MİKROİŞLEMCİ SİSTEMLERİ

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

Adresleme Yöntemleri

- İşlenenin nerde olacağını belirtmek için kullanılır.
- Buyruk çözme aşamasında adresleme yöntemi belirlenir ve işlenenin nerede bulunacağı hesaplanır.
- Genel olarak 6 temel adresleme yöntemi vardır.
- Bir mikroişlemcide bulunan adresleme yöntemlerinin sayısının çokluğu yüksek düzeyli dillerdeki karmaşık işlemleri daha kolay yerine getirmesini sağlar.

Örnek MİB ile Adresleme

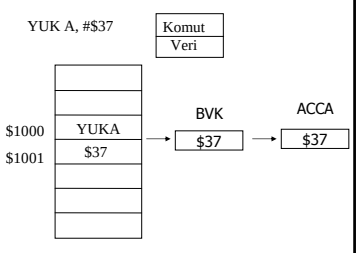
- Bir mikroişlemcinin adresleme yeteneği adres yollarını sayısı ile sınırlıdır.
  - 16 bit ->2^{16} farklı adres -> 65.536 byte
- Bellek adresleri, doğrudan buyrukta verilebilir ya da bir işaretçiden (başlangıç adresi) kaç adım ileride veya geride olduğu belirtilir.

Adresleme Yöntemleri

- Temel adresleme yöntemleri
  - İvedi Adresleme
  - Doğal Adresleme
  - Doğrudan Adresleme
  - Dolaylı Adresleme
  - Sıralı Adresleme
  - Bağıl Adresleme

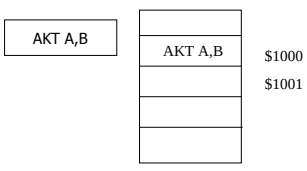
İvedi Adresleme

- İşlenen yerine yazılan bilgi bir adres değil, işlenenin kendisidir.
- Bellek okuma yazma işlemi yapılmaz.
- Hızlı
- Örnek:
  YUK B, $41
  YUK SK, #1000



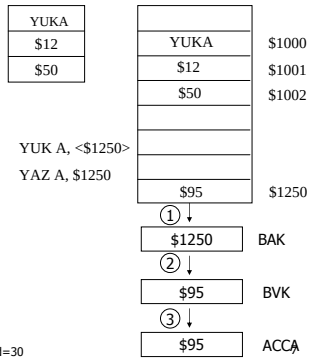
Doğal Adresleme

- Buyrukta bellek adresi kullanılmaz.
- İşlenen alanında bulunan bilgi, işlenenin bulunduğu kütüğün adıdır.
- Sınırlı sayıda kütük adresleme
- Hızlı işlem
- Örnek:
  AKT SK, CD SK←CD
  AKT B, C A ←B
  AKT CD, AB C←A;D←B



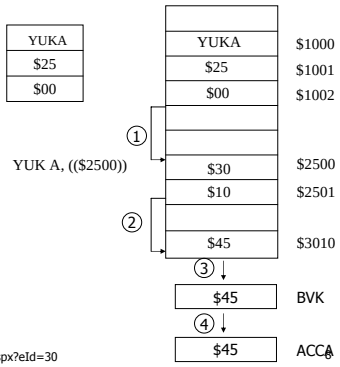
Doğrudan Adresleme

- İşlenen yerinde, işlenecek verinin bulunduğu ya da bulunucağı bellek gözünün adresi yazılıdır.
- Veriye ulaşmak için sadece bir bellek işlemi
- Etkin adres hesaplama yapılmaz



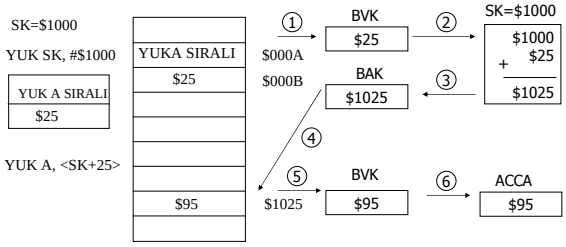
Dolaylı Adresleme

- İşlenen yerinde verinin yazılı olduğu bellek gözünün adresi bulunmaktadır.
- İşlenene ulaşmak için bellek işlemi fazla
- Yavaş
- Bağlantılı liste, kuyruk gibi işaretçi kullanılan veri yapılarını yaratmak için kullanılır.



