3.DERS
İŞLETİM SİSTEMLERİNDE GÖREV YÖNETİMİ

Ders içeriği

Bu ünite içinde İşletim sisteminin en önemli görevlerinden biri olan Görev Yönetimi kavramı açıklanmaya çalışılacaktır.

GÖREV YÖNETİMİ -1

Bilgisayar sistemlerinin verimli kullanımı, Mikroişlemci, ana bellek ve giriş/çıkış birimleri gibi kaynakların, programlar arasında paylaşılmasını gerektirir.

Mikroişlemcinin paylaşılması, programların işletimlerinin birlikte sürdürülmesi yoluyla sağlanır.

Mikroişlemcinin bir programın işletimini, ileride sürdürmek üzere, bırakıp diğer bir programın işletimine geçmesi *anahtarlama* olarak adlandırılır.

GÖREV YÖNETİMİ -2

Mikroişlemcinin paylaşımı, değişik programların bu birime belirli bir sıra ile anahtarlaması yoluyla gerçekleştirilir. Programların Mikroişlemciye hangi sıra ve kurallar çerçevesinde anahtarlanacağı, görev yönetimi kapsamında ele alınır.

Programlar işletimleri sırasında görev olarak adlandırılırlar. Mikroişlemcinin yönetimine, paylaşılan kaynak yerine, bu kaynağı paylaşan görevler yönünden bakılarak görev yönetimi de denir.

GÖREV YÖNETİMİ -3

Aşağıdaki şekilde Ekrana ASCII kodlarını yazan basit bir Assembly programı gözükmektedir.

```
Adres   Makina Kodu

0000   00 20 24
0003   2D 41 53 43 49 49
       20 4B 4F 44 4C 41
       52 49 2D 0D 0A 24

0000
0000   B8 ---- R
0003   8E D8
0005   8D 16 0003 R
0009   E8 0020 R

000C   B9 0100
000F   8D 16 0000 R
0013   E8 0020 R
0016   FE 06 0000 R
001A   E2 F3
001C   B4 4C
001E   CD 21
0020
0020
0020   B4 09
0022   CD 21
0024   C3
0025
```

```
Assembly Kodu
        .STACK 64
        .DATA
CHAR    DB    00,' $'
MESAJ1 DB '--ASCII KODLARI',0DH,0AH,'$'


              .CODE
ANA PROC FAR
        MOV AX,@DATA
        MOV DS,AX
        LEA DX,MESAJ1
        CALL YAZ

        MOV CX,256
DON:    LEA DX,CHAR
        CALL YAZ
        INC CHAR
        LOOP DON
        MOV AH,4CH
        INT 21H
ANA ENDP
YAZ PROC NEAR
        MOV AH,09H
        INT 21H
        RET
YAZ ENDP
```

.

NOT: Assembly dilinin açıklanması bu dersin içinde olmadığı için detaylara girilmeyecektir

GÖREV YÖNETİMİ -4

Derlenmiş kodun Adres ve Makine kodu olarak adlandırılan iki alanı vardır. Adres alanında o kodun hafızada saklandığı yerin adresi gözükür, Makine kodu alanında ise o kodun makin dili karşılığı olan hexa decimal (onlatılı) sayı karşılığı vardır.

Bunlar Adres ve Makine kodu alanlarıdır.

Buna göre **işaretlenmiş olan** (0003 8E D3 MOV DS,AX) satırında mikroişlemci içindeki AX isimli saklayıcının içeriğinin DS isimli saklayıcıya kopyalanması istenmiştir.

GÖREV YÖNETİMİ -5

1.Mikroişlemci içindeki PROGRAM SAYACI isimli saklayıcı işlenecek olan adresi üretir (0003). Bu adres ADRES YOLU üzerinden hafızaya iletilir.

2. Aynı anda mikroişlemci hafızadan okuma işlemi yapmak için OKU işareti üretir. Bu işaret KONTROL YOLU üzerinden hafızaya iletilir. Bu durumda istenilen adres bölgesinden bir Byte lık bilgi okumak için hazırlık yapılmış olur.

3.Bu oku işareti ile belirlenen adres bölgesinden ilk Byte lık bilgi VERİ YOLU üzerinden okunur. (8E) .Okunan bu değer MOV DS,AX sembolik kodunun makine dili karşılığıdır.

GÖREV YÖNETİMİ -6

4.Mikroişlemci içindeki kod çözme ünitesi (8E) nin kodunu çözdüğünde bunun MOV --, AX olduğunu anlar.

5.AX isimli saklayıcı içeriğinin hangi saklayıcıya aktarılacağını anlamak için mikro işlemci bir adım daha işlemelidir. Bu durumda PROGRAM SAYACI otomatik olarak 0004 adresini üretir. Bu adres aynı şekilde ADRES YOLU üzerinden iletilirken KONTROL YOLU üzerinden tekrar OKU işareti üretilir.

