6. PROGRAMCI KLAVUZU VE KOMUT SETLERİ

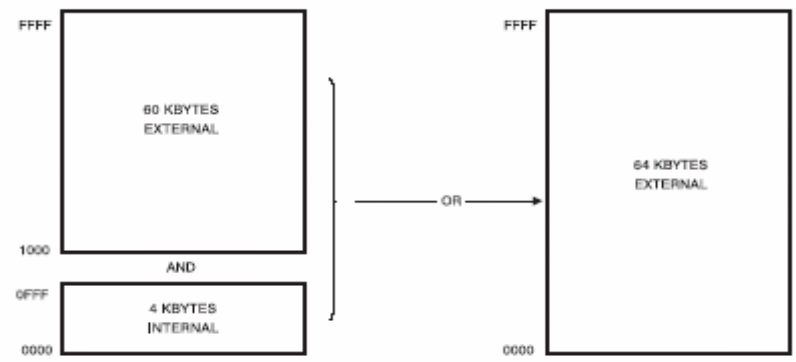
Programcının klavuzu bölümünde hafıza organizasyonuna ait özet bilgi ve özel fonksiyon registerlarına ait detaylı bilgiler yer almaktadır. Ayrıca komut setlerinde yer alan tüm komutlar teker teker açıklanmıştır.

6.1. HAFIZA ORGANİZASYONU

Standart 8051 mikrokontrolörlerinin hafıza yapısı bölüm 3.2 'de detaylı olarak açıklanmıştır. Bu bölümde programcının klavuzu kapsamında özet olarak tekrar verilmiştir.

6.1.1. PROGRAM HAFIZASI

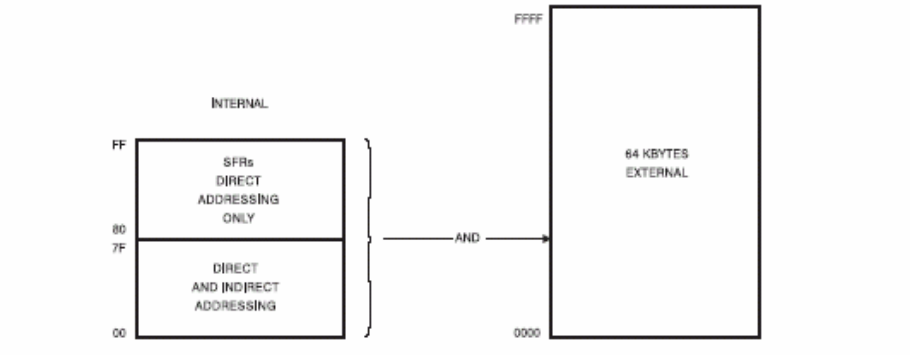
8051 mikrokontrolörlerinde program belleği (ROM,EPROM) yalnızca okunabilirdir.Boyutu 64 Kbyte'a kadar çıkabilir. 8051, dahili 4 Kbyte ROM belleğe sahiptir. ROMsuz (ROMless) versiyonlarda ise tüm program belleği haricidir.



Şekil 6.1 8051 Program Hafızası

6.1.2. VERİ HAFIZASI

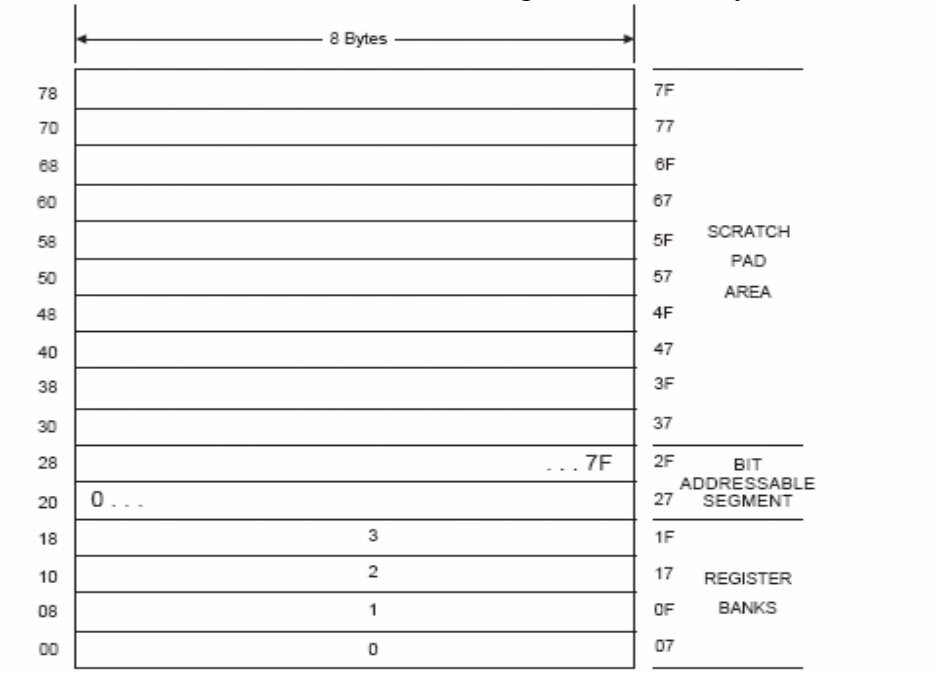
Veri belleği (RAM) program hafızasından ayrı olarak bir yer işgal eder. 8051 dahili 128 Byte RAM hafizasına sahiptir. Bu rakam harici veri belleği ile 64 Kbyte'a kadar artırılabilir.



Şekil 6.2 Veri Hafızası

6.1.3. DOĞRUDAN VE DOLAYLI ADRESLENEBİLİR BÖLGE (DAHİLİ RAM)

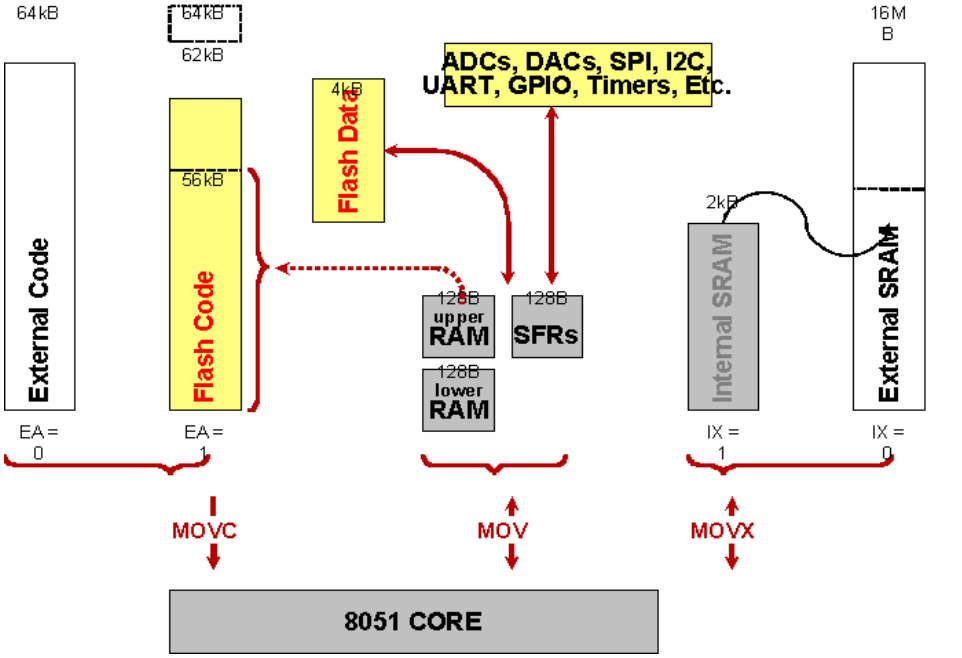
Dahili veri belleği adresleri her zaman bir byte genişliğindedir ve sadece 256 byte'lık adres alanını gösterir. Bu alanda 080H - 0FFh arasında kalan 128 byte'lık alan sadece dolaylı adreslenebilir. 00H - 07FH arasında kalan alan hem doğrudan hem dolaylı adreslenir.



Şekil 6.3 128 Byte Doğrudan ve Dolaylı adreslenebilir RAM

6.1.4. MICROCONVERTER PROGRAMLAMA MODELİ

MicroConverter'lar 8051 Mikrokontrolör hafıza yapısına sahip olmanın yanı sıra başka bellek birimlerini de barındırabilirler. MicroConverter ailesine ait programlama modeli şekil 6.4 de gösterimiştir.



Şekil 6.4

6.2. ÖZEL FONKSİYON REGİSTERLERİ (Special Function Registers/SFRs)

8051 Mikrokontrolör yapısı içerisinde yer alan I/O portları, zamanlayıcı/sayıcı'lar, harici ve dahili kesmeler vb birimlerin kontrol edilmesi, bunlara kullandığı yada ürettiği verilere erişim vb için kullanılan özel amaçlı register'lardır. Bu register'lara doğrudan adresleme ile ulaşılabilir. Ayrıca adreslerinin sonu 0 ve 8H ile bitenler (örneğin 90H, 98H vb) bit adreslenebilir özelliktedirler, yani her bir bitlerine tek tek erişilebilir, okunup-yazılabilir.

6.2.1. SFR SEMBOLLERİ, İSİMLERİ, ADRESLERİ

8051 Mikrokontrolör özel fonksiyon register'larının tümünün sembolleri, isimleri, işlevleri ve bellek adresi bilgileri Tablo 6.1 de verilmiştir.

Symbol	Name	Address
ACC(1)	Accumulator	0E0H
B(1)	B Register	0F0H
PSW(1)	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
P0(1)	Port 0	80H
P1(1)	Port 1	90H
P2(1)	Port 2	0A0H
P3(1)	Port 3	0B0H
IP(1)	Interrupt Priority Control	0B8H
IE(1)	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
TCON(1)	Timer/Counter Control	88H
T2CON(1)(2)	Timer/Counter 2 Control	0C8H
T2MOD(2)	Timer/Counter 2 Mode Control	0C9H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
TH2(2)	Timer/Counter 2 High Byte	0CDH
TL2(2)	Timer/Counter 2 Low Byte	0CCH
RCAP2H(2)	T/C 2 Capture Reg. High Byte	0CBH
RCAP2L(2)	T/C 2 Capture Reg. Low Byte	0CAH
SCON(1)	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

Notes: 1. Bit addressable
2. AT89C52 only

Tablo 6.1 Özel Fonksiyon Registerleri

6.2.2. POWER-ON RESET SONRASI SFR DEĞERLERİ

Power-on reset işlemi sonrasında SFR register'larının aldıkları değerler Tablo 6.2 de gösterilmiştir.

