BÖLÜM 5. KOMUT SETİ MİMARİSİ

- **Aritmetik Mantık Ünitesi (ALU)**
  - Aritmetik İşlemler Kısmı
  - Mantıksal İşlemler Kısmı
  - Çarpma Kısmı
  - Bölme Kısmı

- **Temel Bilgisayar Sistemimiz için Örnek Bir Uygulama**

Aritmetik Mantık Ünitesi (ALU)

Bir mikroişlemcide bulunan ALU birimi, bütün aritmetik ve mantıksal işlemlerin yapıldığı yerdir. Temel bilgisayar sistemimizde tasarlanan ALU, çarpma ve giriş/çıkış işlemleri haricinde, 16 bitlik veriler üzerinde işlem yapabilecek şekilde tasarlanmıştır. Kullanılan kaydediciler 16 bitlik olduğundan, çarpma işleminde iki tane 8 bitlik sayı kullanılmıştır.

Tasarlanan ALU dört bölümden oluşmaktadır;

1. Çarpma ve bölme işlemleri hariç, diğer aritmetik işlemlerin yapıldığı bölüm
2. Mantıksal işlemlerin yapıldığı bölüm
3. Çarpma işleminin yapıldığı bölüm
4. Bölme işleminin yapıldığı bölüm

Aritmetik İşlemler Kısmı

Bu kısımda 16 adet tam toplayıcı kullanılmıştır. Tam toplayıcının tasarım adımları aşağıda gösterilmiştir.

Tam Toplayıcı

Tam toplayıcı, 2 adet 1 bitlik veriyi ve elde bitini (c_i) giriş olarak alır, toplam ve elde çıkışı (c_0) üretir.

Girişler			Çıkışlar	
AC	**DR**	**c_i**	**Toplam**	**c_o**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

