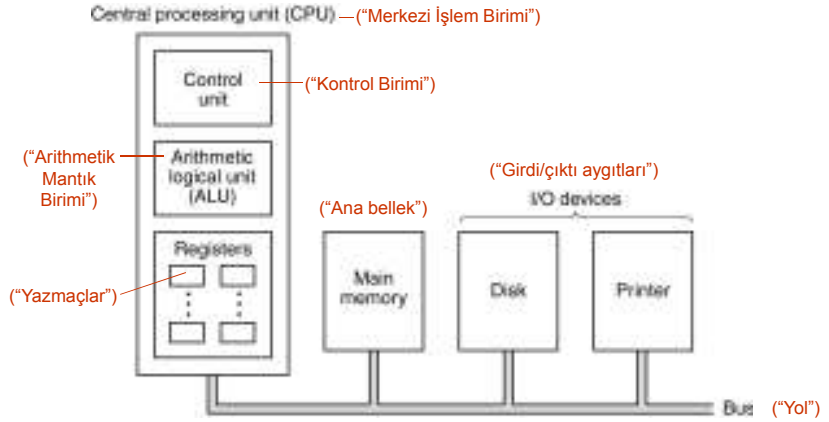
BİL 362 Mikroişlemciler: 8086 Mimarisi

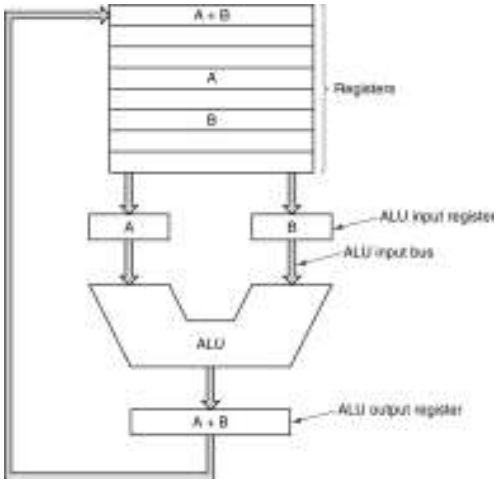
Ahmet Burak Can
abc@cs.hacettepe.edu.tr

Temel Bilgisayar Yapısı



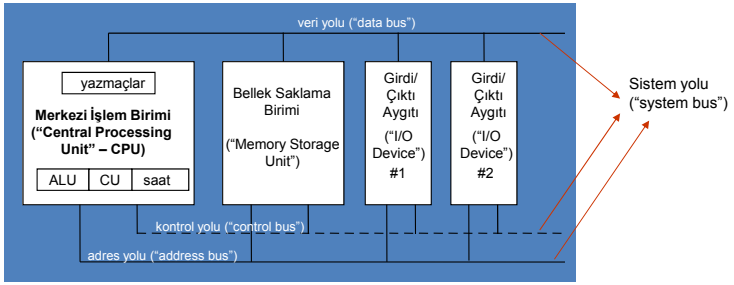
Bir işlemci ve 2 giriş/çıkış aygıtı içeren basit bir bilgisayarın yapısı

Merkezi İşlem Birimi (CPU) – 1



Merkezi İşlem Birimi (CPU) – 2

- Saat ("clock"), Merkezi İşlem Biriminin (CPU) iç işlemlerini, diğer sistem bileşenleri ile senkronize eder.
- Kontrol Birimi ("Control Unit" -- CU), komut işletme adımlarının sırasını koordine eder.
- Aritmetik Mantık Birimi ("Artihmetic Logic Unit" -- ALU), ekleme ve çıkarma gibi aritmetik işlemler ile AND ve OR gibi mantıksal işlemleri gerçekleştirir.



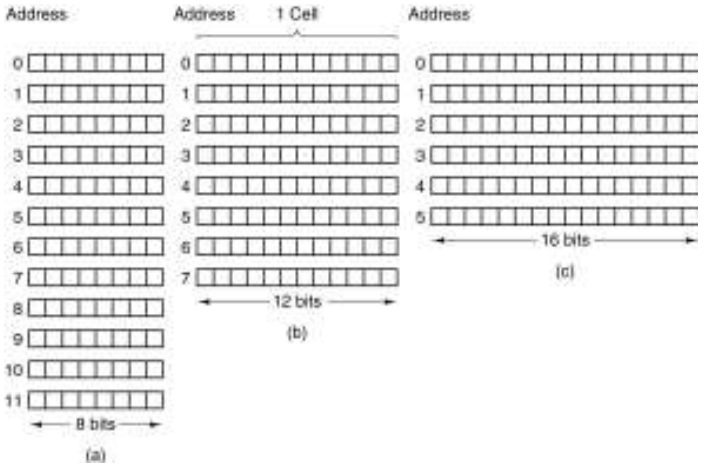
Yazmaçlar
(“Registers”)

- Ana bellekteki veriler işlenmek üzere merkezi işlem birimine taşınırlar.
- Burada geçici olarak verilerin saklandığı hücrelere yazmaç denir.
- Merkezi işlem biriminde çeşitli amaçlar için kullanılan yazmaçlar mevcuttur:
  - Ana bellekte erişilmek istenen verinin adresini tutan adres yazmaçları
  - Programda çalıştırılmakta olan komut adresini tutan adres yazmaçlar
  - Komutu tutan komut yazmacı (“instruction pointer - IP”)
  - Üzerinde işlem yapılan veriyi tutan akümulatör (“accumulator”)

Ana Bellek
(“Main Memory”)

- Bilgisayarlarda geçici olarak veri saklamak için her biri bir bit saklayabilen çok fazla sayıda devre (“flip-flop”) mevcuttur. Bilgisayarlardaki bu "bit depolarına" ana bellek denir.
  - Ana bellekteki devreler gruplanarak hücreleri oluştururlar. Her hücre genellikle 8 bitten (1 bayt) oluşur. Bu hücrelerin her birinin adresi vardır.
  - Hücreler adreslerine göre ardışık olarak dizilir. Hücre adresi hücrenin ana bellek içindeki yerini belirler.
  - Bellekteki hücrelerin dizilme sırasının erişime bir etkisi yoktur (en baştaki hücreye erişmekle en sondaki hücreye erişmek aynıdır). Bu yüzden ana belleğe Rastgele Erişimli Bellek ("Random Access Memory – RAM”) denir.
- Bilgisayarın ana belleğindeki hücrelerin toplam sayısı, bilgisayarın bellek kapasitesini gösterir (örneğin, 1 MB= 2^{20} bayt, 1 GB= 2^{30} bayt)