Register	Value in Binary
ACC(2)	00000000
B(2)	00000000
PSW(2)	00000000
SP	00000111
DPTR	
DPH	00000000
DPL	00000000
P0(2)	11111111
P1(2)	11111111
P2(2)	11111111
P3(2)	11111111
IP(2)	80C51 XXX00000, 80C52 XX000000
IE(2)	80C51 0XX00000, 80C52 0X000000
TMOD	00000000
T2MOD(3)	XXXXXX00
TCON(2)	00000000
T2CON(2)(3)	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
TH2(3)	00000000
TL2(3)	00000000
RCAP2H(3)	00000000
RRAP2L(3)	00000000
SCON(2)	00000000
SBUF	Indeterminate
PCON	CMOS 0XXX0000

Notes: 1. X = Undefined
2. Bit Addressable
3. AT89C52 only

Tablo 6.2 Power-On Reset Sonrası SFR Değerleri

6.2.3. SFR HAFIZA HARİTASI

Veri belleğinin 080H-0FFh alanı içerisinde yer alan özel fonksiyon registerlarının bellek içi yerleşim şekilleri Şekil 6.5 de gösterilmiştir.

8 Bytes

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW(1)								D7
C8	T2CON(1)(2)	T2MOD(2)	RCAP2L(2)	RCAP2H(2)	TL2(2)	TH2(2)			CF
C0									C7
B8	IP(1)								BF
B0	P3								B7
A8	IE(1)								AF
A0	P2								A7
98	SCON(1)	SBUF							9F
90	P1								97
88	TCON(1)	TMOD(1)	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON(1)	87

↑ Bit Addressable

Notes: 1. SFRs converting mode or control bits
2. AT89C52 only

Şekil 6.5 SFR Hafıza Haritası

6.2.4. PROGRAM STATUS WORD (PSW)

Çalışma anında CPU 'nun durumunu gösteren çeşitli durum bitlerini içerir.

CY	AC	F0	RS1	RS0	OV	—	P

CY: Elde bayrağı. Aritmetik işlemlerde elde durumunu göstermenin yanı sıra boolean işlemlerde de kullanılabilir.

AC: Yardımcı elde bayrağı.Örnek ile açıklarsak 8 bitlik bir toplama işleminde düşük dört bitin toplamından sonra bir elde oluşursa A_c biti 1 lenir.

F0: Genel amaçlı durum bayrağıdır.

RS1, RS0: Register bankları seçici bitleri

OV: Taşma bayrağı

---: Kullanılamayan bit

P: Parity Bit

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

6.2.5. GÜÇ KONTROL (POWER CONTROL/PCON)

Sistemin güç ayarlamalarının gerçekleştirildiği register'dır.

SMOD	—	—	—	GF1	GF0	PD	IDL

SMOD: Asenkron seri haberleşmede Baudrate timer/counter ile elde edilir. SMOD 1 olduğu durumda boudrate iki katı hızına çıkar sistem resetlendiğinde SMOD 0 ile çalışmaya başlar.

BİT 6, BİT 5, BİT 4: Kullanılmıyor.

GF0 ve GF1: Genel amaçlı bayrak bitleridir.

PD: CMOS entegrelerde, entegrenin güç tüketimini azaltmakta kullanılır.

IDLE: CMOS entegrelerin güç tüketim kontrolü için kullanılan diğer bir bit.

6.2.6. KESMELER (INTERRUPTS / IE, IP)

Program akışı esnasında beklenilen herhangi bir olayın meydana gelmesi (örneğin sayıcının tamalanması) nedeniyle program akışının durdurulması ve bu olay için istenilen işlemler tamamlandıktan sonra programa kalındığı yerden devam edilmesini sağlayan mekanizmayı kesme olarak tanımlayabiliriz. Bazı kesme registerları sekilde gösterilmiştir.

Interrupt Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
R1 & T1	0023H
TF2 & EXF2(1)	002BH

IE: KESME İZİN REGİSTERİ

EA	—	ET2	ES	ET1	EX1	ET0	EX0

EA: Genel kesme izin biti.Herhangi bir kesmeyi aktif etmek için öncelikle bu bitin aktif edilip daha sonra istenilen kesmenin özel izin biti aktif edilmelidir.

ET2 : Sadece 8052'de bulunan Timer-2 aktif etme bitidir.

ES: Seri port kapısının özel izin bitidir.Seri portun kesmesinin açılması için "1" yapılmalıdır.

ET1: Timer-1'in kesmesinin özel izin bitidir. Timer-1 kesmesini aktif etmek için bu bit "1" yapılmalıdır.

EX1: Harici kesme-1'in özel izin bitidir. Harici kesme -1'i aktif etmek için "1" yapılır.

ET0 : Timer-1'in kesmesinin özel izin bitidir. Timer-0'in kesmesini aktif etmek için bu bit "1" yapılmalıdır.

EX0 : Harici kesme-0'in özel izin bitidir. Harici kesme -0'i aktif etmek için "1" yapılır.

KESME ÖNCELİĞİ

En yüksekten düşük seviyeye öncelik seviyesi;

IE0
TF0
IE1
TF1
RI or TI
TF2 or EXF2

IP : KESME ÖNCELİK REGİSTERİ

—	—	PT2	PS	PT1	PX1	PT0	PX0

EA: Genel kesme izin biti.Herhangi bir kesmeyi aktif etmek için öncelikle bu bitin aktif edilip daha sonra istenilen kesmenin özel izin biti aktif edilmelidir.

ET2: Sadece 8052'de bulunan Timer-2 aktif etme bitidir.

ES : Seri port kapısının özel izin bitidir.Seri portun kesmesinin açılması için "1" yapılmalıdır.

ET1: Timer-1'in kesmesinin özel izin bitidir. Timer-1 kesmesini aktif etmek için bu bit "1" yapılmalıdır.

EX1: Harici kesme-1'in özel izin bitidir. Harici kesme -1'i aktif etmek için "1" yapılır.

ET0: Timer-1'in kesmesinin özel izin bitidir. Timer-0'in kesmesini aktif etmek için bu bit "1" yapılmalıdır.

EX0: Harici kesme-0'in özel izin bitidir. Harici kesme -0'i aktif etmek için "1" yapılır.

6.2.7. ZAMANLAYICILAR (TIMERS/TCON, TMOD, T2CON)

8051 MikroKontrollör 2 adet 16 bitlik Sayıcı/Zamanlayıcı içerir.

TCON:

16 bitlik iki zamanlayıcının ayarlarının yapılmasında kullanılır.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1: (Timer-1 taşma bayrağı). Zamanlayıcı (Timer) taştığı anda bu bayrak 1 (SET) olur. Mikroişlemci ilgili kesme programına sıçradığı anda bu bayrak tekrar temizlenir. Eğer kesme programı yoksa bu bayrak program tarafından temizlenmelidir. Sayıcı (Counter) FFFFH değerinden 0000H değerine atladığında bir taşma bayrağı oluşur.

TR1: (Timer-1 çalışma kontrol biti). Timer-1' i çalıştırma bitidir. Bu bit '1' yapıldığında Timer-1 çalışmaya başlar.

TF0: (Timer-0 taşma bayrağı). Timer taştığı anda bu bayrak SET olur. Mikroişlemci ilgili kesme programına sıçradığı anda bu bayrak tekrar temizlenir. Eğer kesme programı yoksa bu bayrak program tarafından temizlenmelidir. Sayıcı FFFFH değerinden 0000H değerine atladığında bir taşma bayrağı oluşur.

TR0: (Timer-0 çalışma kontrol biti). Timer-0' ı çalıştırma bitidir. Bu bit '1' yapıldığında Timer0 saymaya başlar.

IE1: (Harici kesme 1 kenar bayrağı). INT1 pininde yüksekten alçağa düşen bir sinyal görüldüğünde, program INT1 kesme adresi 0013h'e sıçrar.

IT1: (Harici kesme 1 INT1 tip belirleme biti). Eğer sinyal yüksekten düşüğe geçtiğinde kesme aktiflenmesi isteniyorsa bu bit SET edilir. Bu bit 0 olduğunda pindeki 0 sinyali kesmeyi aktifler.

IE0: (Harici kesme 0 kenar bayrağı). INT0 pininde yüksekten alçağa düşen bir sinyal görüldüğünde, program INT0 kesme adresi 0003h'e sıçrar.

IT0: (Harici kesme 0 INT0 tip belirleme biti). Eğer sinyal yüksekten düşüğe geçtiğinde kesme aktiflenmesi isteniyorsa bu bit SET edilir. Bu bit 0 olduğunda pindeki 0 sinyali kesmeyi aktifler.

TMOD:

16 bit'lik iki register'ın hangi modda çalışacağını belirlemek için kullanılır.

Timer 1				Timer 0			
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0

GATE: OR kapısı enable bitidir. Zamanlayıcı-1 (Timer-1)'in çalışmaya başlayabilmesi için bu bitin değerinin 0 olması gereklidir. Yani GATE biti ve TCON registerindeki TR1 biti timer-1' in çalışmaya başlamasını kontrol eder. GATE biti 1 ve TR1 biti 1 ise, timerın çalışması INT1 pinindeki sinyale bağlıdır. Bu sinyal 1 olduğu anda timer-1 çalışır. Sonuç olarak, INT1 biti ile GATE'in tersi OR kapısı ile bağlıdır. Bu şekilde, INT1 ile Timer/Counter dışarıdan kontrol edilir.

C/T: Sayıcı (Counter) veya zamanlayıcı (Timer) seçme bitidir. Bu bit '1' (SET) olduğu zaman timer/counter-1 sayıcı modunda çalışmaya başlar. Bu durumda T0 pinine bağlı sinyal sayılmaya başlar (chip dışında oluşan bir olay sayılabilir). Eğer '0' seçilirse Timer modunda makine periyoduna göre timer olarak çalışır.

M1: Timer/Counter- 1 mod seçme biti

M0: Timer/Counter-1 mod seçme biti.

GATE: OR kapısı enable biti.. Timer-0' in çalışmaya başlayabilmesi için bu bitin değerinin 0 olması gereklidir. Yani GATE biti ve TCON registerindeki TR0 biti timer-0' ın çalışmaya başlamasını kontrol eder. GATE biti 1 ve TR0 biti 1 ise, timerın çalışması INT0 pinindeki sinyale bağlıdır. Bu sinyal 1 olduğu anda timer-0 çalışır.

