

14. MİKROİŞLEMCİ PROGRAMLAMA TEKNİKLERİ

14.1. Programlamaya Giriş

Örnek Pr. 14-1 0040h bellek adresindeki 8-bit veriyi 0041h adresine transfer eden programı yazınız.

Çözüm:

0040H adresindeki veriyi A yazmacına yükle	LDAA 0040H
A yazmacındaki veriyi 0041H adresine sakla	STAA 0041H

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 veya	STAA 0040H
0040H adresindeki veriyi sıfırla	CLR 0040H

14.2. Mikroişlemcilerin Gelişmiş Komutları

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

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

Çözüm:

Çevirici giriş kaynak dosyası:

```

org 0d019h ; programın başlangıç adresi
basla2:    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:

```

00028      D019          org 0d019h ; programın başlangıç adresi
00029      (2) D019 0C   basla2:    clc          ; Elde bayrağını sıfırla
00030      (2) D01A 86 32   ldaa #32h    ; A akümülatörüne 32H yükle
00031      (2) D01C 89 29   adca #29h    ; A aküm. Elde ile 29h topla
00032      (2) D01E 19     daa          ; sonucu BCD'ye dönüştür.
00033      0000          end          ; programın sonu

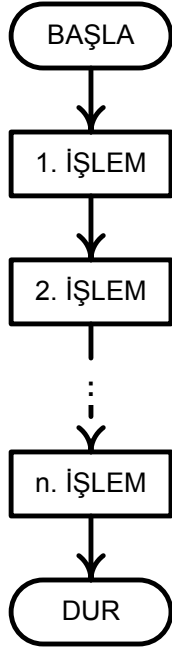
```

$32h + 29h = 5Bh \rightarrow DAA \rightarrow 61h$

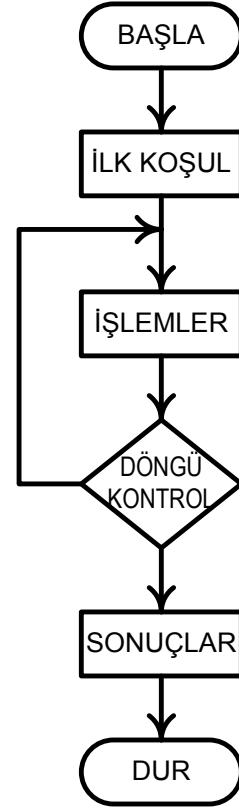
