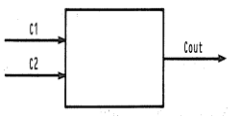
OCCAM
(II)

ALT PROSESİ

- ALT prosesi seçeneklerinden sadece bir tanesinin yürütülmesine izin verir.
- Örnek: C1 ve C2 ‘den gelen veriler Cout’a yönlendirilecekler.




```
INT X:
WHILE TRUE
  ALT
    C1 ? X
      Cout ! X
    C2 ? X
      Cout ! X
```

1. seçenek

2. seçenek

- PAR kullanılamaz çünkü Cout kanalı iki paralel proses içinde yer alamaz. ALT prosesi esnek bir çözüm getirir.

ALT PROSESİ

- Genel:

```
ALT
  G1
    P1
  G2
    P2
  .
  .
  .
  Gn
    Pn
```

Gi :Koruma (guard)

Pi :Proses

guard = input | Boolean & input | Boolean & SKIP

- Koruma (guard) en basit haliyle bir input işlemidir; hazır yazılmış bir veri varsa koruma hazır sayılır. Önce giriş işlemi yapılır, daha sonra da bu korumaya ilişkin proses yürütülür.
- Koruma bir lojik ifade ve onu izleyen input işlemi veya sadece bir lojik ifade (&SKIP)'den oluşabilir. (Sadece lojik ifade meşgul bekleme yaratabilir!!)
- Hazır bir koruma bulunamaz ise ALT bloke olur.
- Birden fazla hazır koruma var ise rastgele bir tanesi seçilir.

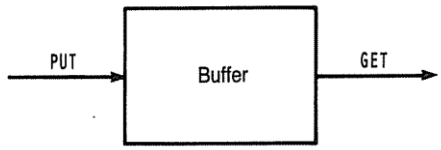
ALT PROSESİ

- Koruma için örnekler:
  - chan?X
  - A > B & chan?X
  - A < B & SKIP
  - Time AFTER T -- Time Timer tipinden kanal, gerçek zaman >= T olunca koruma hazır olur
  -

```
TIMER Clock:
SEQ
  Clock ? time
  ALT
    ch ? Some.Variable
      -- normal action
    Clock ? AFTER time PLUS 1000
      -- error recovery action
```

1000 zaman birimi süresince ch kanalından bilgi bekle, bu süre içinde input gerçekleşmez ise ikinci koruma doğrulanacaktır, hata rutinini yürüt

ÜRETİCİ-TÜKETİCİ PROBLEMİ




```
CHAN OF INT PUT, GET:
PAR
  VAL INT Buf.Size IS 32:
  INT TOP, BASE, CONTENTS:
  [Buf.Size]INT BUFFER:
```

ÜRETİCİ-TÜKETİCİ PROBLEMİ

```
SEQ
  CONTENTS := 0
  TOP := 0
  BASE := 0
  WHILE TRUE
    ALT
      CONTENTS < Buf.Size & PUT ? BUFFER [TOP]
        SEQ
          CONTENTS := CONTENTS + 1
          TOP := (TOP + 1) REM Buf.Size
      CONTENTS > 0 & GET ! Buffer [BASE]   -- HATA: KORUMA İÇİNDE OUTPUT
        SEQ
          CONTENTS := CONTENTS - 1
          BASE := (BASE + 1) REM Buf.Size
```

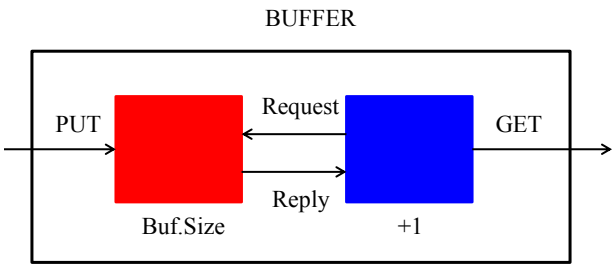
ÜRETİCİ-TÜKETİCİ PROBLEMİ

```
      CONTENTS > 0 & Request ? Any
        SEQ
          Reply ! BUFFER[BASE]
          CONTENTS := CONTENTS - 1
          BASE := (BASE + 1) REM Buf.Size
INT Temp:   -- single buffer process
VAL INT Any IS 0:   -- dummy value
WHILE TRUE
  SEQ
    Request ! Any
    Reply ? Temp
    GET ! Temp
```

