MİKROİŞLEMCİ SİSTEMLERİ

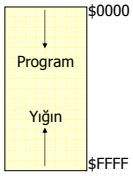
Yrd.Doç.Dr. Şule Öğüdücü
http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Yığın

- Geçici olarak veri saklamak amacıyla kullanıcı tarafından bellek içinde ayrılmış bir alandır.
- Yığında en son saklanan veri yığından ilk olarak çekilir. (LIFO)
- Bellekte küçülen adreslere doğru büyür.
  - Kullanıcı yığının dip adresini tanımlar.

Yığın

- Yığın için bellekte yüksek adreslerden başlayarak yer ayrılması, yığın ve kullanıcı programlarının birbirini ezmesini önler.
- İki şekilde kullanılır:
  - Program içinde verileri saklamak
    - `YIĞ` ve `ÇEK` buyrukları ile
  - MİB tarafından altprogramlara ve kesme hizmet programlarına dallanırken dönüş adresini saklamak ve parametre aktarmak



Örnek MİB için Yığın

- Örnek MİB'de 16 bitlik bir yığın göstergesi (YG) vardır.
- YG'nin değeri yığının en üstündeki boş bellek gözünü işaret eder.
- MİB akümülatör içeriklerini yığına aktarılabilir, ya da yığının en üstündeki veri akümülatörlere çekilebilir.
- Yığın büyük adreslerden küçük adreslere doğru büyür.
  - Genelde bellekteki en büyük adres yığın göstergesine yüklenir.

Yığma İşlemi

- Yığın bellek içinde istenen bir adresten başlayarak kurulur:

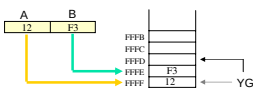
```
YÜK YG, $FFFF
```

- Yığına yeni bir veri yazmak için: `YIĞ A`
  - Yığın göstergesinin gösterdiği bellek gözüne veri yazılır.

```
<YG>  ←  A
```

  - Yığın göstergesi değeri bir azalır.

```
YG  ←  YG-1
```



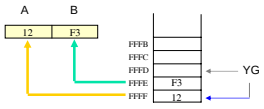
Çekme İşlemi

- Yığından bir veri çekmek için: `ÇEK B`
  - Yığın göstergesinin değeri bir artırılarak işlem yapılacak bellek gözünü işaret eder.

```
YG ← YG + 1
```

  - Yığın göstergesinin işaret ettiği bellek gözündeki veri alınır.

```
B  ← <YG>
```



Yığın İşlemleri

- Yığma ve çekme işlemlerinde işlenenlerin sırası ters olmalıdır.

```
YIĞ A
YIĞ B
:
:
ÇEK B
ÇEK A
```

- Eğer çekme işlemi yapmadan çok fazla yığma işlemi yapılırsa, yığın programları ve verileri ezer. Yığının üst adresini kontrol etmek gerekir.

Altprogram

- Program içinde tekrarlanan program parçaları altprogramlara dönüştürülür.
  - Programın çeşitli yerlerinde aynı program kodunu yazmak yerine altprogram çağrılır.
  - Aynı işlem farklı parametrelerle yürütülebilir.
- Asembler dilinde altprogram ana programın herhangi bir yerinde yer alabilir.
  - Genellikle ana programdan farklı bir alanda yazılır.
- Altprogramı çağırmak için `ALT` komutu kullanılır.
- Altprogramdan ana programa dönmek için `DÖN` komutu kullanılır.

Örnek-1

Bellekte $1000 adresinden başlayan 10 elemanlı bir dizi var. Dizi aynı adresten başlayarak ters sırada saklanacaktır.

