
MSC-1 Ailesi

Mikrodenetleyicilerin Komut Kümesi

Ekim 2008
Yar. Doç. Dr. Mustafa Engin

Mikroişlemci Programlama

- Mikroişlemci ikilik komutlar kabul eder ve sonuçlarını ikilik olarak kullanıcıya verir.
- İkilik komutlarla program yazma tekniğine **Makine dili** adı verilir.
- Kullanımı zor olduğu için makine dili yerine konuşma diline yakın komutlarla program yazma tekniği ortaya çıkmış ve adına **Assembly dili** adı verilmiştir.
- Assembly dilinin dilbilgisi kurallarına **adresleme kipi** adı verilir.
- Adresleme kipleri komutların hangi yazaç veya bellek alanında işlem yapacağını, sonucu nereye yazacağını ve bir sonraki komutun hangi adresten okunacağını belirtir.

8051'in Adresleme Kipleri

8051'in veri işleyen adresleme kipleri.

- Doğrudan
- İvedi
- Yazaç
- Dolaylı
- Sıralı

8051'in Program akışını düzenleyen adresleme kipleri.

- Bağlı
- Mutlak
- Uzun

Doğrudan (Direct) Adresleme

8051'in iç veri belleğinde alt 128 bayt ve SFR bölgesinde veri işleyen komutlarda kullanılır.

Komut **8 bit adres**

Örnek:

Mov A, 20h ;(20h) → A

Mov A, 0F0h ;(0F0h) → A

Mov 20h, A ; A → (20h)

Mov 0F0h, A ;A → (0F0h)

MoV A, P1 ;P1 → A

Not: P1 assembler programı tarafından doğrudan adresi ile değiştirilerek makine diline dönüştürülür.

İç RAM



İvedi (Immediate) Adresleme

8051'in iç veri belleğinde yer alan bellek satırlarına ve yazaçlara sabit sayı yüklemek için kullanılır.

Komut **# 8 Sayı**

“#” (diez olarak okunur) işareti İvedi adresleme belirteçidir.

Örnek:

Mov A, #20h	;A=20h
Mov A, #0F0h	;A=0F0h
Mov 20h, #02	;(20h)=02
Mov R0, #0	;R0=0
Mov P1, #55	;P1 = 55

Yazaç (Register) Adresleme

8051'in yazaçları ile akümülatör arasındaki veri işleme komutlarında kullanılır.

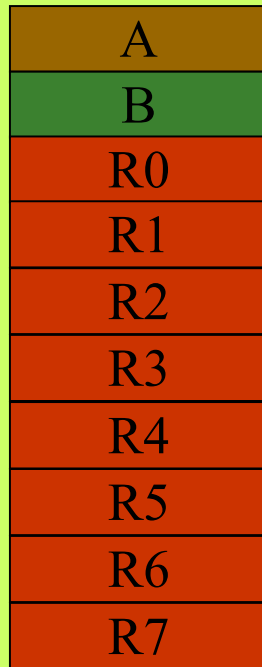
Komut **Rn**

n=0,1,2,3,4,5,6,7.

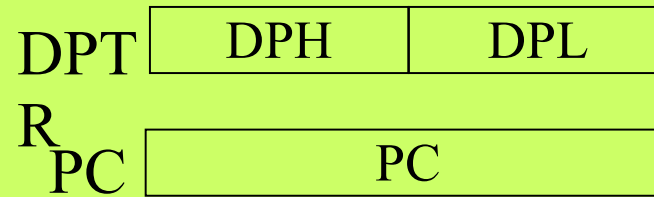
Örnek:

Mov A, R1	;R1 → A
Mov R0, A	;A → R0
Add A, R2	;A + R2 → A
Mov R0, #0	;R0=0
MoV R3, P1	;P1 → R3

Yazaçlar



8-bitt
Yazaçlar



16-bit Yazaçlar

Dolaylı (Indirect) Adresleme

8051'in üst 128 bayt ile yazaçlar ve SFR arasındaki veri işleme komutlarında ve ayrıca Akümülatör ile dış veri belleği arasında veri aktaran komutlarda kullanılır.

