

Mikroişlemciler ve Intel Mikroişlemcileri Alt Yapısı

Haftanın Amacı:

1. Temel mikroişlemcileri ve yapılarını tanıtmak
2. Intel Mikroişlemciler Tanıtmak
3. Bilgisayar Temel mimarisini vermek
4. Bilgisayar Temel Mimarisindeki Kavramları tanıtmak

Basit Mikroişlemci Mimarisi

Mikroişlemci, bellek ve I/O elemanlarının birleşiminden meydana gelen mikroişlemciye dayalı bilgisayar sistemlerine mikrobilgisayar denir.

Mikroişlemci, bir mikrobilgisayarın merkezi işlem birimidir. İşlemcinin fonksiyonu, program komut kodlarının bellekten alınıp getirilmesi, kodunun çözülmesi ve çalıştırılması ve giriş-çıkış işlemlerinde kullanılan kontrol sinyallerinin üretilmesi ve senkronizasyonun sağlanmasıyla sonuçların gözlenmesi işlemleridir. Bir mikroişlemcinin konfigürasyonuna kaydediciler, komut kod-çözücüsü, aritmetik ve mantık birimi, işlemlerde ardışıklığı sağlamak için frekans üretici, bölücü ve sayıcı gibi zamanlama ve kontrol elemanları dâhildir.

Mikroişlemciler bilgisayarın kalbidir, aynı zamanda Merkezi İşlem Birimi (CPU) olarak anılır. CPU genel olarak aşağıdaki işlemleri yapar.

- Sistemdeki tüm elemanlar ve birimlere zamanlama ve kontrol sinyali sağlar
- Bellekten komut alıp getirir.
- Komutun kodunu çözer.
- Komutun operandına göre, veriyi kendisine veya G/Ç birimine aktarır.
- Aritmetik ve mantık işlemlerini yürütür.
- Program işlenirken, diğer donanım birimlerinden gelen kesme taleplerine cevap verir.

Mikroişlemcinin Mimari Yapısı en basit şekliyle

- Aritmetik ve Mantık Birimi(ALU)
- Zamanlama ve Kontrol Birimi
- Kaydediciler

Birilerinden oluşur.

Aritmetik ve Mantık Birimi(ALU)

ALU, mikroişlemcide aritmetik ve mantık işlemlerinin yapıldığı yerdir. Bu birime giriş işlemleri, akümülatör kaydedicisiyle bellekten alınan veri arasında veya akümülatörle diğer kaydediciler arasında olabilir.

ALU' da yürütülecek işlemler, Aritmetik işlemler olarak toplama, çıkarma, bölme ve çarpma, mantık işlemleri denince, AND, OR, EXOR, ve NOT dır. ALU da yapılan bütün bu işlemler kontrol sinyalleri vasıtası ile, zamanlama ve kontrol biriminin gözetiminde eşzamanlı olarak yapılır.

ALU' da basit matematik komutlar zorlanmadan işlenebilir fakat karmaşık aritmetik işlemleri gerçekleştirmek için ayrı altıyordam gruplarına veya elektronik devrelere ihtiyaç duyulur. Eğer ek devre konulmamışsa, mevcut devrelerle bu işlemleri gerçekleştirmek için birbiri ardına aynı komutu defalarca işlemek gereklidir, bu da zaman kaybı demektir. Gelişmiş mikroişlemcilerde bu devreler yerleşik vaziyettedir. Gelişmiş mikroişlemcilerde aynı zamanda, Kesirli sayılarla işlem yapmayı sağlayan FPU(Floating Point Unit) denilen bir işlemci daha yerleştirilmiştir.

Zamanlama ve Kontrol Birimi

Sistemin tüm işleyişinden ve işlemin zamanında yapılmasını sağlar. Zamanlama ve Kontrol birimi, bellekte program bölümünde bulunan komut kodunun alınıp getirilmesi, kodunun çözülmesi, ALU tarafından işlenmesi ve sonucunun alınıp belleğe geri konulması için gerekli olan kontrol sinyallerini üretir. Bilgisayar sisteminde bulunan dahili ve harici bütün elemanlar bu kontrol sinyalleri ile denetlenir.

Kaydediciler

Bir işlemcinin tasarımında, mühendisler tamamıyla insanoğlunun günlük yaptığı iş ve davranışlara göre düzen yapmışlardır. Buna göre, yapılan iş ham bilgi girdisinin hızlı bir şekilde işlenerek kullanılabilir çıktıya dönüştürmesi olacağına göre, bu sistemde mutlaka verileri Gerçekçi olarak üzerinde tutacak, bir grup veri saklayıcıya yani kaydediciye ihtiyaç olacaktır. İşlemci içerisinde bir program yürütülürken, işlemcinin içerisinde yani yanı başında, kaydedicilere gerek duyulur.