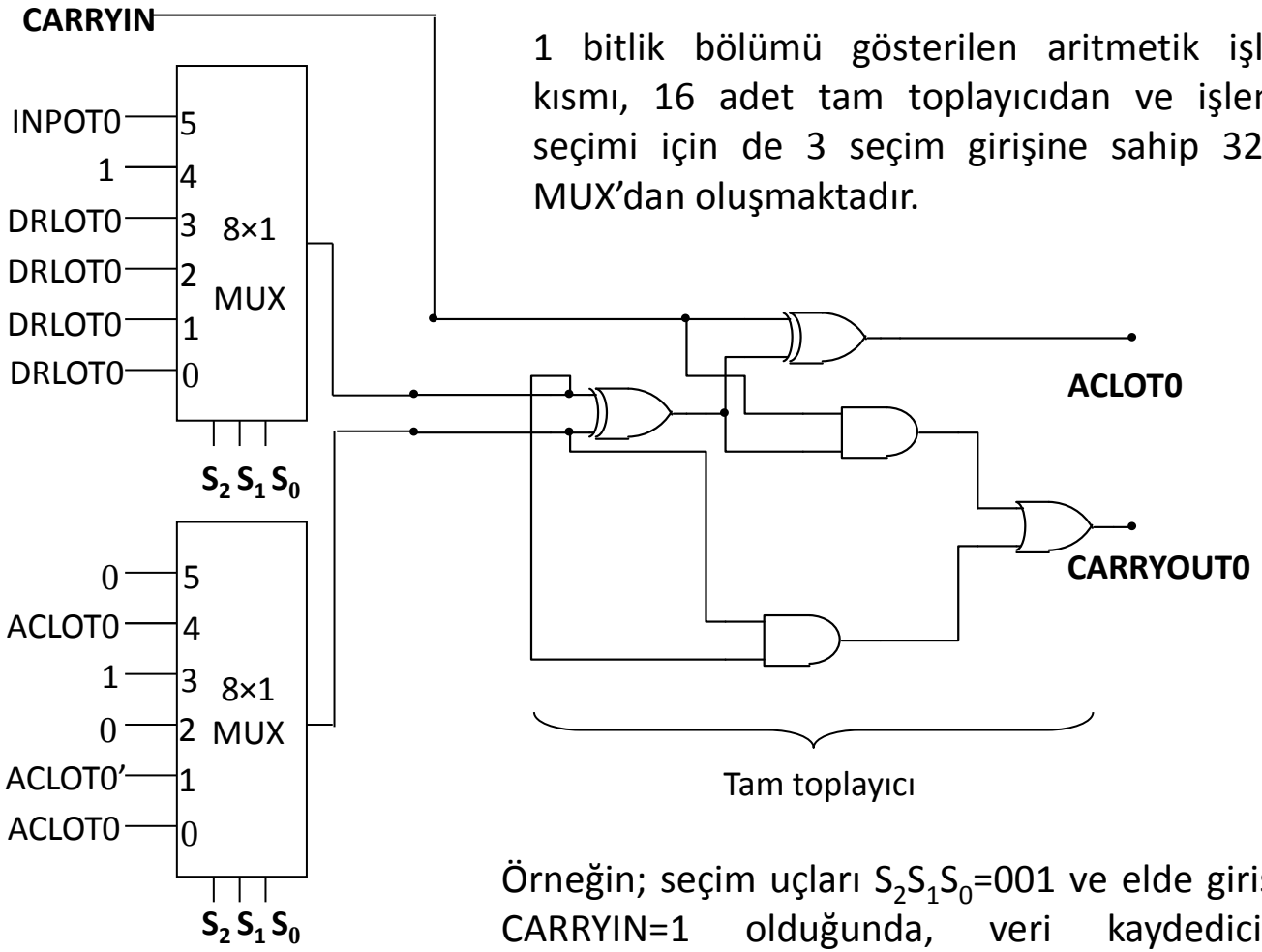
$$\textbf{Toplam} = AC'.DR'.c_i + AC'.DR.c_i' + AC.DR'.c_i' + AC.DR.c_i$$
$$= c_i.(AC'.DR' + AC.DR) + c_i'.(AC'.DR + AC.DR')$$
$$= c_i.(AC \oplus DR)' + c_i'.(AC \oplus DR)$$
$$= AC \oplus DR \oplus c_i$$

$$\mathbf{c_o} = AC'.DR.c_i + AC.DR'.c_i + AC.DR.c_i' + AC.DR.c_i$$
$$= c_i.(AC \oplus DR) + AC.DR$$

ALU'da Gerçekleştirilen Aritmetik İşlemler

Seçim			Girişler			Çıkış	Açıklaması
S_2	S_1	S_0	CARRYIN	Y_0	Y_1	Q	
0	0	0	0	DR	AC	AC←DR+AC	Toplama
0	0	0	1	DR	AC	AC←DR+AC+1	Elde ile toplama
0	0	1	0	DR	AC'	AC←DR+AC'	Borç ile çıkarma
0	0	1	1	DR	AC'	AC←DR+AC'+1	Çıkarma
0	1	0	0	DR	0..0	AC←DR	DR'in aktarımı
0	1	0	1	DR	0..0	AC←DR+1	DR'nin 1 fazlasını aktarma
0	1	1	0	DR	1..1	AC←DR-1	DR'nin 1 eksiğini aktarma
0	1	1	1	DR	1..1	AC←DR	DR'nin aktarımı
1	0	0	0	1..1	AC	AC←AC-1	AC'yi 1 azaltma
1	0	0	1	1..1	AC	AC←AC	AC'nin aktarımı
1	0	1	0	INPR	0..0	AC←INPR	Giriş Kaydedicisinin aktarımı
1	0	1	1	INPR	0..0	AC←INPR+1	Giriş Kaydedicisinin 1 fazlasının aktarımı

Aritmetik İşlemler Kısmının 1 Bitlik Bölümünün Gerçekleştirimi



1 bitlik bölümü gösterilen aritmetik işlemler kısmı, 16 adet tam toplayıcıdan ve işlemlerin seçimi için de 3 seçim girişine sahip 32 adet MUX'dan oluşmaktadır.