6.Bu durumda (D3) Byte ı okunur. Bu değer mikroişlemci içindeki kod çözme ünitesine geldiğinde bunun karşılığının DS saklayıcısı olduğu anlaşılır.

GÖREV YÖNETİMİ -7

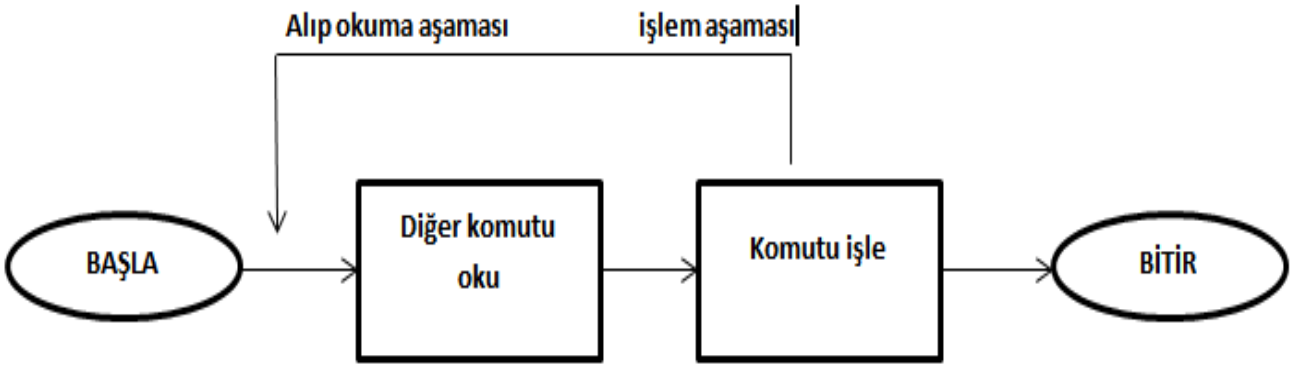
7.Bu durumda mikroişlemci içindeki kontrol ünitesi AX saklayıcının içeriğini DS saklayıcına aktarır.

8.Aynı anda PROGRAM SAYACI bir sonraki kodu okumak için (0005) adresini üretimiştir.

Mikroişlemci çok kısa bir sürede ne çok iş yapıyormuş!!

GÖREV YÖNETİMİ -8

Bu işlemlerin sürekli olarak çalışması aşağıdaki şekilde verilmiştir.



GÖREV YÖNETİMİ -9

Bu şekilde yazılmış olan bir programın çalışması için ise öncelikli olarak hafızaya yüklenmesi gerekir. Hafızaya yüklenmiş ve çalışan bir program aşağıdaki şekilde gösterildiği gibi 4 farklı hafıza alanına ihtiyaç duyar;

Kod Alanı: Kalıcı depolama alanından okunan ve derlenmiş program kodunu içeren bölüm. (.CODE ile başlayıp END ANA ile biten programın derlenmiş şeklini tutan alan. Programda bu alan 25 Hex =37 Byte büyüklüğündedir.)

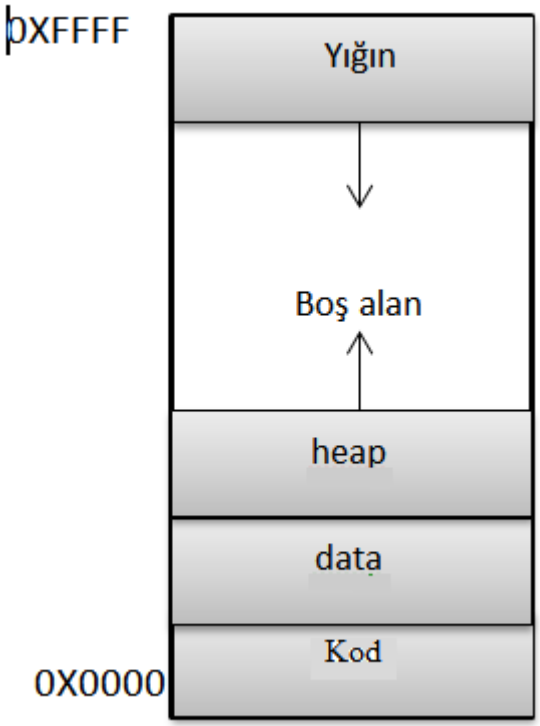
Data alanı: Program çalıştırılmadan önce atanan global ve statik değişkenleri depolar.(DATA şeklinde tanımlanmış olan alan. Programda bu alanının büyüklüğü 21 Byte dır)

GÖREV YÖNETİMİ -9

Heap alanı: Dinamik bellek tahsisi için kullanılan bir alandır. Programda bu alan tanımlanmamıştır.

Yığın Alanı: Yerel değişkenler için kullanılır. Bu alan yerel değişkenler tanımlandığında kullanılmak amacıyla rezerv edilir. Bu alan aynı zamanda fonksiyonların geri dönüş değerini saklamak için kullanılır. Yığının ters yönde büyüdüğüne dikkat etmek gerekir.(.STACK diye tanımlanan alandır ve programda 64byte olarak tanımlama yapılmıştır.