C/T: Counter veya timer seçme biti. Bu bit SET olduğu zaman timer/counter-0, sayıcı modunda çalışmaya başlar. Bu durumda T0 pinine bağlı sinyal sayılmaya başlar.

M1: Timer/Counter-0 mod seçme biti.

M0: Timer/Counter-0 mod seçme biti. Çalışma modları:

M1	M0	Operating Mode	
0	0	0	13-bit Timer
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	Split Timer Mode: (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

MODE	TIMER 0 FUNCTION	TMOD	
		INTERNAL CONTROL(1)	EXTERNAL CONTROL(2)
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Reload	02H	0AH
3	two 8-bit Timers	03H	0BH

Tablo 6.3 Zamanlayıcı olarak Timer/Counter-0

MODE	TIMER 0 FUNCTION	TMOD	
		INTERNAL CONTROL(1)	EXTERNAL CONTROL(2)
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Reload	06H	0EH
3	one 8-bit Counter	07H	0FH

Notes: 1. The Timer is turned ON/OFF by setting/clearing bit TR0 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on $\overline{INT0}$ (P3.2) when TR0 = 1 (hardware control).

Tablo 6.4 Sayıcı olarak Timer/Counter-0

MODE	TIMER 1 FUNCTION	TMOD	
		INTERNAL CONTROL(1)	EXTERNAL CONTROL(2)
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Reload	20H	A0H
3	does not run	30H	B0H

Tablo 6.5 Zamanlayıcı olarak Timer/Counter-1

MODE	COUNTER 1 FUNCTION	TMOD	
		INTERNAL CONTROL(1)	EXTERNAL CONTROL(2)
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Reload	60H	E0H
3	not available	—	—

Notes: 1. The Timer is turned ON/OFF by setting/clearing bit TR1 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on $\overline{INT1}$ (P3.3) wen TR1 = 1 (hardware control).

Tablo 6.6 Sayıcı olarak Timer/Counter 1

6.2.8. SERİ PORT (SCON, SBUF)

8051 'in önemli özelliklerinden biri de seri haberleşme birimi UART a sahip olmasıdır. Seri haberleşmenin gerçekleştirilmesi aşamasında seri portların kontrolü SFR ile yapılır.

SCON:

8051 seri giriş/çıkış biriminin davranışını kontrol etmekte kullanılır

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit	İsim	Bit Adresi	Açıklama
7	SM0	9Fh	Seri port mod bit 0
6	SM1	9Eh	Seri port mod bit 1.
5	SM2	9Dh	Birden çok işlemci ile haberleşme modu
4	REN	9Ch	Alıcı aktif biti. Karakter okuması için bu bit aktif yapılmalıdır.
3	TB8	9Bh	Yolla biti, bit 8. Mod 2 ve 3 de karakter yollamak için kullanılan bit.
2	RB8	9Ah	Mod 2 ve 3 de karakter okumak için kullanılan bit.
1	TI	99h	Yolla bayrağı. Bir karakter yollandığı zaman bu bit aktif olur.
0	RI	98h	Alıcı bayrağı. Bir karakter okunduğu zaman bu bit aktif olur.

SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 OR Fosc./32
1	1	3	9-Bit UART	Variable

MODE	SCON	SM2 VARIATION
0 1 2 3	10H 50H 90H D0H	Single Processor Environment (SM2 = 0)
0 1 2 3	NA 70H B0H F0H	Multiprocessor Environment (SM2 = 1)

Baud Rate Table

Crystal Frequency	7.3728 MHz	8.00 MHz	11.0592 MHz	12.00 MHz	14.75156 MHz	16.00 MHz
TH1						
E0	600	651	900	976	1,200	1,302
F0	1,200	1,302	1,800	1,953	2,400	2,604
F8	2,400	2,604	3,600	3,906	4,800	5,208
F9	2,743	2,976	8,299	4,464	5,486	5,952
FA	3,200	3,472	9,600	5,208	6,400	6,944
FF	19,200	20,833	57,600	62,500		41,666

Table 10. Baud Rate Summary

Baud Rate	Crystal Frequency	SMOD	TH1 Reload Value	Actual Baud Rate	Error
9600	12.000 MHz	1	-7 (F9H)	8923	7%
2400	12.000 MHz	0	-13 (F3H)	2404	0.16%
1200	12.000 MHz	0	-26 (E6H)	1202	0.16%
9200	11.059 MHz	1	-3 (FDH)	19200	0
9600	11.059 MHz	0	-3 (FDH)	9600	0
2400	11.059 MHz	0	-12 (F4H)	2400	0
1200	11.059 MHz	0	-24 (E8H)	1200	0

Note: Due to rounding, there is a slight error in the resulting baud rate. Generally, a 5% error is tolerable using asynchronous (start/stop) communications. Exact baud rates are possible using an 11.059 MHz crystal. The table above summarizes the TH1 reload values for the most common baud rates, using a 12.000 MHz or 11.059 MHz crystal.

Tablo 6.7 Baud Rate Özeti

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

6.3. KOMUT SETİ

8051 Komut Seti anlatımında kullanılan operandlar ve anlamları Tablo 6.8 de gösterilmiştir.

Rn	Seçilen register bank içerisindeki R0-R7 arası register'lar (n = 0,1,2..7)
direct	8 bit'lik dahili veri alanı adresi.Dahili Veri Belleği (0-127) ya da SFR olabilir(ör: I/O port, kontrol register, .. vb →128-255)
@Ri	R0 veR1 tarafından dolaylı adreslenen 8 bit dahili veri belleği alanı.
#data	8-bit sabit değer
#data 16	16-bit sabitr değer
addr 16	16-bit hedef adres.LCALL ve LJMP'la kullanılır.Dallanma 64Kbyte program belleği içerisinde herhangi bir yere gerçekleşebilir.
addr 11	11-bit hedef adres. ACALL ve AJMP ile kullanılır.2Kbyte program belleği içerisinde istenilen yere dallanma gerçekleştirilebilir.
rel	İşaretli 8-bit offset adresi.SJMP ile ya da koşullu dallanmalarda kullanılır.-128 ile 127 byte bağıl adres.
bit	Dahili Veri Belleği ya da SFR ' de direk adreslenebilen bit.

Tablo 6

Komut Listesi

Mnemonic		Description	Byte	Cycle
Arithmetic Operations				
ADD	A,Rn	Add register to accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#Data	Add immediate data to accumulator	2	1
ADDC	A,Rn	Add register to accumulator with carry flag	1	1
ADDC	A,Direct	Add direct byte to Accumulator with carry flag	2	1
ADDC	A,@Ri	Add indirect Ram to Accumulator with Carry flag	1	1
ADDC	A,#Data	Add immediate data to Accumulator with carry flag	2	1
SUBB	A,Rn	Subtract register from Accumulator with borrow	1	1
SUBB	A,direct	Subtract direct byte from Accumulator with borrow	2	1
SUBB	A,@Ri	subtract indirect RAM from accumulator with borrow	1	1
SUBB	A,#data	Subtract immediate data from Accumulator with borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1

INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	Direct	Decrement direct data	2	1
DEC	@Ri	Decrement indirect data	1	1
INC	DPTR	Increment data pointer	1	2
MUL	AB	Multiply A and B	1	4
DIV	AB	Divide A and B	1	4
DA	A	Decimal Adjust Accumulator	1	1

Logical Operations				
ANL	A,Rn	AND register to Accumulator	1	1
ANL	A,direct	AND Register byte to Accumulator	2	1
ANL	A,@Ri	AND İndirect RAM to Accumulator	1	1
ANL	A,data	AND immediate data to Accumulator	2	1
ANL	direct,A	AND Acccumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to Accumulator	1	1
ORL	A,direct	OR direct data to Accumulator	2	1
ORL	A,@Ri	OR indirect RAM to Accumulator	1	1
ORL	A,#data	OR immediate data to Accumulator.	2	1
ORL	Direct,A	OR Accumulator to data byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to Accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	1
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	1
XRL	Direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	Direct,#data	Exclusive-OR immediate data to direct byte	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate Accumulator left through the carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate Accumulator right through the carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

Data Transfer				
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,Direct	Move direct byte to Accumulator	2	1
MOV	A,@Ri	Move indirect Ram to Accumulator	1	1
MOV	A,#data	Move immediate data to Accumulator	2	1
MOV	Rn,A	Move Accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move Accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct byte	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move Accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Rİ,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data16	Load Data pointer with a 16-bit Constant	3	2
MOVC	A,@A+DPTR	Move code byte relative to DPTR to Accumulator	1	2
MOVC	A,@A+PC	Move code byte relative to PC to Accumulator	1	2
MOVX	A,@Ri	Move external RAM(8 Bit addr) to Accumulator	1	2
MOVX	@Ri,A	Move Accumulator to external RAM (8bit address)	1	2
MOVX	@DPTR,A	Move Accumulator to external RAM (16 bit address)	1	2
PUSH	direct	Push direct byte on to stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	1
XCHD	A,@Ri	Exchange low-order digit indirect with Accumulator	1	1
Boolean Variable Manipulation				
CLR	C	Clear carry Flag	1	1
CLR	Bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	Bit	Set direct bit	2	1
CPL	C	Complement carry Flag	1	1
CPL	Bit	Complement direct bit	2	1

ANL C,bit	AND Direct bit to carry flag	2	2
ANL C,/bit	AND complement of direct bit to carry	2	2
ORL C,bit	OR direct bit to carry flag	2	2
ORL C;/bit	OR complement of direct bit to carry	2	2
MOV C,bit	Move direct bit to carry flag	2	1
MOV bit,C	Move carry flag to direct bit	2	2
Program And Machine Control			
ACALL addr11	Absolute subroutine call	2	2
LCALL addr16	Long subroutine call	3	2
RET	Return from subroutine	1	2
RETI	Return from ınterrupt	1	2
AJMP addr11	Absolute jump	2	2
LJMP addr16	Long Jump	3	2
SJMP rel.	Short jump (Relative Addr)	2	2
JMP @A,+DPTR	Jump indirect relative to DPTR	1	2
JZ rel.	Jump if Accumulator is zero	2	2
JNZ rel.	Jump if Accumulator is not zero	2	2
JC rel.	Jump if carry flag is set	2	2
JNC rel.	Jump if carry flag is not set	2	2
JB bit,rel.	Jump if the bit is set	3	2
JNB bit,rel.	Jump if the bit is not set	3	2
JBC bit,rel.	Jump if direct bit is set and clear bit	3	2
CJNE A,direct,rel.	Compare direct to Accumulator and jump if not equal	3	2
CJNE A,#data,rel.	Compare immediate data to Accumulator jump if not equal	3	2
CJNE Rn,#data,rel.	Compare immediate data to register jump if not equal.	3	2
CJNE @Ri,#data,rel	Compare immediate data to indirect and jump if not equal	3	2
DJNZ Rn,rel.	Decrement register and jump if not zero	2	2
DJNZ direct ,rel.	Decrement direct and jump if not zero	3	2
NOP	No Operation	1	1

ACALL addr11

Fonksiyon: Çağırma (Absolute call)

Açıklama: Bir duruma bağlı olmadan, adresle belirtilen alt programı çağırır.PC, yığına atılır. PC'nin düşük 11-bit adresi, addr11 ile değiştirilir ve yüksek 5 bit değişmez. Altprogram Program Hafızasının aynı 2K'lık bloğu içinde olmalıdır.

