12. MİKROİŞLEMCİ KOMUT KÜMESİ

Mikroişlemci girişine uygulanan komutları yerine getiren karmaşık mantıksal devrelere sahiptir. Mikroişlemci komut kümesi, mikroişlemcinin üretim sırasında tanımlanmış, anlamlı olan ikili girişlerinin fonksiyonu olan kümedir.

12.1. 6800 Komut Kümesi

6800 mikroişlemcisi 72 çeşit komuta sahiptir.
Aşağıda bu komutlar için kullanılan kısa komut adları (**mnemonic**) ve açıklamaları verilmiştir.

ABA	B akümülatörün içeriğini A akümülatörüne ekle
ADC	Bellek içeriğini, elde ile birlikte A veya B akümülatörüne ekle
ADD	Bellek içeriğini, A veya B akümülatörüne ekle
AND	Bellek içeriği ile A veya B akümülatörünü lojik VE işlemi yap
ASL	Bellek içeriğini, A veya B akümülatörünü Aritmetik sola öteleme yap
ASR	Bellek içeriğini, A veya B akümülatörünü Aritmetik sağa öteleme yap
BCC	Eğer elde bayrağı “0” ise dallan
BCS	Eğer elde bayrağı “1” ise dallan
BEQ	Eğer sonuç sıfır ise (sıfır bayrağı “1”) dallan
BGE	Eğer sonuç sıfıra eşit veya sıfırdan büyük ise dallan
BGT	Eğer sonuç sıfırdan büyük ise dallan
BHI	Eğer sonuç işaretli olarak büyük ise dallan
BIT	A veya B akümülatör için bit test

BLE	Eğer sonuç sıfıra eşit veya sıfırdan küçük ise dallan
BLS	Eğer sonuç aynı veya küçük ise dallan
BLT	Eğer sonuç sıfırdan küçük ise dallan
BMI	Eğer sonuç eksi ise dallan
BNE	Eğer sonuç sıfıra eşit değilse dallan
BPL	Eğer sonuç artı ise dallan
BRA	Daima dallan
BSR	Alt programa dallan
BVC	Eğer taşma bayrağı “0” ise dallan
BVS	Eğer taşma bayrağı “1” ise dallan
CBA	A ile B akümülatörünü karşılaştır
CLC	Elde bayrağını “0” yap
CLI	Kesme örtme bayrağını “0” yap
CLR	Bellek içeriğini, A veya B akümülatörüne temizle
CLV	Taşma bayrağını “0” yap
CMP	Bellek içeriği ile A veya B akümülatörünü karşılaştır
COM	Bellek içeriğini, A veya B akümülatörünü 1’e tümle
CPX	Bellek içeriği ile X dizin yazmacını karşılaştır
DAA	A akümülatörünü ondalığa ayarla
DEC	Bellek içeriğini, A veya B akümülatörünü azalt

DES	Yığın işaretçi yazmacını azalt
DEX	Dizin yazmacını azalt
EOR	Bellek içeriği ile A veya B akümülatörünü lojik ÖZEL VEYA işlemi yap
INC	Bellek içeriğini, A veya B akümülatörünü azalt
INS	Yığın işaretçi yazmacını artır
INX	Dizin yazmacını artır
JMP	Koşulsuz sıçra
JSR	Alt programa sıçra
LDA	Bellek içeriğini, A veya B akümülatörüne yükle
LDS	Bellek içeriğini, yığın işaretçi yazmacına yükle
LDX	Bellek içeriğini, dizin yazmacına yükle
LSR	Bellek içeriğini, A veya B akümülatörünü lojik sağa öteleme yap
NEG	Bellek içeriğini, A veya B akümülatörünü 2'ye tümle (eksi işaretli yap)
NOP	İşlem yok (yalnız program sayıcıyı artırır)
ORA	Bellek içeriği ile A veya B akümülatörünü lojik VEYA işlemi yap
PSH	A veya B akümülatörünü yığına it
PUL	A veya B akümülatörünü yığından çek
ROL	Bellek içeriğini, A veya B akümülatörünü elde ile birlikte sola döndür
ROR	Bellek içeriğini, A veya B akümülatörünü elde ile birlikte sağa döndür
RTI	Kesme hizmet programından geri dön

RTS	Alt programdan geri dön
SBA	A akümülatöründen B akümülatörünü çıkar
SBC	Bellek içeriğini, ödünç ile birlikte A veya B akümülatöründen çıkar
SEC	Elde bayrağını "1" yap
SEI	Kesme örtme bayrağını "1" yap
SEV	Taşma bayrağını "1" yap
STA	A veya B akümülatörünün içeriğini bellekte sakla
STS	Yığın işaretçi yazmacının içeriğini bellekte sakla
STX	Dizin yazmacının içeriğini bellekte sakla
SUB	Bellek içeriğini, A veya B akümülatöründen çıkar
SWI	Yazılım ile kesme
TAB	A akümülatörünü B akümülatörüne transfer et
TAP	A akümülatörünü durum yazmacına transfer et
TBA	B akümülatörünü A akümülatörüne transfer et
TPA	Durum yazmacını A akümülatörüne transfer et
TST	Bellek içeriğini, A veya B akümülatörünü test et
TSX	Yığın işaretçiyi dizin yazmacına transfer et
TXS	Dizin yazmacını yığın işaretçiye transfer et
WAI	Donanım kesmesi bekle

12.2. 6800 Komut Tablosu

Tablo 12.2-1 6800 mikroişlemcisinin işlem kodu haritası

									Akümülatör A				Akümülatör B			
	İçerik	İçerik	Bağıl	İçerik	Akü A	Akü B	Dizin	Geniş	Hemen	Doğru.	Dizin	Geniş	Hemen	Doğru.	Dizin	Geniş
L H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		SBA	BRA	TSX	NEG				SUB							
1	NOP	CBA		INS					CMP							
2			BHI	PULA					SBC							
3			BLS	PULB	COM											
4			BCC	DES	LSR				AND							
5			BCS	TXS					BIT							
6	TAP	TAB	BNE	PSHA	ROR				LDA							
7	TPA	TBA	BEQ	PSHB	ASR					STA				STA		
8	INX		BVC		ASL				EOR							
9	DEX	DAA	BVS	RTS	ROL				ADC							
A	CLV		BPL		DEC				ORA							
B	SEV	ABA	BMI	RTI					ADD							
C	CLC		BGE		INC				CPX							
D	SEC		BLT		TST				BSR		JSR					
E	CLI		BGT	SWI			JMP		LDS				LDX			
F	SEI		BLE	WAI	CLR					STS				STX		

Tablo 12.2-2 6800 komut tablosu

12.2.1. 6800 Komut Tablosu Açıklamaları

Komut tablosunun sonunda, tabloda tekrarlanarak kullanılan ortak işaretler, kısaltmalar, kodların anlamlarını kısaca açıklanmıştır.

Kullanılan işaretlerin açıklamaları

Op	İşlem kodu (onaltılık)	*Komut Kodu Açıklamaları*
~	Mikroişlemci çevrim sayısı	
#	Program bayt adedi	
+	Aritmetik artı	*Komut Çalışması Açıklamaları*
-	Aritmetik eksi	
·	Lojik VE (AND)	
+	Lojik VEYA (OR)	
⊕	Lojik ÖZEL VEYA (XOR)	
M	Bellek adresi (adres değeri M)	
(M)	Belleğin içeriği (M adresinde saklanan veri değeri)	
M_{SP}	Yığın İşaretçinin gösterdiği adres	
$(\overline{M})$	Bellek içeriğinin tümleyeni (M adresinde saklanan veri değerinin tümleyeni)	
→	Transfer işlemi, yönü	
0	Bit = sıfır	
00	Bayt = sıfır	
H	Yarım elde (3. Bitten)	*Durum Kodu Yazmacı Bitleri Açıklamaları*
I	Kesme örtme	
N	Eksi (işaret biti)	
Z	Sıfır (bayt)	
V	Taşma, 2'ye tümleyen	
C	Elde (7. bitten)	

0	İşlem sonucu ne olursa olsun bit değeri daima "0" olur.	
1	İşlem sonucu ne olursa olsun bit değeri daima "1" olur.	
×	İşlem sonucuna göre bit değeri "1" ya da "0" olur.	
•	İşlem sonucu ne olursa olsun bit değeri etkilenmez!	
CCR	Durum Kodu Yazmacı	*Komut Çalışması Açıklamaları*
L	16-bit değerin alt baytı	
H	16-bit değerin üst baytı	

Durum Yazmacı açıklamaları

a	sonuç = 10000000 ise V = 1
b	sonuç ≠ 00000000 ise C = 1
c	BCD değerin büyük ağırlıklı karakteri > 9 ise C = 1
d	önceki çalışmada işlenen = 10000000 ise V = 1
e	önceki çalışmada işlenen = 01111111 ise V = 1
f	Ötelemede N ⊕ C işlemi sonucu V = 1 olur
g	Sonucun en büyük ağırlıklı biti =1 ise N = 1
h	Çıkarmada 8-bit'ten 2'ye tümleyen taşma varsa V = 1
i	Sonuç sıfırdan küçük (bit 15=1) ise N = 1
j	Durum yazmacının içeriği yığından yüklenir
k	Kesme oluştuğunda I=1 olur.
m	Durum yazmacı içeriğinin tamamı A Akümülatöründen yüklenir

12.3. Yükleme, Saklama ve Transfer Komutları

Tablo 12.3-1 Yükleme, Saklama ve Transfer Komutları

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
Temizle	CLR	00 → (M)	•	•	0	1	0	0
	CLRA	00 → A	•	•	0	1	0	0
	CLRB	00 → B	•	•	0	1	0	0
Akümülatöre yükle	LDAA	(M) → A	•	•	×	×	0	•
	LDAB	(M) → B	•	•	×	×	0	•
Akümülatörü sakla	STAA	A → (M)	•	•	×	×	0	•
	STAB	B → (M)	•	•	×	×	0	•
Akümülatördeki veriyi yığına it	PSHA	A → (M_{SP}) , SP - 1 → SP	•	•	•	•	•	•
	PSHB	B → (M_{SP}) , SP - 1 → SP	•	•	•	•	•	•
Yığındaki veriyi akümülatöre çek	PULA	SP + 1 → SP , (M_{SP}) → A	•	•	•	•	•	•
	PULB	SP + 1 → SP , (M_{SP}) → B	•	•	•	•	•	•
Akü. A → Akü. B	TAB	A → B	•	•	×	×	0	•
Akü. B → Akü. A	TBA	B → A	•	•	×	×	0	•

Örnek 12.3-1 CLRA

İşlemden önce bellek ve yazmaçlar:
A= 41H=01000001 olsun.