GÖREV YÖNETİMİ -10



ÇOK GÖREVLİ PROGRAMLAMA -1

Çok görevli programlamada ise mikroişlemcinin zaman paylaşımlı olarak birden çok programı çalıştırması istenmektedir.

Aslında her bir programın çalışması yukarıda açıklandığı şekilde olmaktadır. Burada programların, çalışabilmek için Mikroişlemcilere anahtarlanmaları zorunludur.

Ancak bunun yanı sıra ana belleğe yüklenmeleri, işletimleri sırasında gereksinim duyacakları giriş/çıkış türü kaynakları, diğer programlarla yarışarak elde etmeleri de gerekir

ÇOK GÖREVLİ PROGRAMLAMA -2

Programlar, mikroişlemci dışındaki, ana bellek, giriş/çıkış birimleri gibi kaynakları, yine ana işlem birimi aracılığıyla tüketebilirler.

Programlarının, ana belleğe yüklenebilmeleri, giriş/çıkış birimlerinden okuma-yazma yapabilmeleri, işletimleri sırasında gereksinim duyabilecekleri ek bellek alanlarını elde edebilmeleri, işletim sistemi içinde yer alan, ilgili yönetici ve sürücü görevlerin mikroişlemciye anahtarlanarak çalıştırılmaları sonucu gerçekleşebilir.

ÇOK GÖREVLİ PROGRAMLAMA -3

Mikroişlemcinin, birden çok işlemi zaman paylaşımlı olarak yapabilmesini sağlamak üzere bir işlemi bırakıp diğer bir işleme geçmesi belirli önlemler alınmadan yapılamaz.

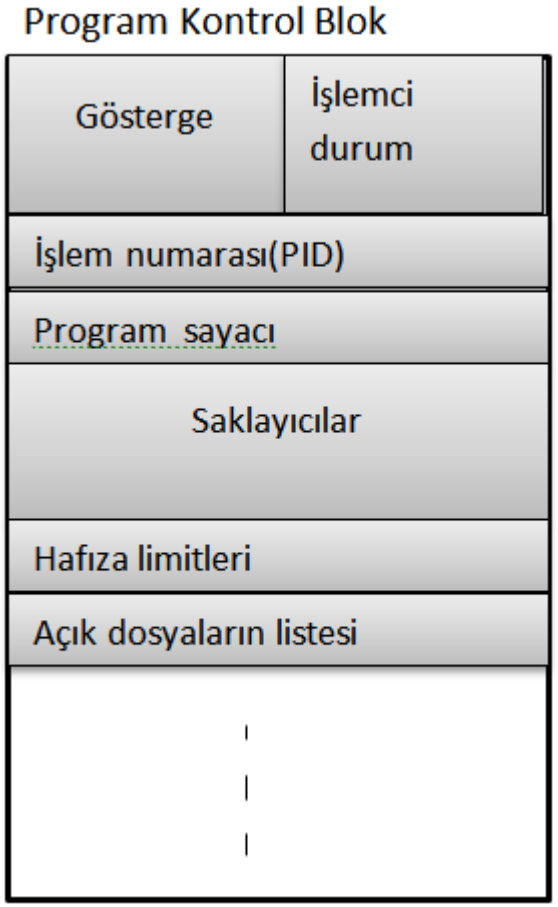
Yarım bırakılan bir işletimin, tutarlı bir biçimde, kalınan noktadan sürdürülebilmesi, işletimin bırakıldığı andaki durum bilgilerinin saklanması yoluyla sağlanır.

Bu nedenle, her görev için işletim sistemi tarafından bir veri yapısı tutulur.