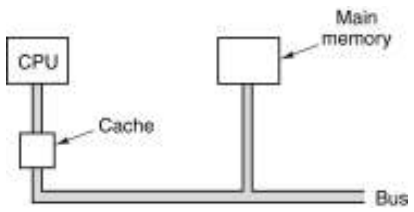
Ana Bellek (“Main Memory”)
Bellek Adresleri



96-bitlik bir belleğin 3 değişik yapılandırması

Önbellek (Cache Memory)

- Birincil bellek CPU’dan oldukça yavaş kaldığı için, yakın zamanda kullanılan komutları ve verileri taşımak amacıyla, yüksek-hızlı önbellek kullanılır.
  - Veri birincil bellekten ilk defa okunduğunda, kopyası önbelleğe alınır. Program aynı veriyi ikinci kez okumak isterse, veriyi önbellekte arar.
    - Önbellek vuruşu (“cache hit”): Veri önbellektedir.
    - Önbellek kaçırma (“cache miss”): Veri önbellekte değildir ve birincil bellekten tekrar okunur.



Örnek:

İki tip önbellek:

- 1. seviye önbellek (CPU’nun içinde; daha küçük, daha hızlı, daha pahalı)
- 2. seviye önbellek (önceleri CPU’nun dışında iken şimdi yongaya entegre edilmiştir.)

Veriyolu

- Merkezi işlem birimi ve ana bellek arasında bitlerin geliş gidişi, adına "veriyolu" dediğimiz bağlantı yığını ile sağlanır.
  - Ana bellekteki veriler, veriyolu ile merkezi işlemciye alınır. Burada işlemler tamamlanınca da ana bellekte istenen adrese yerleştirilir.
  - Kontrol birimi veriyolu üzerinden istenen verileri getirip götürürken, aritmetik-mantık birimi veriler üzerinde program tarafından belirlenen işlemleri yapar.

Komutlar

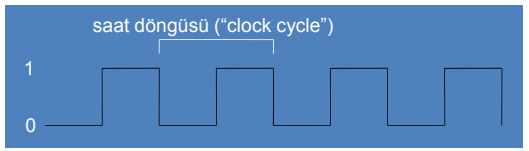
- Merkezi İşlem Biriminde program gereği yerine getirilmesi gereken işlemleri yapmak için makine komutları kullanılır.
  - Komutlar, bilgisayarların özelliklerine ve kullandıkları Merkezi İşlem Birimine göre farklılık gösterir.
  - Kullanılan komutlar farklı yapıda olmakla birlikte benzer işlevleri yerine getirirler; komutların uzunlukları da değişiklik gösterir.
- Örnek: 16 bitten oluşan bir bilgisayar komutu aşağıdaki alanları içerebilir.
  - İşlem alanı ("OP-CODE: operation code"): Her bir komut için ayrılan alanın ilk dört biti (ilk onaltılı sayı) işlem alanıdır.
  - İşlenen ("operand") alanı: Bu alanda 12 bit (3 onaltılı sayılık) yer mevcuttur. Komutun tamamlayıcı bilgilerini karşılarlar.
    - Örneğin: işlemimiz toplama ise işlenen alanı kaynak işlenen ve hedef işlenenlerin bellekteki adres bilgilerini içerir.

Komut Türleri

- Kontrol
  - Program akışını yönlendirir
- Veri transfer
  - Veriyi bir konumdan diğerine taşır
  - Bellekten belleğe taşıma sırasında yazmaçları kullanır
    - Bellekten belleğe taşıma direkt olarak yapılmaz
- Aritmetik/mantık
  - Mevcut bit gruplarından yeni bit grupları oluşturur
    - AND, OR, XOR
    - Çevirme ve kaydırma
    - Toplama, çıkarma, çarpma, bölme

Saat ("Clock")

- CPU ve sistem yolunu ilgilendiren her işlem, saat ("clock") tarafından senkronize edilir.
  - Makine komutları için temel zaman birimi saat döngüsüdür ("clock cycle").
  - Saat döngüsünün uzunluğu, saat hızının tersidir ve saniyedeki salınımla ölçülür.
    - Örnek: Saniyede bir milyar kez salınan bir saat (1 GHz), saniyenin milyarda biri uzunluğunda (1 nano-saniye) saat döngüsü üretir.
- Bir makine komutu, işletilmek için en az bir saat döngüsü gerektirir.

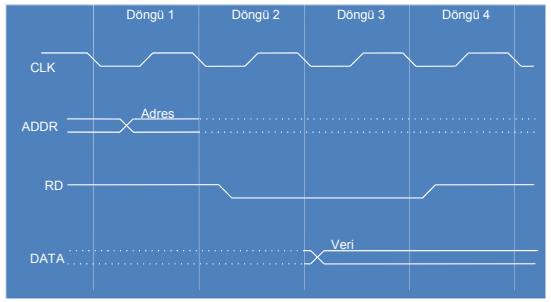


Bellekten Okuma - 1

- Program üretkenliği, çoğunlukla belleğe erişim hızına bağlıdır.
  - CPU saat hızı birkaç GHz iken, belleğe erişim çok daha yavaş çalışan veriyolu üzerinden olur.
- Bu sebeple CPU komutlar işletilmeden önce, işlenenlerin bellekten getirilmesi için birkaç saat döngüsü beklemek zorunda kalır.
  - Boşa giden döngülere kayıp döngüler ("wasted cycles") denir.
- Bellekten okuma saat tarafından kontrol edilir.
- Bellekten komut veya veri okuma birkaç adımda gerçekleşir.

Bellekten Okuma - 2

- Bellekten okuma sırasında her bir saat döngüsünde (saat vuruşunda) aşağıdakiler gerçekleşir:
  - Döngü-1: Bellek adresi, adres yoluna (ADDR) koyulur.
  - Döngü-2: Okuma çizgisi (RD), belleğe bir değerin okunacağını belirtmek için, 0 atanır.
  - Döngü-3: CPU, belleğin karşılık vermesi için bir döngü bekler. Bu döngü sırasında, bellek yöneticisi işleneni veri yoluna (DATA) koyar.
  - Döngü-4: Okuma çizgisi (RD), CPU'ya veri yolundaki veriyi okumasını belirtmek için, 1 atanır.