Adresleme: Doğrudan

Etkilenen Bayraklar: Yok

Çevrim: 2

Kodlama:

A10	A9	A8	1	0	0	0	1	A7	A6	A5	A4	A3	A2	A1	A0

İşlem: (PC)← (PC)+2
(SP)← (SP)+1
((SP))←(PC_{7-0})
(SP)← (SP)+1
((SP))←(PC_{15-8})
(PC_{10-0})← sayfa adresi

ÖRNEK:

```
ACALL LABEL ;LABEL adresinden başlayan alt programı çağırır.
```

ADD A,<kaynak-byte>

Fonksiyon: Kaynak byte Akümülatör ile toplanır(ADD).

Açıklama: Kaynak byte ile belirtilen değişken, ACC'ye toplanır ve sonuç ACC'de bulunur.

Adresleme: Kaynak için dört adresleme modu vardır: reg, dir,int,imm.

Etkilenen Bayraklar: CY, AC,OV.

Çevrim:1.

Komut	Kodlama	İşlem
ADD A,Rn	0 0 0 1 1 r r r	(A)←(A)+(Rn)
ADD A,direct	0 0 1 0 0 1 0 1 Direkt Adres	(A)←(A)+(dir)
ADD A,@Ri	0 0 1 1 0 1 1 i	(A)←(A)+((Ri))
ADD A,#data	0 0 1 0 0 1 0 0 İvedi Adres	(A)←(A)+#data

ÖRNEKLER:

```
ADD  A,R0        ;R0'ın içeriğini ACC ile topla.
ADD  A,45H       ;45H adresindeki hafıza içeriğini ACC ile topla.
ADD  A,@R0       ;R0 ile belirlenen hafıza içeriğini ACC ile topla.
ADD  A,#12H      ;12H'ı ACC ile topla.
```

ADDC A,<kaynak byte>

Fonksiyon: Kaynak byte'ı elde ile ACC'yi topla (ADD with Carry).

Açıklama: Kaynak byte ile belirtilen değişken ve elde ACC'ye toplanır ve sonuç ACC'de bulunur.

Adresleme: Kaynak için dört adresleme modu vardır; reg,dir,ind,imm.

Etkilenen Bayraklar: CY, AC,OV.

Çevrim:1.

Komut	Kodlama	İşlem
ADDC A,Rn	0 0 1 1 | 1 r r r	(A)←(A)+(Rn)+C
ADDC A,direct	0 0 1 1 | 0 1 0 1 Direkt Adres	(A)←(A)+(dir) +C
ADDC A,@Ri	0 0 1 1 | 0 1 1 i	(A)←(A)+((Ri)) +C
ADDC A,#data	0 0 1 1 | 0 1 0 0 İvedi Adres	(A)←(A)+#data+C

ÖRNEKLER:

```
ADDC  A,R0    ;R0'ın içeriğini ACC ve C ile topla.
ADDC  A,45H   ;45H adresindeki hafıza içeriğini ACC ve C ile topla.
ADDC  A,@R0   ;R0 ile belirlenen hafıza içeriğini ACC ve C ile topla.
ADDC  A,#12H  ;12H'ı ACC ve C ile topla.
```

AJMP addr11

Fonksiyon: Dallanma (Absolute Jump)

Açıklama: Program akışı adresle belirlenen yere dallanır. PC'nin düşük 11-bit

adresi,ADDR11 ile değiştirilir ve yüksek 5-bit değişmez. Altprogram Program Hafızasının aynı 2K'lık bloğu içinde olmalıdır.

Adresleme: Doğrudan

Etkilenen Bayraklar: Yok

Çevrim:2.

Kodlama:

A10	A9	A8	0	0	0	0	1	A7	A6	A5	A4	A3	A2	A1	A0

İşlem: (PC)← (PC)+2

(PC_{10-0})← sayfa adresi

ÖRNEK:

```
AJMP LABEL   ;Label ile belirtilen yere dallan.
```

ANL <Hedef byte>,<kaynak byte>

Fonksiyon: Byte değişkenleri için lojik AND (Logical-AND for byte variables).

Açıklama: Belirtlen değişkenler arasında karşılıklı bit-bit (bit-wise) lojik And işlemini yerine getirir ve sonuç hedef değişkende saklanır.

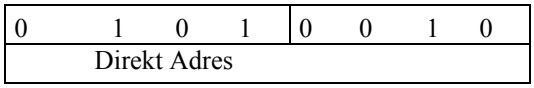
Adresleme: İki operant, 6 adresleme modu oluşturur.Hedef ACC olduğunda, kaynak, register, doğrudan, register-dolaylı ivedi adresleme modu kullanabilir.Hedef doğrudan bir adres olduğu zaman kaynak ACC veya ivedi veri olabilir.

Etkilenen Bayraklar: Yok.

Komut	Çevrim	Kodlama	İşlem
ANL A,Rn	1	0 1 0 1 1 r r r	(A)←(A)^(Rn)
ANL A,direct	1	0 1 0 1 0 1 0 1 Direkt Adres	(A)←(A)^(dir)
ANL A,@Ri	1	0 1 0 1 0 1 1 i	(A)←(A)^((Ri))
		0 1 0 1 0 1 0 0 İvedi Data	

ANL A,#data (A)←(A)^#data 1

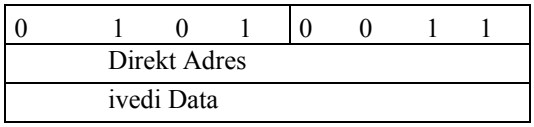
ANL direct,A



(dir)←(dir)^(A)

1

ANL direct,#data



(dir)←(dir)^#data

2

ÖRNEKLER:

```
ANL  A,R0              ;R0 ile ACC'yi ANDle. Sonuç ACC'de
ANL  P1,#01110011B     ;P1'in 7,3 ve 2'inci bitlerini temizle.
ANL  A,@R0             ;ACC ve R0 ile işaretli hücreyi AND'le.,
ANL  35H,#45H          ;35H hücresinin içeriği ile 45H AND'lenir ve
                       ;sonuç 35H'a  yerleştirilir.
ANL  35H,A             ;35H hücresinin içeriği ile ACC AND'lenir.
                       ;Sonuç 35H'a yerleştirilir.
```

ANLC <Hedef byte>

Fonksiyon: Bit değişkenleri için lojik AND (Logical-AND).

Açıklama: Operantlardan biri ve sonucu gideceği kaynak elde bayrağıdır.(\) işareti kaynağın önüne tersini belirtmek için gelebilir. Kaynak bit, herhangi ibr adreslenebilir bit olabilir.

Adresleme: Doğrudan

Etkilenen Bayraklar: C

Çevrim: 2.

ANL C,bit

Kodlama :

İşlem: (C) ← (C)^(bit)

ANL C,/bit

1	0	1	1	0	0	0	0	Bit adres

İşlem: (C) ← (C)^l (bit)

ÖRNEKLER:

```
ANL C,24H.7       ;Elde bayrağı ile 24H'taki biti, yani dahili RAM
                  ;hücresi ;24H'ın 7'inci biti AND' lenir.
                  ;Sonuç elde bayrağına yerleştirilir.
ANL C,/27H        ;Elde bayrağı ile 27H'daki bitinin yani dahili Ram
                  ;hücresi ;24H'ın 7'inci bitinin tersi AND'lenir.
                  ;Sonuç elde bayrağına yerleştirilir
ANL ACC.2,C       ;Elde bayrağı CC'nin 2'inci biti ileAND'lenir.Sonuç
                  ;ACC'nin  ikinci bitine yerleştirilir.
```

CJNE <hedef b yte>,<hedef byte>, rel.

Fonksiyon: Karşılaştır ve eğer eşit değil ise dallan (Compare and jump İf not equal)

Açıklama: İlk iki operantı karşılaştırır ve eğer değerleri eşit değil ise dallanır.Dallanırsa, hedefin üçüncü operant ile belirlendiği göreceli dallanma (Relative Jump) gerçekleştirilir.

Adresleme: ilk iki operant, dört adresleme modu oluşturur.ACC doğrudan adreslenmiş byte , ivedi veri ile; herhangi dolaylı adreslenmiş bir RAM hücresi veya bir register, ivedi sabit ile karşılaştırılabilir.

Etkilenen Bayraklar: C

Çevrim: 2.

İşlem: (PC)← (PC)+3
IF (hedef-byte)<> (kaynak-byte)
THEN
(PC)←(PC)+relative offset
IF (hedef-byte)< (kaynak –byte)
THEN
(C)←1
ELSE

(C)←0

CJNE A,direct,rel

1	0	1	1	0	1	0	0
Direkt Adres							
Relative adres							

CJNE A,#Data,rel

1	0	1	1	0	1	0	0
İvedi data							
Relative adres							

CJNE Rn,#data,rel

1	0	1	1	0	r	r	r
İvedi data							
Relative adres							

CJNE @Ri,#data,rel

1	0	1	1	0	1	1	İ
İvedi data							
Relative adres							

ÖRNEKLER :

```
CJNE A,#0FH,LABEL       ;ACC'deki değer, sabit 0FH ile
                        ;karşılaştırılır.Eğer eşit değil ise LABEL
CJNE R4,#35,LABEL       ;Hafıza alanından devam eder.R4'deki değer
                        35ile karşılaştırılır. Eğer
                        ;sabit ise LABEL hafıza alanından devam eder.

CJNE  @R1,#67,LABEL     ;R1 ile işaretli hücre, sabit 67 ile
                        ; karşılaştırılır, eğer eşit değil ise LABEL
                        ;hafıza alanından devam eder.
```

CLR A

Fonksiyon: ACC'yi temizle(Clear Accumulator).