Örneğin; seçim uçları $S_2S_1S_0$=001 ve elde girişi de CARRYIN=1 olduğunda, veri kaydedicisinin içeriğinden akümülatörün içeriği çıkartılır.

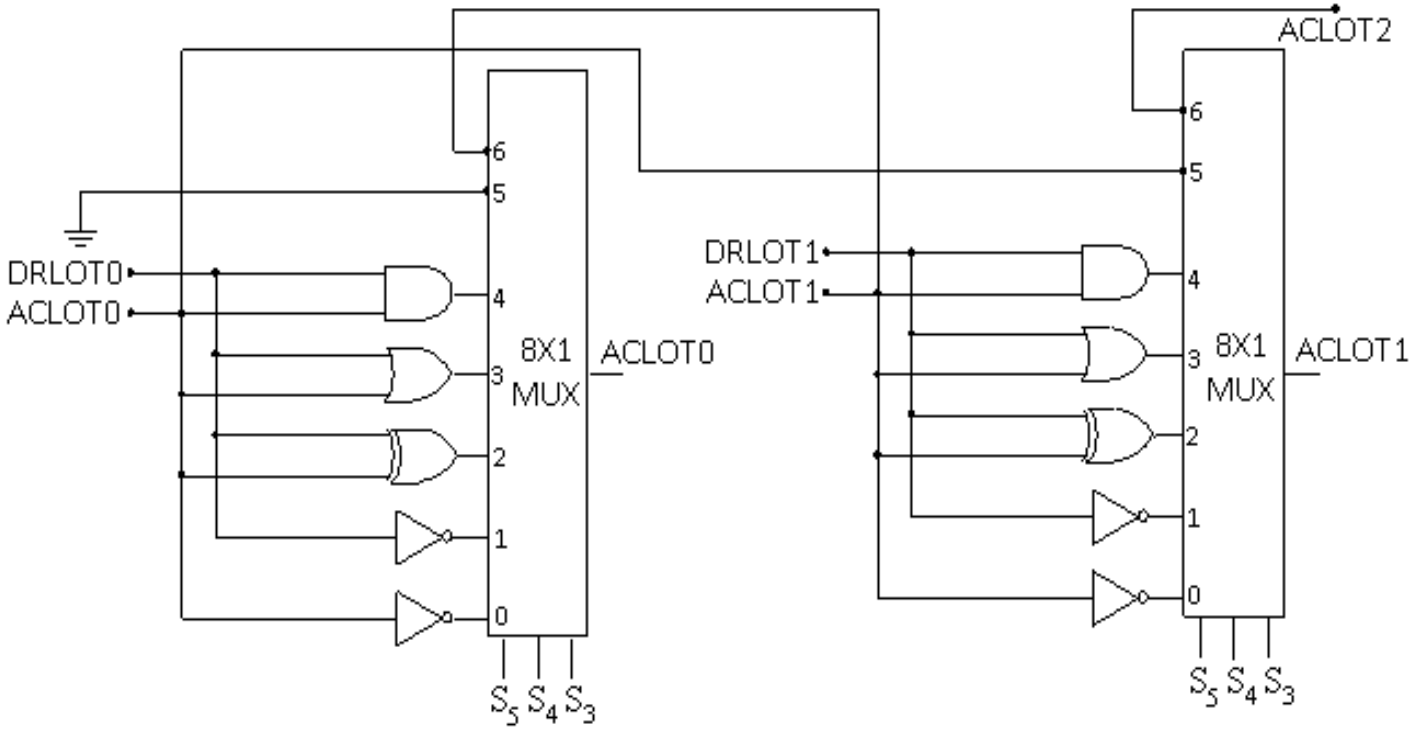
Mantıksal İşlemler Kısmı

Bu kısım, 7 farklı işlemi yerine getirebilmektedir. Aşağıdaki tabloda yerine getirilen işlemler verilmiştir;