ÜRETİCİ-TÜKETİCİ PROBLEMİ

```
CHAN OF INT PUT, GET, Request, Reply:
PAR
  VAL INT Buf.Size IS 32:
  INT TOP, BASE, CONTENTS:
  [Buf.Size]INT BUFFER:
  SEQ
    CONTENTS := 0
    TOP := 0
    BASE := 0
    INT Any:
    WHILE TRUE
      ALT
        CONTENTS < Buf.Size & PUT ? BUFFER [TOP]
          SEQ
            CONTENTS := CONTENTS + 1
            TOP := (TOP + 1) REM Buf.Size
        CONTENTS > 0 & Request ? Any
          SEQ
            Reply ! BUFFER[BASE]
            CONTENTS := CONTENTS - 1
            BASE := (BASE + 1) REM Buf.Size
  INT Temp:   -- single buffer process
  VAL INT Any IS 0:   -- dummy value
  WHILE TRUE
    SEQ
      Request ! Any
      Reply ? Temp
      GET ! Temp
```

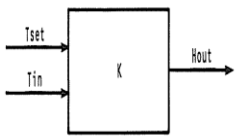
ÜRETİCİ-TÜKETİCİ PROBLEMİ



- Üretici: PUT!x
- Tüketici: GET?y
- Tampon alan kapasitesi: Buf.Size +1

ALT: ÖRNEK

- Dış ortamdan okunan bir veri olması istenen değer ile karşılaş-tırılacak, duruma uygun bir kontrol işareti üretilecek.

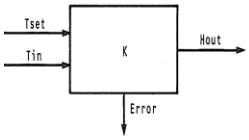


Tset: olması gereken ısı değeri
Tin: okunan ısı değeri
Hout: kontrol işareti (ısı + / ısı-)

```
SEQ
  SET.VALUE ? Tset
  WHILE TRUE
    ALT
      SET.VALUE ? Tset
        SKIP
      Treading ? Tin
        H.VALUE ! K*(Tset - Tin)
```

ALT: ÖRNEK

- Dış ortamdan veri okumada bir sorun olması durumunda hata mesajı üret.



Tin kanalından veri gelmez ise TimeOut aralıklarıyla Error kanalına hata mesajları gönderilecektir.

```
TIMER clock:
VAL INT Any IS 0:
SEQ
  clock ? Time
  SET.VALUE ? Tset
  WHILE TRUE
    ALT
      SET.VALUE ? Tset
        SKIP
      Treading ? Tin
        SEQ
          clock ? Time
          H.VALUE ! K*(Tset - Tin)
      clock ? AFTER Time PLUS TimeOut
        SEQ
          clock ? Time
          Error.Channel ! Any
```

ASENKRON HABERLEŞME

- Denetim görevi gören prosesler hizmet verdikleri giriş işaretlerine her zaman yanıt verebilir durumda olmalıdırlar.
- Occam'da haberleşme senkron olduğu için, denetçi prosesin bir çıkış işlemi üzerinde bloke olması mümkündür.
- Örnek: H.VALUE kanalından bilgi okunana kadar proses bekleyecek ve bu arada gelen yeni Tin değerlerine yanıt gecikecektir.

Denetçi proses:

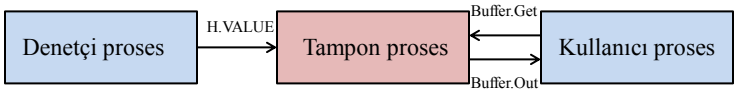
```
Treading ? Tin
  SEQ
    clock ? Time
    H.VALUE ! K*(Tset - Tin)
```

Kullanıcı proses:

```
H.VALUE ? New.Setting
```

ASENKRON HABERLEŞME

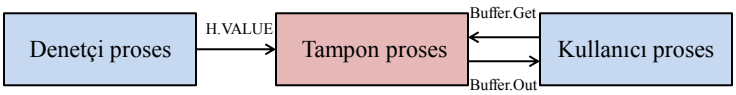
- Çözüm: Asenkron haberleşme modelini oluşturmak için denetçi ile kullanıcı proses arasına bir tampon proses eklemek gerekir.
- Tampon proses denetçinin ürettiği bilgileri okur ve kullanıcı hazır olduğunda, bunların en güncel olanını aktarır .



