Mikrobilgisayar Mimarisi ve Programlama

8085 Adresleme ve Komutlar

Assembly Formatı

Assembly komut satırı biçimi

Etiket	İşlem Kodu Komut (OpCode)	İşlenen (Operand)	Açıklama
Basla:	MVI	A, 20H	; A kaydedicisine 20H değerini yükle

Açıklama Satırı

- Assembly dili (;) ile başlayan satırları açıklama satırı olarak kabul eder.
- Bu satırları yorumlamaz ve makine kodu üretmez.
- Yazılan uygulamanın anlaşılırlığını arttırır

Assembly Formatı

- **Etiket:**
  - Komut satırının ilk bilgisidir ve sembolik isimlerden oluşur.
  - Program içerisindeki belirli işlevlerin gerçekleştiği bölümlerin başlangıcını göstermek amacı ile kullanılır.
  - Etiket, program alanının kolaylıkla bulunmasını ve hatırlanmasını sağlar.
  - Etiketlere isim verirken aşağıdaki semboller kullanılabilir.
    - Alfabetik karakterler : A'dan Z'ye ve a'dan z'ye (assembler büyük-küçük harf ayrımı yapmaz)
    - Sayılar : 0'dan 9'a kadar sayısal değerler (ilk karakter olamaz)
    - Özel karakterler : soru işareti (?), alt çizgi (_), Dolar ($), at (@)
    - Etiket ismi olarak mikroişlemci komut setinde tanımlı olan bir komut ismi verilemez. Yine etiket bir harf ile başlamak zorundadır.

Assembly Formatı

- **Komut:**
  - Mnemonik olarak ta adlandırılan, komut seti içerisinde mikroişlemcinin belirli bir işi yapmasını sağlayan tanımlanmış sembollerdir.
  - Komut alanına etiketten sonra 1 boşluk ya da tab ile girilir.

- **Operand:**
  - Bu alan, işlemciye işlenecek veriyi ya da verinin nerede olduğunu gösterir.
  - Tek başına bir anlam ifade etmez.
  - Bu alan **hedef, kaynak** şeklinde tanımlanır.
  - Burada hedef ve kaynak alanı birbirinden '**,**' ile ayrılır. kullanılabilir.

Veri Formatları

- Veriler 4 farklı formatta gösterilebilmektedir:
  - ASCII
  - Hexadesimal
  - İşaretli tamsayı (Signed Integer)
  - İşaretsiz tamsayı (Unsigned Integer).

- Örneğin akünün taşıdığı değer: 0100 0001.
  - İkili formatta işaretsiz bir tamsayıdır.
  - Onlu sayı sisteminde 65'e eşittir.
  - 16'lı sayı sisteminde ise 41 değerine eşittir.
  - ASCII karşılığı ise A harfidir.

Assembly Dilinde Sayı Sistemlerinin Kullanımı

Ön Takı	Son Takı	Anlamı	Örnek
(Boşluk)	D	Onlu sayı (decimal)*	55 – 55D
%	B	İkili sayı (binary)	%01010101 – 01010101B
@	O	Sekizli sayı (octal)	@33 – 33O
$	H	Onaltılık sayı (hexadecimal)	$FB – FBH
‘’		ASCII	‘A’

Yönergeler

ORG

- Kod bellek içerisinde programın başlangıç adresini belirtmek için kullanılan adres konumlandırma talimatıdır.
- ORG 'Adres' şeklinde kullanılır.
- Bir program içerisinde birden fazla ORG komutu kullanılabilir.

Talimat	Açıklama
`ORG 0000h`	`;program 0000h adresinden başlasın PC=0000`
`ORG 0030h`	`;program 0030h adresinden başlasın PC=0030`

Yönergeler

EQU

- EQU (Equal = eşittir) bir sayısal değerin istenilen sembol adına atanması işlemini gerçekleştirir

- Bu tanımlama program içerisinde bir ifadenin ya da değerin çok fazla tekrar edildiğinde programın anlaşılırlığını arttırmak için kullanılır.

İsim	Talimat	Değer	Açıklama
Pi	EQU	3.14	;sabit değer tanımlama
Bilgi	EQU	55h	;55h adresindeki veriyi bilgi değişkenine ata

Yönergeler

END

- Programın bitiğini gösteren talimattır.

DB (Define Bayt)

- Kod bellek içerisinde sayı ve kelime (string) dizilerinin tanımlanmasını sağlar.