İşlemden sonra bellek ve yazmaçlar:
A= 00H=00000000 olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonuç sıfır olduğu için Z=1, Sıfır artı işaretli kabul edildiği için N=0,
Elde ve taşma olamayacağı için C=0 ve V=0 olur.
Yarım elde(H) ve kesme örtme(I) biti etkilenmez.

Örnek 12.3-2 LDAA #7AH

İşlemden önce bellek ve yazmaçlar:
A= 41H
(M)=7AH olur.

İşlemden sonra bellek ve yazmaçlar:
A=7AH olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonuç sıfırdan farklı olduğu için Z=0, Yüklenen değer artı işaretli olduğu için N=0,
Taşma olamayacağı için V=0 olur.
Elde(C), Yarım elde(H) ve kesme örtme(I) biti etkilenmez.

Örnek 12.3-3 LDAA 7AH

İşlemden önce bellek ve yazmaçlar:
A= 41H
M=7AH, (M)=(7AH)=12H olsun.

İşlemden sonra bellek ve yazmaçlar:
A=12H olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonuç sıfırdan farklı olduğu için Z=0, Yüklenen değer artı işaretli olduğu için N=0,
Taşma olamayacağı için V=0 olur. Elde(C), Yarım elde(H) ve kesme örtme(I) biti etkilenmez.

Örnek 12.3-4 LDAA 40H,X

İşlemden önce bellek ve yazmaçlar:
A=12H
X=5300H
M=5300H+40H, (M)=(5340H)=85H olsun.

İşlemden sonra bellek ve yazmaçlar:
A=85H olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonuç sıfırdan farklı olduğu için Z=0, Yüklenen değer eksi işaretli olduğu için N=1,
Taşma olamayacağı için V=0 olur. Elde(C), Yarım elde(H) ve kesme örtme(I) biti etkilenmez.

Örnek 12.3-5 STAB 4723H

İşlemden önce bellek ve yazmaçlar:
B=37H
M=4723H, (M)=(4723H)=45H olsun.

İşlemden sonra bellek ve yazmaçlar:
(4723H)=37H olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonuç sıfırdan farklı olduğu için Z=0,
Saklanan değer artı işaretli olduğu için N=0,
Taşma olamayacağı için V=0 olur. Elde(C), Yarım elde(H) ve kesme örtme(I) biti etkilenmez.

Örnek 12.3-6 PSHA

İşlemden önce bellek ve yazmaçlar:
A=28H
SP=00FFH
M_{SP}=00FFH, (M_{SP})=(00FFH)=05H olsun.

İşlemden sonra bellek ve yazmaçlar:
(00FFH)=28H
SP=00FEH olur.

İşlemden sonra Durum Kodu Yazmacı bitleri: Hiçbiri etkilenmez.

Örnek 12.3-7 PULB

İşlemden önce bellek ve yazmaçlar:
B=34H
SP= 00FEH
M_{SP+1}=00FFH,(M_{SP+1})=(00FFH)=28H olsun.

İşlemden sonra bellek ve yazmaçlar:
SP=00FFH
B=28H olur.

İşlemden sonra Durum Kodu Yazmacı bitleri: Hiçbiri etkilenmez.

Örnek 12.3-8 TAB

İşlemden önce bellek ve yazmaçlar:
A=28H
B=94H olsun.

İşlemden sonra bellek ve yazmaçlar:
A=28H
B=28H olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonuç sıfırdan farklı olduğu için Z=0, Sonuç artı işaretli olduğu için N=0,
Tablodan V=0 olur. Elde(C), Yarım elde(H) ve kesme örtme(I) biti etkilenmez.

12.4. Aritmetik İşlem Komutları

Tablo 12.4-1 Aritmetik İşlem Komutları

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı 5 H	4 I	3 N	2 Z	1 V	0 C
Topla	ADDA	A + (M) → A	×	•	×	×	×	×
	ADDB	B + (M) → B	×	•	×	×	×	×
Akümülatörleri Topla	ABA	A + B → A	×	•	×	×	×	×
Eldeli Topla	ADCA	A + (M) + C → A	×	•	×	×	×	×
	ADCB	B + (M) + C → B	×	•	×	×	×	×
Çıkar	SUBA	A - (M) → A	•	•	×	×	×	×
	SUBB	B - (M) → B	•	•	×	×	×	×
Akümülatörleri Çıkar	SBA	A - B → A	•	•	×	×	×	×
Eldeli Çıkar	SBCA	A - (M) - C → A	•	•	×	×	×	×
	SBCB	B - (M) - C → B	•	•	×	×	×	×
2'ye tümle	NEG	00 - (M) → (M)	•	•	×	×	a	b
	NEGA	00 - A → A	•	•	×	×	a	b
	NEGB	00 - B → B	•	•	×	×	a	b
Akü. Ondalığa Ayarla	DAA	BCD toplama için A akü. ayar.	•	•	×	×	×	c
Azalt	DEC	(M) - 1 → (M)	•	•	×	×	d	•
	DECA	A - 1 → A	•	•	×	×	d	•
	DECB	B - 1 → B	•	•	×	×	d	•
Artır	INC	(M) + 1 → (M)	•	•	×	×	e	•
	INCA	A + 1 → A	•	•	×	×	e	•
	INCB	B + 1 → B	•	•	×	×	e	•

Örnek 12.4-1 ADDB 82H

İşlemden önce bellek ve yazmaçlar:

B=34H
M=82H, (M)=(82H)=7AH olsun.

İşlemden sonra bellek ve yazmaçlar:

```
    B =    34H     0011 0100
+ (82H) = + 7AH   + 0111 1010
    B =    AEH     1010 1110
```

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç sıfırdan farklı olduğu için Z=0, Sonuç eksi işaretli olduğu için N=1 olur.
34H+7AH=AEH<256 olduğu için Elde yoktur ve C=0 olur.
4+10=14<16 olduğu için Yarım elde yoktur ve H=0 olur.
İşaretli işlem yapılırsa: 34H + 7AH = AEH>+127 olduğu için taşma vardır ve V=1 olur.

Örnek 12.4-2 ADCA #7AH

İşlemden önce bellek ve yazmaçlar:

A=3CH
(M)=7AH
C=1 olsun.

İşlemden sonra bellek ve yazmaçlar:

```
    A =    3CH     0011 1100
  7AH =    7AH     0111 1010
+   C = + 1     +           1
    A =    B7H     1011 0111
```

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç eksi işaretli olduğu için N=1, sıfırdan farklı olduğu için Z=0 olur.
3CH+7AH+1=B7H<256 olduğu için Elde yoktur ve C=0 olur.
12+10+1=23>15 olduğu için Yarım elde vardır ve H=1 olur.
İşaretli işlem yapılırsa: 3CH + 7AH + 1 = B7H>+127 olduğu için taşma vardır ve V=1 olur.

Örnek 12.4-3 SBCA 0F100H

İşlemden önce bellek ve yazmaçlar:

A=25H
C=1 olsun.
M=F100H, (M)=(F100H)=50H

İşlemden sonra bellek ve yazmaçlar:

	A	= 25H	0010 0101
-	(F100H)	= - 50H	- 0101 0000
-	C	= - 1	- 1
	A	= D4H	0010 1010

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç eksi işaretli olduğu için N=1, sıfırdan farklı olduğu için Z=0 olur.
Ödünç alındığı için C=1 olur. Yarım elde(H) biti etkilenmez.
İşaretli işlem yapılırsa: +25H – 50H = -2BH, sonuç -1 ile -128 arasında olduğu için taşma yoktur ve V=0 olur.

Örnek 12-17 NEGA

İşlemden önce bellek ve yazmaçlar:

A=50H olsun.

İşlemden sonra bellek ve yazmaçlar:

	0	= 00H	0000 0000
-	A	= - 50H	- 0101 0000
	A	= B0H	1011 0000

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç eksi işaretli olduğu için N=1, sıfırdan farklı olduğu için Z=0,
Komut tablosunda **a** sonuç=10000000 ise V=1 açıklamasıyla işaretli 8-bit değerin yalnız -128 olması durumunda ters işaretli karşılığı +128 > +127 olduğu için taşma vardır ve V=1 olur.
Komut tablosunda **b** sonuç≠00000000 ise C=1 açıklamasıyla işleme giren değer yalnız sıfır olması durumunda ödünç alınır. Burada değer sıfırdan farklı olduğu için C=1 olur.
Yarım elde(H) biti etkilenmez.