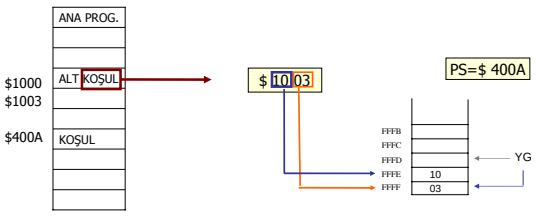
```
BAŞLA   YÜK   YG,$2000
        YÜK   SK,$1000
        YÜK   B,$0A
GERI    DEE   ILERI
        YÜK   A,<SK+0>
        ART   SK
        YIĞ   A
        AZT   B
        DAL   GERI
ILERI   YÜK   SK,$1000
        YÜK   B,$0A
DONGU   DEE   SON
        ÇEK   A
        YAZ   A,<SK+0>
        ART   SK
        AZT   B
        DAL   DONGU
SON     KES
```

Altprogram için Kullanılan Buyruklar

- Örnek MİB'de altprograma dallanmak ve altprogramdan dönmek için kullanılan buyruklar:
  - ALTD ADRES: Verilen adresteki altprograma dallan.
    <YG> ← PS(düşük), YG ← YG-1
    <YG> ← PS(yüksek), YG ← YG-1
    PS ← ADRES
  - ALT ADIM: Adım miktarı kadar ötedeki altprograma dallan.
    <YG> ← PS(düşük), YG ← YG-1
    <YG> ← PS(yüksek), YG ← YG-1
    PS ← PS + ADIM
  - DÖN: Anaprograma dön.
    YG ← YG + 1, PS(yüksek) ← YG
    YG ← YG + 1, PS(düşük) ← YG

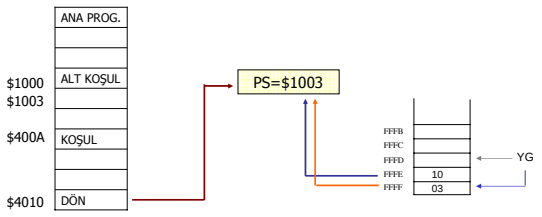
Altprogram

- Ana programda `ALT` komutu görüldüğü zaman:
  - `ALT` komutunu izleyen buyruğun adresi yığında saklanır.
  - Program sayacı altprogramın başladığı adres ile yüklenir.

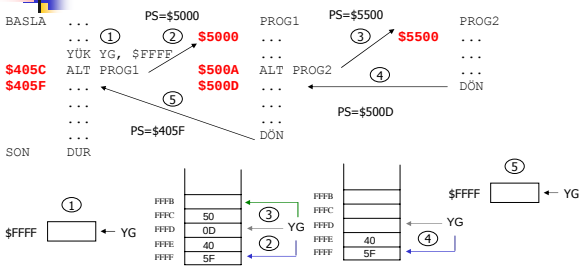


Altprogram

- Altprogramdan `DÖN` komutu ile ana programa dönülür.
  - Yığının en üstünden ana programa dönüş adresi çekilir.
  - Program sayacına ana programa dönüş adresi yüklenir.



Altprogram



Altprograma gitmeden önce yığın göstergesine başlangıç değeri atamak gerekir. Eğer yığın göstergesine değer atanmazsa, dönüş adresi bellekte verilerin bulunduğu yere yazılabilir ya da yığın göstergesi değeri bellek bulunmayan bir adresi işaret ediyorsa anaprograma dönüş adresi saklanmaz.

http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Örnek-2

Bir sayının mutlak değerini bulan altprogram. Sonuç ACCA üzerinden ana programa aktarılıyor.

```
BAŞLA    YÜK    YG,$FFFF
         YÜK    A,$A9
         YAZ    A,<$1000>
         YÜK    A,$75
         YAZ    A,<$100A>
         ALT    MUTDEG
         YÜK    A,<$1000>
         ALT    MUTDEG
SON      KES
-------------------------
MUTDEG   SIN    A,$80
         DEE    BİTTİ
         EKS    A
BİTTİ    DÖN
```