Intel Firması Tarafından Üretilen Mikroişlemciler Ve Özellikleri

Üretici	Kısım	Veri Yolu Genişliği	Hafıza Boyutu
Intel	8048	8	2K İçsel
	8051	8	8K İçsel
	8085A	8	64K
	8086	16	1M
	8088	8	1M
	8096	16	8K İçsel
	80186	16	1M
	80188	8	1M
	80251	8	16K İçsel
	80286	16	16M
	80386EX	16	64M
	80386DX	32	4G
	80386SL	16	32M
	80386SLC	16	32M + 1K Ön Bellek
	80386SX	16	16M
	80386DX/DX2	32	4G + 8K Ön Bellek
	80386SX	32	4G + 8K Ön Bellek
	80386DX4	32	4G + 16K Ön Bellek
	Pentium	64	4G + 16K Ön Bellek
	Pentium (P24T)	32	4G + 16K Ön Bellek
Pentium Pro İşlemci	Pentium Pro İşlemci	64	64G + 16K L1 Ön Bellek + 256K L2 Ön Bellek
	Pentium II	64	64G + 32K L1 Ön Bellek + 512K L2 Ön Bellek
	Pentium II Xeon	64	64G + 32K L1 Ön Bellek + 512K veya 1 M L2 Ön Bellek
	Pentium III, Pentium 4	64	64G + 32K L1 Ön Bellek + 256K L2 Ön Bellek

Pentium Mikroişlemciler

Pentium 1993 de geliştirilmiştir, 80386 ve 80486 mikroişlemcilere benzer bir işlemcidir.

Mikroişlemci orijinal olarak P5 veya 80586 olarak etiketlendi, ama Intel numara kullanmayı düşünmedi bunun yerine Pentium etiketini kullandı.

Pentium un daha sonra üretilen versiyonu ilave komutlar içermekte idi bu komutlar multimedia komutları idi ve MMX olarak ta adlandırılır. Intel'in düşüncesi bu işlemciyi çok sayıda üreticinin kullanması idi, ancak bir kaç yazılım firması kullanmıştır.

Daha sonra, Intel uzun süre beklenen Pentium OverDrive(P24T) eski 80486 sitemleri için üretti. İşlemci 63 MHz veya 83 MHz saat hızında çalışabilmekte idi. Pentium OverDrive, 80486 dan Pentium'a olan ideal geçişi temsil eder.

Pentium un en ustaca özelliği onun çift integer işlemcileridir. Pentium iki komutu eş zamanlı olarak yürütmekte idi. Çünkü superscaler teknoloji olarak adlandırılan iki farklı integer işlemci içermekte idi. Bu özellik Pentium a her bir saat darbesinde iki komutu yürütebilir.

Pentium II ve Pentium XEON Mikroişlemciler

Pentium II 1997 de piyasaya sürülmüştür. Intel işlemciler için yeni bir yön çizmiştir. Önceki işlemcilerde kullanılan entegre devre teknolojilerinin yerine Pentium II daha küçük bir baskılı devreye yerleştirildi. L2 cache ve Mikroişlemcinin aynı baskılı devrede yer almasına Pentium II denmiştir.

Pentium II modülündeki işlemci MMX uzantılı Pentium Pro dur ve L2 cache yoktur.

1998 in ortalarında Pentium II nin yeni veriyonu Xeon duyurulmuştur. Bu model sunucu uygulamaları ve hızlı istemciler için geliştirilmiştir.

Pentium II ile Pentium II Xeon arasındaki ana fark Xeon 32K byte L1 cache ve 512K, 1M veya 2M byte cache özelliğinin mümkün olması

Pentium III Mikroişlemciler

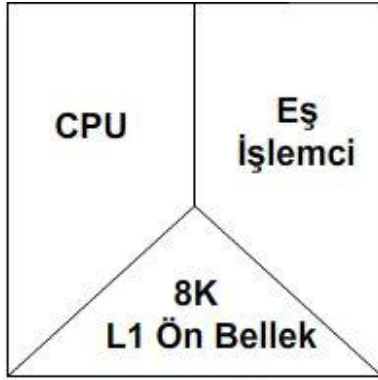
Pentium III Mikro işlemcisi Pentium II den hızlı bir çekirdek kullanır. Ancak hala P6 veya Pentium Pro işlemcidir. Aynı zamanda plastik kılıflı ve 370 versiyon olarak üretilen Flip-chip olarak adlandırılan bir modeli üretilmiştir. Bu eski model Pentium paketlerine benzemekte idi. Intel Flip-chip sürümün daha ucuz olduğunu ileri sürmektedir.

