

**İ.K.Ü. FEN-EDEBİYAT FAKÜLTESİ
MATEMATİK-BİLGİSAYAR BÖLÜMÜ**

**2010-2011 GÜZ YARIYILI "MC 78K İŞLETİM SİSTEMLERİ"
I.VİZE ÇÖZÜMLERİ**

1) Aşağıdaki kavramları kısaca açıklayınız.

- İzlek, iş parçacığı (*Thread*)
- Karşılıklı Dışlama (*Mutual Exclusion*)
- Semafor (*Semaphore*)
- İş Sıralama (*Job/CPU Scheduling*)

İzlek (*Thread*) : Bir prosesin sözde-paralel olarak çalışabilen iş parçacıklarından her biri. Bir prosesin bu şekilde "*mini proseslere*" parçalanmasının temel avantajı, I/O bağımlı bir iş parçacığının I/O işlem isteğinde bloke olup merkezi işlem birimini (CPU) aynı prosese ait başka bir izleğe bırakabilmesi ve bu sebeple sistem performansında oluşacak artıştır. Aynı prosesin izlekleri prosese ait olan ortak adres alanını paylaşırlar; bu şekilde aynı kaynaklara erişebilirler ve kolay haberleşebilirler.

Karşılıklı Dışlama (*Mutual Exclusion*) : Paylaşılan (ortak erişilen) bir kaynağı kullanan (veya kendi kritik bölgesinde olan) bir proses, diğer proseslerin o kaynağa erişimini (veya kritik bölgelerine girmesini) engellemeli; diğer bir ifadeyle yarış durumunun oluşmasını önlemek için diğer prosesleri "dışlamalıdır".

Semafor (*Semaphore*) : Kritik bölge kontrolü ve proses senkronizasyonu için kullanılan ve iki adet kesilemeyen işleme (*up (V)*, *down (P)*) sahip olan tamsayı bir değişkendir. **P** işlemi ile semaforun değeri sıfırdan büyükse bu değer bir azaltılır ve işleme devam edilir; semaforun değeri 0 ise proses bu semafor üzerinde bloke olur (*sleep*). **V** işlemi ile semaforun değeri 0'dan farklı ise (veya 0 olup da bu semafor üstünde bloke olmuş hiç proses yoksa) bu değer bir arttırılır; semaforun değeri 0 ise ve üzerinde bloke olmuş en azından bir proses varsa, bu proseslerden biri uyandırılır (*wakeup*), semaforun değeri arttırılmaz. Bu iki işlem atomiktir; kesilemez.

İş Sıralama (*Job/CPU Scheduling*) : Çalışmaya hazır durumda olup hazır kuyruğunda bekleyen proseslerin belirli bir algoritmaya göre seçilip merkezi işlem birimine sırayla atanmaları. İşletim sisteminin iş sıralama yapan modülüne iş planlayıcısı veya iş çizelgeleyicisi (*scheduler*) denir.

2) Yarış durumu (race condition) nedir, hangi durumlarda oluşur ? Aşağıda sembolik kod ile verilen iki prosesin (A ve B) çalışmasında hangi senaryo gerçekleşirse ortaya bir yarış durumu çıkar ? Örnek veriniz.

```
global int a = 0
```

```
// PROSES A : Seri port denetçisinden  
// kesme geldiğinde devreye girer  
gorev A()  
    a = a + 1  
    print "RX"  
gorev sonu
```

```
// PROSES B : Her saniyede bir  
// devreye girer  
gorev B()  
    if (a mod 2 = 0)  
        print a  
    end if  
gorev sonu
```

Yarış Durumu (Race Condition) : Ortak bir bellek alanına (veya ortak bir değişkene) okuma ve/veya yazma amaçlı erişen iki veya daha fazla sayıdaki prosesi göz önüne alalım. Paylaşılan ortak değişkenin son değeri, bu proseslerin merkezi işlem birimindeki çalışma sıralarının farklılıklarına göre değişiklik gösteriyorsa bu istenmeyen duruma **yarış durumu** denir.

Verilen örnekteki **a** değişkeni paylaşılan değişken olup, A prosesi bu değişkenin değerini 1 arttırmaktadır. B prosesi ise **a** değişkeninin değeri bir çift sayı olduğunda bu değeri ekrana çıkarmaktadır; dolayısıyla B prosesinin tek sayıları (ör:1,3,5...) ekrana çıkarma durumu söz konusu olmamalıdır.