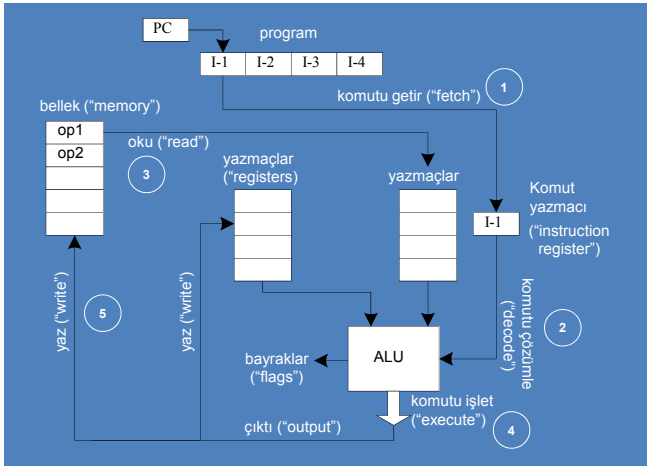


Komut İşletme Adımları

1. Komutu getir ("fetch"):
   - Bellekten bir sonraki komutu getir ve yazmaca koy.
   - Program sayacını ("program counter" ya da "instruction pointer") bir sonraki komutu gösterecek şekilde güncelle.
2. Komutu çözümle ("decode"):
   - Getirilen komutun tipini ve işlevini belirle.
3. İşlenenleri getir ("fetch operands"):
   - Komut bellekteki bir sözcüğü kullanıyorsa sözcüğün nereden getirileceğini belirle, sözcüğü getir ve CPU yazmacına koy.
4. Komutu işlet ("execute"):
   - Yazmaçlardaki işlenenler üzerinde komutu işlet ve sonucu yazmaça koy.
5. Sonucu sakla ("store output"):
   - Çıktı bir bellek işleneniyse belleğe yaz.

   (Bir sonraki komutu işletmek için 1.adıma git.)

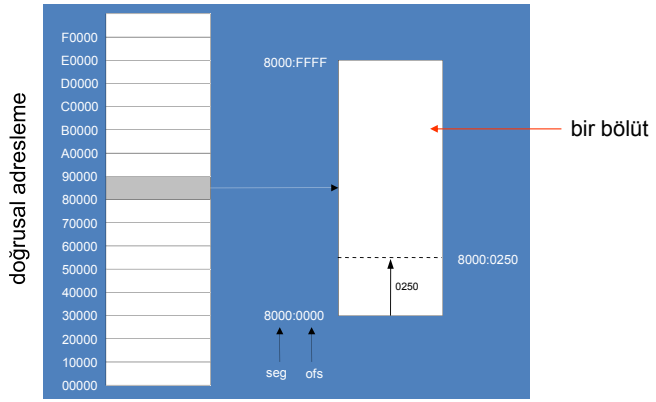
Komut İşletme Diyagramı



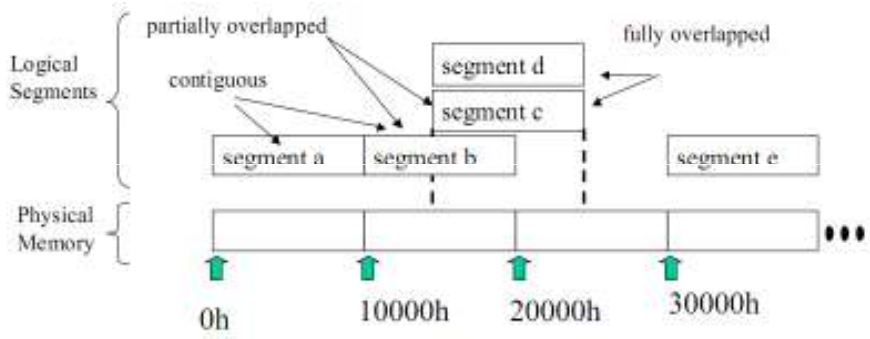
Komut işletme döngüsündeki her komut, en az bir saat döngüsü ("clock cycle") gerektirir.

Bölütlü Bellek (“Segmented Memory”)

- Bölütlü bellek (“segmented memory”): Bellek, bölüt adı verilen kısımlara ayrılmıştır.
  - Örnek: Intel 8086 -- gerçek-adres modunda 64 kilo-baytlık (KB) bölütler
  - Doğrusal adres, 16-bit bölüt değerine eklenen 16-bit ofset değeri ile hesaplanır.

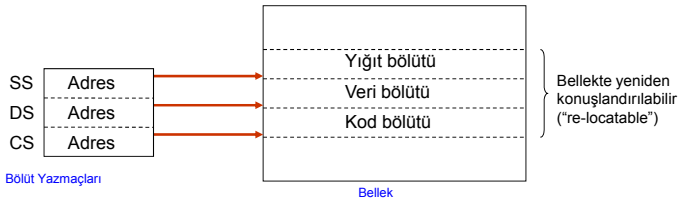


Bölütlerin Örtüşmesi



Bölüt Türleri

- Bir program içinde, 3 tip bölüt bulunur:
  - Kod bölütü (“code segment”): Çalıştırılacak komutları içerir.
    - Tipik olarak, ilk çalıştırılabilir komut, bu bölütün başında yer alır.
    - Kod bölüt yazmacı (“code segment register” – CS) kod bölütünü adresler.
  - Veri bölütü (“data segment”): Programın tanımlı verilerini ve sabitlerini içerir.
    - Veri bölüt yazmacı (“data segment register” – DS) veri bölütünü adresler.
  - Yığıt bölütü (“stack segment”): Programın işletme sırasında geçici olarak saklamaya ihtiyaç duyduğu veriyi ve adresleri içerir.
    - Yığıt bölüt yazmacı (“stack segment register” – SS) yığıt bölütünü adresler.



Fiziksel Adreslerin Hesaplanması

- Bölüt adresini 16 ile çarpın (sonuna onaltılı 0 da koyabilirsiniz) ve ofset değerini ekleyin.

- Örnek: 08F1:0100’i fiziksel adrese çevirin.

```
Adjusted Segment value:  0 8 F 1 0
Add the offset:            0 1 0 0
Linear address:          0 9 0 1 0
```

Adresler için her zaman onaltılı gösterim kullanılır.