Pentium IV Mikroişlemciler

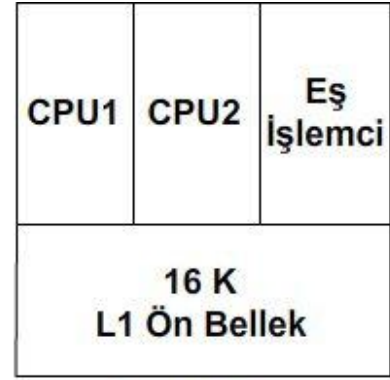
Pentium 4 mikroişlemci, 2000 yılının sonlarında duyurulmuştur. Pentium 4 , Pentium Pro dan başlayarak Pentium III e kadar devam eden işlemciler gibi P-6 yapısında idi. Pentium 4 dün 1.3, 1.4 ve 1.5 GHz hız sürümlerinde olan ana farklılık Entegre yapısının RAMBUS hafıza teknolojisini desteklemesi.

Entegre içerisinde yapılan değişiklikler, boyut düzenlemeleri yüksek işlemci hızları elde edilmesini sağlamıştır. Şaşırtıcı olanı ise Intel'in L1 önbelleğini 32K dan 8 KB düşürmesidir

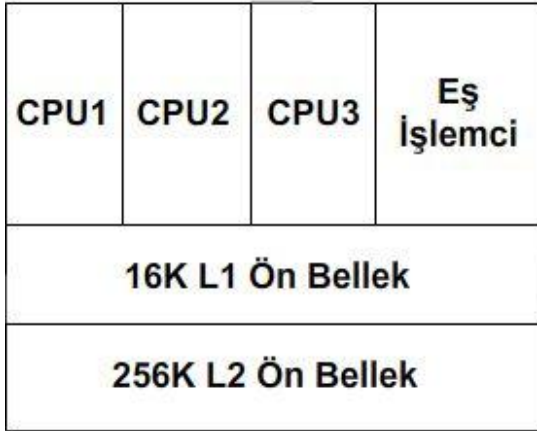
Diğer bir değişiklik Alüminyum yerine bakır iç bağlantılar kullanılmasıdır. Bunun nedeni bakırın iyi bir iletken olmasıdır. Bakır mikroişlemci saat frekansını artırmaktadır.



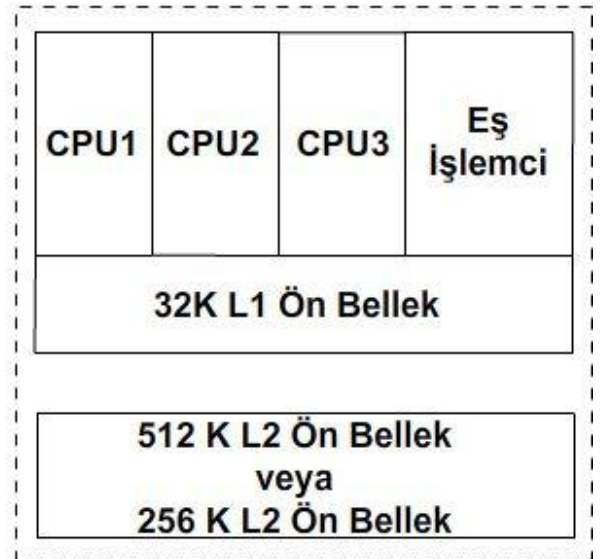
80486DX



Pentium



Pentium Pro



**Pentium II, Pentium III veya
Pentium 4 Modülü**

Motorola Mikroişlemcileri

Motorola firması rakip bir işlemci üretici firmasıdır. Bu firmanın ürettiği belli başlı işlemci ve özellikleri aşağıda verilmiştir.

Mimari Esasları

Bir bilgisayar komut kümesi, programcının makineyi programlarken kullanabileceği ilkel emirleri veya makine komutlarının tamamının oluşturduğu kümeyi belirtir. Bir komut setinin karmaşıklığı, komut ve veri formatlarına, adresleme modalarına, genel amaçlı registrlara, opcode tanımlamalarına ve kullanılan akış kontrol mekanizmalarına bağlıdır. İşlemci tasarımıdaki komut seti mimarileri iki ekol üzerinde geliştirilmiştir. Bunlar CISC(sisk) ve RISC tipi Mimariler.