Örnek 12.4-4 DEC 40H,X

İşlemden önce bellek ve yazmaçlar:

X= 0200H
M=40H olsun. (240H)=80H

İşlemden sonra bellek ve yazmaçlar:

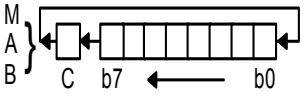
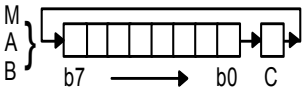
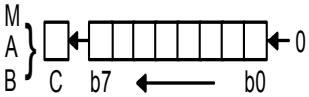
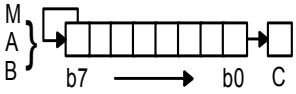
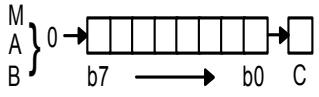
1000 0000
(240H) = 0111 1111 olur

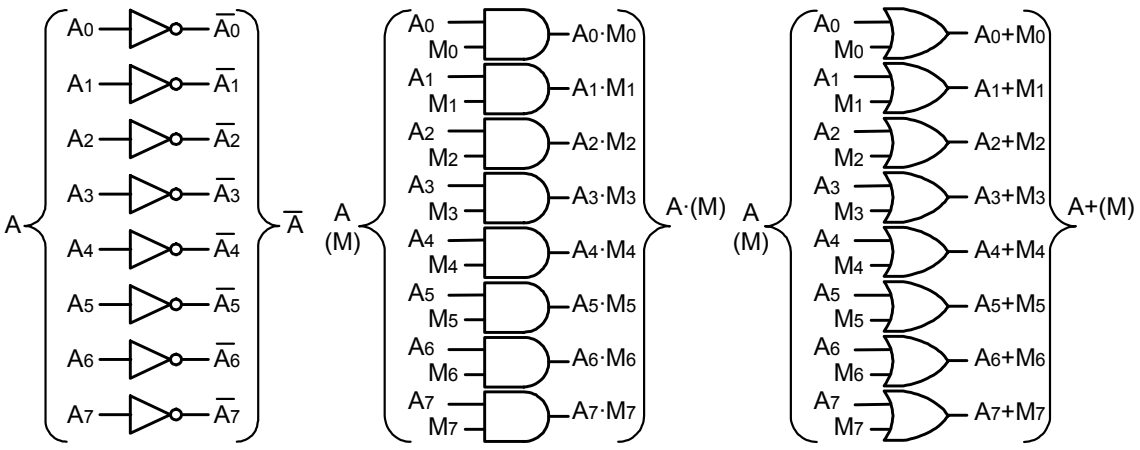
İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç artı işaretli olduğu için N=0, sıfırdan farklı olduğu için Z=0 olur.
Komut tablosunda **d** önceki çalışmada işlenen=10000000 ise V=1 açıklamasıyla işaretli 8-bit değerin yalnız -128 olması durumunda taşma olur. Çünkü -128 azaltıldığında -129 olması gerekirken 8-Bit işaretli değer sınırları içinde -129 yoktur. Burada işaretli değer -128 olduğu için taşma vardır ve V=1 olur.
Elde(C) ve Yarım elde(H) biti etkilenmez.

12.5. Mantıksal İşlem Komutları

Tablo 12.5-1 Mantıksal İşlem Komutları

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı 5 H	4 I	3 N	2 Z	1 V	0 C
1'e tümle	COM	$(\overline{M}) \rightarrow (M)$	•	•	×	×	0	1
	COMA	$(\overline{A}) \rightarrow (A)$	•	•	×	×	0	1
	COMB	$(\overline{B}) \rightarrow (B)$	•	•	×	×	0	1
Lojik VE	ANDA	$A \cdot (M) \rightarrow A$	•	•	×	×	0	•
	ANDB	$B \cdot (M) \rightarrow B$	•	•	×	×	0	•
Lojik VEYA	ORAA	$A + (M) \rightarrow A$	•	•	×	×	0	•
	ORAB	$B + (M) \rightarrow B$	•	•	×	×	0	•
Lojik ÖZEL VEYA	EORA	$A \oplus (M) \rightarrow A$	•	•	×	×	0	•
	EORB	$B \oplus (M) \rightarrow B$	•	•	×	×	0	•
Sola döndür	ROL		•	•	×	×	f	×
	ROLA		•	•	×	×	f	×
	ROLB		•	•	×	×	f	×
Sağa döndür	ROR		•	•	×	×	f	×
	RORA		•	•	×	×	f	×
	RORB		•	•	×	×	f	×
Aritmetik sola kaydır	ASL		•	•	×	×	f	×
	ASLA		•	•	×	×	f	×
	ASLB		•	•	×	×	f	×
Aritmetik sağa kaydır	ASR		•	•	×	×	f	×
	ASRA		•	•	×	×	f	×
	ASRB		•	•	×	×	f	×
Lojik sağa kaydır	LSR		•	•	0	×	f	×
	LSRA		•	•	0	×	f	×
	LSRB		•	•	0	×	f	×



Örnek 12.5-1 COM 20H

İşlemden önce bellek ve yazmaçlar:
M=20H olsun. (20H)=ACH

İşlemden sonra bellek ve yazmaçlar:
AC 10101100
(20H) = 53 01010011 olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonuç artı işaretli olduğu için N=0, sıfırdan farklı olduğu için Z=0 olur.
Komut Tablosunda Taşma V=0, Elde C=1 verilmiştir.
Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.5-2 ORAA #40H

İşlemden önce bellek ve yazmaçlar:
A=2EH

İşlemden sonra bellek ve yazmaçlar:
A = 2EH 00101110
40H = 40H 01000000
A = 6EH 01101110 olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonuç artı işaretli olduğu için N=0, sıfırdan farklı olduğu için Z=0 olur.
Komut Tablosunda Taşma biti V=0, Elde C, Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.5-3 ROLB

İşlemden önce bellek ve yazmaçlar:
B= 3CH
C= 1

İşlemden sonra bellek ve yazmaçlar:
B = 3CH = 00111100
B = 79H = 01111001 ve C= 0 olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Elde biti, sola öteleme sonucu b7 değerini alarak C=0 olur.
Sonuç artı işaretli olduğu için N=0, sıfırdan farklı olduğu için Z=0 olur.
Komut tablosunda **f** Ötelemede N⊕C işlemi sonucu V=1 açıklamasıyla değerin komutdan önceki işareti ötelemeden sonraki işareti karşılaştırılır. Değerin işlemden önceki işareti b7 biti C bitine ve sonraki işareti N bitine aktarıldığından N⊕C=0⊕0=0, taşma biti V=0 olur.
Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.5-4 ASL 4321H

İşlemden önce bellek ve yazmaçlar:
M= 4321H, (M)=(4321H)= 9DH
C= 0

İşlemden sonra bellek ve yazmaçlar:
(4321H) = 9DH = 10011101
(4321H) = 3AH = 00111010 ve C= 1 olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Elde biti sola öteleme sonucu b7 değerini alarak C=1 olur.
Sonuç artı işaretli olduğu için N=0, sıfırdan farklı olduğu için Z=0 olur.
Komut tablosunda **f** Ötelemede N⊕C işlemi sonucu V=1 açıklamasıyla N⊕C=0⊕1=1, taşma biti V=1 olur. Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.5-5 ASRB

İşlemden önce bellek ve yazmaçlar:
B= CBH
C= 1

İşlemden sonra bellek ve yazmaçlar:
B= CBH = 11001011
B= E5H = 11100101 ve C= 1 olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sonucun işaretinin değişmemesi için işaret biti olan b7 değeri korunur.
Elde biti sağa öteleme sonucu b0 değerini alarak C=1 olur.
Sonuç eksi işaretli olduğu için N=1, sıfırdan farklı olduğu için Z=0 olur.
Komut tablosunda **f** Ötelemede N⊕C işlemi sonucu V=1 açıklamasıyla N⊕C=1⊕1=0, taşma biti V=0 olur. Yarım elde H ve kesme örtme I biti etkilenmez.

12.6. Karşılaştırma ve Test Komutları

Tablo 12.6-1 Karşılaştırma ve Test komutları

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
Karşılaştır	CMPA	A - (M)	•	•	×	×	×	×
	CMPB	B - (M)	•	•	×	×	×	×
Aküm. Karşılaştır	CBA	A - B	•	•	×	×	×	×
Sıfır veya eksiliği Test et	TST	(M) - 00	•	•	×	×	**0**	**0**
	TSTA	A - 00	•	•	×	×	**0**	**0**
	TSTB	B - 00	•	•	×	×	**0**	**0**
Bit test	BITA	A · (M)	•	•	×	×	**0**	•
	BITB	B · (M)	•	•	×	×	**0**	•

Örnek 12.6-1 CMPA #25H

İşlemden önce bellek ve yazmaçlar:

A = 98H
(M) = 25H

İşlemden sonra bellek ve yazmaçlar:

A = 98H 1001 1000
- (M) = - 25H - 0010 0101
= 73H 0111 0011

A= 98H ve (M)= 25H olarak kalır.

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç artı işaretli olduğu için N=0, Sonuç sıfırdan farklı olduğu için Z=0 olur.
İşaretli işlem yapılırsa: -68H – 25H = -8DH > 80H sonuç -1 ile -128 arasında olmadığı için taşma vardır ve V=1 olur.
İşaretsiz çıkarmada (98H-25H=73H) ödünç alınmadığı için C=0 olur.
Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.6-2 CMPA 25H

İşlemden önce bellek ve yazmaçlar:

A = 98H
(M) = (25H) = 98H

İşlemden sonra bellek ve yazmaçlar:

A = 98H 1001 1000
- (25H) = - 98H - 1001 1000
= 00H 0000 0000

A= 98H ve (25H)=98H olarak kalır.

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç artı işaretli olduğu için N=0, sıfır olduğu için Z=1 olur.
İşaretli işlem yapılırsa: -68H – (-68H) = 00H olur. Taşma yoktur ve V=0 olur.
Ödünç alınmadığı için C=0 olur. Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.6-3 CMPB 82H,X

İşlemden önce bellek ve yazmaçlar:

B=E8H
X= 0300H
M=82H+X, (0382H)=F6H olsun.

İşlemden sonra bellek ve yazmaçlar:

B = E8H 1110 1000
- (0382H) = - F6H - 1111 0110
= F2H 1111 0010

B= E8H ve (0382H)=F6H olarak kalır.

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç eksi işaretli olduğu için N=1, sıfırdan farklı olduğu için Z=0 olur.
İşaretli işlem yapılırsa: E8H – F6H = - 18H + 0AH = -0EH, sonuç -1 ile -128 arasında olduğu için taşma yoktur ve V=0 olur.
Ödünç alındığı için C=1 olur.
Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.6-4 CBA

İşlemden önce bellek ve yazmaçlar:

A= 75H
B= E8H olsun.

İşlemden sonra bellek ve yazmaçlar:

A = 75H 0111 0101
- B = - E8H - 1110 1000
= 8DH 1000 1101

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç eksi işaretli olduğu için N=1, sıfırdan farklı olduğu için Z=0 olur.
İşaretli işlem yapılırsa: +75H – E8H = 75H + 18H = 8DH, sonuç 0 ile +127 arasında olmadığı için taşma vardır ve V=1 olur.
Ödünç alındığı için C=1 olur.
Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.6-5 TSTA

İşlemden önce bellek ve yazmaçlar:

A= 75H olsun.