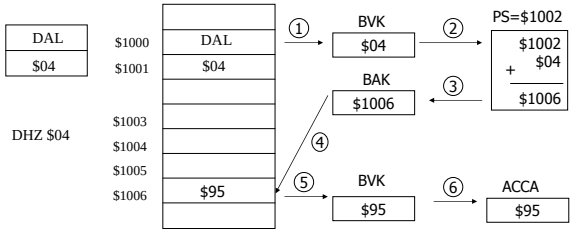
Sıralı Adresleme

- MİB içinde bulunan sıralama kütüğüne dizinin ilk elemanının adresi yazılır. Veriler üzerinde işlem yaparken verinin sıra numarasını belirtmek yeterli olur.



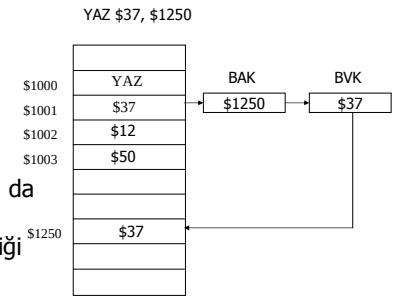
Bağıl Adresleme

- Bu adresleme yöntemi dallanma komutları için kullanılır. İşlenen yerinde görülen sayı bir sonraki komutun program sayacındaki adresten ne kadar uzakta olduğunu gösterir. Dallanma komutları durum kütüğündeki bazı bitlere göre koşullu veya koşulsuz (DHZ) olabilir.



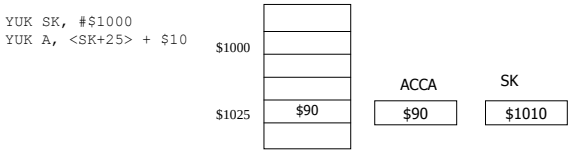
Gelişmiş Adresleme Yöntemleri

- Belleğe İvedi Adresleme:
  - Veri doğrudan bellekte bir adrese yazılır.
  - Akümülatör ya da yardımcı kütüklerin içeriği bozulmaz.



Gelişmiş Adresleme Yöntemleri

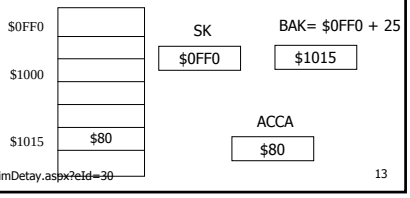
- Artırmalı Sıralı Adresleme
  - Sıralı adreslemede olduğu gibi etkin adres hesaplanır, ardından sıralama kütüğü içeriği buyrukta belirtilen değer kadar artırılır.
  - Artırma değeri $00-$FF arasında olabilir.



Gelişmiş Adresleme Yöntemleri

- Azaltmalı Sıralı Adresleme
  - Etkin adres hesaplanmadan önce sıralama kütüğünün değeri buyrukta verilen değer kadar azaltılır.
  - Azaltma değeri $00-$FF arasında olabilir.

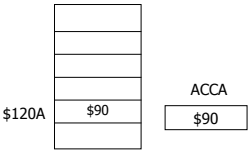
```
YUK SK, #$1000
YUK A, <SK+25> - $10
```



Gelişmiş Adresleme Yöntemleri

- Kütüğe Bağlı Sıralı Adresleme:
  - Etkin adres yardımcı kütüklerin, sıralama kütüğünün içeriklerinin ve buyrukta verilen değerin toplanması ile hesaplanır.
  - Yardımcı kütükler ile sıralama kütüğünün boyu aynı olmalıdır.

YUK A, <CD+SK+$05>
C: $02, D:$05, SK:$1000
A ← <$120A>



Buyruk Kümeleri

- Veri aktarma buyrukları
- Aritmetik işlem buyrukları
- Lojik işlem buyrukları
- İşlem buyrukları
- Öteleme Buyrukları
- Karar verme dallanma buyrukları
- Giriş-çıkış buyrukları

Buyruk Kümeleri

		İvedi	Doğal	Doğruda	Sıralı	Bağıl
Veri Akt.	Aktarma		X			
	Yükleme	X		X	X	
	Yazma			X	X	
	Takas		X			
Aritmetik	Toplama	X	X	X	X	
	Eldeli Top.	X	X	X	X	
	Çıkarma	X	X	X	X	
	Eldeli Çık.	X	X	X	X	
	Çarpma	X	X	X	X	
	Bölme	X	X	X	X	
Lojik	VE	X	X	X	X	
	VEYA	X	X	X	X	
	YADA	X	X	X	X	
İşlem	Silme		X	X	X	
	Artırma		X	X	X	
	Eksiltme		X	X	X	
	Kurma		X			
	Öteleme		X	X	X	
	Döndürme		X	X	X	
	Onluk Ayarı		X			
	Tümleme		X	X	X	
	Eksileme		X	X	X	
Karar	Karşılaştırma	X	X	X	X	
verme	Koşullu Dal.					X
dallanma	Koşulsuz Dal.					X
	Koşullu Bağ.			X		
bağlanma	Altprg. Dal.			X	X	X

Veri Aktarma Buyrukları

- Aktarma: MİB içindeki kütüklerin içeriklerini birbirine aktarılması için kullanılır.