İsim	DB	ifadeler	Açıklama
`Max_sayi:`	`DB`	`255`	`;tek bir değişken tanımlanması`
`Tablo:`	`DB`	`0, 5, 4, 3, -10`	`;dizi olarak tanımlama`
`Yaz:`	`DB`	`"8085 Ogreniyorum"`	`;string olarak tanımlama`

8085 Komutları

- 8085, 8-bitlik bir işlemci olduğundan 2^8 (256) adet komut barındırabilir.
  - Ancak 8085 sadece 246 farklı kombinasyon ile toplam 74 komuta sahiptir.
    - Çoğu komutun birden fazla formatı bulunmaktadır.

- Bu komutlar 5 farklı grupta toplanmaktadır.
  - Veri Transfer Komutları
  - Aritmetik İşlem Komutları
  - Mantıksal Komutlar
  - Dallanma Komutları
  - Program Kontrol Komutları

Komut ve Veri Formatları

- Her bir komut 2 kısımdan oluşmaktadır.
  - İlk kısım gerçekleştirilecek işlemi veya görevi gösterir.
    - Bu kısım "opcode" (operation code) olarak adlandırılır.
  - 2. kısım üzerinde işlenecek veriyi ifade eder.
    - "operand" işlenen olarak adlandırılır.

İşlenen Türleri

- **İşlenenler farklı şekillerde kullanılabilir:**
  - Bazı komutlarda işlenen gözükmeyebilir. (imalı işlenen)
    - CMA (Akünün bitlerini tersle)
  - İşlenen 8-bitlik bir sayı olabilir (ivedi veri).
    - ADI 4FH (A = A + 4FH)
  - İşlenen dahili bir kaydedici olabilir (Kaydedici).
    - SUB B (A=A-B)
  - İşlenen 16-bitlik bir adres olabilir (Bellek adresi).
    - LDA 4000H (A=B[4000H])

Komut Boyutları

- İşlenen türlerine bağlı olarak komut uzunlukları farklı olabilir.
  - Tipik olarak tüm komutlar belekte 1-bayt yer kaplar.
  - İvedi veri veya bellek adresi taşıyan komutlar ise 1-baytan daha büyüktür.
    - İvedi taşıyan komutlar iki bayttır. (1 bayt opkod 1 bayt veri).
    - Bellek adresi içeren bir komut ise bellekte 3-bayt işgal eder. (1-bayt opkod 2-bayt ise 16-bitlik adres).

İvedi sayı işleyen bir komut

- İşlem: 8-bitlik bir sayının aküye kaydedilmesi.

  - MVI A, 32H
    - İşlem: MVI A
    - İşlenen: 32 sayısı
    - Makine kodu:

      0011 1110 3E 1. bayt.

      0011 0010 32 2. bayt.

Bellek adresi içeren bir komut

- İşlem : 2085 adresine git.
  - Komut: JMP 2085H
    - Opkod: JMP
    - İşlenen: 2085H
    - Makine kodu:

1100 0011	C3	1. bayt.
1000 0101	85	2. bayt
0010 0000	20	3. bayt

Adresleme Modları

- Bir komutun işlenmesi için gerekli verilerin bir bellek bölgesinden alınması, bir bellek bölgesine konulması, bellek – kaydedici veya kaydedici – kaydedici arasında değiştirilmesi, vb. işlemlerde farklı bellek erişim yöntemleri kullanılır.

- İşlemciler verilere farklı yollar ile erişebilirler. Buna "adresleme modları" denilir.

- 8085 işlemcisi 4 adet adresleme moduna sahiptir:
  - İmalı — CMA (Aküyü tersle)
  - İvedi — MVI B, 45H (B=45H)
  - Kaydedici — MOV B,C
  - Doğrudan — LDA 4000H (A=Bellek[4000H])
  - Dolaylı — LDAX B (A= Bellek[BC])

Adresleme Modları

- Adresleme mantığını açıklamak için günlük hayattan bir örnek verebiliriz.
  - Bir lokantada yemek sırasında aşağıdaki isteklerde bulunabiliriz:
    - **Tuzu bana verebilir misiniz?** (Ne istendiği doğrudan belirtilmektedir: Kaydediciye veri kaydedilmesi)
    - **Sürahiyi verebilir misiniz.** (İçindeki bilinmesine ve amaç içindekini almak olmasına rağmen sürahi ön plana çıkarılmaktadır: Bir kaydediciden diğerine bilgi aktarımı)
    - **Haydi yiyelim. (**Ne yenileceği)bilinmekte ve işlem açıklanmaktadır: Akümlatörün içeriğinin tersini alma işlemi)
    - **12 nolu menüyü rica ediyorum.** (İstek yerine isteğin bulunduğu menü numarası belirtilmektedir: Bir bellek bölgesinden veri transferi).
    - **Hasan'ın siparişinin aynısını istiyorum.** (istenilen dolaylı olarak belirtilmektedir: Bir kaydedicinin veya kaydedici çiftinin içeriğinin işlem yapılacak bellek bölgesini belirtmesi.