İşlemden sonra bellek ve yazmaçlar:

A = 75H = 0111 0101

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç artı işaretli olduğu için N=0, sıfırdan farklı olduğu için Z=0 olur.
Komut tablosundan Taşma V=0, Elde C=0. Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.6-6 BITA #80H

İşlemden önce bellek ve yazmaçlar:

A= 83H
(M)= 80H olur.

İşlemden sonra bellek ve yazmaçlar:

A = 83H 1000 0011
(M) = 80H 1000 0000
= 80H 1000 0000
(M) = 80H, A = 83H olarak kalır.

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç eksi işaretli olduğu için N=1 olur.
Sonuç sıfırdan farklı olduğu için Z=0 olur. Bu aynı zamanda A akümülatörünün b_7 bitinin sıfırdan farklı, b_7=1 olduğunu gösterir.
Komut tablosundan Taşma V=0. Elde C, Yarım elde H ve kesme örtme I biti etkilenmez.

Örnek 12.6-7 BITA 40H

İşlemden önce bellek ve yazmaçlar:

A= 83H
(40H)= 01H olsun.

İşlemden sonra bellek ve yazmaçlar:

A = 83H 1000 0011
(40H) = 01H 0000 0001
= 01H 0000 0001
(40H) = 01H, A = 83H olarak kalır.

İşlemden sonra Durum Kodu Yazmacı bitleri:

Sonuç artı işaretli olduğu için N=0 olur. Sonuç sıfırdan farklı olduğu için Z=0 olur.
A akümülatörünün b_0 bitiyle 40H adresinin b_0 bitine bakılır. b_0=1 olduğunu gösterir.
Komut tablosundan Taşma V=0. Elde C, Yarım elde H ve kesme örtme I biti etkilenmez.

12.7. Dizin Yazmacı ve Yığın İşlem Komutları

Tablo 12.7-1 Dizin yazmacı ve Yığın işlem komutları

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
Dizin yazmac. karşılaş.	CPX	$X_H : X_L$ - (M : M+1)	•	•	**g**	×	**h**	•
Dizin yazmacı azalt	DEX	X - 1 → X	•	•	•	×	•	•
Yığın işaretcisini azalt	DES	SP - 1 → SP	•	•	•	•	•	•
Dizin yazmacı artır	INX	X + 1 → X	•	•	•	×	•	•
Yığın işaretcisini artır	INS	SP + 1 → SP	•	•	•	•	•	•
Dizin yazmacına yükle	LDX	(M : M+1) → $X_H : X_L$	•	•	**i**	×	**0**	•
Yığın işaretçisine yükle	LDS	(M : M+1) → $SP_H : SP_L$	•	•	**i**	×	**0**	•
Dizin yazmacını sakla	STX	$X_H : X_L$ → (M : M+1)	•	•	**i**	×	**0**	•
Yığın işaretcisini sakla	STS	$SP_H : SP_L$ → (M : M+1)	•	•	**i**	×	**0**	•
Dizin Yaz.→ Yığın İşar	TXS	X - 1 → SP	•	•	•	•	•	•
Yığın İşar → Dizin Yaz	TSX	SP + 1 → X	•	•	•	•	•	•

Örnek 12.7-1 CPX #86H

İşlemden önce bellek ve yazmaçlar:
(M:M+1) = 00:86
X = 5126H olsun.

İşlemden sonra bellek ve yazmaçlar:
X = 5126H 01010001 00100110
- (M:M+1) = - 0086H - 00000000 10000110
= 50A0H 01010000 10100000
İşaretli işlem yapılırsa:
26H – 86H = 26H + 7AH = A0H > 7FH

İşlemden sonra Durum Kodu Yazmacı bitleri:
Elde C, Yarım elde H ve kesme örtme I biti etkilenmez.
Sonuç artı işaretli olduğu için (b15=0) N=0,
Komut tablosunda; **g** Sonucun en büyük ağırlıklı biti =1 ise N = 1
Sonuç sıfırdan farklı olduğu için Z=0,
İşaretli 8-bit değer için taşma olduğu için V=1 olur.
Komut tablosunda; **h** Çıkarmada 8-bit'ten 2'ye tümleyen taşma varsa V = 1

Örnek 12.7-2 DEX

İşlemden önce bellek ve yazmaçlar:
X = 1234H olsun.

İşlemden sonra bellek ve yazmaçlar:
X = 1233H 00010010 00110011

İşlemden sonra Durum Kodu Yazmacı bitleri:
Sıfır Z biti dışında hiçbir biti etkilenmez. Sonuç sıfırdan farklı olduğu için Z=0 olur.

Örnek 12.7-3 LDX #1453H

İşlemden önce bellek ve yazmaçlar:
(M:M+1) = 14:53
X = 1234H olsun.

İşlemden sonra bellek ve yazmaçlar:
X = 1453H 00010100 01010011

İşlemden sonra Durum Kodu Yazmacı bitleri:
Elde C, Yarım elde H ve kesme örtme I biti etkilenmez.
Sonuç artı işaretli olduğu için (b15=0) N=0,
Komut tablosunda; **i** Sonuç sıfırdan küçük (bit 15=1) ise N = 1
Sonuç sıfırdan farklı olduğu için Z=0,
Taşma olamayacağı için V=0 olur.

Örnek 12.7-4 STS 2000H

İşlemden önce bellek ve yazmaçlar:
(M:M+1) = (2000H:2001H) = 8F:FF
SP = 007FH olsun.

İşlemden sonra bellek ve yazmaçlar:
(2000H:2001H) = 00H:7FH
(2000H) = 00H ve (2001H) = 7FH olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Elde C, Yarım elde H ve kesme örtme I biti etkilenmez.
Sonuç artı işaretli olduğu için (b15=0) N=0,
Komut tablosunda; **i** Sonuç sıfırdan küçük (bit 15=1) ise N = 1
Sonuç sıfırdan farklı olduğu için Z=0,
Taşma olmayacağı için V=0 olur.

Örnek 12.7-5 TSX

İşlemden önce bellek ve yazmaçlar:
X = 1234H
SP = 00F8H olsun.

İşlemden sonra bellek ve yazmaçlar:
X = 00F9H
SP = 00F8H olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Hiçbir biti etkilenmez.

12.8. Durum Kodu Yazmacı İşlem Komutları

Tablo 12.8-1 Durum Kodu Yazmacı İşlem Komutları

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
Elde bitini "0" yap	CLC	0 → C	•	•	•	•	•	**0**
Kesme bitini "0" yap	CLI	0 → I	•	**0**	•	•	•	•
Taşma bitini "0" yap	CLV	0 → V	•	•	•	•	**0**	•
Elde bitini "1" yap	SEC	1 → C	•	•	•	•	•	**1**
Kesme bitini "1" yap	SEI	1 → I	•	**1**	•	•	•	•
Taşma bitini "1" yap	SEV	1 → V	•	•	•	•	**1**	•
Aküm. CCR'ye yükle	TAP	A → CCR	**m**	**m**	**m**	**m**	**m**	**m**
CCR'yi Aküm. yükle	TPA	CCR → A	•	•	•	•	•	•

Örnek 12.8-1 CLC

İşlemden önce bellek ve yazmaçlar:
CCR = FFH , C = 1 olsun.

İşlemden sonra bellek ve yazmaçlar:
CCR = FEH = 11111110, C = 0 olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Elde C biti dışında hiçbir biti etkilenmez. C = 0 olur.

Örnek 12.8-2 CLI

İşlemden önce bellek ve yazmaçlar:
CCR = FFH , I = 1 olsun.

İşlemden sonra bellek ve yazmaçlar:
CCR = EFH = 11101111, I = 0 olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:
Kesme örtme I biti dışında hiçbir biti etkilenmez. I = 0 olur.

Örnek 12.8-3 TPA

İşlemden önce bellek ve yazmaçlar:
A = 55H, CCR = C0H
H=0, I=0, N=0, Z=0, V=0, C=0 olsun.

İşlemden sonra bellek ve yazmaçlar:
A = C0H, CCR = C0H
H=0, I=0, N=0, Z=0, V=0, C=0 olarak kalır.

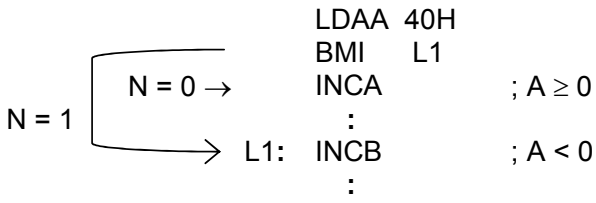
İşlemden sonra Durum Kodu Yazmacı bitleri:
Hiçbir biti etkilenmez.

12.9. Dallanma Komutları

Tablo 12.9-1 Dallanma Komutları

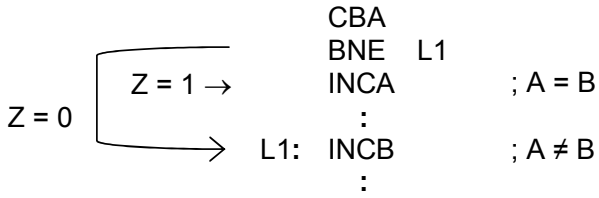
İşlem, Açıklama	Kısa Komut Adı	Dallanma Koşulu	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
Daima dallan	BRA	YOK	•	•	•	•	•	•
Artı işaretli ise dallan	BPL	N = 0	•	•	•	•	•	•
Eksi ise dallan	BMI	N = 1	•	•	•	•	•	•
Sıfırdan ≠ ise dallan	BNE	Z = 0	•	•	•	•	•	•
Sıfıra = ise dallan	BEQ	Z = 1	•	•	•	•	•	•
Taşma "0" ise dallan	BVC	V = 0	•	•	•	•	•	•
Taşma "1" ise dallan	BVS	V = 1	•	•	•	•	•	•
Elde "0" ise dallan	BCC	C = 0	•	•	•	•	•	•
Elde "1" ise dallan	BCS	C = 1	•	•	•	•	•	•
Yüksek ise dallan	BHI	C + Z = 0	•	•	•	•	•	•
Aynı veya düşük ise	BLS	C + Z = 1	•	•	•	•	•	•
Sıfırdan ≥ ise dallan	BGE	N ⊕ V = 0	•	•	•	•	•	•
Sıfırdan < ise dallan	BLT	N ⊕ V = 1	•	•	•	•	•	•
Sıfırdan > ise dallan	BGT	Z + (N ⊕ V) = 0	•	•	•	•	•	•
Sıfırdan ≤ ise dallan	BLE	Z + (N ⊕ V) = 1	•	•	•	•	•	•

Örnek 12.9-1 BMI



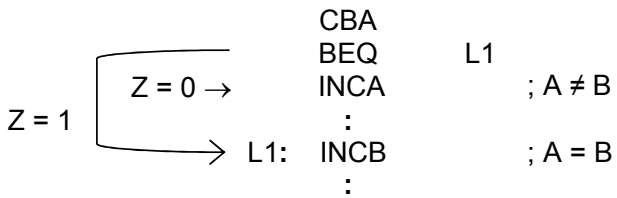
Bu örnekte bellekte 40H adresindeki 8-bit değer A akümülatörüne yüklenir. Bu değerin işareti eksi ise N=1 olur ve L1 etiketinin gösterdiği adrese gidilerek INCB komutu çalıştırılarak programa devam edilir. Değerin işareti artı ise N=0 olur ve INCA komutu çalıştırılır. Sayısal örnekler: (40H)=8FH, B=20H ise A=8FH, N=1 olur ve L1 etiketine gidilerek B artırılır. Sonuç olarak B=21H olur. (40H)=14H, B=20H ise A=14H, N=0 olur ve devam edilerek A artırılır. Sonuç olarak A=15H olur.