ÇOK GÖREVLİ PROGRAMLAMA -4

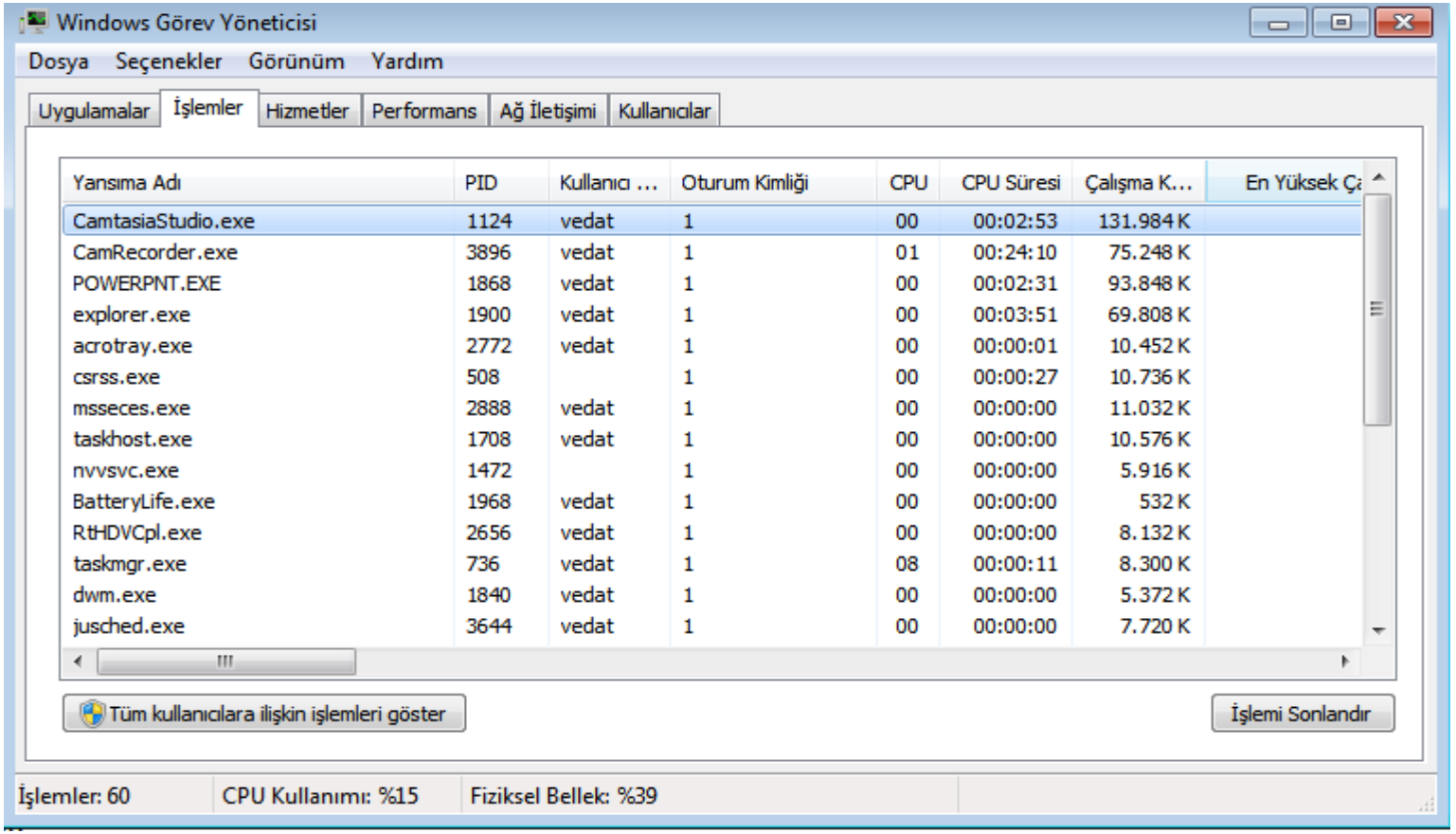
Bu veri yapısı, örneğin işletimin hangi komuttan başlayarak sürdürüleceği bilgisini de içeren mikroişlemci saklayıcı (register) içeriklerini, varsa kullanılan kütüklerle ilgili (kılavuz kütük, açık kütükler gibi) kimi bilgileri içermek durumundadır.

Program kontrol bloğu (PCB) olarak adlandırılır ve



ÇOK GÖREVLİ PROGRAMLAMA -5

Windows altında çalışan farklı görevler ait durumlar.



ÇOK GÖREVLİ PROGRAMLAMA -6

Unix/ Linux altında çalışan farklı görevler ait durumlar.

ps –e

```
UID        PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
root         1     0  0  2101   792   6 11:17 ?        00:00:02 init [2]
root         2     0  0     0     0   1 11:17 ?        00:00:00 [kthreadd]
root         3     2  0     0     0   0 11:17 ?        00:00:00 [ksoftirqd/0]
root         6     2  0     0     0   0 11:17 ?        00:00:00 [migration/0]
root         7     2  0     0     0   0 11:17 ?        00:00:00 [watchdog/0]
root         8     2  0     0     0   1 11:17 ?        00:00:00 [migration/1]
root        10     2  0     0     0   1 11:17 ?        00:00:00 [ksoftirqd/1]
root        12     2  0     0     0   1 11:17 ?        00:00:00 [watchdog/1]
root        13     2  0     0     0   2 11:17 ?        00:00:00 [migration/2]
root        15     2  0     0     0   2 11:17 ?        00:00:00 [ksoftirqd/2]
```