Açıklama: ACC 00H ile yüklenir.

Adresleme: Register-özel (Sadece ACC).

Etkilenen Bayraklar: Yok.

Çevrim: 1

Kodlama:

1	1	1	0	0	1	0	0

İşlem: (A)← 0

CLR bit

Fonksiyon: Bir bit temizle (Clear bit).

Açıklama: Belirtilen bit sıfırlanır.

Adresleme: Doğrudan.

Etkilenen Bayraklar: C (CLR C komutunda)

Çevrim: 1.

CLR C

Kodlama:

1	1	0	0	0	0	1	1

İşlem: (C)← 0

CLR BİT

Kodlama:

1	1	0	0	0	0	1	0	Bit adres

İşlem: (bit)← 0

CPL A

Fonksiyon: ACC'yi tersle (Complement accumulator).

Açıklama: ACC'nin bitleri tersleri."1" ler "0"lanır ve "0"lar "1"lenir.

Adresleme: Register özel (sadece ACC)

Etkilenen Bayraklar: Yok

Çevrim: 1.

Kodlama:

1	1	1	1	0	1	0	0

İşlem:

(A) ← l (A)

ÖRNEK:

```
CPL  A      ; ACC'de 01011100B bulunsa komut yürütüldükten sonra ACC
            ; 10100011B olur.
```

CPL BIT

Fonksiyon: Belirtilen biti tersle.(Complement bit)

Açıklama: Belirtilen bit bir ise sıfırlanır, sıfır ise birlenir.

Adresleme: Doğrudan.

Etkilenen Bayraklar: CPL C ile C terslenir.

Çevrim: 1

CPL C

Kodlama:

1	0	1	1	0	0	1	1

İşlem:

(C) ← l (C)

CPL BIT

Kodlama:

1	0	1	1	0	0	1	0	Bit adres

İşlem:

(Bit) ← l (Bit)

ÖRNEKLER:

```
CPL C       ;Eldse bitini tersle.
CPL P1.2    ;Çıkış portun 2. bitini tersle.
CPL 20      ;Bit 20'yi (RAM bölgesi 22H'in bit 4'ü) tersle.
```

DA A

Fonksiyon: ACC'yi ondalık olarak toplama için ayarla (Decimal Adjust)

Açıklama: BCD sayılar üzerindeki ADD veya ADCC komutlarından sonra daha fazla işlem gerekebilir.Eğer düşük 4 bit (nibble) , 9'dan büyük değere sahipse , elde biti C yüksek 4 bite eklenir. Benzeri şekilde , eğer yüksek 4-bit 9'dan büyükse PSW'deki elde biti 1 lenir.

Adresleme: Register-özel(sadece ACC)

Etkilenen Bayraklar: C

Çevrim : 1

Kodlama:

1	1	0	1	0	1	0	0

İşlem: İF [[(A_{3-0})>9]v[(AC)=1]]
THEN (A_{3-0}← A_{3-0})+6
AND
İF[[[[(A_{-7-4})>9]v[(C)=1]]
THEN (A_{-7-4})← (A_{-7-4})+6

ÖRNEKLER:

```
MOV   R3,#35
ADD   A,R3
DA    A
```

DEC byte

Fonksiyon: Byte'ın içeriğini azalt.(Decrement byte)

Açıklama: Belirtilen değişken byte,1azaltılır.Değişken 00h ise,ilk değeri 0FFh olur.(underflow)

Adresleme: Operand için 4 adresleme modu vardır : acc , reg , dir , ind.

Etkilenen Bayraklar: Yok

Çevrim: 1

Komut	Kodlama	İşlem	Çevrim
DEC A	0 0 0 1 | 0 1 0 0	(A)←(A)-1	1
DEC RN	0 0 0 1 | 0 1 0 0	(Rn)←(Rn)-1	1
DEC direct	0 0 0 1 | 0 1 1 1 Direct Adres	(dir)←(dir)-1	1
DEC @Ri	0 0 0 1 | 0 1 0 i	(Ri)←((Ri))-1	1

ÖRNEKLER:

```
DEC   A          ; ACC'nin içeriğini 1 azalt.
DEC   R7         ; Register R7'nin içeriğini 1 azalt.
DEC   35H        ; Dahili RAM bölgesi 35'in içeriğini 1 azaltır.
DEC   @R0        ; R0 ile işaretli RAM bölgesinin içeriğini 1 azalt.
```

DIV AB

Fonksiyon: Böl (Divide)

Açıklama: ACC ve B registerlarının içerikleri,8-bit işaretsiz tamsayı olarak işlem görür. ACC, B register' ı ile bölünür.Tam sayı sonuç ACC'ye yerleştirilir. Kalan B' ye yazılır.

Adresleme: Register-Özel(Sadece ACC ve B)

Etkilenen Bayraklar: C,OV.Elde bayrağı her zaman temizlenir. Sıfır ile bölme yapılmaya kalkışılmadığı takdirde, taşma (Overflow) sıfır volt bayrağı temizlenir.Eğer B registerinde 00H bulunursa, taşma bayrağı komutu DIV tarafından birlenir

Çevrim:4.

Kodlama:

1	0	0	0	0	1	0	0

İşlem: $(A)_{15-8} \leftarrow (A)/(B)$
$(B)_{7-0}$

DJNZ <byte>,<rel-adrr>

Fonksiyon: Bir azalt ve eğer sıfır değilse dallan(Decrement And Jump If Not Zero)

Açıklama: Belirtilen byte değişken bir azaltılır.Eğer sonuç sıfırdan farklı ise ,ikinci değişkenle belirtilen adrese dallanma olur.00h ilk değeri 0FFh ile sonuçlanır (underflow)

Adresleme: Operand için 2 adresleme modu vardır: reg ve dir.

Etkilenen Bayraklar : Yok

Çevrim : 1

İşlem : (PC)←(PC)+2
(byte)←(byte)-1
IF (byte)>0 or (byte)<0
THEN
(PC)←(PC)+rel

DJNZ Rn,rel

Kodlama:

1	1	0	1	1	r	r	r	relative adres

DJNZ direct , rel

Kodlama:

1	1	0	1	0	1	0	1	direkt adres	relative adres

ÖRNEKLER:

```
DJNZ 40H,LABEL    ; 40H'ın içeriğini 1 azalt, sonuç 0'dan farklı ise
                  ; LABEL'a dallan.
```

```
DJNZ R6,LABEL     ; R6'ın içeriğini 1 azalt, sonuç 0'dan farklı ise
                  ; LABEL'a dallan.
```

NC byte

Fonksiyon: Byte arttır.(Increment).

Açıklama: Değişken 1 arttırılır.00h ilk değeri 0FFh ile sonuçlanır (underflow)

Adresleme: Operand için dört adresleme modu vardır:acc , reg , dir , ind.

Etkilenen Bayraklar : Yok

Komut	Kodlama	İşlem
INC A	0 0 0 0 | 0 1 0 0	(A)←(A)+1
INC Rn	0 0 0 0 | 1 r r r	(Rn)←(Rn)+1
INC direct	0 0 0 0 | 0 1 0 1 Direkt Adres	(dir)←(dir)+1
INC @Ri	0 0 0 0 | 0 1 1 i Direkt Adres	((Ri))←((Ri))+1

ÖRNEKLER:

```
INC   A           ;ACC'nin içeriğini arttır.
INC   R7          ;R7'nin içeriğini arttır.
INC  35H          ;Dahili RAM bölgesi 35'in içeriğini arttır.
INC   @R0         ;R0 ile işaretli RAM bölgesinin içeriğini arttır.
```

INC DPTR

Fonksiyon : Veri işaretçisini arttır. (Increment Data Pointer)

Açıklama : 16-bit veri işaretçisi DPTR 1 arttırılır.Bu arttırılabilen tek 16 bit'lik registerdır.

Adresleme : Register – özel (sadece DPTR)
Etkilenen Bayraklar : Yok

Çevrim: 1

Kodlama : 1 byte

1	0	1	0	0	0	1	1

(DPTR)←(DPTR)+1

ÖRNEKLER:

```
INC   DPTR        ;16-bit veri işaretçisini bir arttır.
```

JB bit,rel

Fonksiyon : Eğer bit 1 ise dallan (Jump if Bit set)

Açıklama : Belirtilen bit 1 ise , komuttaki adrese dallan, aksi durumda bir sonraki komuta devam et.

Adresleme: Doğrudan

Etkilenen Bayraklar : Yok

Çevrim : 2

Kodlama :

0	0	1	0	0	0	0	0	Bit adres	relative adres

İşlem :
(PC)←(PC)+3
IF(bit)=1
THEN(PC)←(PC)+rel

ÖRNEKLER:

```
JB    P1.2,LABEL        ;P1'in ikinci bit'i 1 ise,LABEL'a dallan.
```

JBC bit,rel

Fonksiyon : Eğer bit, 1 ise dallan ve biti temizle (Jump if Bit set and Clear bit)

Açıklama : Belirtilen bit 1 ise , komuttaki adrese dallan, aksi durumda bir sonraki komuta devam et.Bit durumdan bağımsız olarak, 0 değil ise temizlenir ; 0 ise temizlenmez.

Adresleme: Doğrudan

Etkilenen Bayraklar : Yok

Çevrim : 2

Kodlama :

0	0	0	1	0	0	0	0	Bit adres	relative adres

İşlem :

(PC)←(PC)+3
IF(bit)=1
THEN
(bit)←0
(PC)←(PC)+rel

ÖRNEKLER:

```
JBC   P1.2,LABEL        ;P1'in ikinci bit'i 1 ise,bit'i temizler ve
                        ; LABEL'a dallanır.Alsi halde bit'i
                        ; temizler,bir sonraki komutla devam eder.
```

JC rel

Fonksiyon : Elde 1 ise dallan İ(jump ıf carry is set)

Açıklama : Elde bayrağı 1 ise, belirtilen adrese dallan, aksi halde bir sonraki komutla devam er.