Örnek 12.9-2 BNE



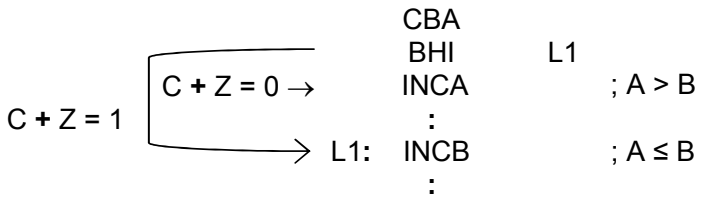
Bu örnekte A akümülatörü, B akümülatörüyle karşılaştırılmakta A=B ise karşılaştırma komutunda A-B çıkarma işleminde değerlerin birbirinden farklı olması durumunda sonuç sıfırdan farklı olur. Bu durumda Sıfır bayrağı Z=0 olur ve L1 etiketinin gösterdiği adrese gidilerek INCB komutu çalıştırılarak programa devam edilir. Değerlerin birbirine eşit olması durumunda Z=1 olur ve INCA komutu çalıştırılır. Sayısal örnekler: A=24H, B=05H ise Z=0 olur ve L1 etiketine gidilerek B artırılır. Sonuç olarak B=06H olur. A=24H, B=24H ise Z=1 olur ve devam edilerek A artırılır. Sonuç olarak A=25H olur.

Örnek 12.9-3 BEQ



Bu örnekte A akümülatörü, B akümülatörüyle karşılaştırılmakta A=B ise karşılaştırma komutunda A-B çıkarma işleminde değerlerin birbirine eşit olması durumunda sonuç sıfır olur. Bu durumda Sıfır bayrağı Z=1 olur ve L1 etiketinin gösterdiği adrese gidilerek INCB komutu çalıştırılarak programa devam edilir. Değerlerin birbirinden farklı olması durumunda Z=0 olur ve INCA komutu çalıştırılır. Sayısal örnekler: A=24H, B=24H ise Z=1 olur ve ve L1 etiketine gidilerek B artırılır. Sonuç olarak B=25H olur. A=24H, B=05H ise Z=0 olur devam edilerek A artırılır. Sonuç olarak A=25H olur.

Örnek 12.9-4 BLS



Bu örnekte A akümülatörü, B akümülatörüyle karşılaştırılmakta A≤B ise karşılaştırma komutunda A-B çıkarma işleminde sonucun sıfır veya sıfırdan küçük olması durumu için Elde C=1 veya Sıfır bayrağı Z=1 olur ve L1 etiketinin gösterdiği adrese gidilerek INCB komutu çalıştırılarak programa devam edilir. Aksi takdirde INCA komutu ile program çalışmasına devam eder. Sayısal örnek verilirse; A=72H ve B=92H ise C=1, Z=0 olur ve L1 etiketine gidilerek B artırılır. Sonuç olarak B=93H olur.

Örnek 12.9-5 BGE

Bu örnekte A akümülatörü, B akümülatörüyle işaretli olarak karşılaştırılmakta ve karşılaştırma komutunda A-B çıkarma işleminde A≥B olması durumu için Eksi bayrağı N ve Taşma bayrağı V aynı olur ve L1 etiketinin gösterdiği adrese gidilerek INCB komutu çalıştırılarak programa devam edilir. Aksi takdirde INCA komutu ile program çalışmasına devam eder. Sayısal örnek verilirse; A=4FH ve B=86H ise N=1, 4FH-86H=4FH+7AH=C9H >7FH V=1, N ⊕ V = 0 olur ve L1 etiketine gidilerek B artırılır. Sonuç olarak B=87H olur.

```
                    CBA
                    BGE     L1
N ⊕ V = 1 →         INCA            ; A ≥ B
                    :
N ⊕ V = 0
                L1: INCB            ; A < B
                    :
```

12.10. Sıçrama Komutu

Tablo 12.10-1 Sıçrama Komutu

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
Koşulsuz dallan, Sıçra	JMP	özel işlem	•	•	•	•	•	•

Tablo 12.10-2 Adresleme şekline göre sıçrama komutlarının açıklaması

Komutun Çalışmasından Önce ve Sonra Yığın İşaretçisi (SP) , Yığın bölgesi ve Program Sayıcısının (PC) Durumu

JMP, Koşulsuz Dallan, Sıçra :

a) Dizinlenmiş Adresleme

PC	Ana Program
n	Komutun işlem kodu = 6EH
n+1	K = uzaklık (İşaretsiz 8-bit)
n+2	Bir Sonraki Komut

SP	Yığın
SP-2	
SP-1	
→ SP	

PC	Ana Program
X+K	Gidilen adresteki Komut
X+K+1	

b) Genişletilmiş Doğrudan Adresleme

PC	Ana Program
n	Komutunun işlem kodu = 7EH
n+1	Gidilecek yerin Adresi (N_H)
n+2	Gidilecek yerin Adresi (N_L)
n+3	Bir Sonraki Komut

SP	Yığın
SP-2	
SP-1	
→ SP	

PC	Ana Program
N	Gidilen adresteki Komut
N+1	

Örnek 12.10-1 JMP 20H,X

İşlemden önce bellek ve yazmaçlar:
PC=1200H
X=1234H olsun.

İşlemden sonra bellek ve yazmaçlar:
X=1234H
PC=1234H+20H, PC=1254H olur.

Örnek 12.10-2 JMP 1800H

İşlemden önce bellek ve yazmaçlar:
PC=1200H olsun.

İşlemden sonra bellek ve yazmaçlar:
PC=1800H olur.

12.11. Alt program çağırma ve Dönüş Komutları

Tablo 12.11-1 Alt program çağırma ve dönüş komutları

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
Altprograma dallan	BSR	özel işlem	•	•	•	•	•	•
Altprograma sıçra	JSR	özel işlem	•	•	•	•	•	•
Altprog. geri dön	RTS	özel işlem	•	•	•	•	•	•

Tablo 12.11-2 Adresleme şekline göre alt program çağırma ve dönüş komutları

JSR, Ana Programdan Alt Programa Sıçra :

a) Dizinlenmiş Adresleme

PC	Ana Program
n	Komutun işlem kodu = ADH
n+1	K = uzaklık (İşaretsiz 8-bit)
n+2	Bir Sonraki Komut

SP	Yığın
→ SP-2	
SP-1	$(n+2)_H$
SP	$(n+2)_L$

PC	Alt Program
X+K	Alt programın ilk Komutu
X+K+1	

b) Genişletilmiş Doğrudan Adresleme

PC	Ana Program
n	Komutunun işlem kodu = BDH
n+1	Alt programın Adresi (N_H)
n+2	Alt programın Adresi (N_L)
n+3	Bir Sonraki Komut

SP	Yığın
→ SP-2	
SP-1	$(n+3)_H$
SP	$(n+3)_L$

PC	Alt Program
N	Alt programın ilk Komutu
N+1	

BSR, Ana Programdan Alt Programa Dallan :

PC	Ana Program
n	Komutunun işlem kodu = 8DH
n+1	K = uzaklık (İşaretli 8-Bit)
n+2	Bir Sonraki Komut

SP	Yığın
→ SP-2	
SP-1	$(n+2)_H$
SP	$(n+2)_L$

PC	Alt Program
n+2+K	Alt programın ilk Komutu
n+3+K	

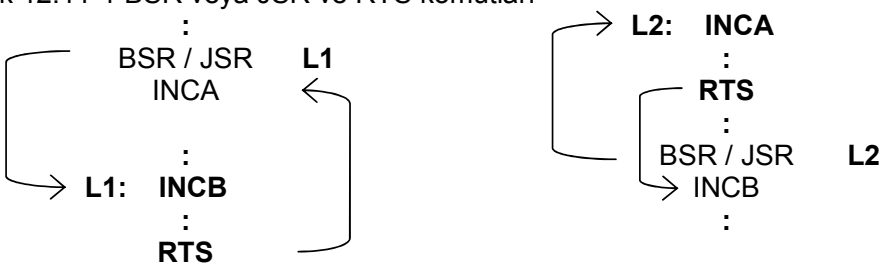
RTS, Alt Programdan Ana Programa Dönüş :

PC	Alt Program
n	Komutunun işlem kodu = 39H
n+1	Bir Sonraki Komut

SP	Yığın
SP-2	
SP-1	N_H
→ SP	N_L

PC	Ana program
N	Bir Sonraki Komut
N+1	

Örnek 12.11-1 BSR veya JSR ve RTS komutları



Örnek 12.11-2 BSR 1200H

İşlemden önce bellek ve yazmaçlar:
PC=1257H
SP=007FH
M_{SP}=007FH, (M_{SP})=(007FH)=05H
M_{SP-1}=007EH, (M_{SP-1})=(007EH)=00H olsun.