AKT K_i, K_j $K_i \leftarrow K_j$

AKT A,B
AKT A,C
AKT B,DK
AKT SK,YG

Veri Aktarma Buyrukları

- Yükleme: Bir verinin veya bir bellek gözünde bulunan bir verinin akümülatöre veya kütüklere aktarılma işlemidir. Kütüğün boyu ve verinin boyu her zaman eşit olmalıdır. Yönü her zaman MİB'ne doğrudur.

YUK K_i, <BELLEK> $K_i \leftarrow$ <BELLEK>

YUK A, <$1000>	ACCA ← <$1000>
YUK C, <$2000>	C ← <$2000>
YUK B, #$25	B ← $25
YUK SK, <$A000>	SK ← <$A000> + <$A001>
YUK SK, #$C000	SK ← $C000

Veri Aktarma Buyrukları

- Yazma: MİB içindeki kütüklerin içerikleri bellek gözlerine aktarılır.

YAZ K_i, <BELLEK>	<BELLEK> ← K_i
YAZ A, $1000	$1000 ← ACC A
YAZ C, $E000	$E000 ← C
YAZ SK, $C000	$C000 + $C001 ← SK

Veri Aktarma Buyrukları

- Takas: MİB içindeki bazı kütüklerin içeriklerini birbirleri ile takas etmelerini sağlar.

TKS K_i, K_j	Ki ⟷ Kj
TKS A, B	ACC A ⟷ ACC B
TKS C, D	C ⟷ D
TKS SK, YG	SK ⟷ YG

Aritmetik İşlem Buyrukları

- Toplama: İki akümülatörün içeriği, bir akümülatör ile bir yardımcı kütüğün içeriği, bir akümülatörün içeriği ile bir veri veya bir akümülatör ile bir bellek gözünün içeriği eldeli veya eldesiz olarak toplanabilir. Sonuç buyrukta birinci işlenen durumunda olan akümülatöre yazılır.

TOP A, B	ACCA ←	ACCA+ACCB
TOP A, Ki	ACCA ←	ACCA+K_i
TOP A, VERİ	ACCA ←	ACCA+VERİ
TOPA, <BELLEK>	ACCA ←	ACCA+ <BELLEK>

- Eldeli Toplama

TOPE A, B	ACCA ←	ACCA+ACCB+E
TOPE A, Ki	ACCA ←	ACCA+K_i+E
TOPE A, VERİ	ACCA ←	ACCA+VERİ+E
TOPE A, <BELLEK>	ACCA ←	ACCA+ <BELLEK>+E

Toplama İşlemi

TOP A, B	ACC A ← ACC A + ACC B
TOPE A, B	ACC A ← ACC A + ACC B + E
TOP A, #$25	ACC A ← ACC A +$25
TOP A, <$1000>	ACC A ← ACC A + <$1000>
TOPE A, <SK+10>	ACC A ← ACC A + <SK+10> + E

Toplama İşlemi

- Örnek: Bellekteki iki sayıyı (N1+N2) toplayıp sonucu (S) bir bellek gözüne yazan program. Programın başlangıç adresi $c000

N1->$c200
N2->$c201
S->$c202

	Adres	Bellek
YÜK A, <$C200>	C000	00
	C001	20
	C002	C2
	C003	00
YÜK B, <$C201>	C004	00
	C005	21
	C006	C2
	C007	01
TOP A,B	C008	43
	C009	01
YAZ A, <$C202>	C00A	01
	C00B	20
	C00C	C2
	C00D	02
	:	:
N1	C200	28
N2	C201	55
S	C202	7D

Toplama İşlemi

- Örnek: X=$40A0 ve Y=$1BC1 sayılarının toplanması

<10>	<11>	X	<10>:0100 0000
<12>	<13>	Y	<11>:1010 0000
+			<12>:0001 1011
<14>	<15>	SONUÇ	<13>:1100 0001

YUK A, <$0010>	ACCA ←	0100 0000
YUK B, <$0011>	ACCB ←	1010 0000
TOP B, <$0013>	ACCB ←	0110 0001 E=1
TOPE A, <$0012>	ACCA ←	0101 1100
YAZ A, <$0014>	$0014 ←	0101 1100
YAZ B, <$0015>	$0015 ←	0110 0001

Aritmetik İşlem Buyrukları

- Çıkarma: İki akümülatörün içeriği, bir akümülatör ile bir yardımcı kütüğün içeriği veya bir akümülatör ile bir bellek gözünün içeriği eldeli veya eldesiz olarak birbirinden çıkarılabilir. Sonuç buyrukta birinci işlenen durumunda olan akümülatöre yazılır.