Adresleme : Doğrudan

Etkilenen Bayraklar : Yok

Çevrim : 2
Kodlama :

0	1	0	0	0	0	0	0	relative adres

İşlem :
(PC)←(PC)+2
IF(C)=1
THEN
(PC)←(PC)+rel
ÖRNEKLER:

```
JC    LABEL            ;Eğer elde bit'i bir ise LABEL'a dallan.
```

JMP @A+DPTR

Fonksiyon : İndisli dallanma .(Jump indirect).

Açıklama : Program, ACC ve DPTR registerlarındaki değerlerin toplanmasıyla oluşan adresin belirlediği hafıza alanına dallanır.

Adresleme : Dolaylı

Etkilenen Bayraklar : Yok

Çevrim : 2

Kodlama :

0	1	1	1	0	0	1	1

İşlem : (PC)←(A)+(DPTR)
ÖRNEKLER:

```
;Aşağıda LABEL bir dallanma tablosunun (jump table) yani,
;dallanma komutlarının taban adresidir.
;Dallanma ACC'deki değere bağlıdır.

            MOV   DPTR,#LABEL
            JMP   @A+DPTR
            .
            .
LABEL:      JMP   L1
            JMP   L2
            JMP   L3
```

JNB bit,rel

Fonksiyon : Eğer bit, 1 değil ise dallan.(Jump if bit not set)

Açıklama :Belirtilen bit sıfırsa, komuttaki adrese dallan, aksi durumda bir sonraki komuta devam et.

Adresleme: Doğrudan

Etkilenen Bayraklar : Yok

Çevrim:2

Kodlama :

0	0	0	1	0	0	0	0	Bit adres	relative adres

İşlem :

(PC)←(PC)+3
IF(bit)=0
THEN
(PC)←(PC)+rel

ÖRNEKLER:

```
JNB   P1.2,LABEL        ;P1'in ikinci bit'i 0 ise,LABEL'a dallan.
```

JNC rel

Fonksiyon: Elde 1 değil ise dallan.(Jump if carry is not set)

Açıklama : Elde biti sıfırsa, belirtilen adrese dallan , aksi durumda bir sonraki komuta devam et.

Adresleme : Doğrudan

Etkilenen Bayraklar: Yok

Çevrim : 2

Kodlama :

0	1	0	1	0	0	0	0	relative adres

İşlem :

(PC)←(PC)+2
IF(C)=0
THEN
(PC)←(PC)+rel

ÖRNEKLER:

```
JNC   LABEL             ;Eğer elde biti sıfırsa,LABEL'a dallan.
```

JNZ rel

Fonksiyon : ACC sıfır değil ise dallan (Jump if ACC is not zero)

Açıklama : ACC'nin bitlerinden herhangi biri 1 ise , belirtilen adrese dallan, aksi durumda bir sonraki komuta devam et.

Adresleme : Doğrudan

Etkilenen Bayraklar : Yok

Çevrim : 2

Kodlama :

0	1	1	1	0	0	0	0	relative adres

İşlem:

(PC)←(PC)+2
IF(A)≠0
THEN
(PC)←(PC)+rel

ÖRNEKLER:

```
JNZ   LABEL            ;Eğer ACC 00h değil ise ,LABEL'a dallan.
```

JZ rel

Fonksiyon : ACC sıfır ise dallan (Jump if ACC is zero)

Açıklama : ACC'nin bitlerinin hepsi 0 ise , belirtilen adrese dallan, aksi durumda bir sonraki komuta devam et.
Adresleme : Doğrudan

Etkilenen Bayraklar : Yok

Çevrim : 2

Kodlama : 2 byte

0	1	1	0	0	0	0	0	relative adres

İşlem:

(PC)←(PC)+2
IF(A)=0

THEN
(PC)←(PC)+rel

ÖRNEKLER:

```
JZ    LABEL            ;Eğer ACC 00h ise ,LABEL'a dallan.
```

LCALL addr16

Fonksiyon : Uzun çağırma (long call)

Açıklama : 16- bit adresle belirtilen yerdeki altprogram çağrılır.Altprogram, program hafızasında herhangi bir yerde olabilir. Bir sonraki komuta işaret eden adres, önce düşük byte olmak üzere yığına yazılır .

Adresleme : Doğrudan

Etkilenen Bayraklar: Yok

Çevrim : 2

Kodlama:

0	0	0	1	0	0	1	0	addr15-addr8	addr7-addr0

İşlem:
(PC)←(PC)+3
(SP)←(SP)+1
((SP))←(PC_{7-0})
(SP)←(SP)+1
((SP))←(PC_{15-8})
(PC)←$addr_{15-0}$

ÖRNEKLER:

```
LCALL       SUBRTN      ;SUBRTN ile belirlenen altprogramı çağır.
```

LJMP addr16

Fonksiyon : Uzun dallanma (long jump)

Açıklama : 16-bit adresle belirtilen yerde olan bir bölgeye durumdan bağımsız dallanma gerçekleştirilir.(unconditional branch).Bir sonraki komut, program hafızasında herhangi bir yerde olabilir.

Adresleme : Doğrudan

Etkilenen Bayraklar: Yok

Çevrim : 2

Kodlama:

0	0	0	0	0	0	1	0	addr15-addr8	addr7-addr0

İşlem:

(PC)←(PC)+3
(SP)←(SP)+1
((SP))←(PC_{7-0})
(SP)←(SP)+1
((SP))←(PC_{15-8})
(PC)←$addr_{15-0}$

ÖRNEKLER:

```
LJMP        LABEL       ;LABEL ile belirlenen hafıza bölgesine dallan.
```

MOV <Hedef byte>,<kaynak byte>

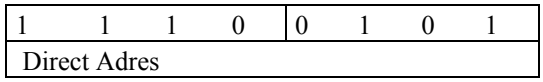
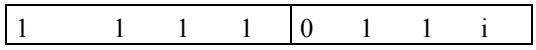
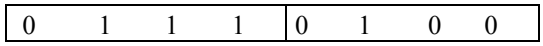
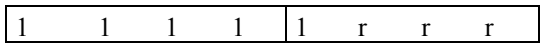
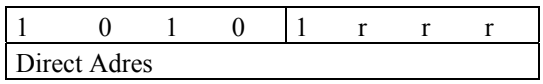
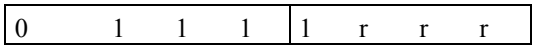
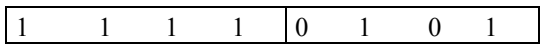
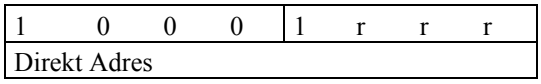
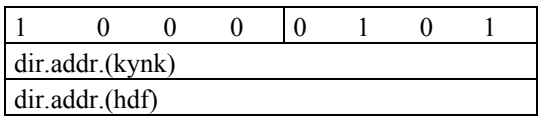
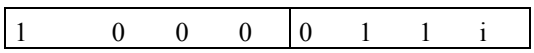
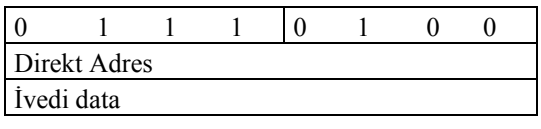
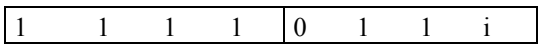
Fonksiyon : Byte değişkenini kopyala (Move byte variable)

Açıklama : Kaynak byte ile belirtilen byte değişkeni , hedef byte'a (ilk operand) kopyalanıp kaynakta bir değişiklik olmaz.

Adresleme : Bu işlem en esnek ,en çok kullanılan ve 15 farklı kaynak ve hedef adresleme modu kombinasyonuna sahip olan komuttur.

Etkilenen Bayraklar: Yok

Komut	Kodlama	İşlem	Çevrim
MOV A,Rn	1 1 1 0 1 r r r	(A)←(Rn)	1

MOV A,direct		(A)←(dir)	1
MOV A,@Ri		(A)←((Ri))	1
MOV A,#data		(A)←#data	1
MOV Rn,A		(Rn)←(A)	1
MOV Rn,direct		(Rn)←(dir)	2
MOV Rn,#data 1		(Rn)←#data	
MOV direct,A		(dir)←(A)	1
MOV direct, Rn		(dir)←(Rn)	1
MOV direct,direct		(dir)←(dir)	2
MOV direct,@Ri		(dir)←((Ri))	2
MOV direct,data		(dir)←#data	2
MOV @Ri,A		((Ri))←(A)	1

MOV @Ri,direct	1 0 1 0 0 1 1 i Direct Adres	((Ri))←(dir)	1
MOV @Ri,#da	0 1 1 1 0 1 1 i	((Ri))←#data	1

ÖRNEKLER:

```
MOV   A,R7
MOV   R7,A
MOV   R0,#30H     ;30H sabitini register R0'ya kopyala.
MOV   R0,30H      ;RAM bölgesi 30H'ının içeriğini R0'a kopyala.
MOV   30H,R0
MOV   30H,#45H    ;RAM bölgesi 30H'ı sabit 45H ile yükle.
MOV   A,@R0       ;R0 ile gösterilen hücre içeriğini ACC'ye kopyala.
MOV   @R0,A
MOV   @R0, 30H    ;30H 'in içeriğini R0 ile işaretli yere kopyala.
MOV   @R0, #30H   ;R0 işaretli yere sabit 30H'i Yükle.
MOV   A,#35H      ;ACC'ye 35H'yi yükle.
MOV   A,35H       ;RAM bölgesi 30H'in içeriğini ACC'ye kopyala.
MOV   35H,A
MOV   35H,30H     ;30H'in içeriğini, RAM Bölgesi 35H'ye kopyala.
```

MOV <Hedef bit>,<kaynak bit>

Fonksiyon : Veri bit'i kopyalama (move bit data)

Açıklama : Kaynak bit ile belirtilen Boolean (1 veya 0) değişken, hedef bite kopyalanır. Kaynak veya hedeften biri elde bayrağı olmalıdır.