İşlemden sonra bellek ve yazmaçlar:
PC=1200H
(007FH)=59H
(007EH)=12H
SP=007DH olur

Örnek 12.11-3 JSR 45H,X

İşlemden önce bellek ve yazmaçlar:
PC=1259H
X=1234H
SP=007FH
M_{SP}=007FH, (M_{SP})=(007FH)=00H
M_{SP-1}=007EH, (M_{SP-1})=(007EH)=00H olsun.

İşlemden sonra bellek ve yazmaçlar:
X=1234H
PC=1234H+45H=1279H
(007FH)=5BH
(007EH)=12H
SP=007DH olur

Örnek 12.11-4 JSR 5600H

İşlemden önce bellek ve yazmaçlar:
PC=125BH
SP=007FH
M_{SP}=007FH, (M_{SP})=(007FH)=00H
M_{SP-1}=007EH, (M_{SP-1})=(007EH)=00H olsun.

İşlemden sonra bellek ve yazmaçlar:
PC=5600H
(007FH)=5EH
(007EH)=12H
SP=007DH olur

Örnek 12.11-5 RTS

İşlemden önce bellek ve yazmaçlar:
PC=5640H
SP=007DH
M_{SP}=007FH, (M_{SP})=(007FH)=5EH
M_{SP-1}=007EH, (M_{SP-1})=(007EH)=12H olsun.

İşlemden sonra bellek ve yazmaçlar:
PC=125EH
(007FH)=5EH
(007EH)=12H
SP=007FH olur

12.12. Kesme İşlem Komutları

Tablo 12.12-1 Kesme işlem komutları

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
Yazılım ile kesme	SWI	özel işlem	•	1	•	•	•	•
Kesme bekle	WAI	özel işlem	•	k	•	•	•	•
Kesmeden geri dön	RTI	özel işlem	j	j	j	j	j	j

Şekil 12.12-1 6800 mikroişlemcisi kesmeler için program akış diyagramı

Tablo 12.12-2 Kesme Hizmet Programından Ana Programa Dönüş, RTI komutu

RTI, Kesme Servis Programından Ana Programa Dönüş :

Örnek 12.12-1 SWI

İşlemden önce bellek ve yazmaçlar:

A=45H, B=89H, CCR=C5H, I=0
PC=C250H, X=1234H SP=007FH
M_{SP}=007FH, (M_{SP})=(007FH)=00H
M_{SP-1}=007EH, (M_{SP-1})=(007EH)=00H
M_{SP-2}=007DH, (M_{SP-1})=(007DH)=00H
M_{SP-3}=007CH, (M_{SP-1})=(007CH)=00H
M_{SP-4}=007BH, (M_{SP-1})=(007BH)=00H
M_{SP-5}=007AH, (M_{SP-1})=(007AH)=00H
M_{SP-6}=0079H, (M_{SP-1})=(0079H)=00H
(FFFAH:FFFBH)=E200H olsun.

İşlemden sonra bellek ve yazmaçlar:

(007FH)=51H
(007EH)=C2H
(007DH)=34H
(007CH)=12H
(007BH)=45H
(007AH)=89H
(0079H)=C5H
A=45H, B=89H, CCR=D5H, I=1
PC=E200H, X=1234H, SP=0078H olur.

İşlemden sonra Durum Kodu Yazmacı bitleri:

Kesme örtme I biti dışında hiçbir biti etkilenmez. I = 1 olur.

Örnek 12.12-2 WAI

İşlemden önce bellek ve yazmaçlar:

A=45H, B=89H, CCR=C5H, I=0
PC=C250H, X=1234H SP=007FH
M_{SP}=007FH, (M_{SP})=(007FH)=00H
M_{SP-1}=007EH, (M_{SP-1})=(007EH)=00H
M_{SP-2}=007DH, (M_{SP-1})=(007DH)=00H
M_{SP-3}=007CH, (M_{SP-1})=(007CH)=00H
M_{SP-4}=007BH, (M_{SP-1})=(007BH)=00H
M_{SP-5}=007AH, (M_{SP-1})=(007AH)=00H
M_{SP-6}=0079H, (M_{SP-1})=(0079H)=00H
(FFF8H:FFF9H)=E300H
(FFFCH:FFFDH)=E400H olsun.

İşlemden sonra bellek ve yazmaçlar:

(007FH)=51H
(007EH)=C2H
(007DH)=34H
(007CH)=12H
(007BH)=45H
(007AH)=89H
(0079H)=C5H
A=45H, B=89H, CCR=D5H, I=1
X=1234H, SP=0078H olur.
IRQ kesme işareti için PC=E300H
NMI kesme işareti için PC=E400H

İşlemden sonra Durum Kodu Yazmacı bitleri:

Kesme örtme I biti dışında hiçbir biti etkilenmez. I = 1 olur.

Tablo 12.12-3 Kesme Hizmet Programından Ana Programa Dönüş, RTI komutu

RTI, Kesme Servis Programından Ana Programa Dönüş :

PC	Kesme Servis Programı
n	Komutunun işlem kodu = 3BH
n+1	Bir Sonraki Komut

SP	Yığın
SP-7	
SP-6	Durum Kodu (CCR)
SP-5	Akümülatör B
SP-4	Akümülatör A
SP-3	Dizin Yaz.(X_H)
SP-2	Dizin Yaz.(X_L)
SP-1	PC_H(N_H)
→ SP	PC_L(N_L)

PC	Ana program
N	Bir Sonraki Komut
N+1	

Örnek 12.12-3 RTI

İşlemden önce bellek ve yazmaçlar:

A=12H, B=34H, CCR=FFH, I=1
PC=E435H, X=5678H SP=0078H
M_{SP}=007FH, (M_{SP})=(007FH)=51H
M_{SP-1}=007EH, (M_{SP-1})=(007EH)=C2H
M_{SP-2}=007DH, (M_{SP-1})=(007DH)=34H
M_{SP-3}=007CH, (M_{SP-1})=(007CH)=12H
M_{SP-4}=007BH, (M_{SP-1})=(007BH)=45H
M_{SP-5}=007AH, (M_{SP-1})=(007AH)=89H
M_{SP-6}=0079H, (M_{SP-1})=(0079H)=C5H olsun

İşlemden sonra bellek ve yazmaçlar:

(007FH)=51H
(007EH)=C2H
(007DH)=34H
(007CH)=12H
(007BH)=45H
(007AH)=89H
(0079H)=C5H
A=45H, B=89H, CCR=C5H, I=0
X=1234H, SP=007FH, PC=C251H olur.

İşlemden sonra Durum Kodu Yazmacı bitleri: Bütün bitleri yığından yüklenir.

12.13. Özel Komutlar

Tablo 12.13-1 İşlem yok, NOP komutu

İşlem, Açıklama	Kısa Komut Adı	Boole / Aritmetik İşlem	Durum Yazmacı					
			5	4	3	2	1	0
			H	I	N	Z	V	C
İşlem yok	NOP	Yalnız PC yazmacını artır	•	•	•	•	•	•

14. MİKROİŞLEMCİ PROGRAMLAMA TEKNİKLERİ

14.1. Programlamaya Giriş

Örnek Pr. 14-1 Bellekte bir adreste bulunan verinin başka bir adrese transfer edilmesi.

a) 0040H bellek adresindeki 8-bit veriyi 0042H adresine transfer eden programı yazınız.

b) 0040H bellek adresindeki 16-bit veriyi 0042H adresine transfer eden programı yazınız.

Çözüm:

a)

0040H adresindeki 8-Bit veriyi A yazmacına yükle	LDAA 0040H
A yazmacındaki 8-Bit veriyi 0042H adresine sakla	STAA 0042H

b)

0040H adresindeki 16-Bit veriyi X yazmacına yükle	LDX 0040H
X yazmacındaki 16-Bit veriyi 0042H adresine sakla	STX 0042H

Örnek Pr. 14-2 0040h bellek adresi ile 0041h adresindeki 8-bit veriyi toplayan ve sonucu 0042h adresinde saklayan programı yazınız.

Çözüm:

Toplamada başlangıç için elde sıfır yapılır	CLC
0040H adresindeki veriyi A yazmacına yükle	LDAA 0040H
A yazmacındaki veriye 41H adresindeki veriyi ekle	ADCA 0041H
A yazmacındaki veriyi 0042H adresinde sakla	STAA 0042H

Örnek Pr. 14-3 0040h bellek adresindeki 8-bit veriden 0041h adresindeki 8-bit veriyi çıkaran ve sonucu 0042h adresinde saklayan programı yazınız.

Çözüm:

Çıkarmada başlangıç için ödünç(elde) sıfır yapılır	CLC
0040H adresindeki veriyi A yazmacına yükle	LDAA 0040H
A yazm.daki veriden 41H adresindeki veriyi çıkar	SBCA 0041H
A yazmacındaki veriyi 0042H adresinde sakla	STAA 0042H

Örnek Pr. 14-4 0040h bellek adresindeki 8-bit veriyi 1-bit sola öteleyen ve sonucu 0041h adresinde saklayan programı yazınız.

Çözüm:

0040H adresindeki veriyi A yazmacına Yükle	LDAA 0040H
A yazmacındaki veriyi 1-bit sola ötele	ASLA
A yazmacındaki veriyi 0041H adresinde sakla	STAA 0041H

Örnek Pr. 14-5 0040h bellek adresindeki 8-bit verinin düşük ağırlıklı 4-bitini 0041h adresinde saklayan programı yazınız. 0041h bellek adresindeki 8-bit verinin yüksek ağırlıklı 4-bitini sıfırlayın.

Çözüm:

0040H adresindeki veriyi A yazmacına yükle	LDAA 0040H
A yazm. veri ile 00001111 değerini VE işlemi yap	ANDA #00001111B
A yazmacındaki veriyi 0041H adresinde sakla	STAA 0041H

Örnek Pr. 14-6 0040h bellek adresindeki veriyi sıfır ile dolduran (temizleyen) programı yazınız.

Çözüm:

A yazmacına 0 değerini yükle	LDAA #0H
A yazmacındaki veriyi 0040H adresinde sakla	STAA 0040H
veya	
0040H adresindeki veriyi sıfırla	CLR 0040H

14.2. Mikroişlemcilerin Gelişmiş Komutları

Mikroişlemcinin bazı komutları program tasarımında sıkça karşılaşılacağı düşünülen bir fonksiyonu yerine getirir.