Şimdi aşağıdaki senaryonun oluştuğunu varsayalım :

- İlk olarak **B** prosesi devreye girsin ve **a** değişkeninin değerini okusun (**a=0**). ($0 \bmod 2 = 0$) olduğu için bu değeri ekrana yazmak üzereyken seri port denetçisinden gelen bir kesme sebebiyle *print* komutunu çalıştırmadan önce kesilsin.
- A** prosesi devreye girer, ve **a=1** yapıp ekrana "RX" mesajını çıkarır.
- B** prosesi tekrar devreye girdiğinde kesildiği noktadan işlemine devam eder. **a** değişkeninin güncel değerini okur ve "*print a*" komutunu işleterek ekrana **1** değerini çıkarır (!!).

Problemin kaynağı paylaşılan **a** değişkenine erişim esnasında proseslerin kesilebilmesidir. (Örnek wikipedia sitesinden alınmıştır).

3) a) İçerik Anahtarlama (context switching) ne demektir, içerik anahtarlama sırasında işletim sistemi tarafından yapılan temel işlemler nelerdir ?

b) Round-robin iş sıralama algoritmasındaki **quantum** süresinin çok uzun veya çok kısa olmasını, içerik anahtarlama ve genel sistem performansı açısından karşılaştırınız.

a) İçerik Anahtarlama (Context Switching) : İşletim sisteminin merkezi işlem biriminde çalışmakta olan bir prosesin yerine başka bir prosesi alması sürecinde yapılan işlemlere denir. Çalışmakta olan proses normal koşullarla sonlanabilir veya çeşitli nedenlerden dolayı (örneğin; prosesin bir I/O isteğinde bulunması, daha yüksek öncelikli bir prosesin hazır kuyruğuna varması, prosese ayrılan sürenin dolması gibi nedenlerle) işletim sistemi tarafından kesilebilir. Kesilen prosesin kesilme anındaki durumu (CPU yazmaçları, program sayacı, ...) prosese ilişkin kontrol bloğuna yazılır (böylelikle bu proses tekrar devreye girdiğinde bu bilgiler geri yüklenerek prosesin kaldığı yerden devam etmesi sağlanır). Merkezi işlem biriminde çalıştırılmaya başlanacak olan yeni prosesin kontrol bloğundaki bilgiler ise sisteme yüklenir ve içerik anahtarlama işlemi tamamlanmış olur. Bu işlemler ek yüküdür (*overhead*); zira işletim sistemi bu süre zarfında yararlı bir iş yapmaz (kullanıcı prosesleri çalışmaz).

b) Round-robin iş sıralama algoritmasındaki **quantum** süresi çok kısa olduğunda prosesler daha sık aralıklarla kesilir, dolayısıyla çok fazla sayıda içerik anahtarlama yapılır. Böyle bir durumda genelde ihmal edilen içerik anahtarlama sürelerinin toplamı ihmal edilemeyecek bir orana yükselebilir. Bu ise kullanıcı prosesleri açısından sistem performansında bir düşüş yaşanmasına sebep olur (CPU işlem zamanının belirli bir yüzdesi sistem ek yükü olarak harcanır). Diğer yandan **quantum** süresi çok uzun tutulduğunda proseslerin bir quanta sürede sonlanma olasılığı artar, içerik anahtarlama sayısı azalır, böylelikle sistemin ek yükü ihmal edilir, sistem performansında artış sağlanır. Fakat quantum süresinin çok uzun olması round-robin algoritmasının FIFO algoritması gibi çalışması anlamına gelir; bu algoritmanın zayıf tarafı ise ortalama proses bekleme süresinin yüksek çıkabilmesidir. Bu sebeplerden dolayı round-robin algoritmasını kullanan işletim sistemlerinde **quantum** makul bir değerde seçilir.

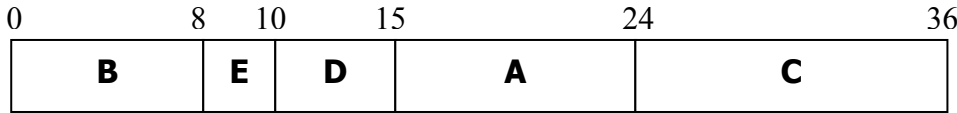
4) Bir işletim sisteminin hazır kuyruğunda aşağıdaki sırada gelen prosesler bulunmaktadır. Buna göre “**en kısa süreli iş ilk**” (*shortest job first*) ve quantum süresi **4 msn** olan “**çevrimsel sıralama**” (*round robin scheduling*) algoritmalarına göre oluşacak zaman diyagramlarını çizip her iki yöntem için proseslerin ortalama bekleme sürelerini (*average waiting time*) hesaplayınız.