Komut kümesi kısmında toplama komutunun çalışma şekli tablosu ile beraber düşünülürse normal toplama işleminde sonuç ikili sayı olarak 5Bh olur. DAA komutu çalıştırıldığında değer 61h olarak bulunur. Bu değer ondalık olarak istenen 32+29 toplama sonucu olan 61 değeridir.

14.3. Programlama İçin Akış Diyagramı Yöntemi

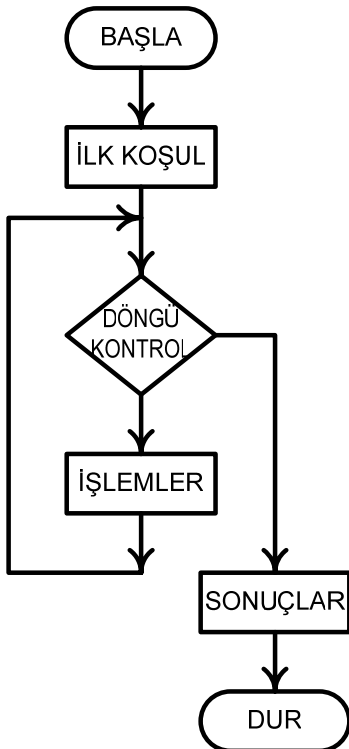
Program tasarımlarında çok karşılaşılan problemlerin genel olarak çözüm yöntemini veren akış diyagramı yapıları vardır.



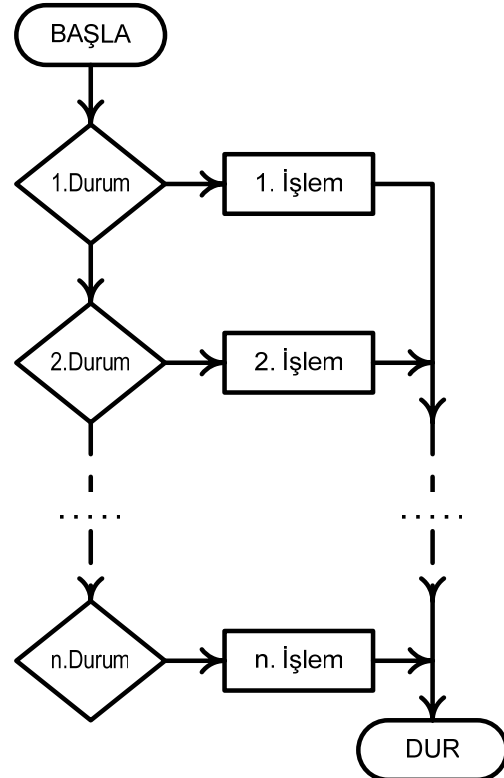
Şekil 14-1 Sıralı yapılan basit işlemler için akış diyagramı



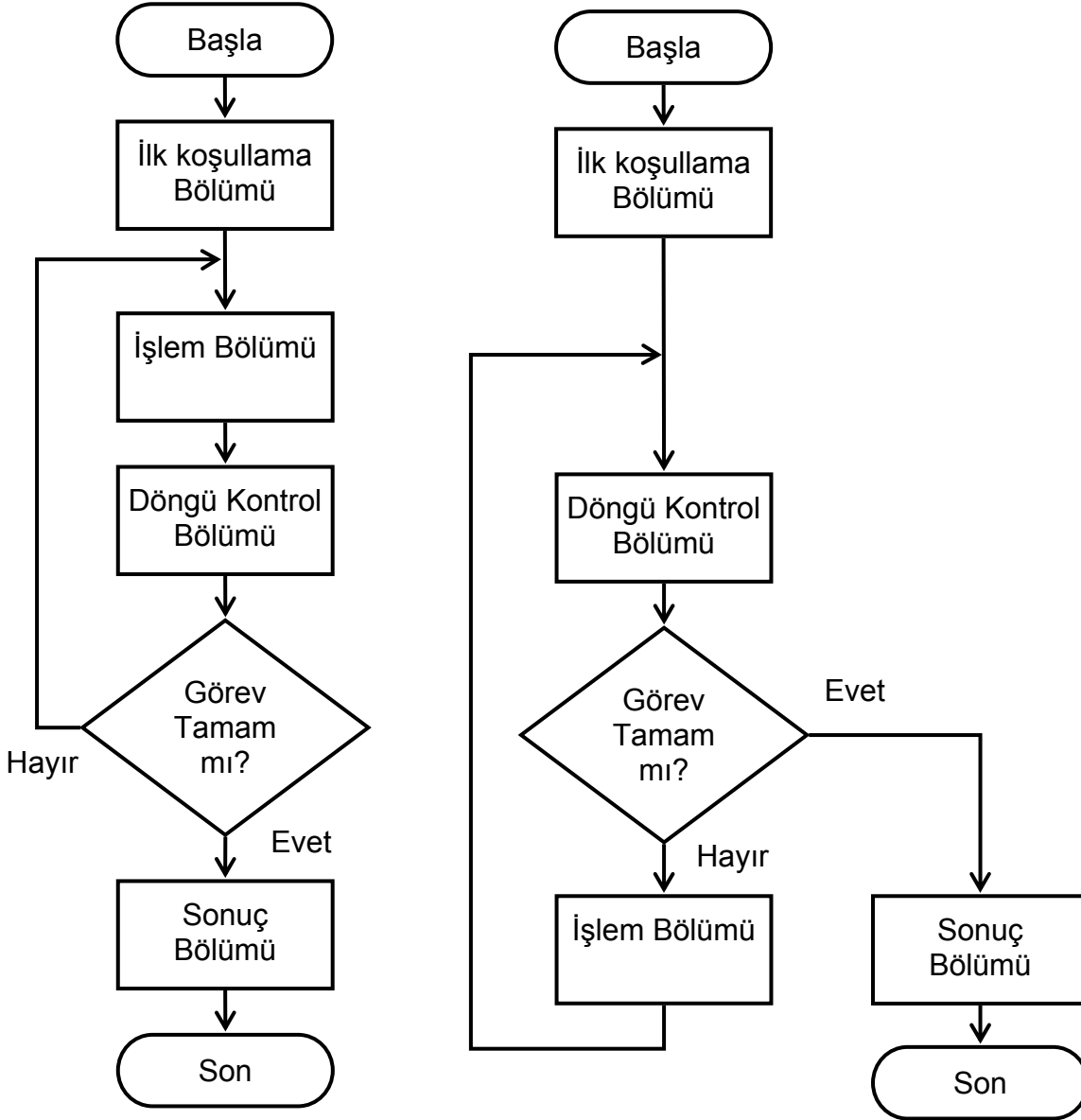
Şekil 14-2 Karar verme, tekrarlı işlemler gerektiren problemlerin çözümü için akış diyagramı



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



a) son işlemden sonra kontrol

b) son işlemden önce kontrol

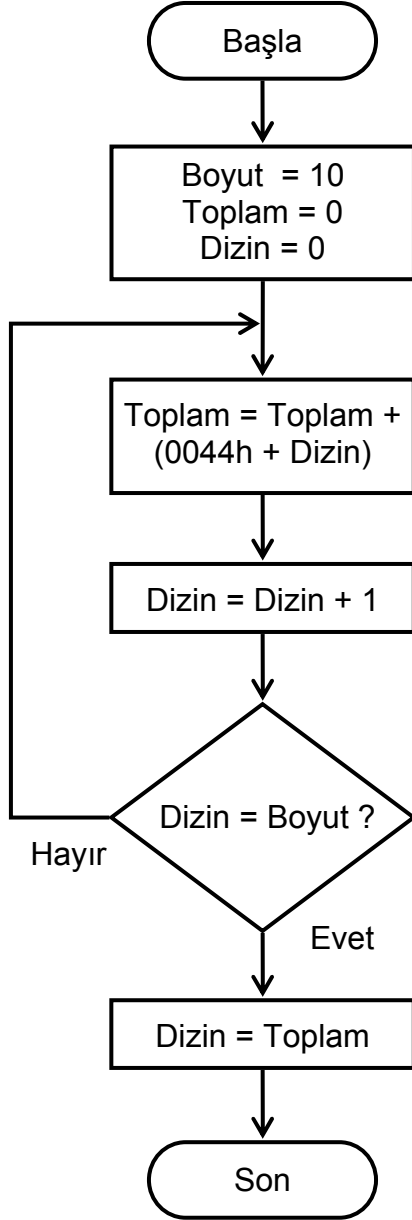
Şekil 14-5 Program döngüleri için akış diyagramları

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.

Akış diyagramı:



Program Karşılığı:

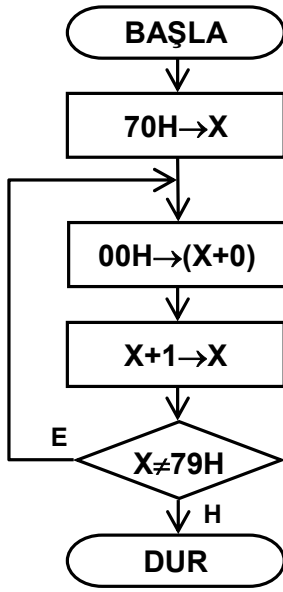
BOYUT	EQU	0040H	
TOPLAM	EQU	0042H	
DIZI_ADR	EQU	0044H	
	LDX	#10	
	STAA	BOYUT	
	LDX	#0	
	STX	TOPLAM	
TOP1:	LDAA	TOPLAM+1	
	ADDA	DIZI_ADR,X	
	BCC	ELD_YOK	
	INC	TOPLAM	
ELD_YOK:	STAA	TOPLAM+1	
	INX		
	CPX	BOYUT	
	BNE	TOP1	
	LDX	TOPLAM	
	END		

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
  