Örnek Pr. 14-7 32+29 İşlemini Toplamadan Sonra Akümülatörü Ondalığa Ayarla (DAA) Komutunu kullanarak yapınız.

Çözüm:

Çevirici giriş kaynak dosyası:

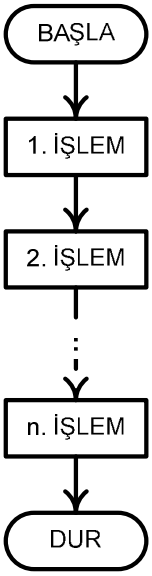
```
; ********************************************************************************
; *  BCD kodlu 2 basamak onaltılık sayıların toplamını DAA komutunu      *
; *  kullanarak bulan program                                            *
; ********************************************************************************
;
            ORG 0D019H      ; programın başlangıç adresi
BASLA:      CLC             ; Elde bayrağını sıfırla
            LDAA  #32H      ; A akümülatörüne 32H yükle
            ADCA  #29H      ; A aküm. Elde ile 29h topla
            DAA             ; sonucu BCD'ye dönüştür
            END             ; programın sonu
```

Çevirici program listesi çıkış dosyası:

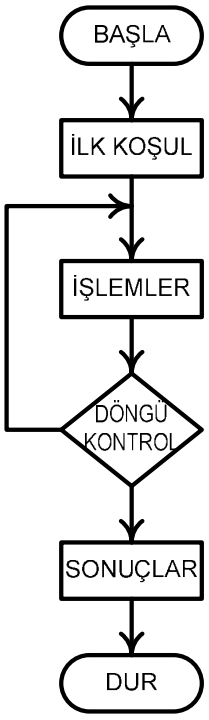
```
D019                  ORG   0D019H  ; programın başlangıç adresi
D019  0C      BASLA:  CLC           ; Elde bayrağını sıfırla
D01A  8632            LDAA  #32H    ; A akümülatörüne 32H yükle
D01C  8929            ADCA  #29H    ; A aküm. Elde ile 29h topla=5Bh
D01E  19              DAA           ; sonucu BCD'ye dönüştür. 5Bh → DAA → 61h
0000                  END           ; programın sonu
```

14.3. Programlama İçin Akış Diyagramı Yöntemi

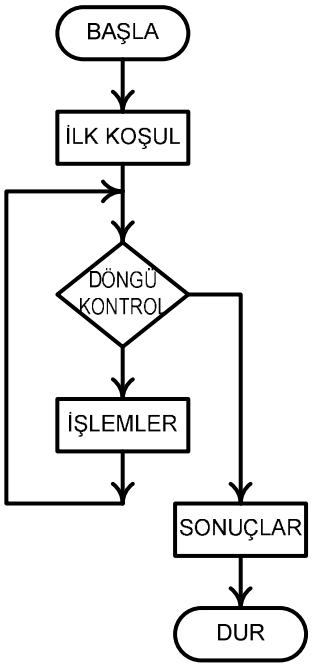
Program tasarımlarında çok karşılaşılan problemlerin genel olarak çözüm yöntemini veren akış diyagramı yapıları vardır.



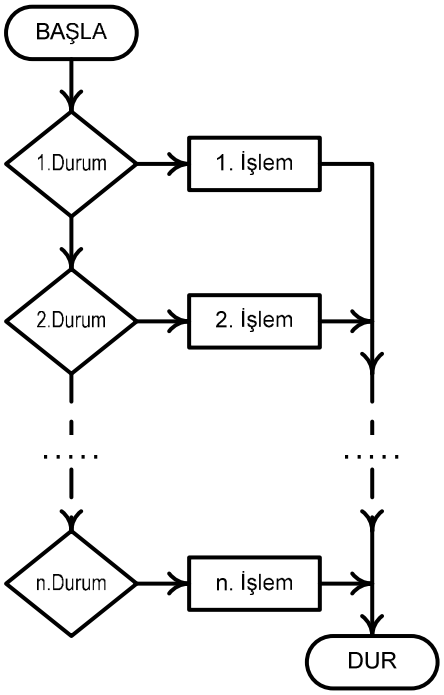
Şekil 14-1 Sıralı yapılan basit işlemler için akış diyagramı



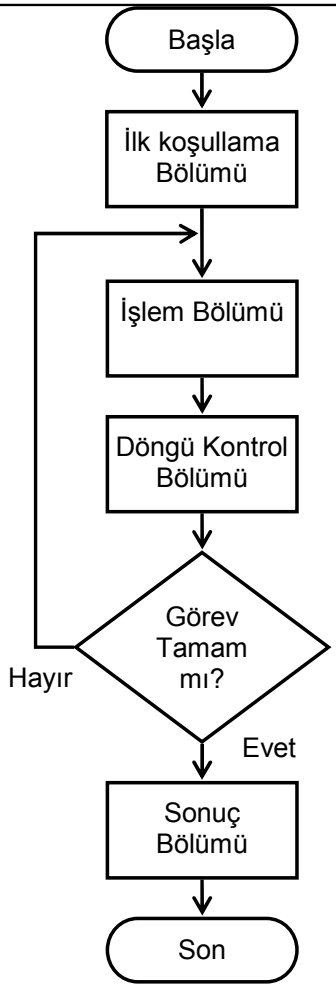
Şekil 14-2 Karar verme, tekrarlı işlemler gerektiren problemlerin çözümü için akış diyagramı



Şekil 14-3 Karar verme, tekrarlı işlemler gerektiren problemlerin çözümü için döngü çıkış kontrolünü önce yapan akış diyagramı



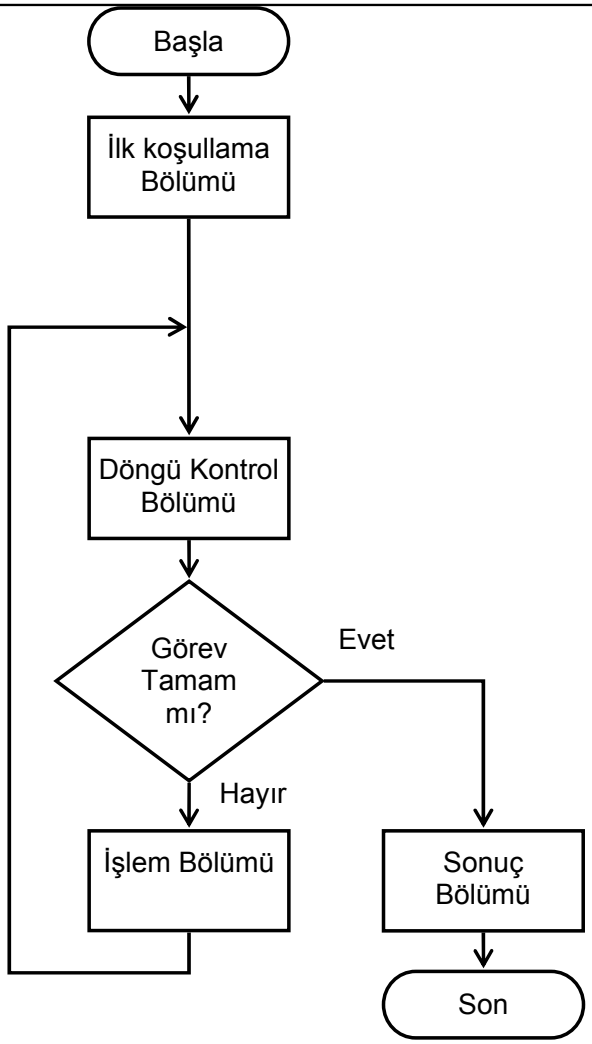
Şekil 14-4 Birden fazla karar verme, çok seçimli işlemler gerektiren problemlerin çözümü için akış diyagramı



Şekil 14-5 Program döngüleri için akış diyagramları

a) son işlemden sonra kontrol

b) son işlemden önce kontrol



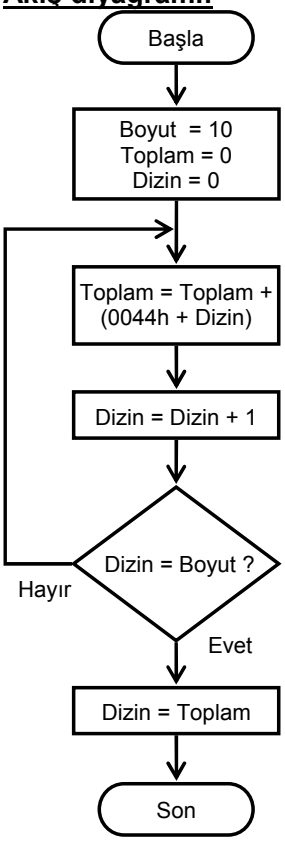
Örnek Pr. 14-8 Sayı dizisinin toplamının hesaplayan programın akış diyagramını çiziniz. 0040h bellek adresinde sayı dizisinin boyutunu, 0042h bellek adresinde sonucu ve 0044h bellek adresinden başlayarak sayı dizisini saklayınız. Ayrıca programın sonunda toplama sonucunu X yazmacında saklayınız.

Çözüm:
Programın akış diyagramı Şekil 14-6'da verilen şekilde tasarlanmıştır. Akış diyagramının sağında ise her bir bloğun gerçekleştirilmesi için 6800 mikroişlemcisi komut veya komut grubu gösterilmiştir.

Sayı dizisinin boyutu, BOYUT 0040H adresinde saklanmıştır. DIZI_ADR 0044H olarak verilmiştir.
Toplama sonucu 16-Bit alınırsa 42H:43H adresleri kullanılır.
TOPLAM =42H ve TOPLAM+1=43H olur.
İlk işlem (TOPLAM:TOPLAM+1)=(42H:43H)=0
A akümülatörüne, önceki toplama sonucu "LDAA TOPLAM+1" komutuyla yüklenir.
(X=0,X+44H=0044H) olarak, Akümülatörde toplanır. "TOPLAM+1" adresinde saklanır.
Her bir ara toplamda, 8-Bit toplamadan üreyen elde, 16-Bit toplama sonucunun yüksek ağırlıklı kısmı TOPLAM adresindeki değer artırılarak toplama sonucuna eklenir.
X dizin yazmacı artırılarak sonraki 8-Bit değerler
(X=1,X+44H=0045H),
(X=2,X+44H=0046H),
…
(X=9,X+44H=004DH) toplanır. İşlemler döngü içinde bu şekilde tekrarlanır.
Bu işlem X dizin yazmacı karşılaştırılarak BOYUT kadar tekrarlanır.
TOPLAM adresindeki 16-Bit sonuç X dizin yazmaca yüklenir ve program sonlandırılır.