http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Değişkenlerin Taşınması

- Altprograma değişkenler kütükler üzerinden gönderilir.
  - Altprograma gönderilmek istenen veriler akümülatör ve yardımcı kütüklere yüklenir.
  - Sonuçlar akümülatör ve yardımcı kütükler üzerinden ana programa gönderilir.
- Bu iş için ayrılmış bellek gözleri kullanılarak değişkenler taşınır.
  - Ana program değişkenleri bellekte ayrılan gözlere yazar, altprogram bu gözlerden değişkenleri okuyarak işlem yapar.
- Yığın göstergesi kullanılarak yapılır.
  - Alt programa gitmeden önce değişkenler yığına aktarılır.

http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Başvuru Aktarımı – Değer Aktarımı

- Başvuru Aktarımı:Parametrelerin adresleri altprograma gönderilir.
- Değer Aktarımı: Parametrelerin değerleri altprograma gönderilir.

Örnek-2'de değişkenlerin değerleri altprograma aktarılmıştır.

Eğer değişkenlerin bulunduğu bellek gözlerinin adresi altprograma aktarılırsa, değişkenlerin değerleri değiştirilebilir.

http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Örnek-3

Örnek-2'de değişkenlerin adresleri altprograma sıralama kütüğü üzerinden gönderiliyor. Altprogram sonucu değişkenin bulunduğu bellek gözüne yazıyor.

```
BAŞLA    YÜK    YG,$FFFF
         YÜK    A,$A9
         YAZ    A,<$1000>
         YÜK    A,$75
         YAZ    A,<$100A>
         YÜK    SK,$100A
         ALT    MUTDEG
         YÜK    SK,$1000
         ALT    MUTDEG
SON      KES
-------------------------------
MUTDEG   YIĞ    A
         YÜK    A,<SK+0>
         SIN    A,$80
         DEE    BITTI
         EKS    A
         YAZ    A,<SK+0>
BITTI    ÇEK    A
         DÖN
```

Sıralama kütüğü üzerinden **başvuru aktarımı**

http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Altprogramlarda Yığın Kullanımı

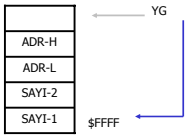
- Ana programa dönüş adresini saklamak
- Parametre aktarımı
  - Ana program değişken değerlerini yığına aktarır. Altprogram ve ana program yığın göstergesi değerini bilir.
- Kütük değerlerini korumak
- Altprogramlarda yerel değişken yaratmak

http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Örnek-4

Iki ikili sayıyı toplayan bir altprogram yazılacaktır. Parametreler **yığın üzerinden değer aktarımı** yöntemi ile aktarılacaktır.

```
BAŞLA   YÜK   YG,$FFFF        TOPLA   YÜK   A,<YG+4>
        YÜK   A,$59                   TOP   A,<YG+3>
        YAZ   A,<$1000>
        YIĞ   A
        YÜK   A,$75
        YAZ   A,<$1005>
        YIĞ   A
        ALT   TOPLA
```

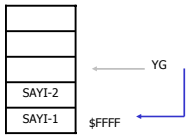


http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Örnek-4

Iki ikili sayıyı toplayan bir altprogram yazılacaktır. Parametreler **yığın üzerinden değer aktarımı** yöntemi ile aktarılacaktır.

```
BAŞLA   YÜK   YG,$FFFF        TOPLA   YÜK   A,<YG+4>
        YÜK   A,$59                   TOP   A,<YG+3>
        YAZ   A,<$1000>               DÖN
        YIĞ   A
        YÜK   A,$75
        YAZ   A,<$1005>
        YIĞ   A
        ALT   TOPLA
        YAZ   A,<$100A>
SON     KES
```

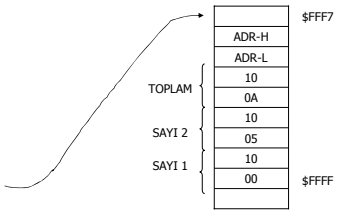


http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Örnek-5

Altprograma değişkenlerin adresleri ve toplamın yazılacağı bellek gözünün adresi **yığın üzerinden** gönderiliyor. Değişkenler altprogramdan dönüldüğü zaman 0 değerini alıyor: **Başvuru Aktarımı**