<u>Proses</u>	<u>Kuyruğa Varış Zamanı</u>	<u>İşlem Süresi</u>
A	0	9
B	0	8
C	2	12
D	5	5
E	7	2

En Kısa Süreli İş İlk (SJF – Shortest Job First)

Bu algoritma *non-preemptive* yapıdadır; dolayısıyla kuyruğa ulaşan proseslerin çalışan prosesi kesme durumu söz konusu değildir.

Zaman diyagramı :



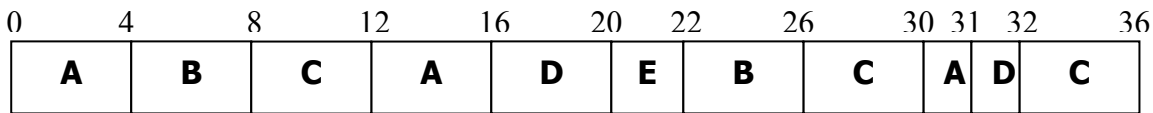
Bekleme Süreleri

$$\begin{array}{lcl}
 A \rightarrow 15 - 0 & = & 15 \text{ mSn} \\
 B \rightarrow 0 - 0 & = & 0 \text{ mSn} \\
 C \rightarrow 24 - 2 & = & 22 \text{ mSn} \\
 D \rightarrow 10 - 5 & = & 5 \text{ mSn} \\
 E \rightarrow 8 - 7 & = & 1 \text{ mSn}
 \end{array}
 \left. \vphantom{\begin{array}{lcl} A \rightarrow 15 - 0 \\ B \rightarrow 0 - 0 \\ C \rightarrow 24 - 2 \\ D \rightarrow 10 - 5 \\ E \rightarrow 8 - 7 \end{array}} \right\} \Rightarrow \text{Ortalama Proses Bekleme Süresi : } (15+0+22+5+1) / 5 = 43/5 = \mathbf{8.6 \text{ mSn}}$$

Çevrimsel Sıralama (RR – Round Robin)

quantum = 4 mSn

Zaman diyagramı :



Bekleme Süreleri

$$\begin{array}{lcl}
 A \rightarrow (12-4)+(30-16) & = & 22 \text{ mSn} \\
 B \rightarrow (4-0) + (22-8) & = & 18 \text{ mSn} \\
 C \rightarrow (8-2)+(26-12)+(32-30) & = & 22 \text{ mSn} \\
 D \rightarrow (16-5)+(31-20) & = & 22 \text{ mSn} \\
 E \rightarrow (20-7) & = & 13 \text{ mSn}
 \end{array}
 \left. \vphantom{\begin{array}{lcl} A \rightarrow (12-4)+(30-16) \\ B \rightarrow (4-0) + (22-8) \\ C \rightarrow (8-2)+(26-12)+(32-30) \\ D \rightarrow (16-5)+(31-20) \\ E \rightarrow (20-7) \end{array}} \right\} \Rightarrow \text{Ortalama Proses Bekleme Süresi : } (22+18+22+22+13) / 5 = 97/5 = \mathbf{19.4 \text{ mSn}}$$

5) Bir sistemdeki prosesler, sistemde bulundukları süre boyunca CPU ve Giriş/Çıkış çevrimleri yaparlar. Aşağıda 4 proses için bu çevrimlerin süresi verilmiştir. Buna göre "**kalan iş süresi en az olan ilk**" (*Shortest-Job-Remaining First*) algoritmasına göre oluşan zaman diyagramını oluşturunuz ve proseslerin sistemde ortalama dönme sürelerini (*average turnaround time*) hesaplayınız. (NOT: Adı geçen algorithma işletim sisteminin çalışma modu preemptive'dir).