Veri Transfer Komutları

- Bu komutlar kaynaktan hedefe verileri **KOPYALAR**.
  - MOV, MVI, LDA, STA, LXI, LDAX, IN, OUT
- Bu komutlar transfer ederler:
  - Sabit bir veriyi bir kaydediciye /kaydedici çiftine
    - MVI B,32H
    - MVI C,00H
    - LXI B,3200H
  - Kaydediciler arasındaki verileri.
    - MOV B,C
  - Bir bellek bölgesinden kaydediciye yada kaydediciden bir belleğe
    - LDA 2000H *;Bellekte 2000h adresindeki bilgiyi aküye kopyala*
    - STA 4000H *;Aküdeki bilgiyi belleğin 4000h adresine kopyala*
    - LDAX B *; BC'nin gösterdiği adresteki bilgiyi aküye kopyala*
    - MOV B,M *;HL'in gösterdiği adresteki bilgiyi B'ye kopyala (Dolaylı)*
  - Bir giriş/çıkış birimi ile akü arasındaki veriyi. (IN FFH, OUT 20H)

Aritmetik İşlem Komutları

- **TOPLAMA(ADD, ADI):** Akünün içeriği ile farklı şekilde toplama yapılabilir ve sonuç aküde tutulur.
  - Herhangi bir 8-bitlik sayı.
    - ADI 20H *; A=A+20H*
  - Bir kaydedicinin değerini.
    - ADD C *;A=A+C*
  - Bir bellek bölgesinin değeri.
    - ADD 0200H *; A=A +Bellek [0200H]*
    - ADD M *;A =A +bellek[HL]*

Aritmetik İşlem Komutları

- **<u>ÇIKARMA(SUB, SUI):</u>** Akünün içeriğinde ile farklı şekilde çıkarma yapılabilir ve sonuç aküde tutulur.
  - Herhangi bir 8-bitlik sayı.
    - SUI F0H *; A =A-F0H*
  - Bir kaydedicinin değerini.
    - SUB E *;A=A-E*
  - Bir bellek bölgesinin değeri.
    - SUB 0300H *; A=A - Bellek [0300H]*
    - SUB M *;A =A -Bellek[HL]*

Aritmetik İşlem Komutları

- **ARTTIRMA (INR)**
  - Herhangi bir bellek bölgesindeki değer yada bir kaydedicinin değeri bir arttırılabilir.
    - INR B ;*B= B+1*
    - INR 2000H ; *Bellek[2000H] +1*
    - INR M ;*Bellek[HL]+1*

- **EKSİLTME (DCR)**
  - Herhangi bir bellek bölgesindeki değer yada bir kaydedicinin değeri bir eksiltilebilir.
    - DCR C ;*C= C - 1*
    - DCR 4000H ; *Bellek[4000H] -1*
    - DCR M ;*Bellek[HL]-1*

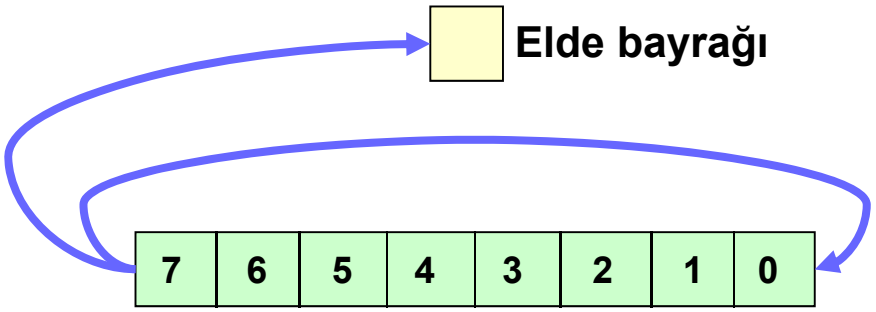
Mantıksal İşlem Komutları

- **VE, VEYA, ÖZEL VEYA:**
  - ANA, ANI, ORA, ORI, XRA ve XRI
    - Akü ile 8-bitlik bir sayı
      - ORI 30H ; A = A VEYA 30H
    - Akü ile bir kaydedicinin içeriği
      - XRA B ; A= A ÖZELVEYA B
    - Akü ile bir belleğin içeriği
      - ANA 1000H ; A= A VE Bellek[1000H]