CISC Mimarisi

Intel'in X86 mimarisine dayalı işlemci serisinin ortaya çıktığı 70'li yıllarda, elektronik veri saklama elemanlarından olan RAM'lerin pahalı ve kısıtlı olması sebebiyle bu kaynakların tasaruflu bir şekilde kullanarak yüksek seviyeli dillerin desteklenmesini isteyen tasarım mimarları tarafından geliştirilmiştir. Bilgisayar performansını ve yapılan işlemleri karmaşıklaştırırsa da yazılımı basitleştirmektedir.

Mimarinin üstünlükleri

1. Mikroprogramlama, assembly dilinin yürütülmesi kadar kolaydır ve sistemdeki kontrol birimlerinden daha ucuzdur.
2. Yeni komutlar ve mikrokod ROM2a eklemenin kolaylığı tasarımcılara CISC makinelerini geriye doğru uyumlu yapmalarına izin verir.
3. Herbir komut daha yetenekli olmaya başladığından, verilen bir görevi yürütmek için daha az komut kullanır. Bu nispeten yavaş ama Ana belleğin daha etkili kullanımını sağlar.
4. Miroprogram komut kümeleri, yüksek seviyeli dillerin yapılarına benzer biçimde yazıldığı için derleyici dizaynı da basit olur.

CISC Mimarinin Dezavantajları

1. İşlemci ailesinin önceki sürümleri kabullendiğinden, komut kodu ve devre tasarımı karmaşılaşır.
2. Farklı komutlar farklı miktarda makine çevrimi tutarlar. Dolayısı ile performans düşer.
3. Çok özel güçlü komular sıklıkla kullanılmıyor.
4. Komutlar genellikle bayrak kodunu komuta bir yan etki olarak kurar. Bu ise ek çevrimler yani beklemler getirir.

RISC Mimarisi

RISC mimarisi CISC mimarili işlemcilerin kötü yanlarını piyasanın tepkisi ve ona bir alternatif olarak, işlemci mimarisinde söz sahibi olan IBM, Motorola, Apple gibi firmalarca sistematik bir şekilde geliştirilmiştir.

Belleklerin ucuzlaması, işlemcilerin hızlanması, CISC mimarisinin avantajını ortadan kaldırmaya başlamıştır. Karmaşık problemler yerine standart ve kısa işlem süresine sahip komutlar ve bunları donanımsal olarak işleyen kontrol birimi daha avantajlı duruma gelmiştir.

RISK Mimarisinin Üstünlükleri

1. Hız: Azaltılmış komut kümesi, kanal ve süperskalar tasarıma izin verdiğinden RISC işlemciler genellikle karşılaştırılabilir yarı iletken teknolojisi ve aynı saat oranları kullanılan CISC işlemcilerinin performansının 2 veya 4 katı yüksek performans gösterirler.
2. Basit Donanım: RISC işlemcinin komut kümesi çok basit olduğundan çok az çip uzayı kullanırlar. Ekstra fonksiyonlar, bellek kontrol birimleri veya kayan noktalı aritmetik birimleri de aynı çip üzerine yerleştirilir.
3. Kısa Tasarım zamanı: RISC işlemciler CISC işlemcilere göre daha basit olduğundan daha çabuk tasarlanabilirler ve diğer teknolojik gelişmelerin avantajlarını CISC tasarımlarına göre daha çabuk kabul edebilirler.

Ön-Belek (cache)

Düşük hızı işlemcilerin kullanıldığı 80 li ve 90 lı yılların başında Ana hafıza işlemciye göre hızlı idi. İşlemci hızlarının artması ile birlikte ana bellek hızlarında paralel bir artış olmamıştır. Mikro işlemci Anabellekten alıp getirdiği programları yürüttüğünden dolayı, sistemin performansında düşüslere sebep olmakta idi. Bu problemi çözmek ve performansı artırmak için işlemci içerisine ve dışına Ön-bellek yerleştirilmiştir. Ön bellek'ler Statik RAM lerdir. Yani güç beslemesi olduğu müddetçe içerisindeki bilgileri dinamik belleklerde olduğu gibi kaybetmez. Dinamik belleklerin hızları 70 ns lerde iken statik belleklerin hızları 8-12 ns ler civarındadır.

Ön-Beleğin Çalışması

Ön-bellek sisteminin amacı, ana bellek ile yapılan veri alış-verişinde bekleme olmaksızın hızlı bir iletişim sağlamaktır. Bütün bunlar sistem içerisindeki ön-bellek denetleyicisi tarafından kontrol edilir. Mikroişlemci bir veri grubu üzerinde çalışacağı zaman ilk önce dâhili olan L1 ön-belleğine bakar. Ön-bellek denetleyicisi, ana bellekte icra edilecek verinin bir kopyasını kapasitelerine uygun olarak alıp ön belleklere yerleştirmişse, her hangi bir sorun olmayacaktır. Yani işlemci hızlı bir şekilde ön bellekten veriyi işleyecektir. L1 de olmadığı zaman işlemci dış veri yoluna çıkacak L2 ön-belleğine bakacaktır. Bu durumda işlemci veri yolundan daha yavaş bir veri yolu kullanılacak dolayısı ile de performans düşecektir. Eğer L2 ön belleğinde de veri bulunamaz ise o zaman ana belleğe gidilecek iş daha da yavaş yürüyecektir.