ÇOK GÖREVLİ PROGRAMLAMA -7

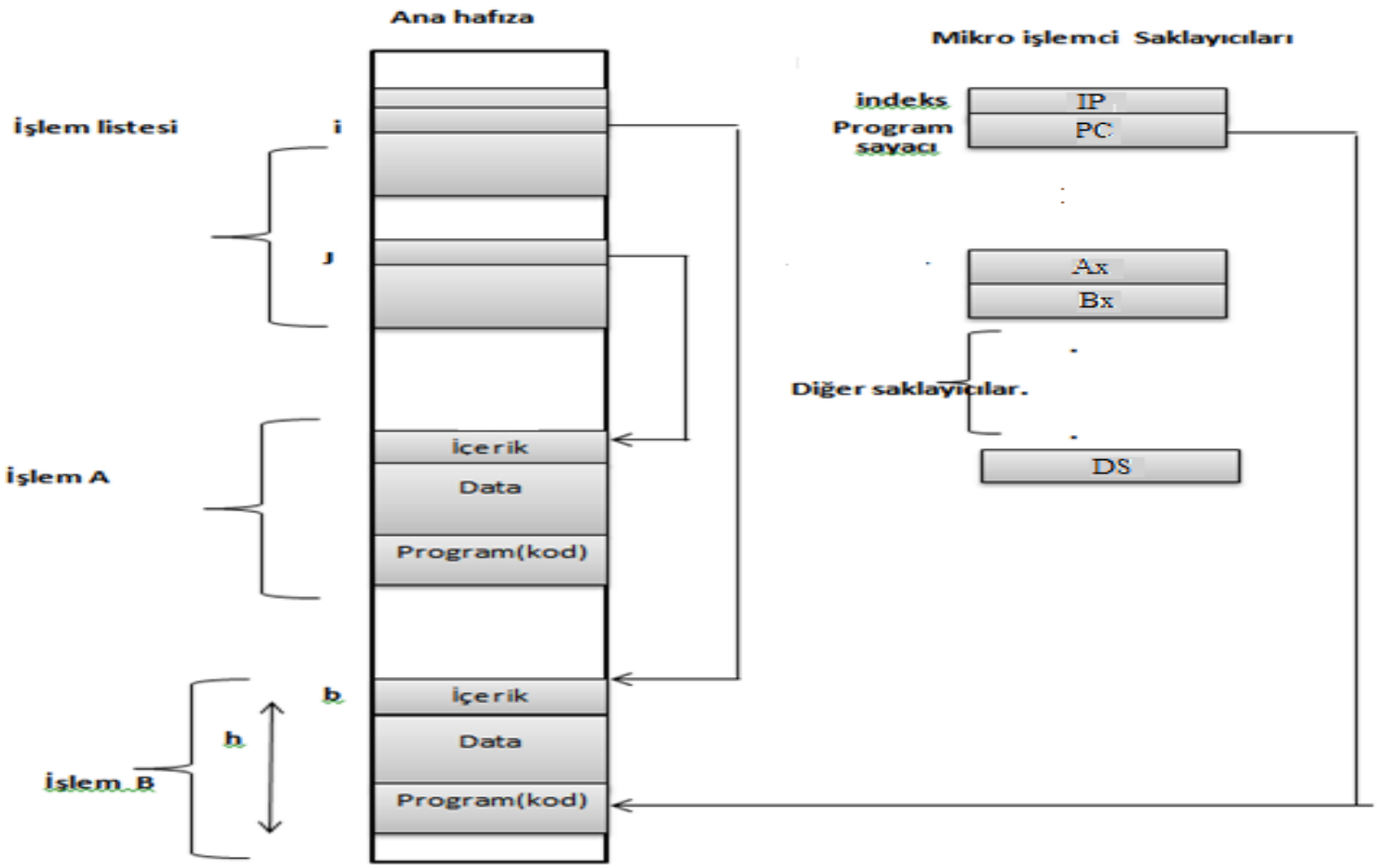
Çok görevi programlamada her bir görevin hafızada ne şekilde yerleştiği aşağıdaki şekilde görülmektedir.

Burada A ve B isminde iki adet görev Mikroişlemci tarafından sırası ile işlenmektedir.

Hangi görevlerin işleneceği işlem listesinde tutulmaktadır.

Mikroişlemcinin sırası ile bu görevleri yerine getirmesine Bu işleme “görev anahtarlama” işlemi denir.