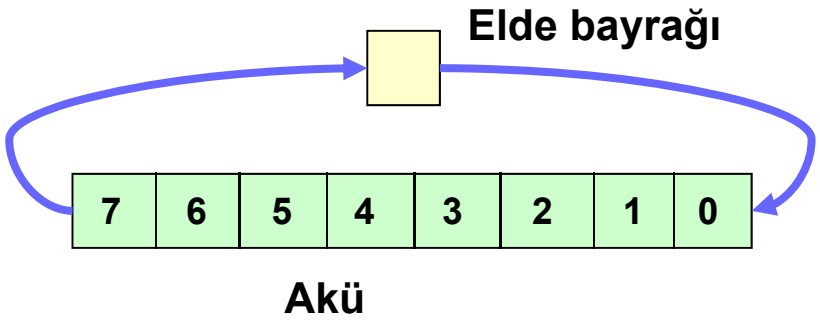
- **TERSLEME:**
  - Akünün 1'e tümleyini alma.
    - CMA ; A =!A

Mantıksal İşlem Komutları

- **DÖNDÜRME:**
  - Akümülatörün içeriğini bir kez sola ve sağa kaydırma.
    - RLC Aküyü bir bit sola kaydır. Bit7, Bit0'a ve eldeye gider.
    - RAL Aküyü elde üzerinden bir bit sola kaydır.
    - RRC Aküyü bir bit sağa kaydır. Bit0,Bit7'ye ve eldeye (CY) gider
    - RAR Akü elde üzerinden sağa kaydır.