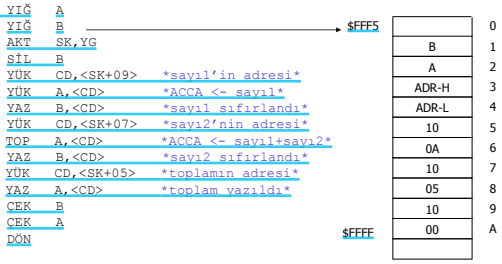
```
BAŞLA   YÜK   YG,$FFFF
        YÜK   A,$59
        YAZ   A,<$1000>
        YÜK   SK,$1000
        AKT   AB,SK
        YIĞ   B
        YIĞ   A
        YÜK   A,$75
        YAZ   A,<$1005>
        YÜK   A,$05
        YIĞ   A
        YÜK   A,$10
        YIĞ   A
        YÜK   SK,$100A
        AKT   AB,SK
        YIĞ   B
        YIĞ   A
        ALT   TOPSİL
SON     KES
```



http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Örnek-5

```
TOPSİL  YIĞ   A
        YIĞ   B
        AKT   SK,YG
        SİL   B
        YÜK   CD,<SK+09>   *sayı1'in adresi*
        YÜK   A,<CD>       *ACCA <- sayı1*
        YAZ   B,<CD>       *sayı1 sıfırlandı*
        YÜK   CD,<SK+07>   *sayı2'nin adresi*
        TOP   A,<CD>       *ACCA <- sayı1+sayı2*
        YAZ   B,<CD>       *sayı2 sıfırlandı*
        YÜK   CD,<SK+05>   *toplamın adresi*
        YAZ   A,<CD>       *toplam yazıldı*
        ÇEK   B
        ÇEK   A
        DÖN
```



http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Yerel Değişkenler

- Altprogram kullandığı yerel değişkenler için yığında yer alır. Yığından yerel değişkenler için yer almak ve yığına geri vermek altprogramın görevidir.

Örnek: 16 bitlik bir sayının yüksek anlamlı ve düşük anlamlı bitlerinin yer değiştirmesi.

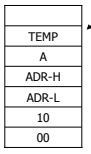
C:

```
void swap(short int *number)
{
   char temp;
   temp = *number;
   *number = *(number + 1);
   *(number + 1) = temp;
}
```

http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Örnek-6

```
BAŞLA   YÜK   YG,$FFFF
        YÜK   SK,$1000
        AKT   AB,SK
        YIĞ   B
        YIĞ   A
        YÜK   A,$44
        ALT   SWAP
SON     KES
```




```
SWAP    YIĞ   A
        AZT   YG      *yerel değişken için yer alındı*
        AKT   SK,YG
        YÜK   CD,<SK+05>  *yüksek anlamlı 8 bitin adresi*
        YÜK   A,<CD>      *ACCA <- yüksek anlamlı 8 bit*
        YAZ   A,<SK+1>    *temp <- yüksek anlamlı 8 bit*
        ART   CD          *düşük anlamlı 8 bitin adresi*
        YÜK   A,<CD>      *ACCA <- düşük anlamlı 8 bit*
        AZT   CD
        YAZ   A,<CD>      *takas*
        YÜK   A,<SK+1>    *ACCA <- temp*
        ART   CD
        YAZ   A,<CD>  *ACCA düşük anlamlı kısma aktarıldı*
        ART   YG   *yerel değişkenin yeri geri verildi*
        ÇEK   A
        DÖN
```

```
void swap(short int *number)
{
     char temp;
     temp = *number;
     *number = *(number + 1);
     *(number + 1) = temp;
}
```

http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30

Dikkat!!