Denetçi proses yine aynı kodu yürütür:

```
Treading ? Tin
  SEQ
    clock ? Time
    H.VALUE ! K*(Tset – Tin)
```

ASENKRON HABERLEŞME



Tampon proses:

```
BOOL New.Value:
INT Buffer.Value:
SEQ
  H.VALUE ? Buffer.Value
  New.Value := TRUE
  INT Any:
  WHILE TRUE
    ALT
      H.VALUE ? Buffer.Value
        New.Value := TRUE
      New.Value & Buffer.Get ? Any
        SEQ
          Buffer.Out ! Buffer.Value
          New.Value := FALSE
```

Kullanıcı proses:

```
VAL Any IS 0:
SEQ
  Buffer.Get ! Any
  Buffer.Out ? New.Setting
```

ALT TÜRETİCİSİ

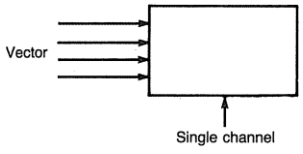
- Bir çok kanaldan istekler kabul edip, her birine benzer yanıt üreten bir sunucu prosesi kodlamak için türetilmiş ALT kullanılabilir. Diğer proses türeticilerine benzer yazımı vardır.
- Giriş kabul edilecek kanallar bir kanal dizisi olarak tanımlanır.
- Türetici indisi aynı zamanda kanal dizisi indisiolarak kullanılır.
- Örnek:

```
VAL INT Max IS 32:
[Max]CHAN OF INT Request:
PAR
  WHILE TRUE
    ALT I = 0 FOR Max
      Request[I] ? temp
        -- some action
```

```
PAR
  WHILE TRUE
    ALT
      Request[0] ? temp
        -- some action
      Request[1] ? temp
        -- same action
      Request[2] ? temp
        -- same action
      Request[3] ? temp
        -- same action
      .
      .
      .
      Request[31] ? temp
        -- same action
```

ALT TÜRETİCİSİ

- Örnek: bir kanal dizisi ile tek bir kanalı ALT prosesi içinde birleştirme.




```
ALT
  ALT I = 1 FOR Max
    Request[I] ? t1
      -- some action
    SingleChannel ? t2
      -- some other action
```

•ALT prosesinin seçeneklerinden birinin bir türetilmiş ALT prosesi olmasına izin verilir (aynı durum IF için de geçerlidir.)
•Tüm seçenekler, iç içe de olsalar, eşit düzeydedirler.

PRI ALT

- ALT çalışma düzeni, hazır korumalar içinden rastgele birinin seçimini öngörür.
- Bazı durumlarda bazı seçeneklerin öncelikle seçilmesi gerekir.
-

```
PRI ALT
  G1
    P1
  G2
    P2
  .
  .
  .
  Gn
    Pn
```

PRI ALT : hazır durumdaki birden fazla seçenek içinden, metinde önde yer alan öncelikle seçilir.
Örnek:Yeni bir Tset değeri hazır ise, bu değer öncelikle okunacaktır.

```
WHILE TRUE
  PRI ALT
    SET.VALUE ? Tset
      SKIP
    Treading ? Tin
      H.VALUE ! K * (Tset - Tin)
```

PRI ALT

- Örnek:

```
PRI ALT
  A > 0 & ch ? X
    -- action
  B > 0 & ch ? Y
    -- different action
```

Durum 1: Lojik ifadelerden sadece biri doğru ve ch kanalına bilgi yazılmış ise, o seçenek seçilir.
Durum 2: Lojik ifadelerden her ikisi de doğru ve ch kanalına bilgi yazılmış ise, ilk seçenek seçilir.

Durum 3: Lojik ifadelerden her ikisi de doğru ancak ch kanalına yazılmış bilgi yok ise PRI ALT prosesi askıya alınır ve ch kanalına bir bilgi yazılana kadar bekletilir. Kanala bilgi yazılır yazılmaz her iki seçenek de hazır olacaktır. Bu durumda hangi seçeneğin seçileceği belirsizdir (öncelik özelliği kaybolur).

PRI ALT

- DİKKAT: korumalarda yer alan lojik ifadelerin tümünün yanlış olması halinde ALT prosesi STOP'a düşer ve durur.
- Koruma sadece giriş prosesi içeriyor ise sorun yok:

  ch?X eşdeğerdir TRUE & ch?X
- Bu durumu engellemek için her zaman açık bir koruma bulundurulmalı → TRUE & SKIP
- Ancak bu çözüm de meşgul beklemeye neden olabilir!!!!!
- Örnek: Bir proses kanala artan sırada değerler gönderir.

```
SEQ
  n := 0
  WHILE TRUE
    SEQ
      out ! n
      n := n + 1
```