Komut **@R_i**

@ işareti dolay adreslemenin belirtecidir ve ed diye okunur.

i=0,1

Örnek:

Mov A, @R1	;	((R1)) → A
Mov @R0, A	;	A → ((R0))
Add A, @R0	;	A + ((R0)) → A
Mov @R0, #0	;	((R0)) = 0

Sıralı (Indexed) Adresleme

8051'in Program belleğinden veri okumak için kullanılır. Kod dönüşüm tablosu veya sensör doğrusallaştırma tabloları program belleğinde saklanır. Bu tablolara ***başvuru tablosu*** adı verilir.

- Bu adresleme sadece Movc komutunda kullanılır.
- DPTR yazıcı Tablonun başlangıç adresini içerir.
- A sıra numarasını
- A ile DPTR toplanır ve işlem yapılacak adres elde edilir.

Movc A,@A+DPTR ;((A+DPTR))→A

Movc A,@A+PC ;((A+PC)) →A

Bağıl (Relative)Adresleme

- 8051'in Program belleğinde program akışını değiştirmek için kullanılır.
- Genellikle bir koşula bağlı olarak komut sonrası yazılan Kayıklık değeri kadar ileri veya geri dallanma olur ve program o noktadan yürütülmeye devam eder.
- Koşul yerine gelmezse bir sonraki komuttan program yürütülmeye devam eder.

Komut 8 bit kayıklık

kayıklık= 00-7Fh aralığında ise ileri dallanma

kayıklık= 80-FFh aralığında ise geri dallanma

Örnek:

JNC, 05h ;C=0 ise 5 adım ileriden devam eder
;C=1 ise bir sonraki komuttan devam eder

- Sadece 1 komut koşulsuz dallanır, SJMP.

Mutlak (Absolute) Adresleme

- 8051'in Program belleğinde program akışını değiştirmek için kullanılır.
- 11 bit adres kullandığı için belleğin 2048 satırında etkilidir.
- Bu adresleme kipi Call ve Jmp komutlarında kullanılır.

ACall 11 bit Adres ;Altprogram çağırır
AJmp 11 bit Adres ;Adrese bağlanır

- ACall ve AJmp komutları 2 bayt uzunluğundadır

Uzun (Long) Adresleme

- 8051'in Program belleğinde program akışını değiştirmek için kullanılır.
- 16 bit adres kullandığı için belleğin tamamında etkilidir.
- Bu adresleme kipi Call ve Jmp komutlarında kullanılır.

LCall 16 bit Adres ;Altprogram çağırır

LJmp 16 bit Adres ;Adrese bağlanır

- ACall ve AJmp komutları 3 bayt uzunluğundadır

MCS-51'in Komut Kümesi

- Veri aktaran
- Aritmetik işlem yapan
- Mantık işlem yapan
- Boolean işlem yapan
- Dallanma ve Bağlanma

Veri Aktaran Komutlar

Mov Hedef, Kaynak ;Hedef ← Kaynak

Kaynağın içeriği hedefe kopyalanır. Kullanıldığı bellek bölümlerine göre sınıflandırırsak;

- İç veri Belleğinde Veri aktaran komutlar
- Dış veri Belleğinde Veri aktaran komutlar
- Program Belleğinden Veri aktaran komutlar

Veri Aktaran Komutlar

Mnemonic	İşlenen	Bayt/Sayı
MOV	A, Rn	1/1
	A, 8 bit adres	2/1
	A, @Ri	1/1
	A, #Say ₁	2/1
	Rn, A	1/1
	Rn, 8 bit adres	2/2
	Rn, #Say ₁	2/1
	8 bit adres, A	2/1
	8 bit adres, Rn	2/2
	8 bit adres, 8 bit adres	3/2
	8 bit adres, @Ri	2/2
	8 bit adres, #Say ₁	3/2

Veri Aktaran Komutlar

Mnemonic	İşlenen	Bayt/Sayk₁
MOV	@Ri, A	1/1
	@Ri, 8 bit adres	2/2
	@Ri, #say₁	2/1
	DPTR, #say₁16	3/2
	C, bit	2/1
	bit, C	2/2
MOVX	A, @DPTR	1/2
	@DPTR, A	1/2
	A, @Ri	1/2
	@Ri, A	1/2

Veri aktaran Komutlar

Mnemonic	İşlenen	Bayt/Saykıl
MOVC	A, @A+DPTR	1/2
	A, @A+PC	1/2
PUSH	8 bit adres	2/2
POP	8 bit adres	2/2
XCH	A, Rn	1/1
	A, 8 bit adres	2/1
	A, @Ri	1/1
XCHD	A, @Ri	1/1

İç veri Belleğinde Veri aktaran komutlar

Hedef Akümülatör ise okuma işlemi olarak adlandırılır.