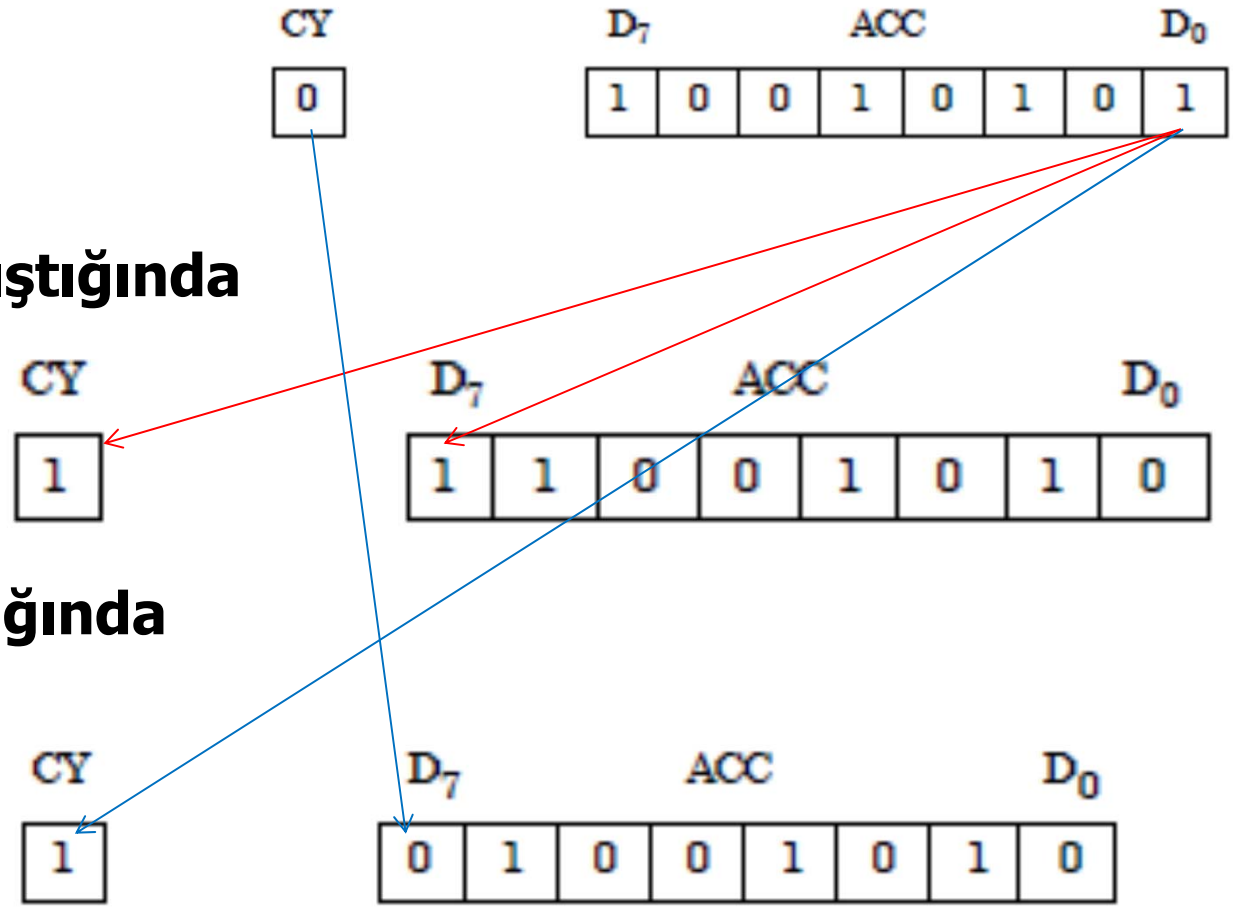
RLC



RAL

Mantıksal İşlem Komutları

- **DÖNDÜRME:**
  - **CY=0 ve A=95H**
  - **RRC komutunu çalıştığında**
  - **RAR komutu çalıştığında**



Mantıksal İşlem Komutları

- **KARŞILAŞTIRMA**
  - **CMP K/B:** Bir bellek adresinin yada bir kaydedicinin içeriğini akü ile kıyaslama.
    - CMP B *;Akü ile B kaydedicisini kıyasla*
    - CMP 2000H *; 2000H adresindeki bilgi ile aküyü kıyasla*
  - **CPI SAYI:** Bir sayı ile aküyü kıyaslama
    - CPI 30H *;Akü ile 30H'ı kıyasla*
  - Karşılaştırma komutu Z, CY, ve S bayrakları kurar.
  - Karşılaştırma bir dahili işlem olarak gerçekleştirilir bu yüzden akünün içeriği değişmez.

    A – (K / B/ Sayı)