Alıştırma - 1

Hangi fiziksel adres 028F:0030 bölüt:ofset adresine karşılık gelir?

028F0 + 0030 = 02920

Alıştırma - 2

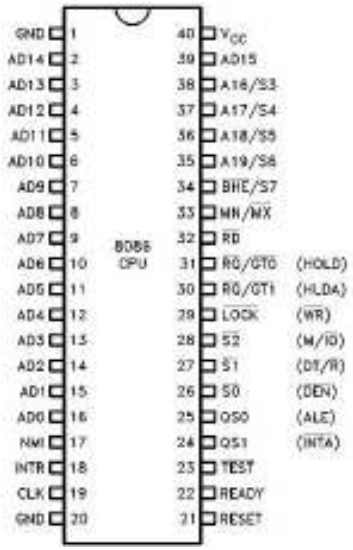
Hangi fiziksel adres 28F30h bölüt adresine karşılık gelir?

Birçok farklı bölüt:ofset adresi 28F30h fiziksel adresini üretebilir.

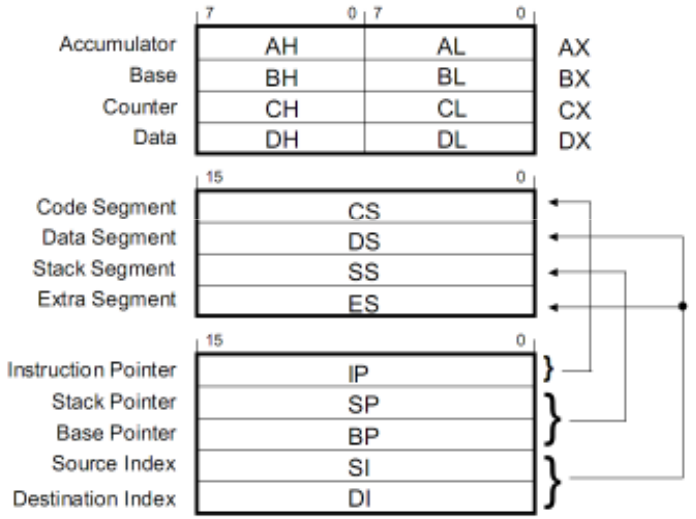
Örneğin:

28F0:0030, 28F3:0000, 28B0:0430, . . .

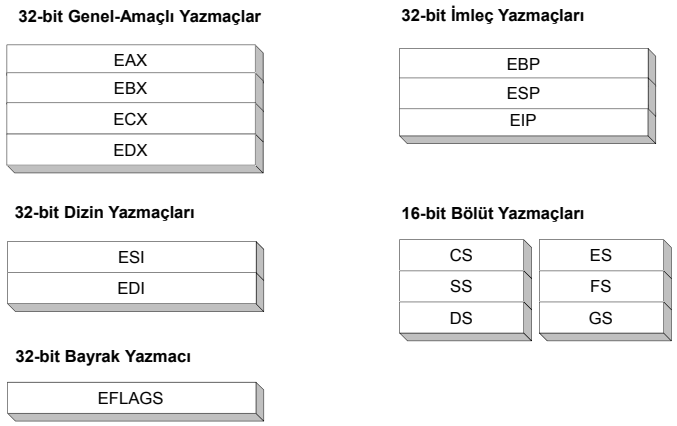
8086 İşlemcisi



8086 Yazmaçları (Registers)



Intel 32 Bit Mimarisi Yazmaçları



8086 Yazmaç Çeşitleri

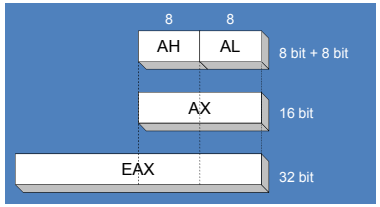
- Genel-amaçlı yazmaçlar ("general-purpose registers"): Girdi/çıktı, artimetik, sayma gibi genel amaçlı işlemlerde kullanılır (AX, BX, CX, DX).
- Bölüt yazmaçları ("segment registers"): Bellek bölütünü adreslemek için kullanılır (CS, DS, SS, ES, FS, GS).
  - 80386/80486/Pentium: Gerçek-adres modunda 16-bit adresleme, korumalı modda 48-bit adresleme
- İmleç yazmaçları ("pointer registers"): Ofset adresleri için kullanılır (IP, SP, BP)
  - Örnek: CS:IP, SS:SP, SS:BP, BP:SI, BP:DI
- Dizin yazmaçları ("index registers"): Genellikle karakter işleme işlemlerinde kaynak ve hedefi adreslemek için kullanılır (SI, DI).
  - Örnek: DS:SI, ES:DI
- Bayrak yazmacı ("flags register"): Çeşitli işlemlerin sonuçlarını taşımak için kullanılır (OF, DF, IF, TF, SF, ZF, AF, PF, CF).

Genel-Amaçlı Yazmaçlar ("General-Purpose Registers")

- AX ("primary accumulator") yazmacı
  - Genellikle girdi/çıktı ve aritmetik işlemler için kullanılır.
  - Örnek: Çarpma ve bölme işlemleri AX yazmacının kullanılmasını bekler.
- BX ("base") yazmacı
  - Adres genişletmek için kullanılabilen tek genel-amaçlı yazmaçtır. Hesaplamalarda da kullanılır.
  - Dizin yazmaçlarıyla birlikte (BX:SI, BX:DI) özel adresleme için taban yazmacı olarak kullanılır.
- CX ("count") yazmacı
  - Döngü sayma ya da bitleri sağa/sola kaydırma işlemlerinde ve hesaplamalarda kullanılır.
- DX ("data") yazmacı
  - Girdi/çıktı işlemlerinde ve bazı aritmetik işlemlerde kullanılır.
  - Örnek: Çarpma ve bölme işlemleri AX ile birlikte DX yazmacının kullanılmasını bekler.

Genel-Amaçlı Yazmaçların Bölümlerine Ulaşmak

- 8-bit, 16-bit veya 32-bit adlar kullanılabilir.
- Sadece EAX, EBX, ECX ve EDX yazmaçları için uygulanabilir.