(TOPLAM) (TOPLAM+1)
+ (X+44H)
(TOPLAM+1)
Eğer elde varsa
(TOPLAM)+1→(TOPLAM)

Akış diyagramı:



Program Karşılığı:

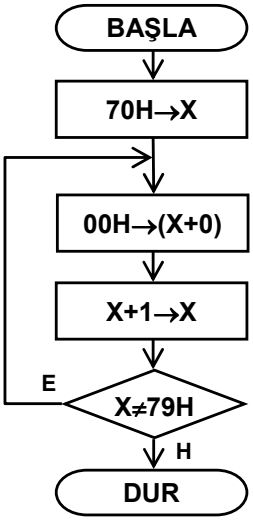
```
BOYUT    EQU   0040H
TOPLAM   EQU   0042H
DIZI_ADR EQU   0044H

         LDX   #10
         STX   BOYUT
         LDX   #0
         STX   TOPLAM

TOP1:    LDAA  TOPLAM+1
         ADDA  DIZI_ADR,X
         BCC   ELD_YOK
         INC   TOPLAM
ELD_YOK: STAA  TOPLAM+1

         INX

         CPX   BOYUT
         BNE   TOP1

         LDX   TOPLAM

         END
```

Şekil 14-6 Boyutu değiştirilebilen 10 elemanlı sayı dizisinin toplamı için akış diyagramı

Örnek Pr. 14-9 Bellekteki 70H ile 78H adresleri arasındaki alandaki baytların sıfır ile doldurulması (temizlenmesi).

Çözüm:

Akış diyagramı:



Program Karşılığı:

```
;programın başlangıcı
ORG     2000H
;başlangıç adres değerini X yazmacına yükle
LDX     #70H
;programdaki döngünün dallanma etiketi
L1:
;dizinlenmiş adresleme ile X+0 adresinin içeriğini sıfırla
CLR     0,X
;adres değerinin artırılması
INX
;son adres sıfırlandı mı?
CPX     #79H
BNE     L1
;programın sonu
END
```

```
                ; CLRMEM1.ASM
                ; 0070H→0078H adresleri arasındaki bellek alanındaki baytların
                ; sıfır ile doldurulması (temizlenmesi).
0000                    CPU   "6800.TBL"
0000                    HOF   "MOT8"
2000                    ORG   2000H     ; programın başlangıcı
2000 CE0070       (3)   LDX   #70H      ; başlangıç adres değerini X yazmacına yükle
2003 6F00   L1:   (7)   CLR   0,X       ; X+0 adresinin içeriğini sıfırla
2005 08           (4)   INX             ; adres değerinin artırılması
2006 8C0079       (3)   CPX   #79H      ; son adres sıfırlandı mı?
2009 26F8         (4)   BNE   L1        ; gelinmediyse L1'e git
0000                    END             ; programın sonu
```

Programın Analiz Tablosu:

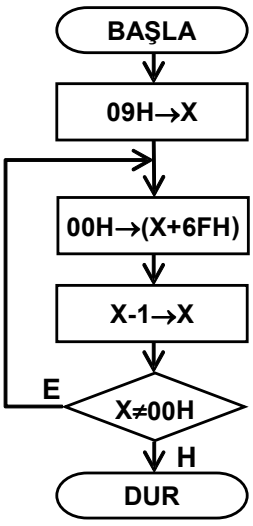
	PC	X	Z	(70H)	(71H)	(72H)	(73H)	(74H)	(75H)	(76H)	(77H)	(78H)
0	2000	?	?	?	?	?	?	?	?	?	?	?
1	2003	0070	0									
2	2009	0071	0	00								
3	2009	0072	0		00							
4	2009	0073	0			00						
5	2009	0074	0				00					
6	2009	0075	0					00				
7	2009	0076	0						00			
8	2009	0077	0							00		
9	2009	0078	0								00	
10	200B	0079	1									00

Programın toplam çalışma süresi = 3 + 9 * (7+4+3+4) = 3 + 9 * 18 = **165 sistem saati.**

Örnek Pr. 14-10 Bellekteki 70H ile 78H adresleri arasındaki baytların temizlenmesini, sıfır ile doldurulmasını sağlayan programın azalan adres sayacı ile tasarlanması.

Çözüm:

Akış diyagramı:



Program Karşılığı:

```
;programın başlangıcı
ORG     3000H
;sıfırla doldurulacak bayt adedini X yazmacına yükle
LDX     #9H
;programdaki döngünün dallanma etiketi
L1:
;dizinlenmiş adresleme ile X+6FH adresinin içeriğini sıfırla
CLR     6FH,X
;adres değerinin azaltılması
DEX
;son adres sıfırlandı mı?
BNE     L1
;programın sonu
END
```

Program Listesi Çıkış Dosyası:

```
                    ; CLRMEM2.ASM
                    ; 0078H→0070H adresleri arasındaki baytların
                    ; sıfır ile doldurulması (temizlenmesi).
0000                      CPU   "6800.TBL"
0000                      HOF   "MOT8"
3000                      ORG   3000H  ; programın başlangıcı
3000 CE0009         (3)   LDX   #9H    ; sıfırlanacak bayt sayısını X yazmacına yükle
3003 6F6F     L1:   (7)   CLR   6FH,X  ; X+6F adresinin içeriğini sıfırla
3005 09             (4)   DEX          ; bayt sayısını azalt
3006 26FB           (4)   BNE   L1     ; bayt sayısı sıfır değilse L1'e git
0000                      END          ; programın sonu
```

Programın Analiz Tablosu:

	PC	X	Z	(78H)	(77H)	(76H)	(75H)	(74H)	(73H)	(72H)	(71H)	(70H)
0	3000	?	?	?	?	?	?	?	?	?	?	?
1	3003	0009	0									
2	3006	0008	0	00								
3	3006	0007	0		00							
4	3006	0006	0			00						
5	3006	0005	0				00					
6	3006	0004	0					00				
7	3006	0003	0						00			
8	3006	0002	0							00		
9	3006	0001	0								00	
10	3008	0000	1									00

Programın toplam çalışma süresi = 3 + 9 * (7+4+4)
= 3 + 9 * 15 = **138 sistem saati.**

Örnek Pr. 14-11 Aşağıda verilen 6800 makine dili programın eksiklerini tamamlayınız ve her satırındaki komutun çalışma süresini ve açıklamasını yanına yazınız. PC program sayıcısı, X dizin yazmacı, A akümülatörü, durum yazmacının Z (sıfır) biti ve etkilenen bellek gözleri üzerinde analizini yapınız ve programın toplam çalışma süresini hesaplayınız.

Yazmaçların ilk durumu :

PC=E000H SP=006FH X=0002H A=25H B=FAH CCR=CCH

Bellek gözlerinin durumu (Bütün değerler onaltılık olarak verilmiştir!):

Adres	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0030	FC	FD	FE	FF	01	02	03	04	05	06	07	08	09	10	11	12
0040	13	14	15	16	17	18	76	91	A1	87	0B	17	43	B7	55	E8

```
E000                ORG   0E000H
E000 ____           LDX   #30H
E003 ____           STX   40H
E005 ____           LDX   #38H
E008 ____           STX   42H
E00A ____     L1:   LDX   40H
E00C ____           LDAA  0,X
E00E ____           INX
E00F ____           STX   40H
E011 ____           LDX   42H
E013 ____           STAA  0,X
E015 ____           INX
E016 ____           STX   42H
E018 ____           CPX   #39H
E01B ____           BNE   L1
E01D ____           NOP
0000                END
```

Çözüm: Programın tamamı:

```
E000                ORG 0E000h    ; programın başlangıç adresi 0E000h
E000 CE0030         LDX   #30h ;(3) X dizin yazmacına 30h değerini yükle.
E003 DF40           STX   40h  ;(5) X dizin yazmacını 40h:41h adresine yükle.
E005 CE0038         LDX   #38h ;(3) X dizin yazmacına 38h değerini yükle.
E008 DF42           STX   42h  ;(5) X dizin yazmacını 42h:43h adresine yükle.
E00A DE40     L1:   LDX   40h  ;(4) X dizin yazmacına 40h:41h adresindeki veriyi yükl
E00C A600           LDAA  0,X  ;(5) A akümülatörüne (X+0) adresindeki veriyi yükle.
E00E 08             INX        ;(4) X dizin yazmacını artır.
E00F DF40           STX   40h  ;(5) X dizin yazmacını 40h:41h adresine yükle.
E011 DE42           LDX   42h  ;(4) X dizin yazmacına 42h:43h adresindeki veriyi yükl
E013 A700           STAA  0,X  ;(6) A akümülatörünü (X+0) adresinde sakla.
E015 08             INX        ;(4) X dizin yazmacını artır.
E016 DF42           STX   42h  ;(5) X dizin yazmacını 42h:43h adresine yükle.
E018 8C0039         CPX   #39h ;(3) X dizin yazmacını 39h değeri ile karşılaştır.
E01B 26ED           BNE   L1   ;(4) sıfır değilse L1'e git
E01D 01             NOP        ;(2) sıfır ise programı bitir.
0000                END
```

Programın analizi:

	PC	X	A	Z	(40)	(41)	(42)	(43)	(38)
0	E000	0002	25	1	13	14	15	16	05
1	E005	0030		0	00	30			
2	E00A	0038					00	38	
3	E00F	0030	FC						
4	E011	0031			00	31			
5	E015	0038							FC
6	E01E	0039		1			00	39	

Programın toplam çalışma süresi = (3+5+3+5)+ 1*(4+5+4+5+4+6+4+5+3+4)+2
= 16 + 1*44 + 2 = 62 sistem saati.