PRI ALT

- Örnek: aynı proses belirsiz zamanlarda "in" kanalından n'i sıfırlayan bir mesaj alacaktır. Her çevrimde bu mesajın bulunup bulunmadığı kontrol edilmeli, ancak mesaj yok ise işlem sürdürülmelidir.

```
SEQ
  n := 0
  WHILE TRUE
    PRI ALT
      in ? n
        SKIP
      TRUE & SKIP
        SEQ
          out ! n
          n := n + 1
```

OCCAM-TİP TANIMLARI

- Programming in OCCAM 2-Alan Burns kitabından aşağıdaki bölümleri okuyun.
  - Chapter 5: Data Types
  - Chapter 6: Channel Protocols

OCCAM PROSEDÜRLERİ

- Occam prosedürü: Parametrelere sahip olabilen ve bir ismi olan proses.
- Tanım:

```
PROC name ([,formal])
  body
:
```

  - name: prosedüre verilen ad
  - formal: parametre listesi (boş da olabilir)
  - body: proses kodu
  - ":" PROC anahtar sözcüğünün P'si ile aynı kolonda yer almalıdır.

OCCAM PROSEDÜRLERİ

- Örnek: Giriş parametresi olarak aldığı değerleri büyükten küçüğe sıralayan prosedür.

```
PROC ARRANGE (INT High,Low)
INT Temp:
IF
  High < Low
    SEQ
      Temp := High
      High := Low
      Low := Temp
  TRUE
    SKIP
:
```

```
INT Number1, Number2:
SEQ
  keyboard ? Number1
  keyboard ? Number2
  ARRANGE (Number1,Number2)
  screen ! Number1
  screen ! Number2
```

PROSEDÜR PARAMETRELERİ

- Tüm formel parametrelerin tipleri tanımlı olmalıdır.
- Parametreler adres bilgisi aktarılarak geçirilirler (by reference)
- Değer aktarılması istenirse VAL anahtar sözcüğü kullanılır
- Örnek:

```
PROC Maximum (VAL INT A, B, INT Max)
  IF
    A > B
      Max := A
    TRUE
      Max := B
:
```

PROSEDÜR PARAMETRELERİ

- Diziler de parametre olarak aktarılabilirler.
- Dizi boyutu belirtilmez; [] işareti eşleştirme için yeterli olur
- SIZE operatörü dizinin boyutunu belirler, böylece genel amaçlı prosedürler yazılabilir.
- Örnek: herhangi bir dizinin ortalamasını bulan prosedür.

```
PROC Average ([]REAL32 Data, REAL32 Res)
  SEQ
    Res := 0.0;
    SEQ i = 0 FOR SIZE Data
      Res := Res + Data[i]
    Res := Res/(REAL32 ROUND (SIZE Data))
:
```

PROSEDÜR PARAMETRELERİ

- Kanallar ve kanal dizileri de parametre olarak aktarılabilirler.
- Örnek: Bir kanal dizisinden gelen verileri tek bir kanala yönlendiren prosedür.

```
PROC Concentrator ([]CHAN OF BYTE IN, CHAN OF BYTE OUT)
  BYTE Element:
  WHILE TRUE
    ALT I = 0 FOR SIZE IN
      IN[I] ? Element
        OUT ! Element
:
```

OCCAM PROSEDÜRLERİ

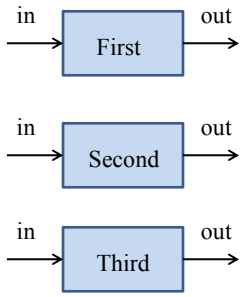
Prosedürlerin kullanım yerleri:

1. Bir SEQ prosesi içinde, bilinen altprogram çağrısı şeklinde. Değişken parametreler alırlar (örnek: PROC Average).
2. Bir PAR prosesi içinde, paralel çalışan bir proses olarak. Bu durumda kanallar da parametre olarak yer alırlar (örnek: PROC Concentrator).

   Bir PAR prosesi içinde yer alan prosesler kanallar üzerinden haberleşirler. Genellikle bu prosesler prosedür olarak yazılır ve haberleştikleri kanallar parametre olarak aktarılır.

