YIĞIT (STACK) KURULUMU

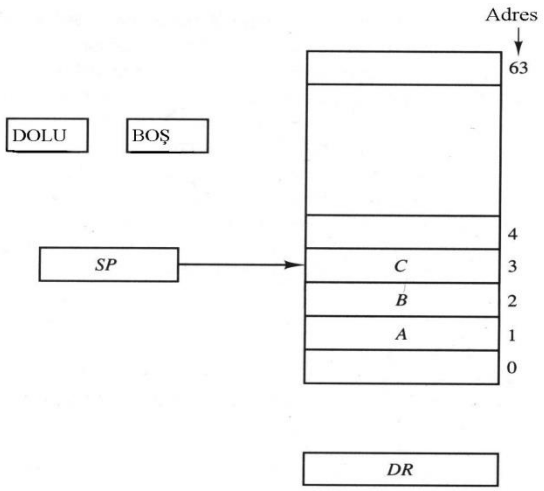
Çoğu bilgisayarın MİB'de yığıt veya LIFO (Last In First Out) bulunur. Yığıt bir bellek parçasıdır ve son depolanan bilgi ilk geri dönen bilgi olur. Yığıta aktarılan son bilgi yığıtın en tepesinde bulunur.

Yığıt Göstergesi (Stack Pointer):

Sayısal bilgisayarlarda yığıt temel olarak bir bellek birimidir. Sadece sayabilen başka hiçbir değerin yazılamadığı 1 adres yazacı vardır. Bu tip adres yazaçlarına *yığıt göstergesi* adı verilir ve aldığı değer daima yığıttaki son kelimenin adresini gösterir.

Yığıt ile ilgili işlemler Ekleme ve Silmedir. Ekleme işlemi *itme* olarak adlandırılır. Çünkü üste bir şey konur ve alttakiler aşağıya gider. Bunun tersi işlem silme işlemidir ve *çekme* olarak adlandırılır. Çünkü bu işlem sonucu alttakilerin bir yukarı çıktıkları varsayılır.

Yazaç Yığıtı (Register Stack):



Şekil. 64 Kelimelik Yığıtın Blok Şeması

Yığıt, ya belleğin bir bölümünde yer alır veya sonlu sayıda bellek kelimeleri ve yazaçların toplamı ile düzenlenir.

Yığıt göstergesi *SP* yığıtın en tepesinde bulunan kelimenin adresinin ikili değerini içerir. Yığıttaki üç bilgiden *C* en tepede olduğundan *SP'*deki adres 3 yani *C'*nin adresidir. Tepedeki bilgiyi silmek için 3 adresindeki bilgi okunur ve *SP* nin içeriği azaltılır. Böylelikle yığıtın tepesine *B* gelir ve *SP* deki adres değeri 2 olur. Yeni bir bilgiyi yığıta yazmak için *SP* bir arttırılır ve yeni adrese yeni bilgi yazılır.

64 kelimelik bir yığıtta $2^6 = 64$ olduğundan S=6 bittir ve değeri 63'ten (ikili 111111) büyük olamaz. 63 1 arttırıldığında 0 sonucuna ulaşılır. 111111 + 1 = 1000000 ikili olarak. Fakat *SP* en küçük 6 önemli biti tuttuğundan 0 elde edilir. Benzer olarak 000000 dan 1 çıkarıldığında sonuç 111111 olur. Bir bitlik *DOLU* yazacı, yığıt dolu ise 1, bir bitlik *BOŞ* yazacı yığıt boş ise 1 yapılır. Veri yazacı *DR* yığıta yazılan veya yığıttan okunan ikili veriyi tutar.

İtme (Yığıt'a Ekleme-PUSH):

Başlangıçta yığıt göstergesi 0 ile temizlenir, *BOŞ* = 1 ve *DOLU* = 0 yapılır. Böylece *SP* = 0 adresini gösterir ve yığıtın boş olduğunu işaret eder. Eğer yığıt dolu değil ise (yani *DOLU* = 0 ise) itme işlemi ile yeni bilgi girilebilir.