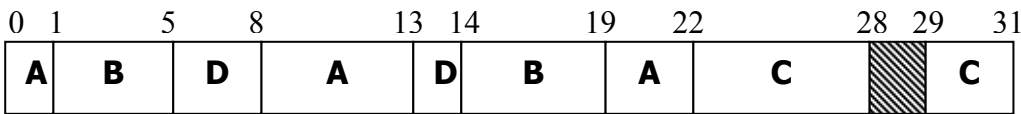
Proses	Sisteme Giriş Zamanı	CPU ve I/O İşlem Süreleri
A	0	6C-4I-3C
B	1	4C-2I-5C
C	1	6C-1I-2C
D	3	3C-5I-1C

Cözüm

Başlangıçta (0. mSn'de) kuyrukta sadece A prosesi olduğundan bu proses çalışmaya başlar. 1. mSn'de B ve C prosesleri hazır kuyruğuna varmaktadır; iş sıralama algoritması *preemptive* olduğundan ve B prosesinin CPU işlem zamanı (4mSn) çalışmakta olan A prosesinin kalan işlem zamanından (6-1=5mSn) daha kısa olduğundan işletim sistemi içerik anahtarlama yapar; yani 1. mSn'de B prosesi devreye girer. 3. mSn'de gelen D prosesi B'yi kesemeyecektir (zira 3. mSn'de B prosesinin 3-1=2mSn'lik işlem zamanı kalmıştır, oysa D prosesi 3 mSn'lik işlem zamanına ihtiyaç duymaktadır). B prosesi 5. mSn'de merkezi işlem birimini terk eder, 2 mSn'lik bir I/O işleminin ardından (yani 5+2=7.mSn'de) hazır kuyruğuna geri gelecektir. 5. mSn'de kuyrukta A, C ve D prosesleri bulunmaktadır. Bu üç proses arasında en kısa işlem zamanı olan D prosesi çalışmaya başlar, 8. mSn'de işlemini bitirir (7. mSn'de kuyruğa tekrar ulaşan B prosesi D'yi kesemez). D prosesi 8+5=13. mSn'de kuyruğa tekrar varacaktır. 8. mSn'de kuyrukta A, B ve C prosesleri vardır. En kısa işlem süresine sahip iki proses vardır : A ve B (5 mSn). Eşitlik durumunda eşitlik çözücü olarak FIFO algoritması kullanıldığından, daha uzun süredir kuyrukta olan A prosesi çalışır. 13. mSn'de A prosesinin işlemi biter; bu proses 13+4=17. mSn'de hazır kuyruğuna tekrar ulaşacaktır. 13. mSn'de kuyrukta B, C ve D prosesleri vardır; en kısa işlem zamanına sahip D prosesi çalışır. 14. mSn'de D prosesi sonlanır; B prosesi devreye girer. B prosesinin sonlanması 19. mSn'yi bulacaktır; bu sırada 17. mSn'de kuyruğa varan A prosesi B'yi kesemez (3 > 2). 19. mSn'de kuyrukta A ve C prosesleri kalmıştır. A prosesinin işlem süresi daha kısa olduğundan (3 < 6) bu proses çalışır ve 22. mSn'de sonlanır. Sistemde kalan tek proses C prosesidir ve henüz hiç çalışmamıştır. Bunun nedeni işlem süresinin o ana kadar yarıştığı diğer proseslere göre hep yüksek kalmasıdır. 22. mSn'de C prosesi çalışmaya başlar, 28. mSn'de merkezi işlem birimini terk eder; 1 mSn'lik I/O işlemi için bloke olur. Bu sırada hazır kuyruğunda hiçbir proses olmadığından CPU boşta kalır. 29. mSn'de kuyruğa tekrar ulaşan C prosesi son CPU işlemini de tamamlayıp 31. mSn'de sonlanır.

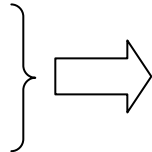
Sistemin zaman diyagramı aşağıda verilmiştir.

Zaman Diyagramı :



Proseslerin Dönme Süreleri

A → (22-0) = 22 mSn
 B → (19-1) = 18 mSn
 C → (31-1) = 30 mSn
 D → (14-3) = 11 mSn



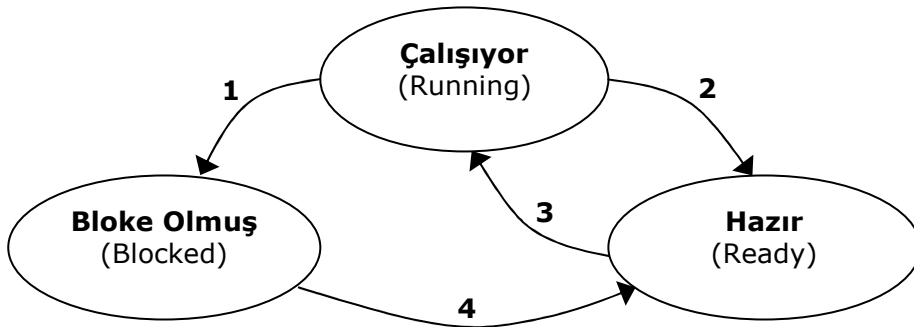
Ortalama Proses Dönme Süresi :
 $(22+18+30+11) / 4 = 81/4 = \mathbf{20.25 \text{ mSn}}$