S_5	S_4	S_3	Q	Açıklaması
0	0	0	AC←AC’	AC’nin tümleyeni
0	0	1	AC←DR’	DR’nin tümleyeni
0	1	0	AC←AC⊕DR	Lojik “XOR” İşlemi
0	1	1	AC←AC∨DR	Lojik “VEYA” İşlemi
1	0	0	AC←AC DR	Lojik “VE” İşlemi
1	0	1	AC←shl(AC)	AC’yi sola kaydırma
1	1	0	AC←shr(AC)	AC’yi sağa kaydırma

Mantıksal İşlemler Kısmının 2 Bitlik Bölümünün Gerçekleştirimi

ALU'nun bu bölümünde yapılan işlemlerin yerine getirilebilmesi için 16 adet 8×1'lik MUX kullanılmıştır.



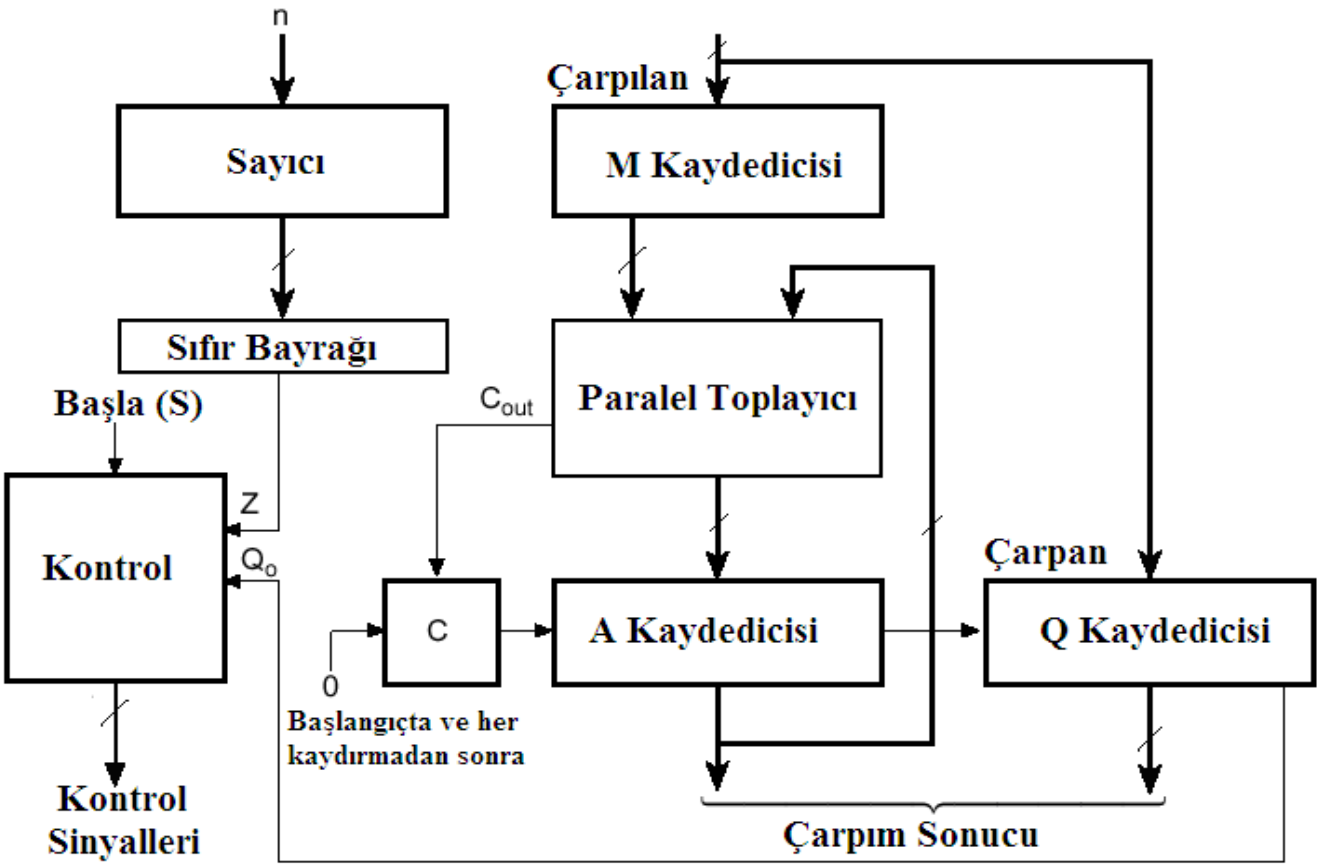
Örneğin; seçim uçları $S_5S_4S_3$ = 011 olduğunda, veri kaydedicisindeki veri ile akümülatördeki veri lojik "VEYA" işlemine tabi tutularak akümülatöre aktarılmaktadır.