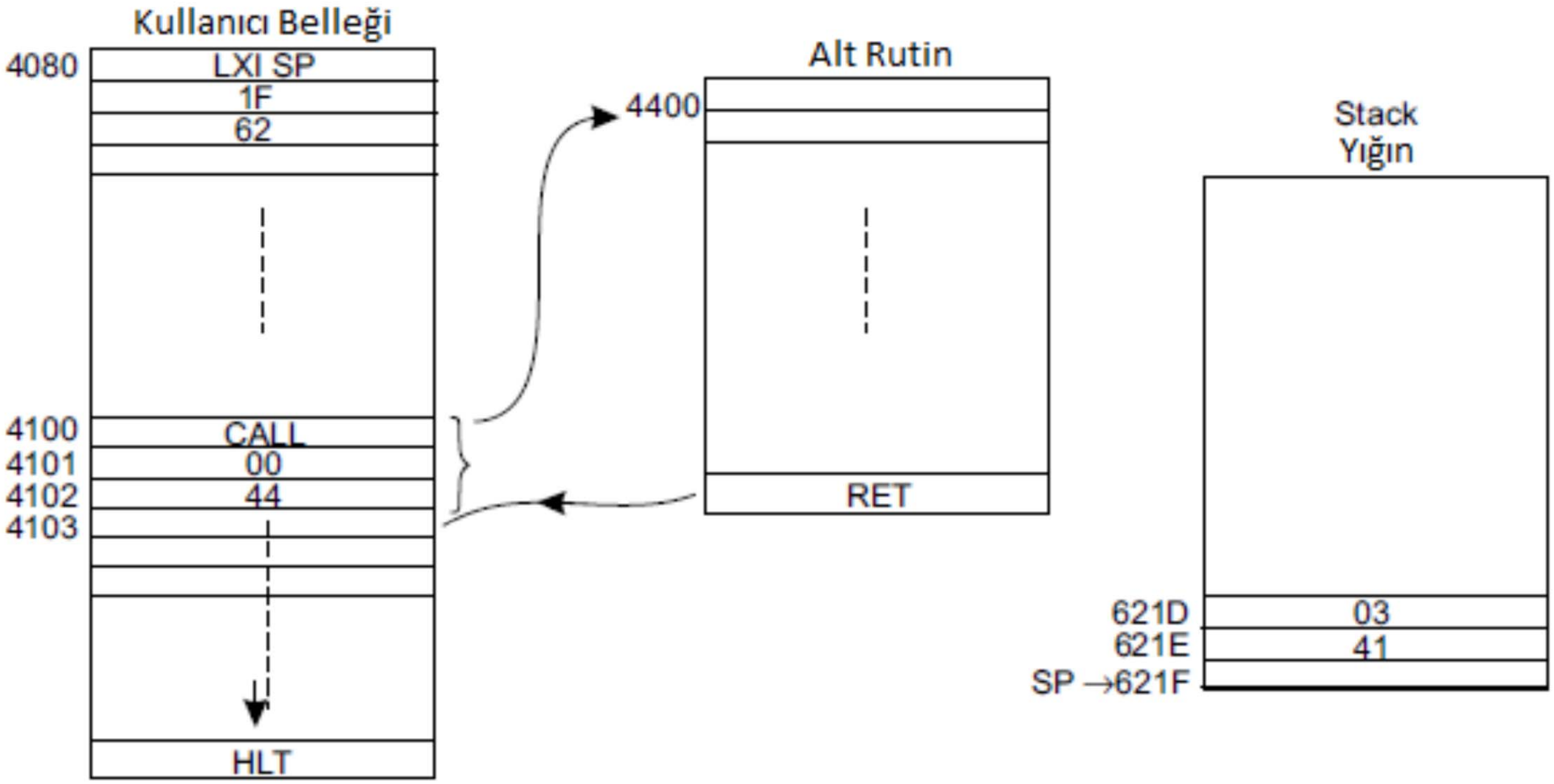
Dallanma Komutları

- 2 tür dallanma komutu vardır:
  - **Şartsız dallanma komutları**: Ne olursa olsun yeni bir bellek bölgesine git.
  - **Şartlı dallanma komutları**
    - Eğer bir şart gerçekleşirse yeni bir bellek bölgesine git.

Şartsız Dallanma Komutları

- **Şartsız dallanma komutları.**
  - JMP Adres *;Belirtilen adrese git*
    - JMP BASLA *;Basla etiketine git*
  - CALL Adres *;Adresteki alt programa git.*
    - CALL BEKLE *;BEKLE adındaki alt program git*
  - RET *;Alt programdan geri dön.*
- **NOT:** Adres değerleri 16-bit olmalıdır.

Şartsız Dallanma Komutları – CALL



Şartlı Dallanma Komutları

- JZ Adres (Jump on Zero)
  - **Zero bayrağı 1 ise adrese git**.
- JNZ Address (Jump on NOT Zero)
  - **Zero bayrağı 1 değil ise adrese git**.
- JC Adres (Jump on Carry)
  - **Elde 1 ise adres git**.
- JNC Addres (Jump on No Carry)
  - **Elde 1 değil ise adres git**.
- JP Adres (Jump on Plus)
  - **İşaret bayrağı 1 değil ise adrese git**
- JM Adres (Jump on Minus)
  - **İşaret bayrağı 1 ise adrese git**

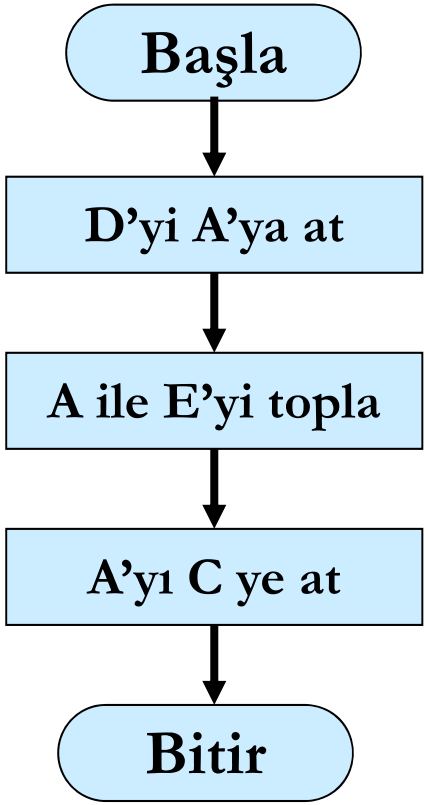
Program Kontrol Komutları

- **HLT**
  - Programın çalışmasını durdurma.

- **NOP**
  - No operation
  - Hiçbir şey yapma.
  - Genelde gecikme amaçlı kullanılır.

Örnek-1

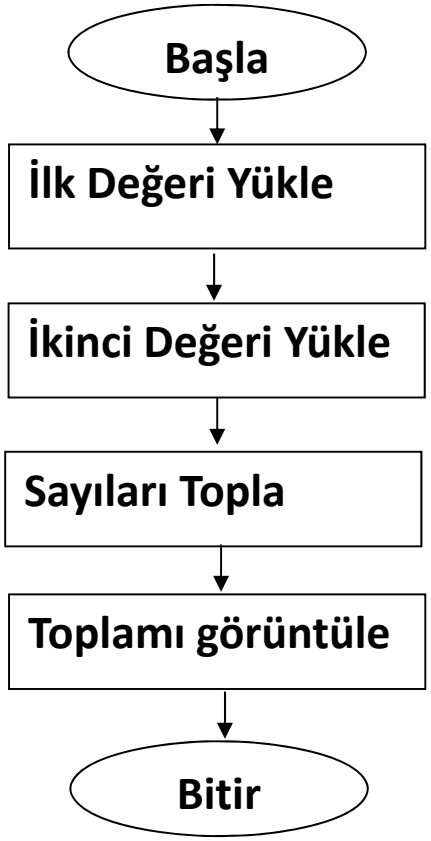
- D ve E kaydedicilerindeki iki sayı toplayarak sonucu C kaydedicisine aktaran programı yazılım.




```
MOV A, D        ;A=D
ADD E           ;A=A+E
MOV C, A        ;C=A  (D+E)
HLT             ;BITIR
```