ÇOK GÖREVLİ PROGRAMLAMA -7

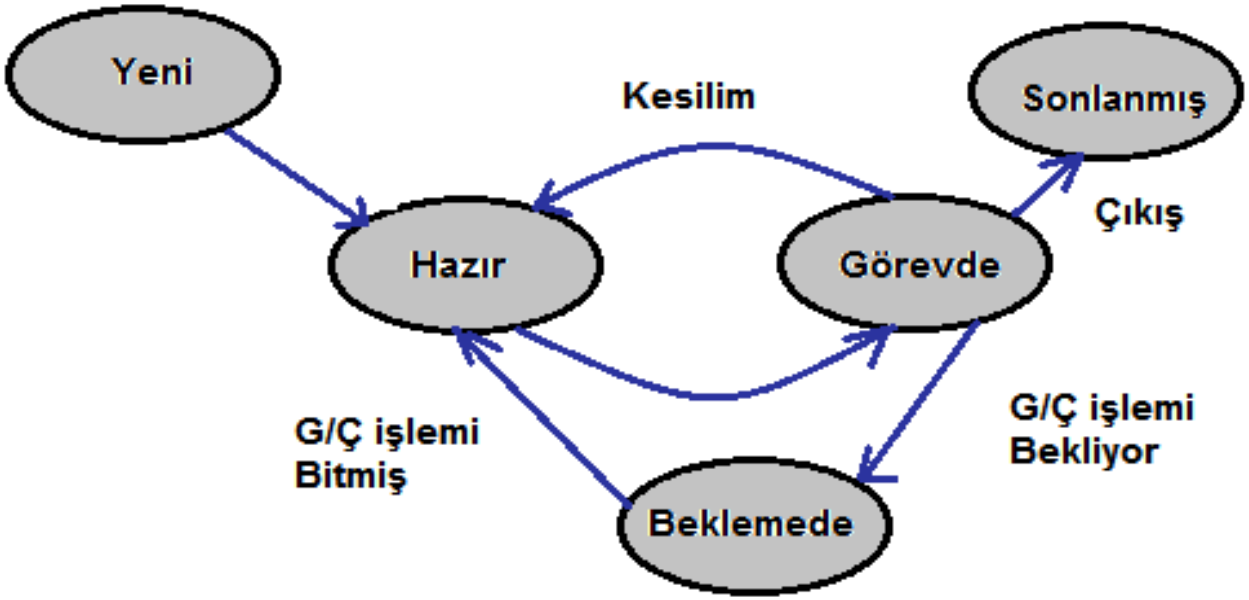


Mikroişlemcinin bir Görevi İşlemesi Aşamaları -1

Görevlerin, işletimleri sırasında bulunduğu durumlar, durum çizelgeleri ile gösterilir.

Durumların her biri, bu çizelgeler üzerinde bir çember ile simgelenir. Bir görev şu beş durumdan birinde bulunabilir.

Mikroişlemcinin bir Görevi İşlemesi Aşamaları -2



Mikroişlemcinin bir Görevi İşlemesi Aşamaları -2

Yeni - süreci oluşturulan olma aşamasındadır.

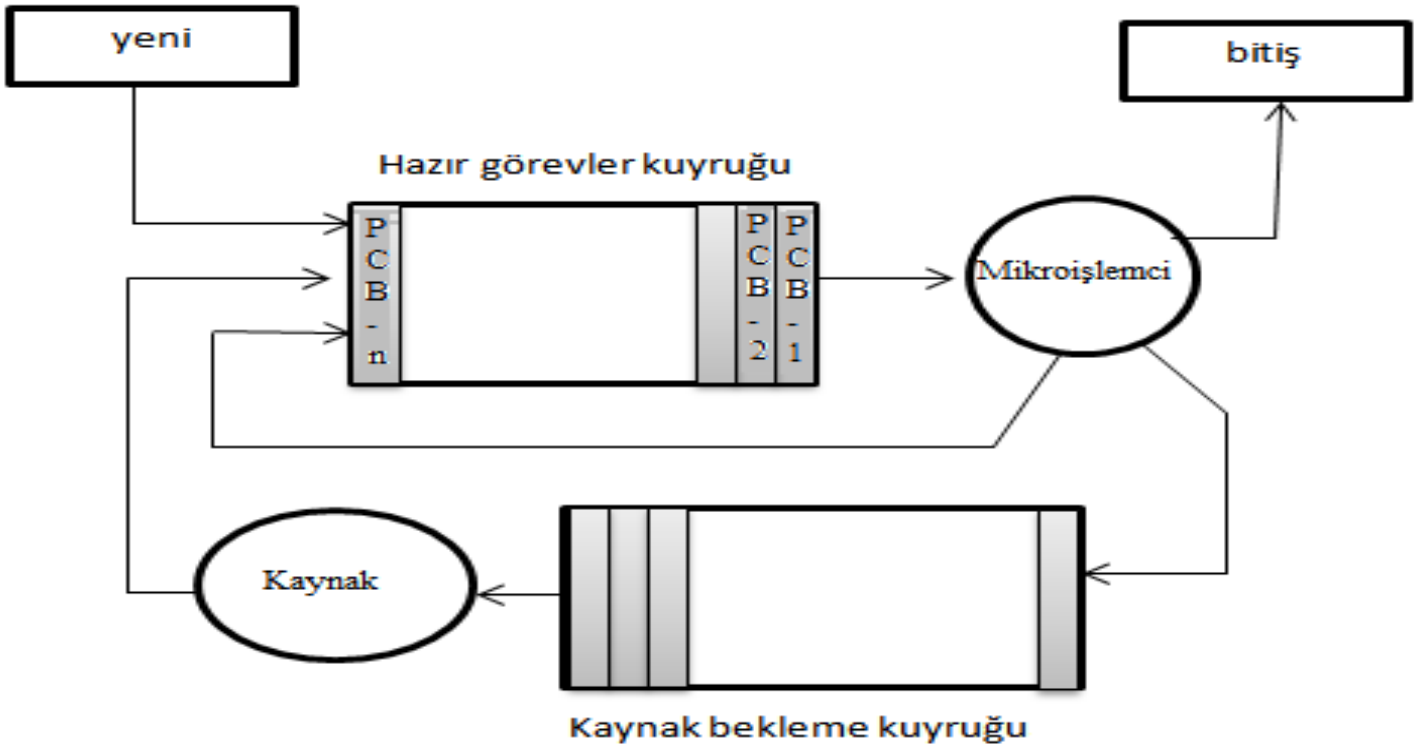
Hazır - sürecini çalıştırmak için gereken tüm kaynaklarını var ama işlemci şu anda bu sürecin talimatlarını çalıştırmıyor.