**İ.K.Ü. FEN-EDEBİYAT FAKÜLTESİ
MATEMATİK-BİLGİSAYAR BÖLÜMÜ**

**2010-2011 GÜZ YARIYILI "MC 78K İŞLETİM SİSTEMLERİ"
II.VİZE ÇÖZÜMLERİ**

- 1) a)** Proseslere ilişkin **yaşam çevrimi** diyagramını çiziniz ve durumlar arası geçişlerin hangi olaylarla sağlandığını belirtiniz.
- b)** **İkili semafor** (*binary semaphore*) ve **sayaç semafor** (*counting semaphore*) kavramlarını açıklayınız ve bu semaforların ne amaçlarla kullanıldığını yazınız.

a)



1 : Çalışıyor durumda olan proses, o an için işlemcinin atandığı procestir; dolayısıyla tek işlemcili sistemlerde çalışıyor durumda en fazla bir proses olabilir. Bu proses çalışmasına devam edebilmek için bir giriş/çıkış işlemine (örneğin; *kullanıcı girişi, disk erişimi*) ihtiyaç duyduğunda veya proses kendisini bloke eden bir sistem çağrısı (örneğin : *sleep*) gerçekleştirdiğinde çalışıyor durumdan bloke duruma geçer.

2 : İşletim sistemi çalışmakta olan prosesi belirli nedenlerden dolayı kesebilir (örneğin; *prosese ayrılan çalışma süresinin dolması, daha yüksek öncelikli bir prosesin hazır kuyruğuna varması*). Böyle bir durumda proses, çizelgeleyici modül (*scheduler*) tarafından çalışıyor durumdan hazır durumuna alınır.

3 : Çizelgeleyici işlemciye yeni bir proses atayacağı zaman hazır durumdaki proseslerden birini belirli bir algoritmaya göre (örneğin; *round-robin, öncelikli sıralama*) işlemcide çalışmak üzere seçer.

4 : Bloke durumdaki proses, bloke olmasına neden olan dış işlemi tamamladıktan sonra (veya *sleep* çağrısı ile bloke olan bir proses, *wakeup* çağrısı ile uyandırıldığı zaman) hazır durumuna geçer.

b) İkili Semafor (Binary Semaphore) : Sadece 0 ve 1 değerlerini alabilen semafor. İkili semafor kritik bölge giriş/çıkış kontrollerinin kilidi olarak (karşılıklı dışlamayı gerçekleyebilmek amacıyla) kullanılabilir. Kritik bölgesine girmek isteyen bir proses, ikili semafor üzerinde *down* işlemi yaparak semaforun değerini 0'a çeker. Bu proses kritik bölgesindeyken, başka prosesler kendi kritik bölgelerine giremeyeceklerdir; zira bu prosesler ikili semafor üzerinde *down* işlemi yaptıklarında bloke olurlar. Kritik bölgesine girmiş olan proses, kritik bölgeden çıkarken ikili semafor üzerinde *up* işlemini gerçekleştirir; eğer bu semafor üzerinde bloke olmuş prosesler varsa biri uyandırılır; yoksa semaforun değeri 1 yapılır.

Sayaç Semafor (Counting Semaphore) : İkili semaforuna aksine 1' den büyük değerler alabilen semafor. Genellikle bir kaynağa erişim kontrolünde kullanılır; semaforun başlangıç değeri boştaki (kullanılabilir) kaynak sayısı olacak şekilde ayarlanır. Kaynağa erişen her proses sayaç semafor üzerinde *down* işlemi yaparak semaforun değerini 1 azaltır; kaynak kullanımını sonlandıran her proses ise sayaç semafor üzerinde *up* işlemi yaparak semaforun değerini 1 artırır.

2) İş sıralama yöntemi olarak *çok seviyeli kuyruk* modeli kullanılan bir sistemde yer alan 3 kuyruğa ilişkin algoritmalar aşağıda verilmiştir. Buna göre yine aşağıda verilen tablodaki proseslerin sisteme dahil olmaları durumunda oluşacak zaman diyagramını çizerek, proseslere ilişkin ortalama bekleme süresini hesaplayınız.