İtme işleminin mikro işlemleri:

SP ← *SP* + 1	*SP* yi 1 arttırır.
M[*SP*] ← *DR*	*DR* deki bilgiyi yığıtın en üstüne yaz.
Eğer (*SP* = 0) ise (*DOLU* ← 1)	Yığıtın doluluğunu kontrol et
BOŞ ← 0	Yığıtın boş olmadığını işaret et

SP bir arttırıldığından bir yukarıdaki kelimenin adresini gösterir durumuna geldi. Belleğe yazma işlemi ile *DR* deki bilgi yığıtın en üst kısmına yazılır. Çünkü *SP* de daima en üst kısmın adresi vardır. Eğer *SP* sıfır olursa yığıt doludur ve *DOLU* = 1 yapılır. *SP* nin 0 olması, son adres 63 iken buna bir eklenince *SP* taşar ve kalan kısım 0 olur. Dolayısıyla yeni bilgi 0 adresine yazılmağa çalışılır. 0 adresine bilgi yazılınca yığıtta yer kalmamış demektir. Dolayısıyla *BOŞ* sıfır yapılır.

Çekme (Yığıt'tan Silme-POP):

Eğer yığıt boş değilse (*BOŞ* = 0) bu taktirde son girilmiş bilgi yığıttan silinebilir.

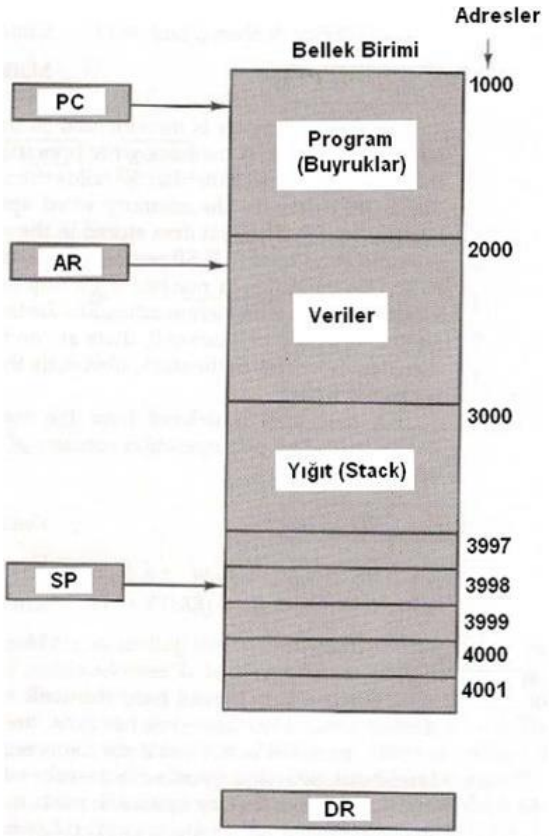
Çekme işleminin mikro işlemleri:

DR ← *M*[*SP*]	Yığıttan bilgiyi oku
SP ← *SP*-1	*SP* yi 1 azalt
Eğer (*SP* = 0) ise (*BOŞ* ← 1)	Yığıtın boşluğunu kontrol et
DOLU ← 0	Yığıtın dolu olmadığını işaretle

İlk mikro işlem yığıttan bilgiyi *DR* ye okur, sonra *SP* bir azaltılır. Eğer *SP* = 0 olursa yığıt boştur ve *BOŞ* ← 1 yapılır. Bu koşul eğer okunan bilgi 1 nolu adreste ise yapılır. Bu adres okunduğu zaman yığıttaki bütün yazaçlar boştur. Eğer okuma işlemi 1 adresinden değil de 0 adresinden yapılırsa,

SP azaltılınca *SP* nin değeri 63 yani ikili 111111 olur, 0 adresi yığıttaki son değeri alır. *DOLU* = 1 iken itme veya *BOŞ* = 1 iken çekme işlemi yapılırsa hatalı işlem yapılmış olur.

Bellek Yığıtı (Memory Stack):



Şekil. Program, Veri ve Yığıt Şeklinde Ayrılmış Bilgisayar Hafızası

Şekil üç bölüme ayrılmış bilgisayar hafızasının bir bölümünü göstermektedir.

PC programda bir sonraki buyruğun adresini gösterir ve al-getir evresi boyunca bir buyruk okumak için kullanılır.

AR adres yazacı bir sonraki veriyi gösterir ve çalışma evresi boyunca bir veri okumak için kullanılır.