Görevde - Mikroişlemci bu görev üzerinde çalışıyor.

Beklemede - bazı kaynak kullanıma hazır hale gelmesi için bekliyor, çünkü görevi, şu anda çalıştıramazsınız. Örneğin süreci klavye, disk erişim isteği bekliyor olabilir.

Sonlanmış - Görev tamamladı.

Mikroişlemcinin bir Görevi İşlemesi Aşamaları -3



Mikroişlemcinin bir Görevi İşlemesi Aşamaları -3

Bir bilgisayar sisteminde yer alan kaynak sayısı işletilen görev sayısından çok daha az olduğundan görevler bu kaynakları belirli öncelik kıstaslarına göre sıralanmak zorundadırlar.

Görevlerin kaynakları kullanmak üzere kurdukları sıralara kuyruklar denir.

İşletim sisteminin temel görevlerinden görev yönetimi, Mikroişlemciyi, sistemde tanımlı görevler arasında paylaştırılmasından sorumludur.

Anahtarlama Algoritmaları-1

Bu paylaştırma yapılırken temel amaç sistem başarımının yükseltilmesidir. Görev yönetimi kapsamında, bu kıstaslardan bir yada birkaçını birlikte gözeten değişik yönetim algoritmaları kullanılır.

- İlk gelen önce işlem görür (First Come First Served)
- En kısa işletim süresi kalan önce (Shortest Remaining Time First)
- Öncelik tabanlı (Priority Based)
- Zaman dilimli (Round-Robin)

Anahtarlama Algoritmaları-1

Bu algoritmaları, işletilmekte olan bir görevin işletimini, bu görevin iradesi dışında **kesen** ya da **kesmeyen** algoritmalar olmak üzere iki değişik sınıfta ele almak olanaklıdır.

Bir görev, uygulanan yönetim gereği bir kez Mikroişlemciye anahtarlandıktan sonra giriş/çıkış, zaman uyumlama gereksinimleri gibi, kendisinden kaynaklanan nedenler dışında(kendi istemi dışında) da mikroişlemciyi bırakmak zorunda kalıyorsa uygulanan yönetim algoritması **kesen (preempetive)** algoritma olarak adlandırılmaktadır.

İlk Gelen Önce İşlem Görür -1

Doğal olarak görevlerin işletimlerinin kendi istemleri dışında kesilmediği durumlarda da kesmeyen (non-preemptive) algoritmalardan söz edilmektedir.

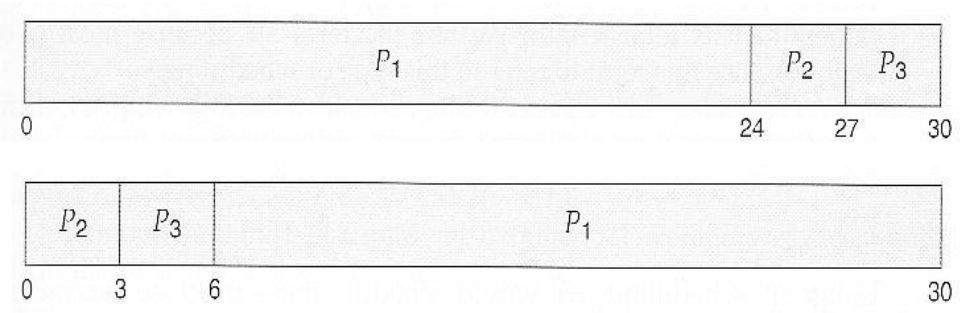
İlk gelen önce işlem görür

Sadece bir (FIFO) ilk gelen ilk çıkar kuyruğu .Bu algoritma kesmeyen bir algoritma olarak gerçekleştirilir. Ancak ne yazık ki, bu yöntem çok uzun ortalama bekleme süreleri alabilir. Örneğin, aşağıdaki üç süreçleri göz önünde bulunduralım:

İlk Gelen Önce İşlem Görür -2

Görev	Çalışma Süresi (msn)
p1	24
p2	3
p3	3

1. Örnekte ortalama bekleme süresi(0 + 24 + 27) / 3 = 17.0 msn. olur.



Ancak öncelikli olarak P2 görevi gelmiş olsa idi ortalama bekleme süresi (0+3+6)/3=3msn olacaktı.