32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

Bölüt Yazmaçları (“Segment Registers”)

- CS (“code segment”) yazmacı
  - Programın kod bölütünün başlangıç adresini içerir. Bu adresle birlikte IP yazmaç değeri (CS:IP), işletilecek bir sonraki komutun adresini gösterir.
    - Normal şartlarda programcı tarafından direkt ulaşılmasına gerek yoktur.
- DS (“data segment”) yazmacı
  - Programın veri bölütünün başlangıç adresini içerir. Bu adresle birlikte komut içindeki ofset değeri (DS:ofset), veriye ulaşılacak adresi gösterir.
- SS (“stack segment”) yazmacı
  - Programın yığıt bölütünün başlangıç adresini içerir. Bu adresle birlikte SP yazmaç değeri (SS:SP), yığıtta ulaşılacak sözcüğün adresini gösterir.
    - Normal şartlarda programcı tarafından direkt ulaşılmasına gerek yoktur.
- ES (“extra segment”) yazmacı
  - Bazı karakter işleme işlemlerinde bellek adresleme için kullanılır (ES:DI).
    - Kullanmadan önce değeri programcı tarafından uygun şekilde atanmalıdır.
- FS ve GS yazmaçları
  - Ek bölüt yazmaçlarıdır. Saklama gereksinimlerini karşılamak için 80386 ile tanıtılmıştır.

Adresleme

- Sözcük adresleme:

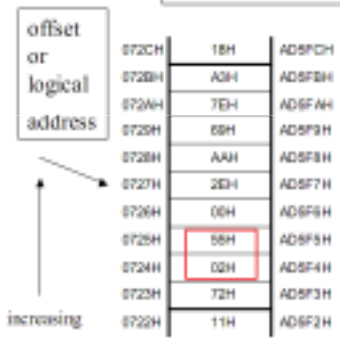
ADRES + 0000H	ADRES + 0001H
Alt Bayt	Üst Bayt

- İkili sözcük adresleme:

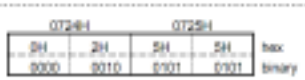
ADRES + 0000H	ADRES + 0001H	ADRES + 0002H	ADRES + 0003H
0. Bayt (En düşük öncelikli bayt)	1. Bayt	2. Bayt	3. Bayt (En yüksek öncelikli bayt)

Bölüt İçi Adresleme

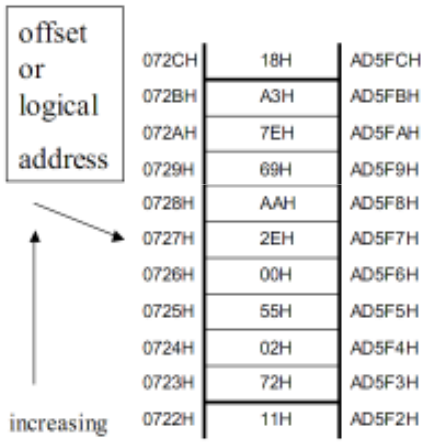
- 1 Word = 16 bits
- *Little Endian* Arrangement
  - MSB (Most Significant Byte) at Higher Address



- Segment Address = ACEDH
- Logical Address = 0724H
- Physical Address
  = ACED0H + 0724H
  = AD5F4H
- M[ACED:0724]
  = M[AD5F4] (word read)
  = 5502H



Bellekten Okuma



Byte Read:
al ← M[0724h]
after: al = 02h

Word Read:
ax ← M[0724h]
after: ax = 5502h

Dword Read
eax ← M[0724h]
after: eax = 2E005502h

Belleğe Byte Yazma

Assume EAX = FAC4237B h

Byte Write: M[0724h] ← al

before:

072CH	18H	AD5FCH
072BH	A3H	AD5FBH
072AH	7EH	AD5FAH
0729H	69H	AD5F9H
0728H	AAH	AD5F8H
0727H	2EH	AD5F7H
0726H	00H	AD5F6H
0725H	55H	AD5F5H
0724H	02H	AD5F4H
0723H	72H	AD5F3H
0722H	11H	AD5F2H

after:

072CH	18H	AD5FCH
072BH	A3H	AD5FBH
072AH	7EH	AD5FAH
0729H	69H	AD5F9H
0728H	AAH	AD5F8H
0727H	2EH	AD5F7H
0726H	00H	AD5F6H
0725H	55H	AD5F5H
0724H	7BH	AD5F4H
0723H	72H	AD5F3H
0722H	11H	AD5F2H

Belleğe Sözcük (Word:16 bit) Yazma

Assume EAX = FAC4237B h

Word Write: M[0724h] ← ax

before:

072CH	18H	AD5FCH
072BH	A3H	AD5FBH
072AH	7EH	AD5FAH
0729H	69H	AD5F9H
0728H	AAH	AD5F8H
0727H	2EH	AD5F7H
0726H	00H	AD5F6H
0725H	55H	AD5F5H
0724H	02H	AD5F4H
0723H	72H	AD5F3H
0722H	11H	AD5F2H

after:

072CH	18H	AD5FCH
072BH	A3H	AD5FBH
072AH	7EH	AD5FAH
0729H	69H	AD5F9H
0728H	AAH	AD5F8H
0727H	2EH	AD5F7H
0726H	00H	AD5F6H
0725H	23H	AD5F5H
0724H	7BH	AD5F4H
0723H	72H	AD5F3H
0722H	11H	AD5F2H

Belleğe İkili Sözcük (Double Word: 32bit) Yazma

Assume EAX = FAC4237B h

DWord Write: M[0724h] ← eax

before:

072CH	18H	AD5FCH
072BH	A3H	AD5FBH
072AH	7EH	AD5FAH
0729H	69H	AD5F9H
0728H	AAH	AD5F8H
0727H	2EH	AD5F7H
0726H	00H	AD5F6H
0725H	55H	AD5F5H
0724H	02H	AD5F4H
0723H	72H	AD5F3H
0722H	11H	AD5F2H

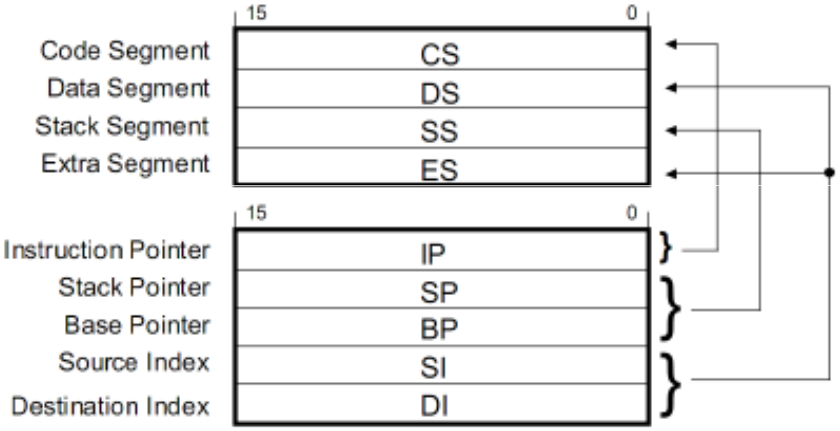
after:

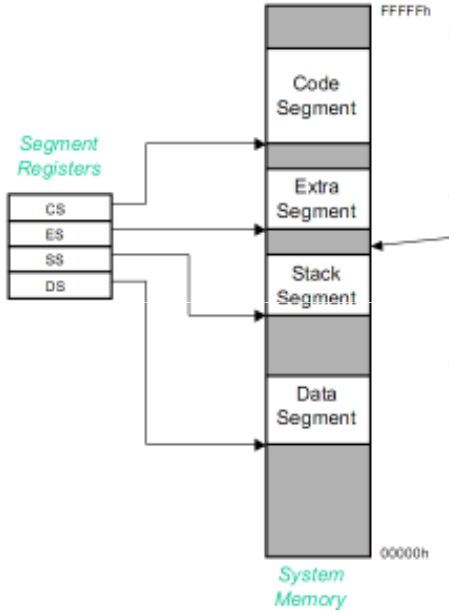
072CH	18H	AD5FCH
072BH	A3H	AD5FBH
072AH	7EH	AD5FAH
0729H	69H	AD5F9H
0728H	AAH	AD5F8H
0727H	FAH	AD5F7H
0726H	C4H	AD5F6H
0725H	23H	AD5F5H
0724H	7BH	AD5F4H
0723H	72H	AD5F3H
0722H	11H	AD5F2H

İmleç Yazmaçları (“Pointer Registers”)

- IP (“instruction pointer”) yazmacı
  - Çalıştırılacak bir sonraki komutun ofset adresini içerir. CS yazmacı ile birlikte (CS:IP), çalışan programın kod bölütünde işletilen komutu gösterir.
  - Komutlar işletildikçe IP ofset değeri sistem tarafından otomatik olarak arttırılır.
- SP (“stack pointer”) yazmacı
  - Yığıtta işlem gören sözcüğün ofset adresini içerir. SS yazmacı ile birlikte (SS:SP), çalışan programın yığıt bölütünde işlem gören sözcüğü gösterir.
  - Sözcükler işlendikçe SP ofset değeri sistem tarafından otomatik olarak azaltılır (“PUSH” işlevi) ya da arttırılır (“POP” işlevi).
- BP (“base pointer”) yazmacı
  - SS yazmacı ile birlikte (SS:BP), programın yığıt aracılığıyla geçirdiği veri veya adres parametrelerine ulaşmak için kullanılır.
  - Taban yazmacı olarak, DI ve SI yazmaçlarıyla birlikte (BP:DI, BP:SI), özel adresleme için kullanılır.

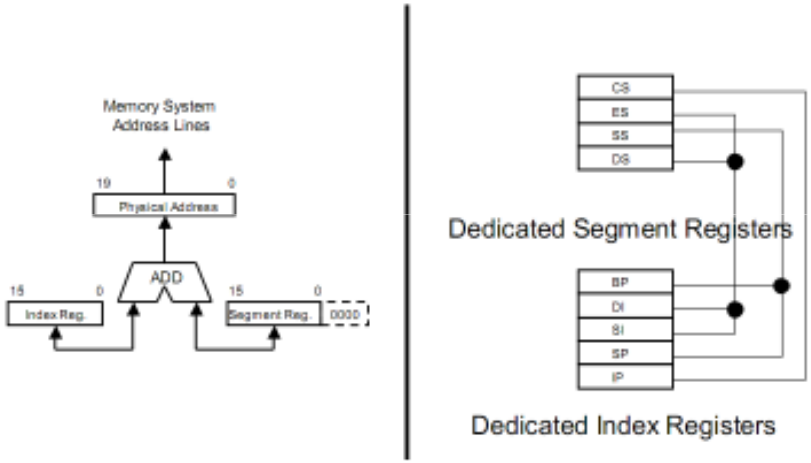
Dizin (“Index”) ve İmleç (“Pointer”) Yazmaçları





- Segment Registers:
  - Point to Base Address
- Index Registers:
  - *fragmentation*
  - Contain Offset Value
- Notation
  - `CS:IP`
  - `DS:SI`
  - `ES:DI`
  - `SS:BP`
  - `SS:SP`

Bölüt ve Dizin Yazmaçları ile Adres Üretimi



Bölüt ve Dizin Yazmacı Kullanımı

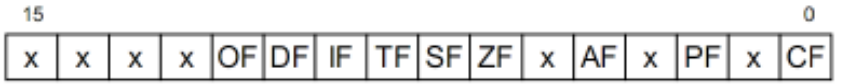
Type of Memory Reference	Default Segment Base	Alternate Segment Base	Offset
Instruction Fetch	CS	None	IP
Stack Operation	SS	None	SP
Variable (except following)	DS	CS, ES, SS	Effective Address
- String Source	DS	CS, ES, SS	SI
- String Destination	ES	None	DI
- BP used as Base Register	SS	CS, DS, ES	Effective Address
- BX Used as Base Register	DS	CS, ES, SS	Effective Address

32 bit Dizin ve İmleç Yazmaçları

- Bu yazmaçların sağ yarıları için 16-bit isimleri vardır.

32-bit	16-bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

Bayrak Yazmacı (“Flags Register- PSW”)



- Durum Bayrakları (“Status Flags”)
  - OF -- Taşma (“overflow”)
  - DF -- Yön (“direction”)
  - IF -- Kesme (“interrupt“)
  - TF -- Yakalama (“trap”)
  - SF -- İşaret (“sign”)
  - ZF -- Sıfır (“zero”)
  - AF -- Yardımcı elde (“auxilary carry”)
  - PF -- Eşlik (“parity”)
  - CF -- Elde (“carry”)

Komutların Çalıştırılması - 1

- Çevirici diliyle yazılmış bir kod kesiminin çalıştırılmasını inceleyim:

```
MOV AL, 'S'
MOV BL, AL
MOV AL, 'C'
MOV BX, 1234H
```

- Bu kod kesimi çevirici tarafından makina komutlarına çevrilir.