SP yığıtın en üstünü gösterir ve yığıta atılacak veya yığıttan alınacak verilerin itme yada içme işleminde kullanılır.

Şekil'de görüldüğü gibi *SP* nin başlangıç değeri 4001'dir. Buna göre birinci veri girildiğinde değeri 4000, ikinci veri girildiğinde 3999 ve en son veride adres değeri 3000 olacaktır.

Yeni bir verinin itme (ekleme) işlemi:

$SP \leftarrow SP\text{-}1$

$M[SP] \leftarrow DR$

SP azaltılarak sonraki kelimenin adresi gösterilir. Belleğe yazma işlemi *DR* den yığıtın en üstüne yazma işlemidir.

Yeni bir verinin çekme (silme) işlemi:

$DR \leftarrow M[SP]$

$SP \leftarrow SP + 1$ şeklinde olur.

Yığıt Limitleri:

Çoğu bilgisayarlar yığıt taşması (dolu yığıt) veya yığıt bitmesi (boş yığıt) denetimini donanım ile sağlamaz. Yığıt limitleri için iki işlemci yazacı kullanarak denetim yapılır. Yazacın birisi üst limiti (burada 3000) ve diğer yığıtta alt limiti (burada 4001) tutar. Bir sonraki itme işleminde *SP* üst-limit yazacı ile, çekme işleminde de alt-limit yazacı ile karşılaştırılır.

İtme ya da çekmeden herhangi biri için ihtiyaç duyulacak iki mikro işlem *SP* vasıtasıyla belleğe giriş ve *SP* yi güncelleştirme işlemleridir.

Bir yığıt göstergesine bir başlangıç değeri yüklenir. Buraya yığıt başlangıç değeri olarak en alttaki adres atanır. Bundan sonra *SP* değeri otomatik olarak her itme veya çekme işlemi ile azaltılır veya arttırılır.

Zıt Polonyalı Gösterimi:

Zıt polonyalı yöntemi, yığıt işletimi için uygun bir yöntemdir.

$A * B + C * D$ ifadesi zıt polonyalı gösteriminde

$AB * CD * +$

biçiminde yazılır ve şu şekilde değerlendirilir. İfade soldan sağa doğru aratılır, bir operatöre ulaşıldığında, operatörün solunda yer alan iki veri çalıştırılır.

Zıt polonyalı gösterimindeki matematiksel ifadelerin değerlendirilmesi için en etkili yol yazaçların bir yığıt düzenlemesi ile bulunmasıdır.

Yığıttaki mikro işlemler:

(1)Yığıtta en üstteki iki veri işlem için kullanılır,

(2) Yığıt çekilir. İşlem sonucu üstteki veri ile yer değiştirir.

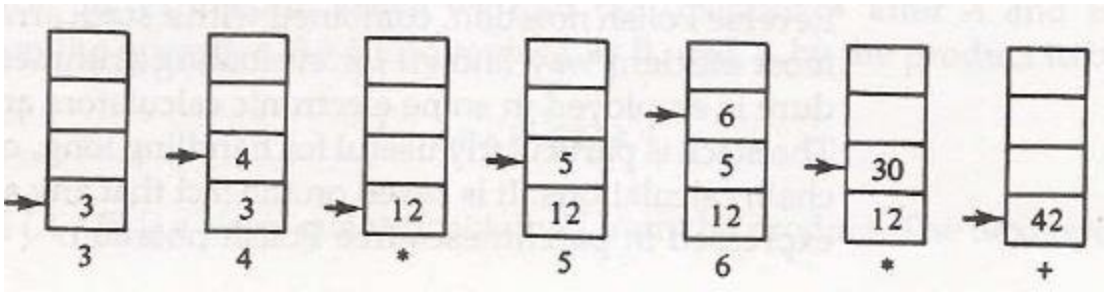
Verileri sürekli olarak yığıt içine iterek ve işlemleri yukarıdaki gibi çalıştırarak, ifadeler uygun sırada değerlendirilir ve sonuç yığıtın en üstünde kalır.

(3*4) + (5*6)

aritmetik ifadesi ele alındığında, zıt polonyalı yönteminde bu ifade

34 * 56 * +

şeklinde gösterilir.