En Kısa İşletim Süresi Kalan Önce -1

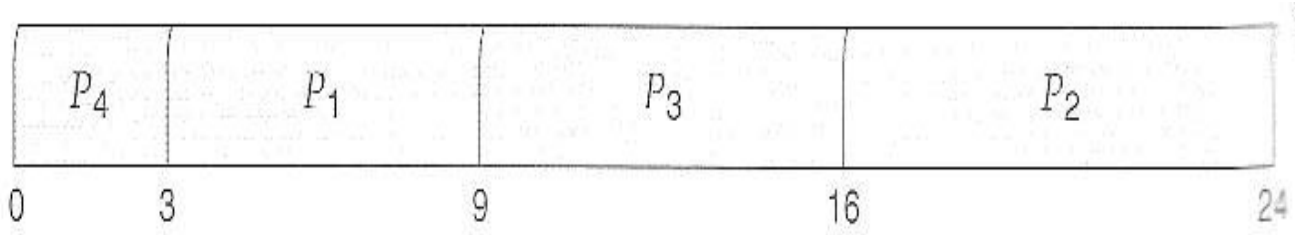
Bu algoritmada arkasındaki fikir ilk önce bitecek görevi öne almaktır. Burada sırası ileP4, P3,P2 ve P1 görevleri yerine getirilir.

Bu algoritma **hem kesen hem de kesmeyen algoritma** olarak gerçekleştirilebilir

En Kısa İşletim Süresi Kalan Önce -2

Görev	Çalışma Süresi (msn)
p1	6
p2	8
p3	7
p4	3

Burada ortalama bekleme süresi (0 + 3 + 9 + 16) / 4 = 7.0 ms. olur



Öncelik Tabanlı -1

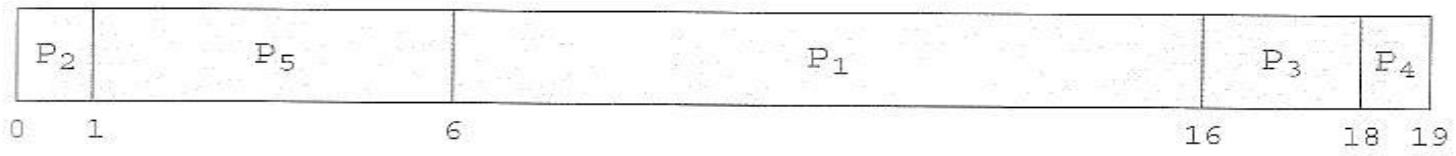
Bu algoritmada çalışan her bir göreve bir öncelik atanır ve önceliği yüksek olan görev diğerlerinden daha önce bitirilmeye çalışılır.

Öncelik tabanlı görev yönetim algoritması, **hem kesen hem de kesmeyen algoritma** olarak gerçekleştirilebilmektedir

Öncelik Tabanlı -2

Görev	Çalışma Süresi (msn)	Öncelik
p1	10	3
p2	1	1
p3	2	4
p4	1	5
p5	5	2

Burada ortalama bekleme süresi (0+1+6+16+18)/5= 8.2 msn. olarak gerçekleşir.



Zaman Dilimli (Round Robin) -1

Zaman dilimli görev yönetim algoritmasıyla, hazır görevler kuyruğunda bekleyen görevler, eşit uzunluktaki zaman dilimleri içinde ana işlem birimine, sırayla anahtarlanır.

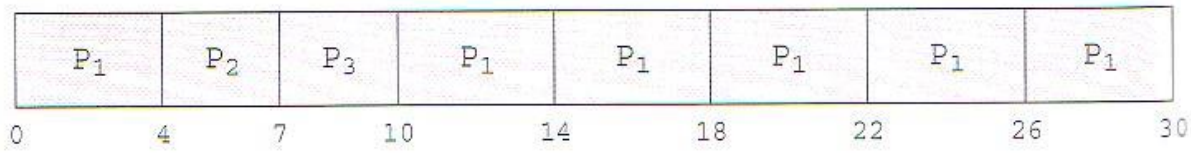
Örneğin, her 10 msn de bir gelen saat uyarılarıyla görev yönetici, çalışmakta olan görevi, hazır görevler kuyruğunun sonuna ekler. Kuyruk başındaki görevi de, kendisinden sonra çalışmak üzere MI ye anahtarlar.

Görevlerin işletimi, giriş/çıkış ve zaman uyumlama istemi gibi nedenlerle kendilerine ayrılan zaman dilimi dolmadan sonlanabilir.

Zaman dilimli görev yönetimi **kesen** bir algoritmadır.

Zaman Dilimli (Round Robin) - 2

Görev	Çalışma Süresi (msn)
p1	24
p2	3
p3	3



Yukarıdaki örnekte zaman dilimi 4 msn olarak seçilmiştir. Bu algoritmanın performansı seçilecek zaman dilimine oldukça bağlıdır.