Mov A, Kaynak ;A ← Kaynak

Mov A, R0 ;A ← R0

Mov A, 20h ;A ← (20h)

Mov A, @R0 ;A ← ((R0))

Mov A, #5 ;A=5

Kaynak Akümülatör ise Yazma işlemi olarak adlandırılır.

Mov Hedef, A ;Hedef ← A

Mov R0, A ;R0 ← A

Mov 20h, A ; (20h) ← A

Mov @R0, A, ; ((R0)) ← A

İç veri Belleğinde Veri aktaran komutlar

Hedef veya Kaynak Akümülatör olmayabilir.

Mov 20h, R0	;(20h) ← R0
Mov R0, 20h	;R0 ← (20h)
Mov 20h, #0	;(20h)=0
Mov R0, #25h	;R0 = 25h
Mov @R0, 85h	;((R0))←(85h)
Mov 35h, 25h	;(35h) ← (25h)
Mov @R0, #25	; ((R0)) = 25

İç veri Belleğinde Veri aktaran komutlar

Mov dışında iç veri aktarma komutları vardır.

Push 8 bit adres ;Yığına belleğin içeriğini atar

Push 20h ;((SP)) ← (20h)

Push Acc ;((SP)) ← Acc

Pop 8 bit adres ;Yığından belleğe çeker

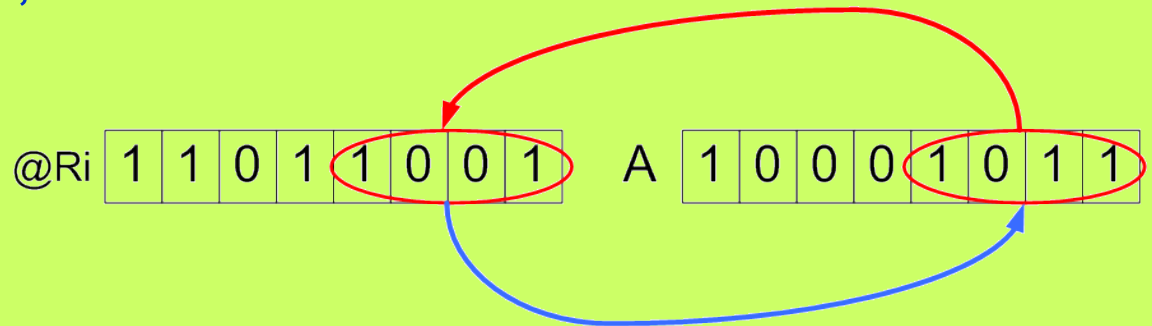
Pop 20h ;((SP)) → (20h)

Pop Acc ;((SP)) → Acc

XCH A Kaynak ;Akümülatör ile kaynağın içerikleri takas edilir

XCH A, R0 ;A ↔ R0

XCHD A, @Ri



Dış veri Belleğinden Veri aktaran komutlar

- Hedef veya Kaynak Akümülatördür.
- 16 bit adres hattı kullanılırsa DPTR işlem yapılacak adresi gösterir.

```
Movx A, @DPTR
Movx @DPTR, A
Mov DPTR,#0600
Movx A, @DPTR      ;(0600)→A
```

- 8 bit adres hattı kullanılırsa R0 veya R1 işlem yapılacak adresi gösterir.

```
Movx @Ri, A
Movx A, @Ri
Mov R1,#150
Movx A, @R1        ;(150) →A
```

Program Belleğinden Veri aktaran komutlar

- Hedef Akümülatördür.
- Program belleğinde oluşturulan başvuru tablosunda değer okur.
- DPTR veya PC temel adresi, Akümülatör ise sıra numarasını içerir.