Haftanın İlave Okuma Kaynağı:

Dr. Nurettin Topaloğlu, "X86 Tabanlı Mikroişlemci Mimarisi ve Assembly Dili", Haziran 2004, Seçkin yayınevi

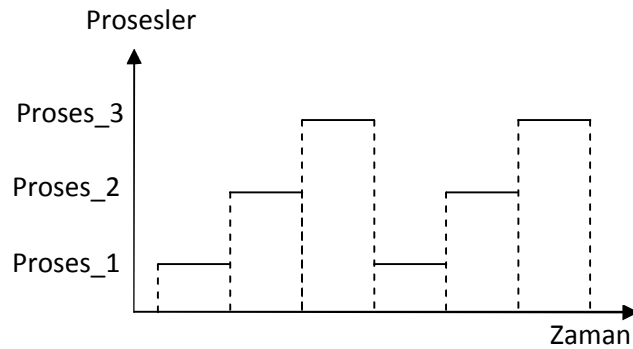
Sayfa **23** den sayfa **170**'e kadar

3. PROSES YÖNETİMİ (PROCESS MANAGEMENT)

Bir bilgisayar sisteminde çalışan tüm yazılımlar, proses olarak anılır. Genel olarak bir sistemde, kullanıcı ya da işletim sisteminin prosesleri vardır. Bir proses, sistem çağrılarını kullanarak başka bir proses oluşturabilir ya da diğer proseslerle etkileşim kurabilir.

Proses daha teknik bir tabirle, çalışır durumda olan programa verilen bir isimdir. CPU tarafından işletilmeyen bir program pasif bir yapıya sahip olup komutlar dizisinden ibarettir. Bu komutlar işleme alınmak suretiyle işlevlerini yerine getirebilirler.

Tüm modern bilgisayarlar aynı anda birkaç işi yapabilirler; kullanıcı programı çalışırken diskten okuma, bir text dokümanı ekranda gösterme veya çıktı işlemlerini bir arada gerçekleştirebilir. Çoklu programlamayı destekleyen sistemlerde CPU, bir programdan diğer bir programa belli zaman dilimlerinde anahtarlanır. Aslında CPU aynı anda bir programı çalıştırabilmesine rağmen, prosesler arasında hızlı anahtarlandığından dolayı bu prosesleri paralel çalıştırdığı izlenimini verir.



CPU'nun proseslere anahtarlanması

Grafikten, belli bir zaman aralığında tüm proseslere çalışma imkanının verildiği görülmektedir.

İşletim sistemine ait görevler (spooling gibi), toplu işler (batch job), kullanıcı programları birer prosesdir. Prosesler CPU, bellek, dosya ya da giriş/çıkış işlemleri gerektirebilirler. Bu kaynaklar ya proses oluşturulurken ya da çalıştırılırken proseslere tahsis edilirler.

3.1. Klasik ve Modern Prosesler

Teknolojik gelişmeler sonucunda proses kavramı iki farklı manaya gelmektedir. Klasik proses; von Neumann mimarisi bilgisayarlarda işletilmekte olan programa ait bir tabirdir. 80'li 90'lı yıllarda uygulama programları için yeterli olmuştur. Daha sonraları modern proses ve thread (iplik) kavramı ortaya çıkmıştır. Modern prosesler bir stüdyo gibi, thread'ler ise müzik yapmak için stüdyoyu kullanan müzisyenler gibi düşünülebilir. Klasik proseslerde, her bir müzisyen, kendi stüdyosuna sahipken, modern proseslerde müzisyenler (thread) ortak bir stüdyoyu paylaşabilirler. Programcılar, programlarının değişik kısımlarını bir thread kümesi olarak tek bir klasik proses çatısı altında geliştirebilirler. Klasik prosesler, tek bir thread'e sahip modern prosesler olarak düşünülebilir.