Çarpma Kısmı

- İki adet ikili sayının çarpımı, kalem kâğıtla yapılacak olursa bir dizi ardışıl kaydırma ve toplama işlemleriyle gerçekleştirilir. İşlem, çarpanın en önemsiz bitinden başlayarak sırayla bütün bitlerine bakılarak yapılır. Eğer çarpanın biti 1 ise çarpılan aşağıya aynen alınır. Aksi taktirde aşağıya sıfırlar yazılır. Daha sonra da sola bir bit kaydırılır. Kısmi çarpımlar toplanarak, çarpım elde edilmiş olur.

- Donanımsal olarak çarpma işlemi yapılacağı zaman, tüm kısmi çarpımları kaydedicilere yazmak yerine, kısmi çarpımların toplamı bir kaydediciye yazılır. Daha sonra da kısmi toplam sağa kaydırılır. Çarpanın biti 0 ise sıfırla toplamaya gerek yoktur sadece sağa kaydırma işlemi yapılır.

Çarpma İşleminin Prensip Şeması



Örnek: 1001 × 0011 İşlemi için Algoritmanın Gerçekleştirilmesi

- M kaydedicisine çarpılanın, Q kaydedicisine çarpanın değeri konulur.
- A kaydedicisi ve elde biti (c) sıfırlanır, sayıcıya 4 değeri atanır.
- Çarpan'ın Q_0 bitine bakılarak kısmi çarpımlar toplamı elde edilir.
- İşlemin sonucunun yüksek anlamlı kısmı A, düşük anlamlı kısmı Q kaydedicisindedir.

Başlangıç Değerleri	**C(0)**	**A (0000)**	**Q (0011) Q0**	**İşlem**
1.Adım	0	1001	0011	Toplama
	0	0100	1001	Kaydırma
2.Adım	0	1101	1001	Toplama
	0	0110	1100	Kaydırma
3.Adım	0	0011	0110	Kaydırma
4.Adım	0	0001	1011	Kaydırma

M = 1001

Çarpım=A Q = 0001 1011 = $1B_h$ = 27

Bölme Kısmı

- İki adet ikili sayının bölme işlemi, karşılaştırma, kaydırma ve çıkarma işlemleri gerektirir. İkili bölme, ondalık bölmeden daha basittir; işleme giren sayıların basamaklarında sadece 0 veya 1 vardır. Ayrıca bölenin, bölünen veya kalan içinde kaç defa olduğunu bulmak kolaydır.

- Sayısal bir bilgisayarda bölme işlemini donanımsal olarak gerçekleştirmek istersek, işlemler biraz değişecektir. Bölünen veya kısmi kalan bir sola kaydırılır. Bölünende bölen varsa gerekli çıkarma işlemi, bölünene bölenin 2'ye göre tümleyenin eklenmesiyle yapılır.

Bölme Kısmının Algoritması

Bölen M ve bölünen Q kaydedicilerine konulur.

Sıra sayıcıya başlangıçta bölünenin bit sayısı atanır.