Adresleme: Doğrudan

Etkilenen Bayraklar: C

Çevrim: 1

MOV C,Bit

Kodlama:

1 0 1 0 0 0 1 0	Bit adres	

İşlem: (C)← (bit)

MOV Bit,C

Kodlama:

1	0	1	0	0	0	1	0	Bit adres

İşlem: (Bit)← (C)

ÖRNEKLER:

```
MOV   C,P1.2      ;P1.2'sini elde bayrağına kopla.
MOV   23H,C       ;Elde bayrağını bit 23H'a yani dahili RAM bölgesi
                  ;24H'a kopyala
```

MOV DPTR,#data 16

Fonksiyon: DPTR'yı 16-bit bilgi ile yükle.(Load DPTR with a 16-bit constant)

Açıklama: DPTR registerı ikinci operand ile belirtilen 16-bit sabit ile yüklenir.Bu komut, 16-bit veri transfer eden tek komuttur.

Adresleme: Doğrudan.

Etkilenen Bayraklar: Yok.

Çevrim: 1

Kodlama:

1	0	0	1	0	0	0	0	ivedi data 15-8	ivedi data 7-0

İşlem: (DPTR)← #data15-0
(DPH) ←#data15-8
(DPL) ←#data 7-0

ÖRNEK:

```
MOV DPTR ,#1234H  ; 1234H değerini DPTR register'ına yükle.
```

MOV A,@A+<Base Reg>

Fonksiyon: Program Hafızadan byte kopya (Move Code Byte).

Açıklama: Program Hafızadan indisli-dolaylı (indexed-indirect) olarak adreslenen kaynak

byte'ı ACC 'ye kopyala.Taban registerı PC veya DPTR'dir..Program Hafızadaki kaynak byte'ın adresini bulmak için, ACC'nin içeriğini taban register ile toplanır.

Adresleme: İndisli – dolaylı

Etkilenen Bayraklar : Yok

Çevrim : 2

MOV A,@A+DPTR

Kodlama :

1	0	0	1	0	0	1	1

İşlem :

(A)←((A)+(DPTR))

MOV A,@A+PC

Kodlama :

1	0	0	0	0	0	1	1

İşlem :

(PC)←(PC)+1

(A)←((A)←(PC))

ÖRNEK:

```
MOVC A,@A+DPTR    ;Program Hafızanın bir byte içeriğini ACC'ye
                  ;kopyala kaynak byte'ın adresi, ACC'nin içeriği ile
                  ;DPTR'ın toplanmasıyla hesaplanır.


MOVC A,@A+PC      ;Program Hafızanın bir byte içeriğini ACC'ye
                  ;kopyala. Kaynak byte'ın adresi, ACC'nin içeriği
                  ;ile PC'nin toplanmasıyla hesaplanır.
```

MOVX <Hedef byte>,<kaynak byte>

Fonkiyon : Harici kopyalama (External Move)

Açıklama : Hari veri hafızasına veya bu hafızadan veri transferi.Kaynak veya hedeften biri ACC olmalı.

Adresleme : Dolaylı adresleme

Etkilenen Bayraklar : Yok

Çevrim : 2

Komut	Kodlama	İşlem
MOVX A,@Ri (A)←((Ri))	1 1 1 0 | 0 0 1 i	
MOVX A,@DPTR	1 1 1 0 | 0 1 0 1	(A)←((DPTR))
MOVX A,@Ri.A	1 1 1 1 | 0 1 1 i	((Ri)) ←(A)
MOVX @DPTR,A	1 1 1 1 | 0 00 0 0	((DPTR))←(A)

ÖRNEKLER:

```
MOVX @DPTR,A          ;DPTR ile işaretli yere ACC'yi kopyala.
MOVX A,@DPTR          ; DPTR ile işaretli yerden ACC'ye kopyala.
MOVX @RO,A            ;R0 ile işaretli yere ACC'yi kopyala.
MOVX A,@RO            ;R0 ile işaretli yerden ACC'ye kopyala.
```

MUL AB

Fonksiyon: Çarpma işlemi(Multiply)

Açıklama: ACC ve B register'ların da bulunan 8-bit tam sayıyı çarpar.16-bit sonucun düşük değerli byte'ı ACC ve yüksek değerlikli byte'I B register'ına yerleştirlir.

Adresleme: Register-Özel(Sadece ACC ve B)

Etkilenen Bayraklar: Eğer sonuç 255'ten(FFH) büyük ise, taşma bayrağı (OV) birlenir aksi durumda temizlenir. Elde bayrağı her zaman temizlenir.

Çevrim: 4

Kodlama :

1	0	1	0	0	1	0	0

İşlem :

$(A)_{7\text{-}0} \leftarrow (A) \times (B)$
$(B)_{15\text{-}8}$

ÖRNEK:

```
MUL AB      ; eğer ACC 50H ve B 0AH içeriyorsa çarpım 3200H olur ve B'de
            ; 32H ve ACC'de 00H bulunur.
```

NOP

Fonksiyon: İşlem yok(no operation)

Açıklama: İşlem bir sonraki komuta devam eder.

Adresleme: -

Etkilenen Bayraklar: Yok

Çevrim: 1

Kodlama :

0	0	0	0	0	0	0	0

İşlem :

$(PC) \leftarrow (PC)+1$

ORL <Hedef Byte>, <kaynak byte>

Fonksiyon: Byte değişkenleri için lojik OR(veya) işlemi. (Logical-OR for byte variables)

Açıklama: Bu komut karşılıklı bit-bit (bit wises) lojik-OR işlemini, belirten değişkenler arasında gerçekleştirir, sonuç, hedef byte'ta saklanır.

Adresleme: İki operant 6 adresleme modu oluşturur.Hedef ACC olduğnda, kaynak register, doğurdan, register-dolaylı ve ivedi adresleme modu kullanabilir.Hedef bir doğrudan adres olduğu zaman kaynak ACC veya ivedi veri olabilir.

Etkilenen Bayraklar: Yok.

ÖRNEKLER:

```
ORL P1,#00110010B       ;P1'in 5,4 ve 1 inci bitlerini "1"le.
ORL A,35H               ;ACC'yi RAM 35H'in içeriği ile OR'la.
ORL 35H,A               ;35H içeriğini ACC ile OR'la.
ORL 35H,#30H            ;35H'in içeriğini sabit 35H ile OR'la.
ORL A,@R0               ;ACC'yi R0' ın içeriği ile OR'la.
```

Komut	Kodlama	İşlem	Çevrim
ORL A,	0 1 0 0 | 1 r r r	(A)←(A)v(Rn)	1
ORL A,direct	0 1 0 0 | 0 1 0 1 Direct Adres	(A)←(A)v(dir)	1
ORL A,@Ri	0 1 0 0 | 0 1 1 i	(A)←(A)v((Ri))	1
ORL A,#data	0 1 0 0 | 0 1 0 0 İvedi Data	(A)←(A)v#data	1
ORL direct,A	0 1 0 0 | 0 0 1 0 Direct Adres	(dir)←(dir)v(A)	1
ORLdirect,#data	1 0 1 0 | 1 r r r Direct Adres İvedi Data	(dir)←(dir)v#data	2

ORL C <kaynak bit>

Fonksiyon: Elde biti 1-bit değişkeni ile lojik OR (veya) işlemi. (Logical-OR for bit variables)

Açıklama: Elde bir her zaman operandlardan biridir. Aynı zamanda işlemin hedefidir.Kaynak, herhangi bir adreslenebilir bittir .Kaynağın tersi (/) işareti ile belitilebilir.

Adresleme: Doğrudan.

Etkilenen Bayraklar: C.

Çevrim: 2.

ORL C bit

Kodlama:

0	1	1	1	0	0	1	0	Bit adres

İşlem:

(C)← (C) v (bit)

ORL C,/bit

Kodlama:

1	0	1	0	0	0	0	0	Bit adres

İşlem:

(C)← (C)v l(bit)

ÖRNEKLER:

```
ORL C,ACC.7       ;ACC.7 ile C'yi OR'la.Sonucu da C'ye yaz.
ORL C,/ACC.6      ;(/) işareti ACC.6'nın tersi anlamına gelir.
```

POP Direct

Fonksiyon: Yığından oku. (Pop from stack)

Açıklama: Yığın işaretçisi (Stack Pointer-SP) tarafından işaretli dahili RAM bölgesinden bir byte
okunur ve yığın işaretçisi bir azaltılır. Okunan değer doğrudan belirtilen yere transfer edilir.

Adresleme: Doğrudan.

Etkilenen Bayraklar:Yok.

Çevrim: 2.

Kodlama :

1	1	0	1	0	0	0	0	Direct adres

İşlem:

(direct)← ((SP))
(SP)←(SP)-1

ÖRNEKLER:

```
POP DPH     ;Yığın üstünden bir byte, DPTR'ın yüksek değerlikli
            ;byte'ına ;transfer edilir. SP bir azaltılır.
POP 35H     ;Yığın üstünden bir byte, dahili Ram hücresi 35H'a
            ;transfer edilir ve SP bir azaltılır.
```

PUSH Direct

Fonksiyon: Yığına yaz.(Push onto stack).

Açıklama: Yığın işaretçisi (SP) bir arttırılır doğrudan adreste belirtilen değişkenin içeriği SP. tarafından işaretli dahili RAM bölgesine kopyalanır.

Adresleme: Doğrudan.

Etkilenen Bayraklar:Yok.

Çevrim: 2.

Kodlama :

1	1	0	0	0	0	0	0	direct adres

İşlem:

(SP)← (SP)+1
((SP))←(direct)

ÖRNEKLER:

```
PUSH DPL    ;SP bir arttırılır ve DPTR'ın düşük değerlikli byte'ı SP
            ;ile  işaretli dahili RAM'e kopyalanır.
PUSH 35H    ;SP bir artırılır ve dahili RAM hücresi 35H'in içeriği SP
            ;ile işaretli RAM'e kopyalanır.
```

RET

Fonksiyon: Alt programdan dönüş(return from subroutine)

Açıklama: ACALL veya LCALL tarafından yürütülen altprogramdan, yığından PC olarak işleme girecek iki byte okunarak, ana programa dönülür.

Adresleme: Dolaylı.

Etkilenen Bayraklar: Yok.

Çevrim: 2

Kodlama:

0	0	1	0	0	0	1	0

İşlem:

$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP)-1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP)-1$

RETI

Fonksiyon: Kesmeden dönüş (return from interrupt)

Açıklama: PC, yığından geri okunur.Kesme hizmet programının bittiği, kesme kontrol lojiğine haber verilir.Örneğin; RETI komutu ile TMOD SFR'ın IE0 biti, donanım tarafından temizlenir.

Adresleme: Dolaylı.

Etkilenen Bayraklar: Yok.

Çevrim: 2.

