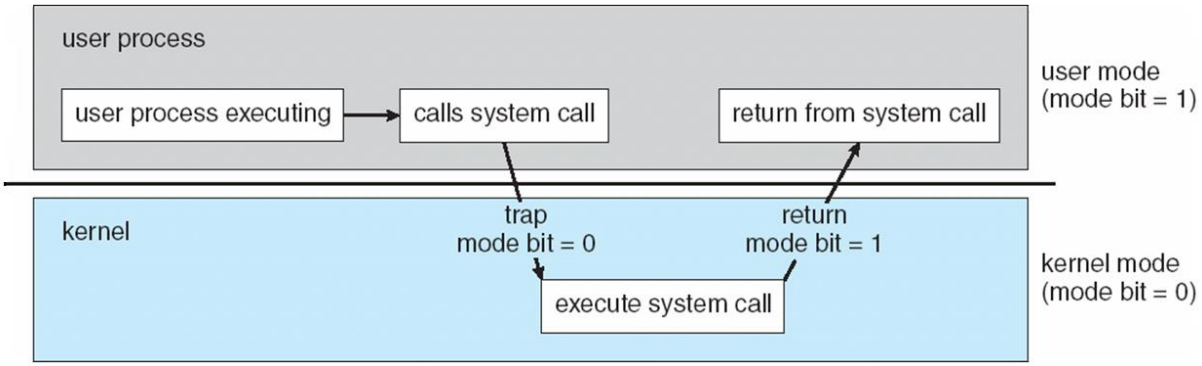
İşletim sistemi ve sistemdeki kullanıcılar, yazılım ve donanım kaynaklarını paylaştıkları için kullanıcının programında meydana gelen bir hatanın sadece o programı etkilediğine (**teoride diğerlerini etkilememeli**) emin olmak isteriz. Paylaşımlı sistemlerde, programın birinde

oluşan bir **bug**, birçok *prosesi* **olumsuz** yönde etkileyebilir. Örneğin bir prosesin **sonsuz döngüye girmesi**, diğer proseslerin işlerini doğru tamamlayamamalarına neden olabilir.

Çoklu programlamanın olduğu işletim sistemlerinde fark edilmesi daha zor hatalar meydana gelebilir, örneğin bir program diğer bir programı veya onun datasını veya işletim sistemini direk etkileyebilir. İşletim sistemi bu tarz hatalara karşı korumalı olmalı ve normal çalışmasını sürdürmelidir. Bunu desteklemek için işletim sistemi yazılım kodu ile kullanıcı kodları farklı seviyelerde çalıştırılabilmelidir. İşletim sistemlerinde bu ayrımı yapabilmek için 2 temel çalışma moduna (**dual mode**) ihtiyaç vardır:

- **Kernel mode** (System mode, Supervisor mode, Privilege mode)
- **User mode**

Güncel çalışma modunu ayırt etmek için bilgisayar **donanımı** seviyesinde **mode biti** eklenmiştir: **kernel (0)**, **user (1)**. Bu mode sayesinde bir prosesin işletim sistemi adına mı yoksa kullanıcı adına mı çalıştığı ayırt edilebilir. Bilgisayar bir kullanıcı uygulaması çalıştırıyorsa, user mode'da çalışılıyor demektir. Bir kullanıcı uygulaması çalışırken işletim sisteminde bir servis talep ediyorsa (system call, sistem çağrısı), işletim sistemi bu talebi gerçekleştirebilmek adına **user mode'dan → kernel mode'a geçiş** yapmalıdır. Bu geçiş Şekil 8'de gösterilmiştir.



Şekil 8. User mode'dan kernel mode'a geçiş

Sistem boot edilirken donanım kernel mode'da çalışmaya başlar. Daha sonra işletim sistemi yüklenir ve kullanıcı uygulamaları user mode'da çalışır. Ne zaman bir trap ya da interrupt gerçekleşse, donanım user mode'dan kernel mode'a geçer.

3.8. İşletim Sisteminden Beklenen Özellikler

Proses Yönetimi

- Kullanıcını ve sistem proseslerini yaratmak, silmek.
- Prosesleri durdurmak ve çalışmaya devam ettirmek.
- CPU'daki proses ve iş parçacıklarının çalışma önceliklerini ve sıralarını organize etmek (scheduling).
- Proses **senkronizasyonu** için bir mekanizma oluşturmak.
- Proseslerin birbirleri ile olan haberleşmesi için bir mekanizma oluşturmak.
- Kilitlenmelerin (**deadlock**) yönetilmesi. Ortak kaynakların kullanımında iki proses de bekleme durumuna geçerse kilitlenme olur. Yani, biri diğerinin sonucunu beklerken, diğeri de ötekinin sonucunu bekler.

Bellek Yönetimi

- Bellekteki alanların, kim tarafından kullanıldığını takip etmek.
- Hangi proseslerin ve verilerin belleğe veya bellekten taşınmasına kara vermek.
- Bellekten yer tahsis etmek veya bellekteki ayrılan alanı serbest bırakmak.

Dosya ve Disk Yönetimi

- Dosya organizasyonu için klasörler yaratmak ve silmek.
- Dosya ve dizinler üzerinde değişiklik yapabilmeyi sağlamak.
- Disk planlaması (scheduling) yapmak.
- Alan tahsis yönetimi gerçekleştirmek (Storage allocation).
- Boş alan (free-space) yönetimi yapmak.

Giriş / Çıkış Birimleri Yönetimi

- Ön belleğe yazmak ve okumak.
- **Spooling** (kuyruklama) işlemlerini gerçekleştirmek.
- DD (Device Driver) ara yüzlerini yönetmek.
- Belirli donanım aygıtları için sürücüleri yönetmek.