Bir bilgisayar sisteminde her bir prosesin kendi adres uzayı vardır. Prosesler bir arada çalışabilirler ancak bazı özel mekanizmalar (shared memory gibi) dışında adres uzaylarını paylaşmazlar. Bir proses çatısı altındaki thread'ler ise aynı adres uzayını, global değişkenleri, açık durumda olan dosyaları paylaşabilirler. Her thread'in kendine has durumları (hazır, bekler,...), program sayacı (PC), stack bölgesi, yerel değişkenleri ve saklayıcıları vardır. Bir prosesin birden fazla yürüteceği işlem varsa, işlemleri bölmek yani thread'lere paylaşmak performansı arttırabilir. Bu durumda bir thread bloke olduğunda diğer thread'ler çalışabilecektir. Thread'lerin kendi kaynakları olmadığı için oluşturulmaları ve sonlandırılmaları, proseslere göre daha kolaydır. Özellikle thread'lerin bir kısmı işlemciye yönelik diğer bir kısmı da I/O işlemlerine yönelik çalıştırılabilirse daha yüksek performans elde edilir. Örneğin bir kelime işlemci programı düşünüldüğünde, bir thread basılan karakterlerin algılanması için, diğer thread basılan karakterlerin ekranda gösterilmesi ve biçimsel değişiklikler için, başka bir thread de belli aralıklarla içeriğin kaydedilmesi için bir arada kullanılırsa, tek bir prosese göre daha yüksek performans gözlenecektir.



Kelime işlemci programında kullanılacak thread'ler

Çok işlemcili sistemlerde, her bir işlemciye bir thread atamak suretiyle gerçek paralellik sağlanabilir. Tek işlemcili sistemlerde ise, bir prosese adanan sürede, prosesin sahip olduğu thread'ler zaman paylaşımı olarak çalışırlar dolayısıyla gerçek paralellik sağlanamaz.

90'lı yılların öncesinde çoklu programlama, senkronizasyon ve kilitlenme gibi kavramlar klasik prosesler üzerinde yorumlanmıştır. Bugün ise tüm bu başlıklar genelleştirilerek, proses ve thread kavramına uygun hale getirilmiştir. Dolayısıyla işletim sistemini klasik proses kavramı üzerine öğrenmek ve daha sonra modern proses ve thread kavramına göre uydurmak daha kolay olacaktır.

Günümüz işletim sistemlerinden olan FreeBSD Unix, modern prosesleri destekler şekilde tasarlanmamasına rağmen, Linux ve Solaris gibi işletim sistemleri klasik proses kavramı üzerine inşa edilmiş ve bazı eklentilerle (kütüphane rutinleriyle) modern proses ve thread kavramlarını destekler hale getirilmiştir. Mach ve Windows işletim sistemleri modern proses ve thread kavramı üzerine kurulmuştur. Unix ailesinin klasik prosesleri, Windows ailesinin de modern prosesleri desteklemek üzere tasarlandığı söylenebilir.

3.2. Sistem Çağruları

Bilgisayar ilk açıldığı zaman işletim sistemi ana belleğe yüklenir ve çalışmaya başlar. İşletim sistemi bir programı çalıştırmak üzere seçtiğinde o programın bulunduğu ana bellek bölgesine dallanır ve programı işletir. Belirli bir zaman diliminden sonra işletim sistemi tekrar CPU'nun kontrolünü kazanır ve kendi kodunu çalıştırır. Ve bu durum sürekli olarak devam eder. Ancak kullanıcı programları ayrıcalıklı komutları çalıştıramazlar; kullanıcı adına işletim sistemi servisleri tarafından yapılır. Uygulama programları, sistem çağrılarını kullanarak bu servisleri kullanırlar. Sistem çağrıları, proses yönetimi, aygıt yönetimi, bellek yönetimi ve dosya yönetimi kapsamında işletim sistemi tarafından ele alınır. Örneğin proses yönetiminde; Unix'te `fork()`, Windows'ta ise `CreateProcess()` sistem çağrıları proses oluşturmak için kullanılır. Benzer şekilde Linux'ta `pthread_create()` ve Windows'ta `CreateThread()` thread oluşturmak için kullanılır. Proses yöneticisi kullanıcı komutlarının ve gerektiğinde ayrıcalıklı komutların işletiminden sorumludur.

Özellikle Unix'in birçok türevinin çıkmasıyla birlikte farklı sistem çağrı arabirimleri kullanılmaya başlanmıştır. Bu farklılıkları gidermek için POSIX.1 standardı geliştirilmiştir. Unix tabanlı sistemler (Linux, OpenBSD, FreeBSD) bu çağrı arabirimini baz alırken Windows ailesinin üyeleri Win32 API'yi kullanır.

Proses ve threadlerin oluşturulması ve sonlandırılması, proseslere kaynak tahsisi, proseslerin giriş/çıkış işlemleri için aygıt yöneticisiyle ve belleğe yüklenmesi için de bellek yöneticisiyle işbirliği yapmak proses yöneticisinin sorumluluğundadır.