1. **kuyruk** : En kısa iş kalan ilk (*Shortest Job Remaning First*)
2. **kuyruk** : Çevrimsel Sıralamalı (*Round Robin, quantum = 4 mSn*)
3. **kuyruk** : FIFO

Prosesler	Varış Zamanı	İşlem Süresi
A	0	8
B	3	3
C	7	6
D	12	2
E	19	6

Sistem Modeli :

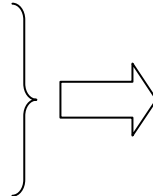


Zaman diyagramı :

0	3	6	10	12	14	18	19	25
A	B	A	C	D	C	A	E	

Bekleme Süreleri :

$$\begin{array}{ll}
 A \rightarrow (0-0) + (6-3) + (18-10) & = 11 \text{ mSn} \\
 B \rightarrow (3-3) & = 0 \text{ mSn} \\
 C \rightarrow (10-7) + (14-12) & = 5 \text{ mSn} \\
 D \rightarrow (12-12) & = 0 \text{ mSn} \\
 E \rightarrow (19-19) & = 0 \text{ mSn}
 \end{array}$$



Ortalama Proses Bekleme Süresi :
 $(11+0+5+0+0) / 5 = 16/5 = \mathbf{3.2 \text{ mSn}}$

3) Bir sistemdeki ana belleğin belirli bir andaki durumu aşağıdaki gibidir :

25K (Boş) - 30K (Dolu-A) - **30K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

a) Aşağıdaki sıraya uygun olarak ana belleğe yerleşmek isteyen prosesleri; *first fit*, *best fit* ve *worst fit* algoritmalarına göre yerleştiriniz; dış parçalanma durumlarını inceleyiniz.

P0 → 15K P1 → 20K P2 → 5K P3 → 25K

b) Sisteme bir **P4** prosesi dahil olduğunda, *next fit* algoritmasına göre bir dış parçalanma oluştuğuna göre bu P4 prosenin minimum boyutunun ne olabileceğini bulunuz.

c) Sistemde daha önceden var olan 3 prosten (A,B,C) sadece **B prosesi** çalışma anında bellekte yer değiştirebildiğine göre ve b şıkkinda oluşan dış parçalanmanın bellek sıkıştırma (*compaction*) yöntemiyle bile çözülemediği bilindiğine göre **P4** prosenin minimum boyu ne olabilir?

a) İlk Uygun Boşluk (First Fit) :

P0 → 15K (Dolu-P0)-10K (Boş) - 30K (Dolu-A) - **30K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P1 → 15K (Dolu-P0)-10K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P2 → 15K (Dolu-P0)- 5K (Dolu-P2) - 5K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P3 → 15K (Dolu-P0)- 5K (Dolu-P2) - 5K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **25K (Dolu-P3) - 10K (Boş)** - 10K (Dolu-C)

En Uygun Boşluk (Best Fit) :

P0 → 15K (Dolu-P0)-10K (Boş) - 30K (Dolu-A) - **30K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P1 → 15K (Dolu-P0)-10K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P2 → 15K (Dolu-P0)- 5K (Dolu-P2) - 5K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P3 → 15K (Dolu-P0)- 5K (Dolu-P2) - 5K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **25K (Dolu-P3) - 10K (Boş)** - 10K (Dolu-C)

En Büyük Boşluk (Worst Fit) :

P0 → 25K (Boş) - 30K (Dolu-A) - **30K (Boş)** - 5K (Dolu-B) - **15K (Dolu-P0)** - **20K (Boş)** - 10K (Dolu-C)

P1 → 25K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **15K (Dolu-P0)** - **20K (Boş)** - 10K (Dolu-C)

P2 → 5K (Dolu-P2) - 20K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **15K (Dolu-P0) - 20K (Boş)** - 10K (Dolu-C)

P3 → 25K boyutundaki bu proses hiçbir boşluğa yerleşemez. **Dış Parçalanma !**

b) **Next-fit** algoritmasına göre proseslerin boşluklara ne şekilde yerleştiklerini bulmamız gerekir.