Movc A, @A+DPTR ;(A+DPTR) → A

Movc A, @A+PC ;(A+DPTR) → A

Örnek:

P1 portundan okuduğu sayının karesini alıp P2'ye yazan programı yazın.

```
ORG 0
MOV DPTR, #TAB1
MOV P1, #0FFH
```

L01:

```
MOV A, P1
MOVC A, @A+DPTR
MOV P2, A
SJMP L01
```

```
ORG 300H
TAB1: DB 0,1,4,9,16,25,36,49,64,81
END
```

Aritmetik İşlem Komutları

Mnemonic	İşlenen	Bayt/Sayı
ADD	A, Rn	1/1
ADDC	A, direct	2/1
SUBB	A, @Ri	1/1
	A, #data	2/1
INC	A	1/1
DEC	Rn	1/1
	direct	2/1
	@Ri	1/1
INC	DPTR	1/2
MUL	AB	1/4
DIV	AB	1/4
DA	A	1/1

Toplama

- 8051 eldeli ve eldesiz toplama yapabilir.

```
ADD A,#25      ;A+25→A
ADDC A,#25     ;A+25+C →A
ADD A,#6       ;A+6 →A
ADD A,R6       ;A+R6 →A
ADD A,6        ;A+(6) →A
ADD A,0F3H     ;A+(0F3H) →A
ADD A,@R1      ;A+((R1)) →A
```

Örnekler

Mov A,#38h

Add A,#2Fh

38		0	0	1	1	1	0	0	0
+ 2F	+	0	0	1	0	1	1	1	1
<hr/>									
67		0	1	1	0	0	1	1	1

Mov A,#38h

Mov A,#30h

Add A,#27h

30		0	0	1	1	0	0	0	0
+ 27	+	0	0	1	0	0	1	1	1
<hr/>									
57		0	1	0	1	0	1	1	1

Mov A,#30h

Mov A,#88h

Add A,#93h

88		1	0	0	0	1	0	0	0
+ 93	+	1	0	0	1	0	0	1	1
<hr/>									
11B		0	0	0	1	1	0	1	1

Mov A,#88h

Mov A,#9Ch

Add A,#64h

9C		1	0	0	1	1	1	0	0
+ 64	+	0	1	1	0	0	1	0	0
<hr/>									
100		0	0	0	0	0	0	0	0

Mov A,#9Ch

Örnek – 16-bit Toplama

1E44H ile 56CAH sayılarını toplayan programı yazın.

```
MOV  A, #44H      ; 1. sayının DDB'ini A'ya al.  
ADD  A, #CAH      ; 2. sayının DDB'ı ile topla.  
MOV  R1, A        ; Toplamın DDB'ını yaz. CY = 1.  
MOV  A, #1EH      ; 1. sayının YDB'ını A'ya al.  
ADDC A, #56H      ; 2. sayının YDB'ı ile topla.  
MOV  R2, A        ; Toplamın YDB'ını yaz. CY = 0.
```

Sonuç: 750EH R2 ve R1'de yazılıdır, CY = 0.

Örnek – BCD Toplama

34 ile 49 BCD sayıları toplayan programı yazın

```
MOV    A, #34H        ; 1. sayıyı A'ya al
ADD    A, #49H        ; 2. sayı ile topla
                        ; A = 7DH
DA     A              ; BCD'ye ayarla A = 83
```

Örnek: İKO (BCD) Toplama

iki adet 4 basamaklı BCD sayılar iç veri belleğinin 40H, 41H ve 42H, 43H satırlarında yüksek değerli basamakları önce gelecek şekilde sırasıyla yazılıdır. Bu iki sayıyı toplayın ve BCD sonucu 40H ve 41H bellek satırlarında yüksek değerli basamak önce olacak şekilde saklayan programı yazın.

BCDTOPLA:

MOV A, 43H	;2. sayının birler ve onlar basamaklarını oku.
ADD A, 41H	;1. sayının birler ve onlar basamakları ile topla
DA A	;Toplamın birler ve onlar basamağını BCD'ye uyarla.
MOV 41H, A	;Toplamın birler ve onlar basamaklarını yaz.
MOV A, 42H	;2. sayının yüzler ve binler basamaklarını oku.
ADDC A, 40H	;1. sayının yüzler ve binler basamakları ile topla.
DA A	;Toplamın yüzler ve binler basamağını BCD'ye uyarla.
MOV 40H,A	;Toplamın yüzler ve binler basamaklarını yaz