OCCAM PROSEDÜRLERİ

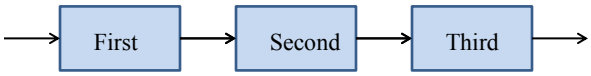
Örnek: Tuş takımından alınan değerleri ekrana taşıyan üç proses




```
PROC First(CHAN OF INT in, CHAN OF BYTE out)
  SKIP
:
PROC Second(CHAN OF BYTE in,out)
  SKIP
:
PROC Third(CHAN OF BYTE in, CHAN OF INT out)
  SKIP
```

OCCAM PROSEDÜRLERİ

- Bu üç proses bir boruhattı şeklinde çalışacaklar (pipeline)



- Bu durumda, First prosesinin çıkış yaptığı kanal ile Second prosesinin giriş yaptığı kanal aynı kanal olmak durumundadır.
- Benzer bir durum Second ve Third prosesleri için de geçerlidir.

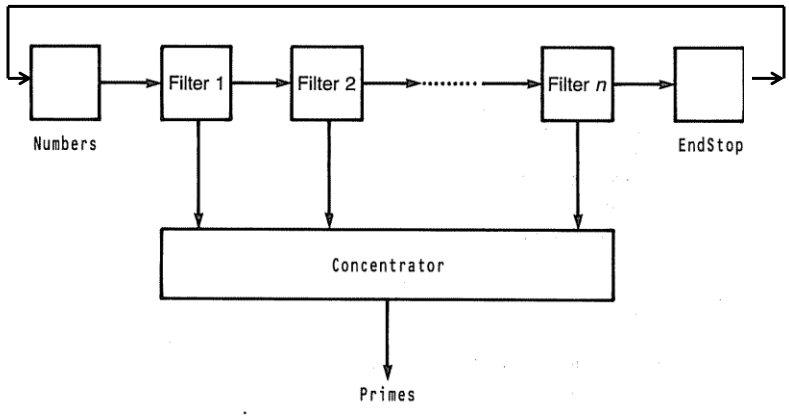
OCCAM PROSEDÜRLERİ

```
PROC First(CHAN OF INT in, CHAN OF BYTE out)
PROC Second(CHAN OF BYTE in,out)
PROC Third(CHAN OF BYTE in, CHAN OF INT out)
CHAN OF INT keyboard:
PLACE keyboard AT 2:
CHAN OF ANY screen:
PLACE screen AT 1:
CHAN OF BYTE ch1,ch2:
PAR
  First(keyboard,ch1)
  Second(ch1,ch2)
  Third(ch2,screen)
```

```
PROC First(CHAN OF INT in, CHAN OF BYTE out)
  SKIP
:
PROC Second(CHAN OF BYTE in,out)
  SKIP
:
PROC Third(CHAN OF BYTE in, CHAN OF INT out)
  SKIP
```

ASAL SAYI ÜRETİCİSİ

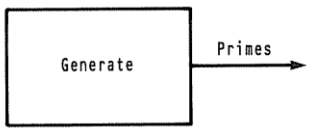
- Sieve of Erathostenes.



ASAL SAYI ÜRETİCİSİ

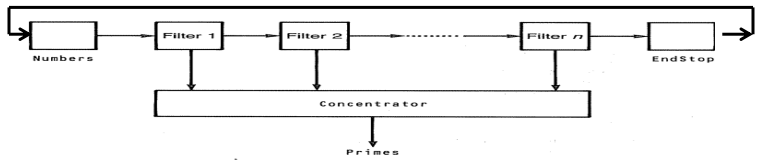
- En üst düzeyde, algoritma asal sayılar üreten bir prosedür olarak tasarlanır.

```
PROC Generate (CHAN OF INT Primes)
```



PROCEDURE GENERATE

```
VAL INT N IS 30:
PROC Generate ( CHAN OF INT Primes)
  VAL INT EndToken IS 0:
  PROC Filter (CHAN OF INT left, right, down)
  PROC Concentrator ([]CHAN OF INT in, CHAN OF INT out)
  PROC Numbers (CHAN OF INT in,out)
  PROC EndStop (CHAN OF INT in,out)
  [N + 1]CHAN OF INT InterFilter:
  [N]CHAN OF INT PC:
  CHAN OF INT OK.To.STOP:
  PAR
    Numbers (OK.To.STOP, InterFilter[0])
    PAR I = 0 FOR N
      Filter (InterFilter[I], InterFilter[I + 1], PC[I])
    EndStop (InterFilter[N],OK.To.STOP)
    Concentrator (PC, Primes)
:
```



PROCEDURE Numbers

```
PROC Numbers (CHAN OF INT in, out)
    INT i:
    SEQ
      i:=2
      WHILE i <> EndToken
        PRI ALT
          in ? i
            SKIP
          TRUE & SKIP
            SEQ
              out ! i
              i:= i+1
      out ! i
:
```