Şekil. (3*4) + (5*6) İşlemi İçin Yığıt İşlemleri

BUYRUK BİÇİMLERİ

Bir buyruğun biçimi bir dikdörtgen içinde verilir. Bir buyruğun bitleri alanlar adı verilen gruplara ayrılır. Bir buyruk biçiminde bulunan alanlar;

1. İcra edilecek işlemi gösteren OPR kod alanı
2. Bir adres alanı: Bellek alanını veya bir işlemci yazacını gösterir
3. Bir kip alanı: Verinin yolunu veya etkin adresin bulunduğunu gösterir

Yazaç Adresi:

Bellek içinde bulunan veriler yazaç adresiyle bellidir. Bir yazaç adresi *k* bitlik bir ikili sayıdır. MİB içindeki 2^k tane yazaçtan bir tanesini belirtir. Eğer MİB da *R*0 dan başlayıp *R*15 e kadar giden 16 tane yazaç varsa yazaç adresleri için 4 bit yeter. Örneğin 0110 adresi *R*5'i gösterir

ADRES SAYILARINA GÖRE BUYRUK BİÇİMLERİ

Üç Adresli Buyruklar:

X = (*A* + *B*) * (*C* + *D*) yi yapan birleştirici dil programı.

ADD R1, A, B	R1 ← M[A] + M[B]
ADD R2, C, D	R2 ← M[C] + M[D]
MUL X, R1, R2	M[X] ← R1 * R2

İki Adresli Buyruklar

X = (*A* + *B*) * (*C* + *D*) işlemini yapan birleştirici dil programı.

MOV R1 A	R1 ← M[A]
ADD R1,B	R1 ← R1 + M[B]
MOV R2,C	R2 ← M[C]
ADD R2,D	R2 ← R2 + M[D]
MUL R1,R2	R1 ← R1 * R2
MOV x, R1	M[X] ← R1

Birinci adresin hem kaynak, hem de hedef yazacı olduğu varsayılıyor. MOV buyruğu veriyi bellekten işlemci yazacına aktarır.

Bir Adresli Buyruklar:

Tüm veri işlemleri için *AC* yi kullanırlar.

LOAD	A	AC ← M[A]
ADD	B	AC ← A + M[B]
STORE	T	M[T] ← AC
LOAD	C	AC ← M[C]
ADD	D	AC ← AC + M[D]
MUL	T	AC ← AC * M[T]
STORE	X	M[X] ← AC

Bütün işlemler *AC* yazacı ile bellekteki veriler arasındadır. *T* bellek adresi ara değerin saklanması içindir.

Sıfır Adresli Buyruklar:

Yığıt kurulumlu bir bilgisayarda buyrukların adrese gereksinimi yoktur. ADD, MUL, PUSH ve POP adressizdir, fakat yığıtla bellek haberleşmesi için bellek adreslerine gerek vardır. (TOS = yığıt göstergesi).

X = (*A* + *B*) * (*C* + *D*) işleminin yığıt kurulumlu bilgisayarda birleştirici dil programı.

PUSH	A	TOS ← A
PUSH	B	TOS ← B
ADD		TOS ← (A + B)
PUSH	C	TOS ← C
PUSH	D	TOS ← D
ADD		TOS ← (C + D)
MUL		TOS ← (C + D) * (A + B)
POP	X	M[X] ← TOS

RISC BUYRUKLARı

RISC buyruk kümesi, bellekle MİB arasındaki aktarımlar için LOAD ve STORE buyrukları ile kısıtlıdır. Diğer bütün buyruklar MİB içinde icra edilir. Yani buyruklar içinde bellek adreslemeli buyruk yoktur. RISC için yazılmış bir programda LOAD ve STORE buyrukları bir bellek ve bir yazaç adresi içerir. Hesaplama tipi buyruklar ise 3 adresli buyruk olup, adreslerin hepsi yazaç adresleridir.

$X = (A + B) * (C + D)$ işleminin RISC buyrukları ile birleştirici dil programı.

```
LOAD    R1, A       R1 ← M[A]
LOAD    R2, B       R2 ← M[B]
LOAD    R3, C       R3 ← M[C]
LOAD    R4, D       R4 ← M[C]
ADD     R1, R1, R2  R1 ← R1 + R2
ADD     R3, R3, R4  R3 ← R3 + R4
MUL     R1, R1, R3  R1 ← R1 * R3
STORE   X, R1       M[X] ← R1
```