Sıradaki Uygun Boşluk (Next Fit) :

P0 → 15K (Dolu-P0) - 10K (Boş) - 30K (Dolu-A) - **30K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P1 → 15K (Dolu-P0) - 10K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **10K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P2 → 15K (Dolu-P0) - 10K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **5K (Dolu-P2) - 5K (Boş)** - 5K (Dolu-B) - **35K (Boş)** - 10K (Dolu-C)

P3 → 15K (Dolu-P0) - 10K (Boş) - 30K (Dolu-A) - **20K (Dolu-P1)** - **5K (Dolu-P2) - 5K (Boş)** - 5K (Dolu-B) - **25K (Dolu-P3) - 10K (Boş)** - 10K (Dolu-C)

Bu son durumda oluşan boşlukların en büyüğü 10K boyutundadır. Bir dış parçalanma oluşması için yeni gelecek olan P4 prosenin boyutunun 10K'dan büyük olması gerekmektedir. Bunu sağlayan minimum değer 10K+1 byte = **10241** byte'dır.

c) Bir önceki şıktaki son durumda B prosesini yer değiştirdiğimizde (başka bir boşluğa taşıdığımızda) burada oluşan yeni boşluk 10K olacaktır (5K= B prosenin boyu + 5K= B prosenin solundaki boşluk). Dolayısıyla b şıkındaki çözüm geçerliliğini korur.

4) Bir sistemde bellek yönetimi için **sayfalama** (paging) yöntemi kullanılmaktadır. Ana belleğin boyunun **2MB**, sayfa boyunun ise **4KB** olduğu bilinmektedir. **71.300 byte** uzunluğundaki bir **P** prosesinin sayfa tablosunun ilk 3 satırında sırası ile 80,26,11 sayıları bulunmaktadır. Buna göre;

- a) Görüntü ve fiziksel adresin ve alt bölümlerinin boyutlarını bulunuz.
- b) **4095.** ve **4096.** görüntü adreslerin ana bellekteki konumları arasında **kaç byte fark** olduğunu bulunuz.
- c) P prosesinin boyutunda **en az kaç adet proses** tamamen belleğe yerleşirse, oluşan toplam iç parçalanma bir P prosesi boyutlarına ulaşır ?

a) Sayfa boyu = 4KB = 2^{12} byte
Ana bellek boyu = 2MB = 2^{21} byte
Ana bellekte toplam $2^{21} / 2^{12} = 2^9$ çerçeve vardır.
Bu bilgiler ışığında fiziksel adresin toplam **21 bit** olduğunu; bu 21 bitin ilk 9 bitinin çerçeve, geri kalan 12 bitinin ise sayfa içi adres olduğunu bulmuş oluruz.

Proses boyu 71300 byte olarak verilmiş. Bu proses toplam $71300/4096 = 17.40$ sayfalık yer kaplar, yani prosesin 18 sayfası vardır. [0-17] arasındaki sayıları **5** bit ile gösterebiliriz. Bu bilgiler ışığında prosese ilişkin görüntü adresin ilk 5 biti sayfa numarası, bundan sonraki 12 biti ise sayfa içi adres bilgisinden oluşur. Görüntü adresin toplam uzunluğu $5+12=17$ bittir. Bu 17 bit sonucuna daha kolay bir yoldan da ulaşabiliriz : 71300 sayısı minimum **17 bit** ile gösterilebilir.

f (9 bit)	d (12 bit)
-----------	------------

Fiziksel Adres

p (5 bit)	d (12 bit)
-----------	------------

Görüntü Adres

b) Sayfa boyu 4096 byte olduğu için **4095.** görüntü adres, 0. sayfanın 4095. (yani son) adresidir. **4096.** adres ise 1. sayfanın 0. (yani ilk) adresidir.

0. sayfanın ana bellekte 80. çerçevede olduğu biliniyor. Buna göre 4095. adres $80 \times 4096 + 4095 = 331775$ fiziksel adrestedir.

1. sayfanın ana bellekte 26. çerçevede olduğu biliniyor. Buna göre 4096. adres $26 \times 4096 + 0 = 106496$ fiziksel adrestedir.

Sonuç : 4095 ve 4096 görüntü adresleri, proses adres uzayında ardışıl adreslerdir. Fakat bu adreslerin farklı sayfalarda olmasından ve bu sayfaların ana bellekte ardışıl olmayan çerçevelerde olmasından dolayı bu iki adresin ana bellekteki konumları arasındaki fark $331775 - 106496 = 225279$ byte'dır.