3.3 Kullanıcı ve Supervisor Mod

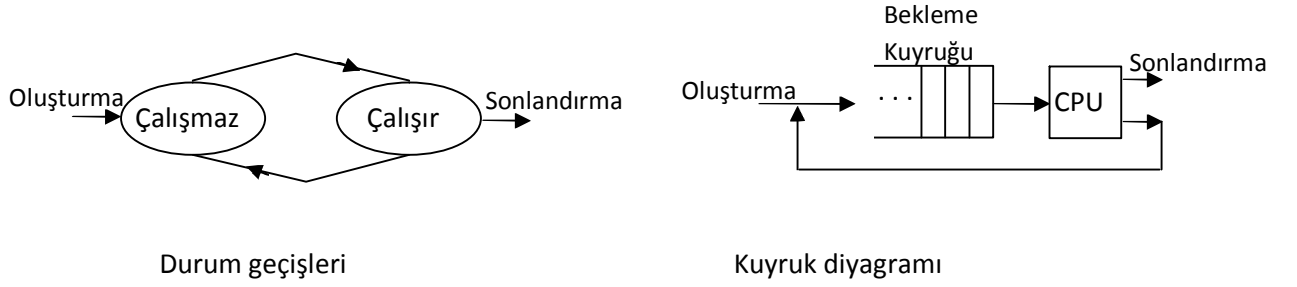
İşletim sistemi, kaynak paylaşımı ve korumanın sağlanabilmesi için giriş/çıkış komutları gibi ayrıcalıklı komutları kendi bünyesinde işler. Böyle olmasaydı, kullanıcı örneğin disk sürücünün herhangi bir kısmını istediği gibi okuyup yazabilirdi. Ayrıcalıklı komutlar, işletim sistemi tarafından CPU'nun supervisor (veya kernel) modunda işletilir. Bir kullanıcı programındaki tüm komutlar, kullanıcı modu komutlarıdır ve donanım üzerinde direkt olarak icra edilir. Kullanıcı programı, örneğin proses oluşturmak için `fork()` komutunu kullandığında donanım tarafından direkt olarak değil CPU'nun supervisor modunda işletim sistemi fonksiyonu olarak işletilir. Bu komutlar fonksiyon çağrıları olarak bilinir. Linux'ta birkaç yüz, Windows'ta ise birkaç bin sistem çağrısı vardır. Windows'ta sistem çağrılarının daha fazla olmasının sebeplerinden biri masaüstü pencere sisteminin işletim sisteminin bir parçası olmasıdır.

3.4. Proses Kontrol Bloğu ve Proses Durum Diyagramları

Bir prosesi sadece program kodlarından ibaret düşünmek yeterli olmaz. Çoklu programlamada, değişik zaman dilimlerinde farklı prosesler çalıştığından proseslere ait bilgilerin tutulduğu proses kontrol blokları (process control block) vardır. Bu bloklar, prosesin ana bellekteki yeri, hafıza limitleri, proses aktivitesi ('hazır', 'çalışır' ya da 'bekler' gibi), program sayacının değeri, işlemcinin kaydedicileri (EAX, EBX,...), prosesin ID numarası, prosesin önceliği, tahsis edilen kaynaklar, açık durumdaki dosyalar gibi bilgileri içerirler.

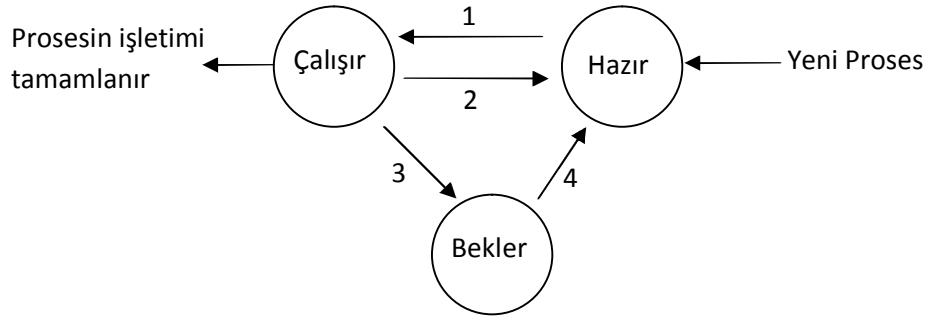
3.4.1. İki durumlu proses modeli

Herhangi bir zaman diliminde bir proses CPU tarafından ya işletiliyordur ya da işletilmiyordur. Yani proses ya çalışır ya da çalışmaz durumdadır. İşletim sistemi tarafından bir proses oluşturulduğunda çalışır durumda değildir. Bir kesme vuku bulduğunda ya da zamanlayıcı işlevi gerçekleştiğinde yeni bir proses çalıştırılmak üzere seçilecektir ve çalışır durumdaki proses çalışmaz durumuna geçecektir. Çalışır durumda olmayan prosesler daha sonra çalıştırılmak üzere kuyrukta tutulurlar. Kuyruktan da ilk gelen ilk çıkar (FIFO) mantığıyla veya proseslere öncelik değeri atamak suretiyle proses seçilebilir.