Komutların Çalıştırılması - 2

- Bu kod kesimine ilişkin ana bellekte oluşturulan sayısal veriler şöyledir:

Bellek Adresi	Adreste Saklanan Veri	Komut
0000:0000	B0H	MOV AL
0000:0001	53H	‘S’
0000:0002	8AH	MOV BL
0000:0003	D8H	AL
0000:0004	B0H	MOV AL
0000:0005	43H	‘C’
0000:0006	BBH	MOV BX
0000:0007	34H	34H
0000:0008	12H	12H

Komutların Çalıştırılması - 3

- Kodların doğru adresten başlanarak çalıştırılması önemlidir!
- İşlemci, önceki sayfada verilen kod kesimini çalıştırmaya, 0000:0000 adresi yerine 0000:0001 başlarsa, işletim şu şekilde değişir.

Bellek Adresi	Adreste Saklanan Veri	Komut
0000:0001	53H	PUSH BX
0000:0002	8AH	MOV BL
0000:0003	D8H	AL
0000:0004	B0H	MOV AL
0000:0005	43H	'C'
0000:0006	BBH	MOV BX
0000:0007	34H	34H
0000:0008	12H	12H

Komutların Çalıştırılması - 4

- Eğer işlemci yanlış adresten işletim başlarsa, kod kesimi aşağıdaki şekilde değişmiş olur:

```
MOV AL, 'S'
MOV BL, AL
MOV AL, 'C'
MOV BX, 1234H
```

→

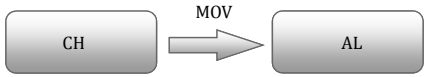
```
PUSH BX
MOV BL, AL
MOV AL, 'C'
MOV BX, 1234H
```


Adresleme Modları

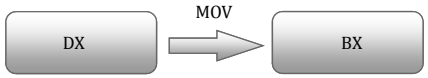
- Yazmaç Adresleme Modları
- Bellek Adresleme Modları

Yazmaç Adresleme Modları

- Bir yazmaçtan diğerine aktarımda, 8 bit veya 16 bit aktarımlar yapılabilir.
- Örnek: MOV AL, CH
  - Bu komut ile CH yazmacındaki veri AL yazmacına kaydedilmektedir.



- Örnek: MOV BX, DX
  - Bu komut ile DX yazmacındaki veri BX yazmacına aktarılır.

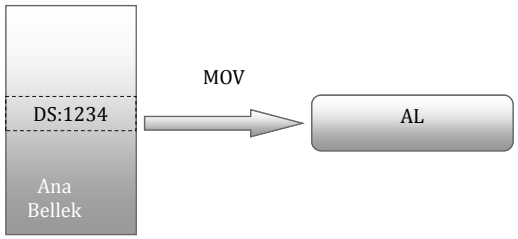


Bellek Adresleme Modları

- Ana belleğe erişimlerde çeşitli adresleme modları vardır:
  - Doğrudan Adresleme Modu
  - Yazmaç Değeriyle Dolaylı Adresleme Modu
  - Eklemeli Dizinle Adresleme Modu
  - Tabanlı Dizinle Adresleme Modu
  - Tabanlı ve Eklemeli Dizinle Adresleme Modu

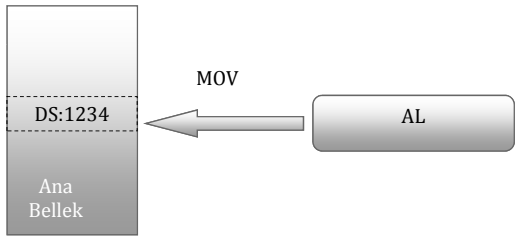
Doğrudan Adresleme Modu

- Doğrudan adresleme modunda, parametere olarak verilen 16 bitlik sabit bir değer adres olarak kullanılarak belleğe erişilir.
- Örnek:
  - Tek baytlık aktarım: MOV AL, DS:[1234H]



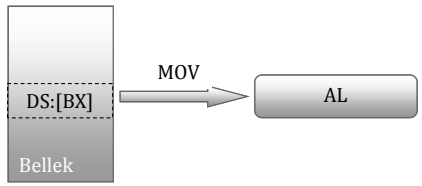
Doğrudan Adresleme Modu - 2

- Örnek:
  - Bir sözcüklük aktarım: MOV AX, DS:[1234H]



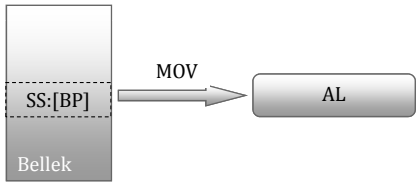
Yazmaç Değeriyle Dolaylı Adresleme Modu -1

- Bu modda, bir yazmacın değeri adres olarak kullanılarak herhangi bir bellek adresindeki verilere erişilir.
  - Bu modda, parametre verilen yazmaca göre, bir bölüt adresi yazmacı dolaylı olarak kullanılır.
- Örnek: MOV AL, [BX]
  - Bu komutta, parametre verilen BX yazmacı ile birlikte DS yazmacı kullanılır.



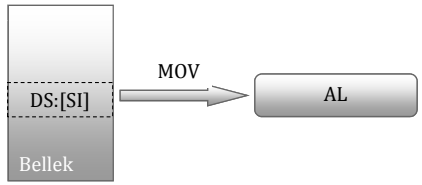
Yazmaç Değeriyle Dolaylı Adresleme Modu -2

- Örnek: MOV AL, [BP]
  - Bu komutta, parametre verilen BP yazmacı ile birlikte SS yazmacı kullanılır. SS ile belirtilen yığıt kesimi içinde erişilecek göreli adresi, BP yazmacı ile belirlenir.



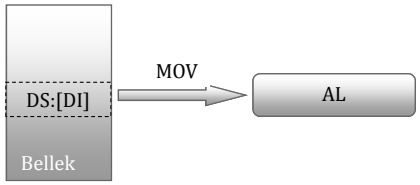
Yazmaç Değeriyle Dolaylı Adresleme Modu - 3

- Örnek: MOV AL, [SI]
  - Bu komutta, parametre verilen SI yazmacı ile birlikte DS yazmacı kullanılır. DS ile belirtilen veri kesimi içinde erişilecek göreli adresi, SI yazmacı ile belirlenir.