PROCEDURE Filter

```
PROC Filter(CHAN OF INT left,right,down)
  INT p,q:
  SEQ
    left ? p
    q := 1   -- dummy value, not EndToken
    PAR
      down ! p
      WHILE q <> EndToken
        SEQ
          left ? q
          IF
            q = EndToken
              SKIP
            (q\p) <> 0
              right ! q
            TRUE
              SKIP
      right ! EndToken
:
```

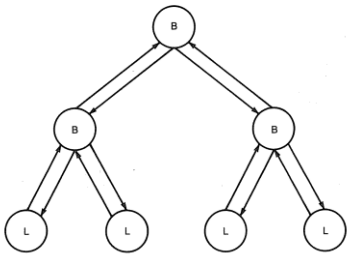
PROCEDURE EndSTOP

```
PROC EndSTOP(CHAN OF INT in,out)
  INT temp:
  SEQ
    in ? temp
    PAR
      out ! EndToken
      WHILE temp <> EndToken
        in ? temp
:
```

PROCEDURE Concentrator

```
PROC Concentrator ([]CHAN OF INT in, CHAN OF INT out)
  INT p:
  SEQ i = 0 FOR N
    SEQ
      in[i] ? p
      out ! p
:
```

PARALEL ARAMA



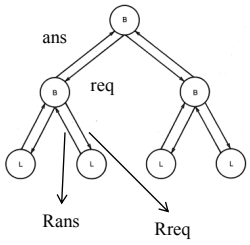
- Bir ikili ağaç yapısında haberleşen paralel prosesler ele alınır.
- Ağacın yapraklarını oluşturan düğümler bilgi taşımaktadır.

Arama işlemi şöyle gerçekleşecektir:

1. Köke aranacak veri aktarılır
2. Veri dallar aracılığı ile tüm yapraklara dağıtılır
3. Her yaprakta verinin varlığı paralel olarak sorgulanır
4. Sonuç yapraklardan dallara aktarılır
5. Sonuç kökten iletilir

PARALEL ARAMA

- Yaprak düğümleri ile ara düğümlerdeki işlemler için iki tür prosedür tasarlanmalıdır (Branch (B) ve Leaf (L)).
- Her ara düğüm (B) sol ve sağ alt çocuklara şu kanallarla bağlıdır:
  1. request---veriyi gönder
  2. answer---yanıt oku



Sol çocuk: Lreq ve Lans

Sağ çocuk: Rreq ve Rans

PROSEDÜR Branch

```
PROC branch(CHAN OF INT req,Lreq,Rreq, CHAN OF BOOL ans,Lans,Rans)
  WHILE TRUE
    INT key:
    BOOL al,ar:
    SEQ
      req ? key
      PAR
        Lreq ! key
        Rreq ! key
      PAR
        Lans ? al
        Rans ? ar
      ans ! al OR ar
:
```

PROSEDÜR Leaf

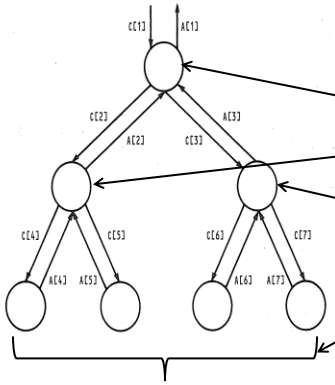
```
PROC leaf(CHAN OF INT req, CHAN OF BOOL ans)
  INT Data,key:
  SEQ
    -- load data
    WHILE TRUE
      SEQ
        req ? key
        ans ! key = Data
:
```

ÖRNEK AĞAÇ

Örnek: 4 adet veri için 4 yapraklı, derinliği 3 olan bir ağaç

```
[8]CHAN OF INT C:
[8]CHAN OF BOOL A:
PROC branch(CHAN OF INT req,Lreq,Rreq, CHAN OF BOOL ans,Lans,Rans)
  -- body of PROC
PROC leaf(CHAN OF INT req, CHAN OF BOOL ans)
  -- body of PROC
PROC User.Interface(CHAN OF INT out, CHAN OF BOOL in)
  -- body of some appropriate user interface process
```

ÖRNEK AĞAÇ




```
PAR
  branch(C[1],C[2],C[3],A[1],A[2],A[3])
  branch(C[2],C[4],C[5],A[2],A[4],A[5])
  branch(C[3],C[6],C[7],A[3],A[6],A[7])
  PAR i = 4 FOR 4
    leaf(C[i],A[i])
  User.Interface(C[1],A[1])
```