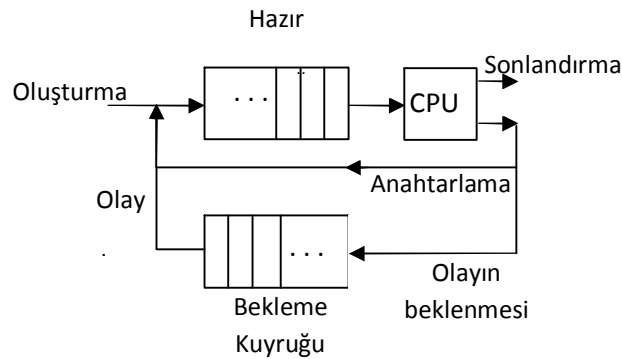


3.4.2. Üç durumlu proses modeli

Eğer tüm prosesler çalışmaya hazır durumundaysa, kuyruk yapısı iki durumlu modeldeki gibi düşünülebilir. Fakat giriş/çıkış işleminin sonlanmasını bekleyen bir proses olduğunda kuyruktan sıra ile prosesleri işleme sokmak mümkün olmayacaktır. İki durumlu modeldeki çalışmaz durumu, hazır ve bekler durumu olarak genişletilebilir.



3 durumlu modelin durum geçişleri



Üç durumlu modelin kuyruk diyagramı

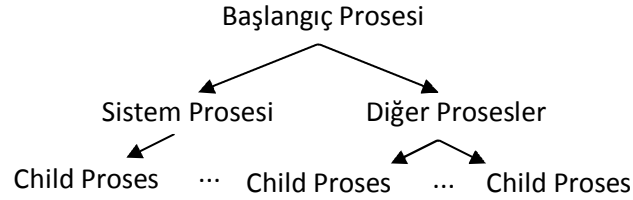
Proses yönetici oluşturduğu prosesi ilk olarak hazır kuyruğına koyar, daha sonra çalıştırmak üzere bir prosesi seçeceği zaman da yine bu kuyruktan seçim yapar. Çalışan prosesin zaman içerisinde ya işlemini sonlandırılır ya da duruma bağlı olarak hazır veya bekler kuyruğına konulur. Örneğin proses bir giriş/çıkış işlemine başladığı düşünülürse veya bir olayın gerçekleşmesini bekliyorsa bekler kuyruğına konur. Prosese ait olay gerçekleştiğinde (veya I/O işlemi tamamlandığında) bekler durumdaki proses hazır durumunu alır. Bir olay vuku bulduğunda (I/O, semafor,...) işletim sistemi bekleme kuyruğunu tarayacak, bu olayı bekleyen prosesleri tespit edecek hazır kuyruğına aktaracaktır. Kuyruқта yüzlerce proses olduğu düşünüldüğünde her bir olay için bir kuyruk tahsisi yapmak daha verimli bir kullanım olacaktır. Böylece olay vuku bulduğunda ilgili kuyruktaki tüm prosesler hazır kuyruğına aktarılacaktır.

3.5. Parent (anne) ve Child (çocuk) Proses Kavramı

Bir proses tarafından oluşturulan proses child proses, child prosesi üreten proses de parent proses olarak isimlendirilir. Proses hiyerarşisinde parent proses çok sayıda child prosese sahip olabilirken child proses sadece bir parent prosese aittir.

İşletim sisteminin kendisi ya da kullanıcı, proses oluşturabilir. Örneğin kullanıcı bir dosyayı yazdırmak istediğinde uygulama programı adına işletim sistemi bu işlemi yönetecek bir proses oluşturabilir. Ya da kullanıcı prosesi veri üretirken, bu proses tarafından oluşturulacak diğer bir proses eşzamanlı olarak verileri uygun bir formata çevirebilir. Bu durum yapısal programlamada oldukça faydalıdır.

Genellikle işletim sistemi yüklendiğinde bir başlangıç prosesi otomatik olarak oluşturulur ve bu proses tüm proses hiyerarşisinin kökü olur. Oluşturulan her proses de bu köke eklenir.



Proses hiyerarşisi