Bölünen, sola 1 kaydırılarak bölünenin en anlamlı kısmı A kaydedicisine aktarılır (Başlangıçta A=0'dır) ve bölenin 2'ye göre tümleyeni A'ya eklenir. Bu işlemin sonucu pozitifse bölüm hanesine 1 yazılır (Q'nun en düşük anlamlı kısmına 1 konulur) ve sıra sayıcı 1 azaltılır. Şayet kalan negatifse bölünende bölen yok demektir (Q'nun en düşük anlamlı kısmına 0 konulur). Bölünen 1 sola kaydırılır ve kısmi kalana bölen eklenerek eski haline getirilir. Yine bölünen de bölenin olup olmadığı araştırılır. Bu işlemler, sayıcı sıfırlanana kadar devam eder.

Örnek: 010110/000011 İşlemi için Algoritmanın Gerçekleştirilmesi

- M kaydedicisine bölenin, Q kaydedicisine bölünenin değeri konulur.

- A kaydedicisi, bölme işlemi sonucu oluşan kalanı tutar ve başlangıçta sıfır değeri atanır, sayıcıya da 6 değeri yüklenir.

- Q kaydedicisi, A kaydedicisine doğru 1 sola kaydırılır, A'da bölen varsa Q'nun 0.bitine 1, yoksa 0 yüklenir. Bu işlemler 6 kere yapılır.

Başlangıç Değerleri	A (000000)	Q (010110) Q0	İşlem
1.adım	000000	10110**0**	Kaydırma
2.Adım	000001	01100**0**	Kaydırma
3.adım	000010	11000**0**	Kaydırma
4.adım	000101	100000	Kaydırma
	000010	10000**1**	Çıkartma
5.adım	000101	000010	Kaydırma
	000010	00001**1**	Çıkartma
6.adım	000100	000110	Kaydırma
	000001	00011**1**	Çıkartma

M= 000011

Bölüm = Q = 000111_2 = 7
Kalan = A = 000001_2=1

Örnek Uygulama

Üst seviyeli bir dil ile yazılan aşağıdaki programın, temel bilgisayar sistemimize göre assembly ve makine dili karşılıklarının bulunması.

```
a = 5;
for i = 1 to 10
  {
    a = a+i;
  }
```

Değişkenler 0030h adresinden itibaren belleğe konulmuştur;

0030h-0031h: a değişkeni

0032h-0033h: i değişkeni

0034h-0035h: i değişkeninin maksimum değeri

Programın Assembly Dili Karşılığının Bulunması

0000h: LDA #0005h / Akümülatöre, a değişkenine konulacak olan decimal 5 değeri atanıyor.
0003h: STA 0030h / a değişkeni için, belleğin 0030h ile 0031h adresleri tahsis edilmiştir.
0006h: LDA #0001h / Akümülatöre, i değişkenine konulacak olan decimal 1 değeri atanıyor.
0009h: STA 0032h / i değişkeni için, belleğin 0032h ile 0033h adresleri tahsis edilmiştir.
000Ch: LDA #000Ah / Akümülatöre i değişkeninin maksimum değeri olan decimal 10 atanıyor.
000Fh: STA 0034h / i'nin maksimum değeri için, belleğin 0034h ile 0035h adresleri tahsis edilmiştir.
0012h: LDA 0030h / a değişkeni bellekten alınıp akümülatöre konuluyor.
0015h: ADD 0032h / a değişkeni ile i değişkeni toplanıyor.
0018h: STA 0030h / toplam sonucu, a değişkeninin tutulduğu bellek bölgesine kaydediliyor
001Bh: LDA 0032h / i değişkeni bellekten alınıyor.
001Eh: SUB 0034h / i'nin maksimum değerinden (decimal 10) i çıkartılıyor
0021h: BZR ~09h / Eğer i değişkeni decimal 10 değerine ulaşmışsa, 002Ch adresine dallanılıyor
0023h: LDA 0032h / i değişkeni bellekten alınıyor
0026h: INCR / i değişkeninin değerinin aktarıldığı akümülatör, 1 artırılıyor
0027h: STA 0032h / i değişkeni belleğe kaydediliyor
002Ah: BRA ~E6h / 0012h adresine dallanılıyor
002Ch: LDA 0030h / toplam sonucu akümülatöre aktarılıyor (Sonucu görebilmek için)
002Fh: HLT / Program sonlandırılıyor.