```

Başlangıç adres değeri X yazmacına yüklenir. Sonra bir döngü kurularak X değişkeni dizinlenmiş adresleme ile beraber sıfırlama komutuyla istenen adres bölgesi sıfırlanır. Bunun için X dizin yazmacının değeri bir sonraki adres içeriğinin sıfırlanması için döngü içinde artırılır. Son adrese gelinip gelinmediği X yazmacındaki adres değeri karşılaştırılarak belirlenir. Değer farklıysa işleme devam edilir. İşlemin bitmesi için X yazmacının değerinin 79h olması gerekir.

Program Analiz Tablosu:

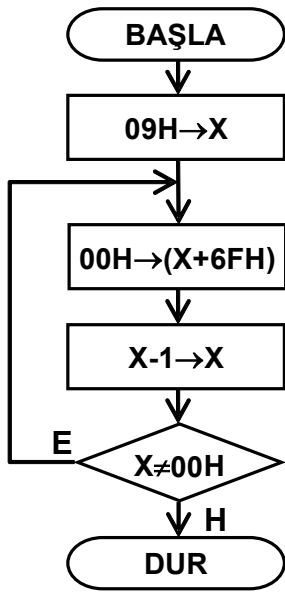
	X	Z	(70h)	(71h)	(72h)	(73h)	(74h)	(75h)	(76h)	(77h)	(78h)
0	?	?	?	?	?	?	?	?	?	?	?
1	0070	0	00								
2	0071	0		00							
3	0072	0			00						
4	0073	0				00					
:	:	:					:	:	:	:	
9	0078	0									00
10	0079	1									

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
  
```

Program Analiz Tablosu:

	X	Z	(78h)	(77h)	(76h)	(75h)	(74h)	(73h)	(72h)	(71h)	(70h)
0	?	?	?	?	?	?	?	?	?	?	?
1	0009	0									
2	0008	0	00								
3	0007	0		00							
4	0006	0			00						
5	0005	0				00					
6	0004	0					00				
7	0003	0						00			
8	0002	0							00		
9	0001	0								00	
10	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 açıklamasını yanına yazınız. A akümülatörü, X dizin yazmacı, 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 ilk durumu (bütün değerler hex olarak verilmiştir.) : \longrightarrow

```

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

```

```

(0030)= FC
(0031)= FD
(0032)= FE
(0033)= FF
(0034)= 01
(0035)= 02
(0036)= 03
(0037)= 04
(0038)= 05
(0039)= 06
(003A)= 07
(003B)= 08
(003C)= 09
(003D)= 10
(003E)= 11
(003F)= 12
(0040)= 13
(0041)= 14
(0042)= 15
(0043)= 16
(0044)= 17
(0045)= 18

```

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:

	A	X	Z	(40)	(41)	(42)	(43)	(38)
0	25	0002	1	13	14	15	16	05
1		0030	0	00	30			
2		0038				00	38	
3		0030						
4	FC	0031		00	31			
5		0038						FC
6		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.