Kodlama:

0	0	1	1	0	0	1	0

İşlem:

$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP)-1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP)-1$

RL A

Fonksiyon: ACC'yi sola döndür(Rotate ACC left)

Açıklama: ACC'deki 8-bit sola bir bit döndürülür, bit7,bit 0 olur.

Adresleme : Register-özel (sadece ACC)

Etkilenen Bayraklar : Yok.

Çevrim : 1
Kodlama:

0	0	1	0	0	0	1	1

İşlem :
$(A_{n+1}) \leftarrow (A_n)$,n=0-6
(A0)← (A7)

ÖRNEK:

```
RL A        ; ACC'nin 11000101B değerine sahip ise, bu komuttan sonra
            ;ACC'nin içeriği 10001011B olur.
```

RLC A

Fonksiyon: ACC'yi sola elde bayrağı üzerinden döndür(Rotate ACC left through Carry flag)

Açıklama: ACC'deki 8-bit sola bir bit döndürülür.Elde bayrağı bit sıfıra ve bit7 elde bayrağına kopyalanır.

Adresleme : Register-özel (sadece ACC)

Etkilenen Bayraklar : C

Çevrim : 1

Kodlama:

0	0	1	1	0	0	1	1

İşlem :
$(A_{n+1}) \leftarrow (A_n)$,n=0-6
(A0)← (C)
(C)← (A7)

ÖRNEK:

```
RLC A       ;ACC, 11000101B değerine sahip ve elde biti sıfır ise,
            ;bu komuttan sonra, ACC 10001010B ve elde bayrağı bir
            ;olur.
```

RR A

Fonksiyon: ACC'yi sağa döndür(Rotate ACC right)

Açıklama: ACC'deki 8-bit sağa bir bit döndürülür, bit0,bit 7'nin konumuna gelir..

Adresleme : Register-özel (sadece ACC)

Etkilenen Bayraklar : Yok.

Çevrim :1

Kodlama:

0	0	0	0	0	0	1	1

İşlem :

$(A_n) \leftarrow (A_{n+1}),n=0\text{-}6$

$(A7) \leftarrow (A0)$

ÖRNEK:

```
RR A        ; ACC'nin 11000101B değerine sahip ise, bu komuttan sonra
            ; ACC'nin içeriği 11100010B olur.
```

RRC A

Fonksiyon: ACC'yi sağa elde bayrağı üzerinden döndür(Rotate ACC right through Carry flag)

Açıklama: ACC'deki 8-bit sağa bir bit döndürülür.Elde bayrağı, bit7 ve bit sıfıra elde bayrağına kopyalanır.

Adresleme : Register-özel (sadece ACC)

Etkilenen Bayraklar : C

Çevrim :1

Kodlama:

0	0	0	1	0	0	1	1

İşlem :

$(A_n) \leftarrow (A_{n+1}),n=0\text{-}6$

$(A7) \leftarrow (C)$

$(C) \leftarrow (A0)$

ÖRNEK:

```
RRC A       ;ACC'nin 11000101B değerine sahip ve elde biti sıfır ise,
            ;bu komuttan sonra, ACC 01100010B ve elde bayrağı bir
            ;olur.
```

SETB <bit>

Fonksiyon: Bit birlenir (set bit)

Açıklama: Belirtilen bit birlenir.SETB elde bayrağı veya herhangi doğrudan adreslenebilen bit üzerinden işlem yapar.Diğer bayraklar etkilenmez.

Adresleme : Doğrudan

Etkilenen Bayraklar: C (SETB C komutunda)

Çevrim : 1

SETB C

Kodlama:

1	1	0	1	0	0	1	1

İşlem:

(C)←1

SETB Bit

Kodlama:

1	1	0	0	0	0	1	0	bit adres

İşlem:

(bit)←1

ÖRNEKLER:

```
SETB C          ;Elde bayrağını birle.
SETB P1.0       ;Çıkış portu P1'in bit 0'ını birle.
```

SJMP rel

Fonksiyon : Kısa dallanma (short jump)

Açıklama: Program, durumundan bağımsız olarak belirtilen adrese dallanır. Hedef aralığı, komuttan 128 byte once ve 127 byte sonra arasında değişmektedir.

Adresleme: Doğrudan

Etkilenen Bayraklar: Yok
Çevrim : 2

Kodlama:

1	0	0	0	0	0	0	0	dolaylı adres

İşlem :

(PC)←(PC)+2
(PC)←(PC)+rel

ÖRNEK:

```
SJMP  LABEL
```

SUBB A,<kaynak- byte>

Fonksiyon: Ödünçlü çıkartma (subtract with borrow)

Açıklama: Kaynak byte ve elde bayrağı ACC'den çıkartılır ve sonuç ACC'ye konur.Eğer bit 7 için bir ödünce (borrow) ihtiyaç varsa elde bayrağı birlenir, aksi halde temizlenir.2'nin tümleyeni (complement) sayılar birbirinden çıkartıldıklarında , bir taşma(over flow), taşma bayrağının (OV) birlenmesi ile belirtilir ve bu , 1 negatif sayıdan bir pozitif sayının çıkartılması sonucu bir negatif sayının üretildiği veya pozitif bir sayıdan negative bir sayının çıkartılmasıyla bir pozitif sayının üretildiği anlamına gelmektedir.

Adresleme: Kaynak için 4 adresleme methodu vardır; reg, dir, ind ve imm

Etkilenen Bayraklar : C, OV

Çevrim: 1

Komut Kodlama İşlem

SUBB A,Rn

1	0	0	1	1	r	r	r	(A)←(A)-(Rn)-C

SUBB A,direct

1	0	0	1	0	1	0	1	(A)←(A)-(dir)-C
Direct Adres								

SUBB A,@Ri

1	0	0	1	0	1	1	i

(A)←(A)-((Ri))-C

SUBB A,#data

1	0	0	1	0	1	0	0
İvedi Data							

(A)←(A)-#data-C

ÖRNEKLER:

```
SUBB A,35H      ;35H'in içeriğini ACC'den çıkar, eğer elde bayrağı
                ;bir ise sonucu da azalt ve ACC'ye yerleştir.
SUBB A,@R0      ;R0 ile işaretli hücrenin içeriğini ACC'den
                ;çıkar,elde bayrağı bir ise sonucu da azalt ve
SUBB A,#35H     ; ACC'ye yerleştir.Sabit 35H' i ACC'den çıkar elde
                ;bayrağı bir ise sonucu da azalt ve ACC'ye
                ;yerleştir.
```

SWAP A

Fonksiyon: ACC'nin iki 4-bit'ini yer değiştir. (Swap Nibbles within the ACC)

Açıklama: Bu komut ACC'nin 4-bit dödürülmesine eşittir.

Etkilenen Bayraklar: Yok.

Çevrim: 2.

Kodlama:

1	1	0	0	0	1	0	0

İşlem:

$(A_{3-0}) \Leftrightarrow (A_{7-4})$

XCH A, <byte>

Fonksiyon: ACC'yi byte değişkeni ile karşılıklı değiştir.(exchange ACC with byte variable)

Açıklama: Bu komut bir byte değişkenin ACC'ye ve ACC'nin bu byte değişkene kopyalandığı iki MOV işlemine eşittir.

Adresleme: Üç adresleme modu vardır; reg,dir,ind.

Etkilenen Bayraklar: Yok..

Çevrim: 1.

Komut	Kodlama	İşlem
XCH A,Rn	1 1 0 0 1 r r r	(A)⇔(Rn)
XCH A,direct	1 1 0 0 0 1 0 1 Direct Adres	(A)⇔(dir)
XCH A,@Ri	1 1 0 0 0 1 1 i	(A)⇔((Ri))

ÖRNEKLER:

```
XCH     A,R7

XCH     A,35H

XCH     A,@R0
```

XCHD A,@Ri

Fonksiyon: Karşılıklı düşük 4-biti değiştir.(Exchange digit)

Açıklama: XCH A,@Ri komutuna benzer. Fark sadece karşılıklı düşük 4-bitlerin değiştirilmesindedir.

Adresleme: Dolaylı.

Etkilenen Bayraklar: Yok.

Çevrim: 1.

Kodlama:

1 1 0 1	0 1 1 i

İşlem:

$(A_{3-0}) \Leftrightarrow ((Ri))_{3-0}$

XRL <hedef byte>,<Kaynak Byte>

Fonksiyon: Byte değişkenleri için lojik exclusive-OR işlemi.

Açıklama: İki byte arasında karşılıklı bit-bit(bit wise) XOR işlemi gerçekleştirilir.Sonuç hedef byte'a yerleştirilir.

Adresleme: İki operand 6 adresleme modu oluşturur. Hedef ACC olduğunda, kaynak, register, doğrudan, register-dolaylı ve ivedi adresleme modu kullanılabilir. Hedef bir doğrudan adres olduğu zaman, kaynak ACC veya ivedi veri olabilir.

Etkilenen Bayraklar: Yok.

Komut	Kodlama	İşlem	Çevrim
XRL	0 1 1 0 1 r r r	A,Rn(A)←(A)∀(Rn)	1
XRL A,direct	0 1 1 0 0 1 0 1 Direct Adres	(A)←(A) ∀ (dir)	1
XR	0 1 1 0 0 1 1 i	A,@Ri(A)←(A) ∀ ((Ri))	1
XRL A,#data	0 1 1 0 0 1 0 0 İvedi Data	(A)←(A) ∀#data	1
XRL direct,A	0 1 1 0 0 0 1 0 Direct Adres	(dir)←(dir) ∀ (A)	1
XRLdirect,#data	0 1 1 0 0 0 1 1 Direct Adres İvedi Data	(dir)←(dir) ∀#data	2

ÖRNEKLER:

```
XRL  A,R6          ;ACC ile R6'yı XOR'la ve sonucu ACC' ye yaz.
XRL  A,#FFH        ;CPL A komutuna eşittir.
XRL  A,35H         ;ACC ile 35H'in içerikleri XOR'lanır.
XRL  35H,A         ;35H içeriği ile ACC'yi XOR'la ve sonucu dahili Ram
                   ;hücresine yaz.
XRL  35H,#30H      ;RAM hücresi 35H ile sabit 30H'I XOR'la.Sonuç
                   ;dahili RAM hücresi 35H' e yaz.
XRL A,@R0          ;ACC ile R0 ile işaretli dahili RAM hücresinin
                   ;içerikleri XOR'lanır.
```