Programın Makine Dili Karşılığının Bulunması

BELLEK ADRESLERİ	MAKİNE DİLİ	ASSEMBLY DİLİ
0000	1A	LDA #0005h
0001	00	
0002	05	
0003	A0	STA 0030h
0004	00	
0005	30	
0006	1A	LDA #0001h
0007	00	
0008	01	
0009	A0	STA 0032h
000A	00	
000B	32	
000C	1A	LDA #000Ah
000D	00	
000E	0A	
000F	A0	STA 0034h
0010	00	
0011	34	
0012	2A	LDA 0030h
0013	00	
0014	30	
0015	20	ADD 0032h
0016	00	
0017	32	
0018	A0	STA 0030h
0019	00	
001A	30	
001B	2A	LDA 0032h
001C	00	
001D	32	
001E	2E	SUB 0034h
001F	00	
0020	34	
0021	53	BZR ~09h
0022	09	
0023	2A	LDA 0032h
0024	00	
0025	32	
0026	03	INCR
0027	A0	STA 0032h
0028	00	
0029	32	
002A	50	BRA ~E6h
002B	E6	
002C	2A	LDA 0030h
002D	00	
002E	30	
002F	0E	**HLT**

Örnek 2

```
k=5;
m=10;
s=0;
if (k>=m)
    s=k+m;
else
    s=k-m;
```

Değişkenler DE00h adresinden itibaren belleğe konulmuştur;

DE00h-DE01h: k değişkeni
DE02h-DE03h: m değişkeni
DE04h-DE05h: s değişkeni

Örnek 2

```
0000h:LDA   #0005H;
0003h:STA   DE00H;
0006h:LDA    #000AH;
0009h:STA    DE02H;
000Ch:LDA    #0000H;
000Fh:STA   DE04H;
0012h:LDA   DE02H;
0015h:SUB    DE00H;
0018h:BCS   ~0AH;
001Ah:LDA  DE02H;
001Dh:SUB   DE00H;
0020h:STA   DE04H;
0023h:HLT
0024h:LDA   DE00H;
0027h:ADD   DE02H
002Ah:STA   DE04H
002Dh: HLT
```

```
k=5;
m=10;
s=0;
if (k>=m)
      s=k+m;
else
      s=k-m;
```

DE00h-DE01h: k değişkeni
DE02h-DE03h: m değişkeni
DE04h-DE05h: s değişkeni

Örnek 2

BELLEK ADRESLERİ	MAKİNE DİLİ	ASSEMBLY DİLİ
0000	1A	LDA #0005h
0001	00	
0002	05	
0003	A0	STA DE00h
0004	DE	
0005	00	
0006	1A	LDA #000Ah
0007	00	
0008	0A	
0009	A0	STA DE02h
000A	DE	
000B	02	
000C	1A	LDA #0000h
000D	00	
000E	00	
000F	A0	STA DE04h
0010	DE	
0011	04	
0012	2A	LDA DE02h
0013	DE	
0014	02	
0015	2E	SUB DE00h
0016	DE	
0017	00	
0018	52	BCS ~0AH
0019	0A	
001A	2A	LDA DE02h
001B	DE	
001C	02	
001D	2E	SUB DE00h
001E	DE	
001F	00	
0020	A0	STA DE04H
0021	DE	
0022	04	
0023	0E	HLT
0024	2A	LDA DE00H
0025	DE	
0026	00	
0027	20	ADD DE02H
0028	DE	
0029	02	
002A	A0	STA DE04H
002B	DE	
002C	04	
002D	0E	**HLT**