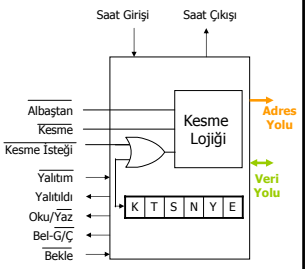
- YIĞ ve ÇEK işlemlerinde işlenenlerin ters sırada olması gerekir.
- YIĞ işlemi sayısı kadar ÇEK işlemi olması gerekir.
  - Aksi halde DÖN komutu ile yığından yanlış adres çekilir ve program çalışmasında hata oluşur.
- Döngü içinde YIĞ ve ÇEK işlemleri yapılmaması iyi olur.
- Altprogramdan DÖN komutu ile dönülmesi gerekir.

Kesme

- Kesme acil durum işareti olarak görülebilir:
  - MİB, o an yapmakta olduğu işe en kısa sürede ara vererek gelen kesmeye cevap vermek üzere Kesme Hizmet programına gider.
  - Kesmenin geldiği birime göre her birimin kendi Kesme Hizmet Programı vardır.

Kesme

- Maskelenemez Kesme: MİB'nin kesme girişine gelen kesmeye o anda yapmakta olduğu işe ara vererek cevap verir.
- Maskelenebilir Kesme: MİB'nin Kesme İsteği girişine gelen kesmeye durum kütüğündeki kesme maskesi bitinin değerine göre cevap verir ya da vermez.
  - K=1: Kesme isteği engellenir.
  - K=0 Kesme isteğine cevap verilir.
- Albaştan (Reset): Ilk açıldığı hale getirir.



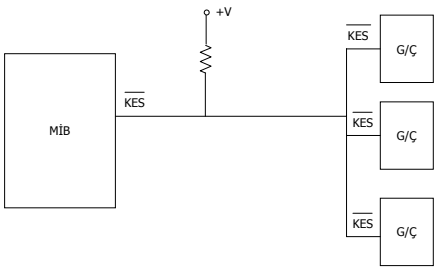
Kesme Hizmet Programına Dallanma

- Kesme Hizmet programları bir altprogram gibi çalışır. MİB kesme hizmet programına gitmeden önce dönüş adresini (kütüklerin içeriklerini) yığında saklar.
- Kesme hizmet programından DÖN (DÖNK) komutu ile dönülür.
- Kesme hizmet programına iki şekilde dallanılır:
  - Başlangıç adresi doğrudan: Kesme hizmet programının başlangıç adresi bellidir.
  - Başlangıç adresi dolaylı: Kesme hizmet programının başlangıç adresinin yazılacağı bellek gözleri bellidir.

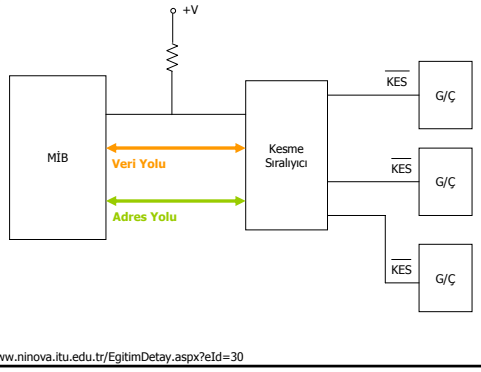
Kesmenin Donanım Yapısı

- Kesme alındığında kesmenin nerden geldiğine iki şekilde karar verilebilir:
  - Yoklama yöntemi: Kesme hizmet programına gidilerek kesme gelebilecek birimler sırayla sınanır.
    - Uzun sürebilir.
  - Kesme Sıralıyıcı: Bir kesme öncelik devresi ile kesmenin hangi birimden geldiği anlaşılır.

Yoklama Yöntemi



Kesme Sıralıyıcı



http://www.ninova.itu.edu.tr/EgitimDetay.aspx?eId=30