MUL & DIV

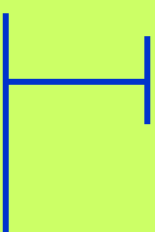
- MUL AB ;B|A = A*B
MOV A,#25H
MOV B,#65H
MUL AB ;25H*65H=0E99
 ;B=0EH, A=99H
- DIV AB ;A = A/B, B = A mod B
MOV A,#25
MOV B,#10
DIV AB ;A=2, B=5
Bölüm =2, kalan=5

Mantık İşlem Yapan Komutlar

Mnemonic	İşlenen	Bayt\Bayt
ANL	A, Rn	1/1
ORL	A, 8 bit Adres	2/1
XRL	A, @Ri	1/1
	A, #Say ₁	2/1
	8 bit Adres, A	2/1
	8 bit Adres, #Say ₁	3/2
	C, bit	2/2
	C, /bit	2/2
CLR	A	1/1
CPL	C	1/1
	bit	2/1

Mantık İşlem Yapan Komutlar

Mnemonic	İşlenen	Bayt/Sayı
RL	A	1/1
RLC	A	1/1
RR	A	1/1
RRC	A	1/1
SWAP	A	1/1
SETB	C	1/1
CLR	bit	2/1
CPL		2/1



Dallanma Komutları

Mnemonic	İşlenen	Bayt/Sayı
CJNE	A, 8 bit adres, kayıklık	3/2
	A, #say ₁ , kayıklık	3/2
	Rn, #say ₁ , kayıklık	3/2
	@Ri, #say ₁ , kayıklık	3/2
DJNZ	Rn, kayıklık	2/2
	8 bit adres, kayıklık	3/2
JZ	kayıklık	2/2
JNZ	kayıklık	2/2
NOP	-	1/1
JC	kayıklık	2/2
JNC	kayıklık	2/2
JB	bit, kayıklık	3/2
JNB	bit, kayıklık	3/2
JBC	bit, kayıklık	3/2

Bağlanma Komutları

Mnemonic İşlenen Bayt/Sayı

LCALL	16 bit adres	3/2
ACALL	11 adres	2/2
RET	-	1/2
RETI	-	1/2
LJMP	16 adres	3/2
AJMP	11 adres	2/2
SJMP	kayıklık	2/2
JMP	@A+DPTR	1/2

DJNZ:

Akümülatörü sıfırlayan ve sonra akümülatöre 10 defa 5 toplayan programı yazın.

```
TPL:   MOV    A,#0           ;A=0
        MOV    R2,#10       ;sayaç=10
TKR:   ADD    A,#05
        DJNZ   R2,TKR ; R2=0 olana kadar yap.
        MOV    R5,A
```

1000 adet döngü oluşturma

```
        Mov R3, #100
L1:    Mov R2, #10 ; 1000 döngü oluştur
L2:    Nop          ; işlem yapma
        Djnz R2, L2 ; R2==0 olana kadar
        Djnz R3, L1 ; R3==0 olana kadar
```

Karşılaştırma

- 8051 basit bir karşılaştırma komutuna sahip değildir. Karşılaştırma komutu akümülatör ile verilen değeri sanal çıkarma işlemine tabi tutar işlem sonucu sıfırdan farklı ise belirtilen adrese bağlanır. Aksi durumda bir sonraki durumdan devam eder.
- CJNE A, doğrudan adres, Kayıklık
- CJNE A, #Veri, Kayıklık
- CJNE Rn, #Veri, Kayıklık
- CJNE @Ri,#Veri,Kayıklıkl

Karşılaştırma

Farklı karşılaştırma örnekleri

```
Test: CJNE A,#Say1,Test ; Dallan eğer, A < Say1
      JC   küçük
```

```
Test: CJNE A, #Say1,Test ; Dallan eğer, A >= Say1
      JNC  büyük_esit
```

```
Test: CJNE A, #Say1,Test ; Dallan eğer, A > Say1
      SJMP baska
baska: JNC  büyük
      -----
```

```
Test: CJNE A, #Say1,Test ;Dallan eğer , A <= Say1
      SJMP küçük_esit
      JC   küçük_esit
```

Ödev-3

R0 ile R1'i karşılaştıran;

Eğer $R0 > R1$ ise P1'e 55h

Eğer $R0 < R1$ ise P1'e 0FFh

Eşitse P1'e 0 yazan programı yazın