ÇIK A, B	ACCA	←	ACCA-ACCB
ÇIK A, Ki	ACCA	←	ACCA-K_i
ÇIK A, VERİ	ACCA	←	ACCA-VERİ
ÇIK A, <BELLEK>	ACCA	←	ACCA- <BELLEK>

- Borçlu Çıkarma

ÇIK E A, B	ACCA	←	ACCA-ACCB-E
ÇIK E A, Ki	ACCA	←	ACCA-K_i-E
ÇIK E A, VERİ	ACCA	←	ACCA-VERİ-E
ÇIK E A, <BELLEK>	ACCA	←	ACCA-<BELLEK>-E

Aritmetik İşlem Buyrukları

- Çarpma:
  - Birinci işlenen A akümülatörü olmalıdır.
  - İkinci işlenen, B akümülatöründe, yardımcı kütüklerden birinde, bir bellek gözünde bulunabilir ya da bir veri olabilir.
  - İşlem sonucu A ve B akümülatör çiftinde yer alır.
  - İşlenenlerin işaretsiz olması gerekir.

ÇAR A, B	ACCA + ACCB	←	ACCA * ACCB
ÇAR A, Ki	ACCA + ACCB	←	ACCA * K_i
ÇAR A, VERİ	ACCA + ACCB	←	ACCA * VERİ
ÇAR A, <BELLEK>	ACCA + ACCB	←	ACCA * <BELLEK>

Aritmetik İşlem Buyrukları

- Bölme:
  - Birinci işlenen A ve B akümülatör çiftinde bulunur.
  - İkinci işlenen yardımcı kütüklerden birinde, bir bellek gözünde bulunabilir ya da bir veri olabilir.
  - Sonuç A ve B akümülatörlerinde, kalan C yardımcı kütüğünde oluşur.
  - İşlenenlerin işaretsiz olması gerekir.
  - 0'a bölme durumunda taşma bayrağı 1 olur.

BÖL AB, Ki	ACCA + ACCB	←	<ACCA + ACCB> / Ki
BÖL AB, VERİ	ACCA + ACCB	←	<ACCA + ACCB> / VERİ
BÖL AB, <BELLEK>	ACCA + ACCB	←	<ACCA +ACCB>/<BELLEK>

Örnek

- Bellekte $1000 ve $1001 bellek gözlerinde bulunan sayı $1002 bellek gözünde bulunan sayıya bölünecek, sonuç $1005-$1006 bellek gözlerine, kalan ise $1007 bellek gözüne yazılacaktır.

```
başla   yük sk, $1000
        yük a, <sk+0> + $01
        yük b, <sk+0> + $01
        böl ab, <sk+0>
        yaz ab, <$1005>
        yaz c, <$1007>
```


Örnek

- Bellekte $1000 ve $1001 bellek gözlerinde bulunan sayılar çarpılarak $1002 bellek gözünde bulunan sayıya bölünecek, sonuç $1005-$1006 bellek gözlerine, kalan ise $1007 bellek gözüne yazılacaktır.

```
başla   yük sk, $1000
        yük a, <sk+0> + $01
        çar a, <sk+0> + $01
        böl ab, <sk+0>
        yük sk, $1005
        yaz ab, <sk+0> + $02
        yaz c, <sk+0>
```


Lojik İşlem Buyrukları

- VE , VEYA, YADA: Lojik buyruklarda birinci işlenen akümülatör olmak zorundadır. Bir akümülatörün içeriği bir veri, diğer bir akümülatör, yardımcı kütük veya bir bellek gözünün içeriği ile lojik işleme sokulabilir.

İŞLEM A, VERİ	ACCA	←	ACCA.İŞLEM.VERİ
İŞLEM A, B	ACCA	←	ACCA.İŞLEM.ACCB
İŞLEM A, Ki	ACCA	←	ACCA.İŞLEM.Ki
İŞLEM A, <BELLEK>	ACCA	←	ACCA.İŞLEM. <BELLEK>

VE A, $25	ACC A	←	ACC A .VE. $25
VEYA A, B	ACC A	←	ACC A .VEYA. B
YADA A, <$1000>	ACC A	←	ACC A .YADA. <1000>