Örnek-2

- 32 ve 48 sayılarını toplanarak sonucu 1H adresindeki 8 adet LED içeren çıkış biriminde görüntüleyecek programı yazalım.




```
MVI A, 32      ;A=32
MVI B, 48      ;B=48
ADD B          ;A=32+48
OUT 01H        ;LEDLER=80
HLT            ;BITIR
```

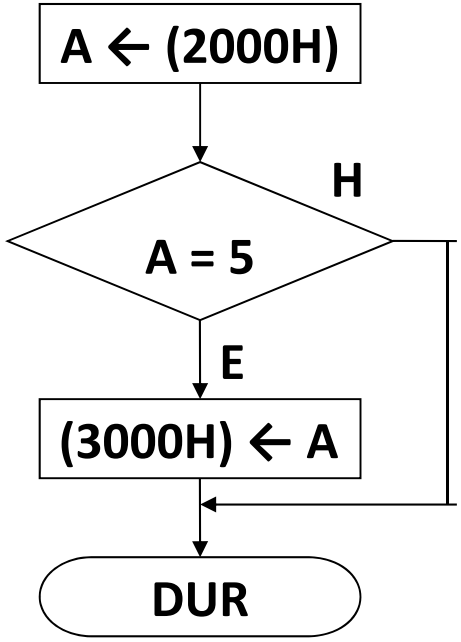
Örnek-3

- 10H Adresli 8 adet anahtar grubu içeren giriş birimindeki değeri okuyarak 20H adresli 8'li LED içeren çıkış birimine aktaran uygulamayı yazalım.


```
BASLA:IN  10H      ;A=GIRIS[10H]
      OUT 20H      ;CIKIS[20H]=A
      JMP BASLA    ;TEKRAR
```

Örnek-4

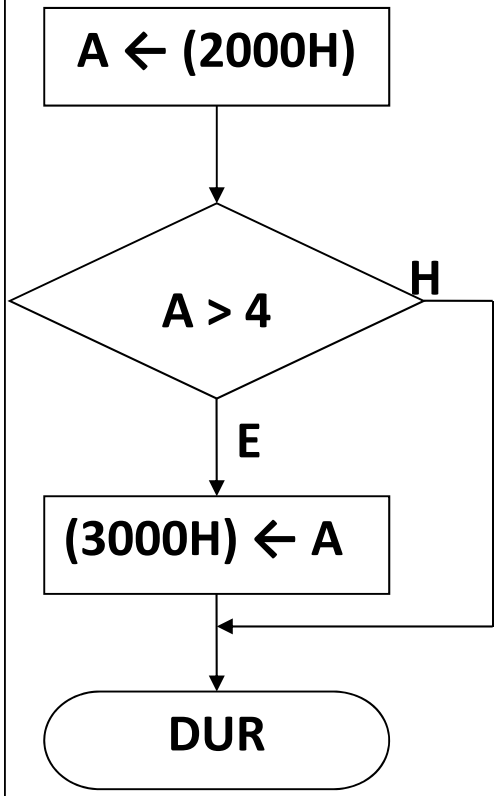
- 2000H adresindeki sayı 5'e eşit ise bu sayıyı 3000H adresine yazan eşit değilse programı bitiren uygulamayı 8085 asembly dili ile yazalım.




```
        LDA 2000H  ;A=B[2000H]
        CPI 5      ;A-5
        JNZ SON    ;!0,SON'a git
        STA 3000H  ;0,B[3000H]=A
SON:    HLT
```