Diğer bir çözüm yolu : Sorulan görüntü adreslerden biri 26. çerçevenin ilk adresi, diğeri 80. çerçevenin son adresidir. 81. çerçevenin ilk adresini baz alırsak, bu iki adres arasında tam olarak $81 - 26 = 55$ çerçeve fark vardır. 55 çerçeve $\rightarrow 55 \times 4096 = 225280$ byte uzunluğundadır. Bu değerden eklediğimiz 1 byte'ı düşersek **225279** sonucuna ulaşırız.

c) P prosesi (a) şıkkında da bulduğumuz gibi 18 sayfaya bölünmüştür. Son (18.) sayfanın tamamı kullanılmamakta olup, kullanılmayan kısım iç parçalanma sebebidir :

$(18 \times 4096) - 71300 = 2428$ byte iç parçalanma vardır.

P prosesi boyutundaki her proses ana bellekte 2428 byte'lık bir iç parçalanma oluşturacaktır.

$71300 / 2428 = 29.36$

Sonuç olarak, P prosesi boyutlarında **en az 30 proses** ana belleğe yerleştiğinde oluşan toplam iç parçalanma boyutu bir P prosesi boyutuna ulaşır.

5) Sayfa boyu **2KB** ve ana belleği **4 çerçeveden** oluşan bir sistemde sayfa hatası oluşması durumunda yapılan sabit disk+bellek işlemi **300 nSn**, sayfa hatası olmaması durumunda yapılan bellek erişimi ise **5 nSn**'dir. Böyle bir sistemde çalışmakta olan **16KB** uzunluğundaki bir **P** prosesi için belirli bir zaman diliminde şu görüntü adreslere referanslar üretildiği bilinmektedir:

1060, 4096, 15200, 8340, 12010, 8192, 8191, 14500, 8842, 8844, 5000, 12750, 4100, 10439, **X**.

Bu sistemde sayfa yerleşim algoritması olarak (LRU-*Least Recently Used*) yöntemi kullanıldığında yukarıdaki adresler için ortalama bellek erişim süresi **182 nSn** olduğuna göre; **X** adresinin alabileceği değer aralıklarını bulunuz.

Cözüm :

Öncelikle verilen görüntü adreslerin prosesin hangi sayfalarında olduklarını bulmak gerekir. Sayfa boyu 2KB (2048 byte) olarak verildiği için her adresin hangi 2048'lik dilimin içine düştüğünü bulmak basit bir bölme işlemi ile bulunabilir.

Adres:1060, 4096, 15200, 8340, 12010, 8192, 8191, 14500, 8842, 8844, 5000, 12750, 4100, 10439, **X**.
Sayfa: 0 , 2 , 7 , 4 , 5 , 4 , 3 , 7 , 4 , 4 , 2 , 6 , 2 , 5 , X.

Ana belleğin 4 çerçeveden oluştuğu bilgisi verilmiş. Sayfa referans dizisi de elimizde olduğuna göre LRU algoritmasına göre sayfa yerleşimini çizebiliriz :

0	2	7	4	5	4	3	7	4	4	2	6	2	5	X
0	0	0	0	5		5				2	2		2	?
	2	2	2	2		3				3	6		6	
		7	7	7		7				7	7		5	
		4	4	4		4				4	4		4	

Toplam 15 bellek referansından 9 tanesi sayfa hatası üretmiş, 5 tanesinde ise sayfa hatası yok. Son bellek referansı (X) hakkında ise henüz bir bilgimiz yok.

Ortalama bellek erişim süresi **182 nSn** olarak verilmiş. Bu durumda toplam bellek erişim süresi : $182 \times 15 = \mathbf{2730}$ nSn'dir.

Bilinen 9 sayfa hatası $\rightarrow 9 \times 300 = 2700$ nSn

Bilinen 5 normal erişim $\rightarrow 5 \times 5 = 25$ nSn

Bilinen toplam erişim süresi : 2725 nSn

Bu sonuçtan hareketle en son bellek referansının $2730 - 2725 = \mathbf{5}$ nSn'lik bir sürede gerçekleştiğini; dolayısıyla **sayfa hatası oluşturmadığını** bulabiliriz. Yani X adresi o an için ana bellekte bulunan 2,6,5 veya 4 numaralı sayfalardan birinin içindedir. Bu adresin alabileceği değer aralıkları :

2. sayfa $\rightarrow [4096-6143]$

4. sayfa $\rightarrow [8192-10239]$

5. sayfa $\rightarrow [10240-12287]$

6. sayfa $\rightarrow [12288-14335]$

şeklindedir.