Yazmaç Değeriyle Dolaylı Adresleme Modu - 4

- Örnek: MOV AL, [DI]
  - Bu komutta, parametre verilen DI yazmacı ile birlikte DS yazmacı kullanılır. DS ile belirtilen veri kesimi içinde erişilecek göreli adresi, DI yazmacı ile belirlenir.



Yazmaç Değeriyle Dolaylı Adresleme Modu - 5

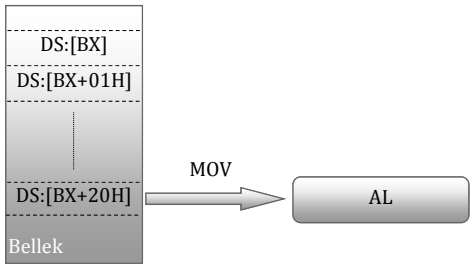
- Eğer varsayılan bölüt adresi yazmacı ([BP] için SS yazmacı, [BX], [SI] ve [DI] için DS yazmacı) yerine, başka bir bölüt adresi yazmacının kullanılması istenirse satırın başında istenilen bölüt yazmacı açıkça belirtilmelidir.
- Örnek:

```
CS MOV AL, [BX]
ES MOV AL, [BP]
DS MOV AL, [SI]
SS MOV AL, [DI]
```

  - Bu örnekte verilen kod, FASM çeviricisi ile bölüt adres yazmacını değiştirme yöntemidir. Başka çeviricilerde, farklı gösterimler olabilir.

Eklemeli Dizinle Adresleme Modu – 1

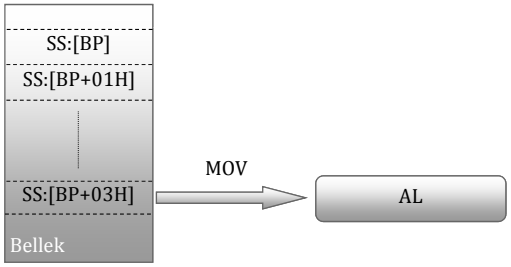
- Bu modda, yazmaçla dolaylı adreslemeye ek olarak, parametre olarak verilen bir sayıyla birlikte, bölüt içi göreli adres oluşturulur ve yazmaçla ilişkilendirilmiş bölüte erişilir.
- Örnek: MOV AL, [BX+20H]



Eklemeli Dizinle Adresleme Modu - 2

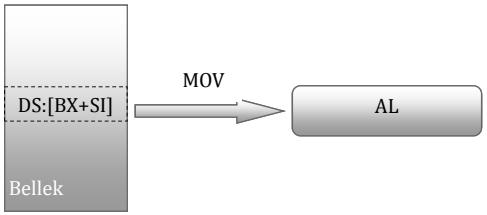
- Örnek:

MOV AL,[BP+03H]



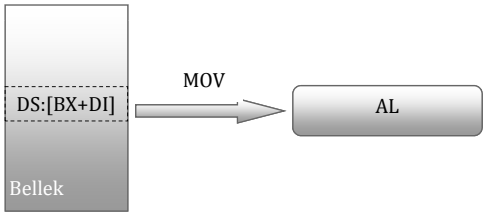
Tabanlı Dizinle Adresleme Modu - 1

- Tabanlı dizinle adresleme kipinde taban yazmaçları olarak adlandırılan BX ya da BP yazmaçlarının değerleri ilk taban değerini oluşturup, diğer bir yazmacın değerinin bu taban değerine eklenmesi ile bölüt içi göreceli adres belirlenir.
- Örnek: MOV AL, [BX+SI]
  - Taban yazmacı için BX kullanıldığından, DS yazmacı ile gösterilen veri bölütüne erişilir.



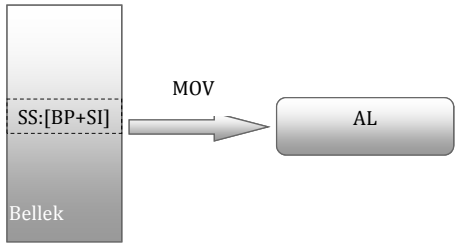
Tabanlı Dizinle Adresleme Modu - 2

- Örnek: MOV AL, [BX+DI]
  - Taban yazmacı için BX kullanıldığından, DS yazmacı ile gösterilen veri bölütüne erişilir.



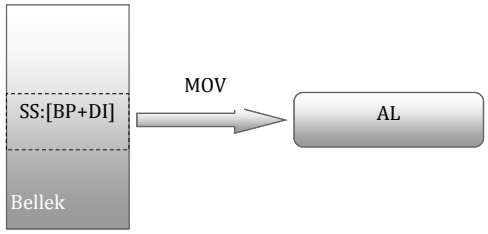
Tabanlı Dizinle Adresleme Modu - 3

- Örnek: MOV AL, [BP+SI]
  - Taban yazmacı için BP kullanıldığından, SS yazmacı ile gösterilen yığıt bölütüne erişilir.



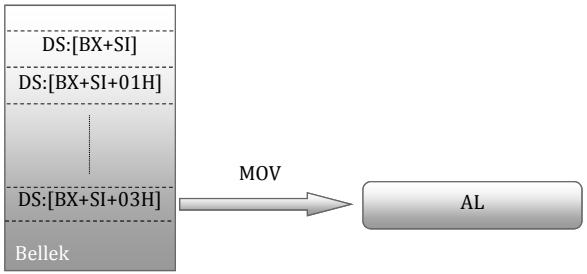
Tabanlı Dizinle Adresleme Modu - 4

- Örnek: MOV AL, [BP+DI]
  - Taban yazmacı için BP kullanıldığından, SS yazmacı ile gösterilen yığıt bölütüne erişilir.



Tabanlı ve Eklemeli Dizinle Adresleme Modu - 1

- Bu modda, hem taban yazmaçlarının değerleri, hem de eklemeler kullanılır.
- Örnek: MOV AL, [BX+SI+03H]
  - Taban yazmacı için BX kullanıldığından, DS yazmacı ile gösterilen veri bölütüne erişilir.



Tabanlı ve Eklemeli Dizinle Adresleme Modu - 2

- Örnek: MOV AL, [BP+SI+25H]
  - Taban yazmacı için BP kullanıldığından, SS yazmacı ile gösterilen yığıt bölütüne erişilir.