Örnek-5

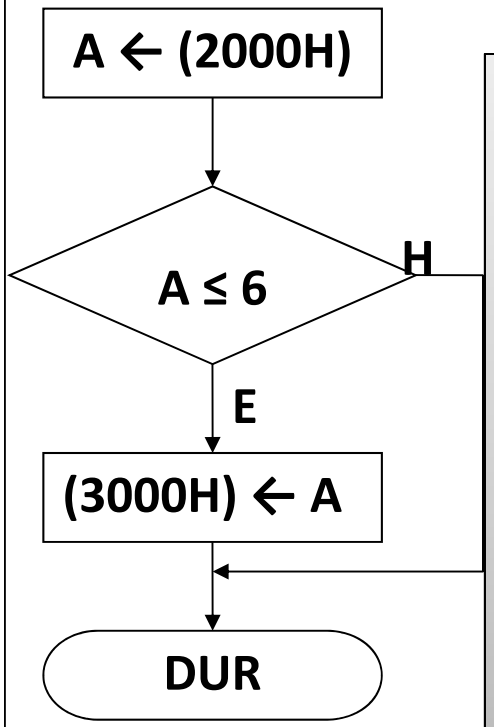
- 2000H adresindeki sayı 4'ten büyük ise bu sayıyı 3000H adresine yazan programı 8085 asembly dili ile yazalım.




```
        LDA 2000H   ;A=B[2000H]
        CPI 4       ;A-4
        JZ SON      ;Eşit SON'a git
        JC SON      ;Küçük SON'a git
        STA 3000H   ;B[3000H]=A
SON:    HLT         ;BITIR
```

Örnek-6

- 2000H adresindeki sayı 6'dan küçük veya 6'ya eşit ise bu sayıyı 3000H adresine yazan programı 8085 asembly dili ile yazalım.




```
        LDA 2000H ;A=B[2000H]
        CPI 6      ;A-6
        JZ DEVAM  ;Eşit DEVAM'a git
        JC DEVAM  ;Küçük DEVAM'a git
        JMP SON   ;SON'a git
DEVAM:  STA 3000H ;[3000H]=A
SON:    HLT       ;BITIR
```

Örnek-7

- 4000H - 4003H aralığındaki sayıları 3000H – 3003H aralığına taşıyan programı 8085 asembly dilinde yazalım.

```
BASLA:   LDA 4000H   ;A=B[4000H]
         STA 3000H   ;B[3000H]=A
         LDA 4001H   ;A=B[4001H]
         STA 3001H   ;B[3001H]=A
         LDA 4002H   ;A=B[4002H]
         STA 3002H   ;B[3002H]=A
         LDA 4003H   ;A=B[4003H]
         STA 3003H   ;B[3003H]=A
         HLT         ;BITIR
```

Örnek-8

- 4000H - 4009H aralığındaki sayıları 3000H – 3009H aralığına taşıyan programı 8085 asembly dilinde döngüler ile yazalım.

```
BASLA:   LXI H, 4000H   ;HL=4000H
         LXI D, 3000H   ;DE=3000H
YENİ:    MOV A, M       ;A=B[4000H]
         STAX D         ;B[DE]=A
         INX M          ;HL=HL+1
         INX D          ;DE=DE+1
         MOV A, L       ;A=L
         CPI 0AH        ;A==10
         JNZ YENİ       ;(H),YENİ'ye git
         HLT            ;(E),Bitir
```

Örnek-9

- 3000H – 3009H aralığındaki sayıların kaç tanesinin 7'den küçük olduğunu bulan ve sonucu 20H adresindeki çıkışa aktaran programı yazalım

```
BASLA:    LXI H,3000H   ;HL=3000H
          MVI C,0H      ;Sayac=0
DEVAM:    MOV A, M      ;A=B[3000H]
          CPI 7         ;A-7
          JNC DEGİL     ;Küçük Değil,DEGIL'e git
          INR C         ;Küçük, sayac=sayac+1
DEGİL:    INX M         ;HL=HL+1
          MOV A,L       ;A=L
          CPI 0AH       ;A-10
          JNZ DEVAM     ;Eşit değil, DEVAM'a git
          MOV A,C       ;Eşit, A=Sayac
          OUT 20H       ;Çıkış[20H]=Sayac
          HLT           ;Bitir
```

Bilmemiz Gerekenler

- Komut nedir?
- Bir komut hangi alanlardan oluşur?
- 8085 hangi adresleme modlarını destekler? Örnekler vereniz?
- 8085 kaç adet komut destekler?
- LDA ile STA arasındaki farkı açıklayınız?
- RAL ve RLC komutları arasındaki fark nedir?
- Koşulsuz dallanma komutları nelerdir?
- CALL komutu çalışırken işlemci ne tür işlemler yerine getirir?
- Koşullu dallanma komutları nelerdir?
- JZ ile JNZ arasındaki fark nedir?
- 8085 assembly dilini kullanarak program yazabilirmisiniz?