14. Hafta Assembler

**Programlama gelişmeleri**

* Programlanabilir makineler geliştirilmeye başlanınca, programlar ve programlama dilleri de gelişmeye başladı.
* Bağlantıların yeri değiştirilerek programlama yapmak oldukça zor olduğu için, bilgisayar dilleri gelişmeye başladı.
* İlk makine dili, birler ve sıfırlar ile geliştirildi.

-Hafızada program denilen bir grup komut biçiminde idi.

* 19501erin başlarında, UNIVAC benzeri sistemlerin gelişmesi ile, digital komutların girilmesini kolaylaştıran assembly dilleri kullanılmaya başlandı.
* Assembler, kullanıcıların anlaşılabilir kodlar yazmasına izin verdi...

-Toplama için **ADD** komutu gibi

* Digital komutlar yerine

- 0100 0111 gibi

* Assembly dili, programlama yapmak için kullanışlı bir araç haline geldi.

**Assembly'nin önemi**

* C/C++, genellikle assembly dili kullanan düşük seviye kontrol yazılımlarında kullanılmaya başlandı.
* Ancak, assembly dili hala önemli bir yere sahiptir.
* Çoğu video oyunu, neredeyse tamamen assembly dilinde yazılır
* Assembly ayrıca, C/C++ ile entegre olarak, bazı kontrol fonksiyonlarının daha hızlı gerçekleştirilmesini sağlar.
* Pentium ve çift çekirdekli işlemcilerde kullanılan bazı paralel komutlar, sadece assembly dilinde yazılır.

**Instruction Set — Komut Seti**

* Mikroişlemcilerin konuştuğu dil olarak düşünülebilir. O
* Her mikroişlemcinin kendine özgü komut seti vardır.
* Komutlar l'ler ve 0’lardan oluşan bir diziden ibarettir. Örn;

|  |  |  |
| --- | --- | --- |
| **Main Memory** | **Hex** | **Instructions** |
| 00111010 | 3A | LDA 003A |
| 00000000 | 00 |  |
| 01000111 | 47 | MOV B,A |
| 00111010 | 3A | LDA 0038 |
| 00111011 | 38 |  |
| 00000000 | 00 |  |
| 10000000 | 80 | ADD B |
| 00110010 | 32 | STA 003C |
| 00111100 | 3C |  |
| 00000000 | 00 |  |
| 01110110 | 76 | HLT |

**Intel 8085 Mikroişlemcisi Komut Seti**

* 8085 mikroişlemci komut seti, 74 farklı işlemi gerçekleştiren 246 komuttan oluşur.
* Her bir kod 8 bitlik binary sayı ile gösterilir. Bu 8 bitlik sayıya Op-Code (Operation Code) denir.
* Bu komutlar yapılan işler referans alınarak beş farklı grup altında toplanabilir:

1-Veri transferi/aktarım komutları,

2-Aritmetik işlemlerle ilgili komutlar,

3- Mantık işlemleriyle ilgili komutlar,

4-Program akış kontrol komutları,

5- Mikroişlemci kontrol komutları.

# Veri transferi/aktarım komutları

<> Bu komutlar register'lar arasında veya hafıza ile register'lar arasında veri aktarırlar.

<> Bu komutlar kaynaktan hedefe veri kopyalarlar.

<> Bu kopyalama işlemi esnasında kaynağın içeriği değiştirilmez.

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **MOV** | **Rd, Rs**  **M, Rs   Rd, M** | **Kaynaktan hedefe kopyalar** |

<> Bu komut kaynak register'ının içeriğini hedef register'a kopyalar.

<> Kaynak register'ının içeriği değiştirilmez.

<> Operand'lardan biri hafıza bölgesi ise, bu bölgenin adresi HL register'ları tarafından gösterilir.

HL çifti bir bellek işaretçisi olarak çalışır.

Örnek:

**MOV B,C** ; C'yi B'ye ata

**MOV B, HL** ; HL adresindeki veriyi B'ye ata

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **MVI** | **Rd. Data**  **M, Data** | **İvedi olarak 8 biti ata** |

<> 8 bitlik veri hedef register'a veya hafızaya kaydedilir.

<> Operand hafıza bölgesi ise, bu bölgenin adresi HL register’ları tarafından gösterilir.

Örnek: **MVI B, 57H**; 57H sayısını B' ye ata

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **LDA** | **16 bit adres** | **Akümülatöre Yükle** |

<> Operand kısmında belirtilen 16 bitlik hafıza adresinin içeriği akümülatöre kopyalanır.

<> Kaynağın içeriği değiştirilmez.

<> Örnek: **LDA 2034H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **LDAX** | **B/D register çifti** | **Dolaylı olarak Akümülatöre yükle** |

<> Operand kısmında register çifti hafıza bölgesini gösterir.

<> Bu komut belirtilen 16 bitlik hafıza adresinin içeriği akümülatöre kopyalar.

<> Kaynağın içeriği değiştirilmez.

<> Örnek: **LDAX B;** BC çiftinin gösterdiği adresteki değeri A'ya yükle

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **LXI** | **Register çifti, 16 bitlik veri** | **Register çiftine hemen yükle** |

<> Bu komut 16 bitlik veriyi register çiftine yükler.

<> B, D, H yazılır bu da BC, DE, HL demektir.

<> Örnek**: LXI H, 2034H;** HL çiftine 2034H'ı yükler

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **LHLD** | **16 bit adres** | **H-L Register’larına doğrudan yükle** |

<> Bu komut 16 bitlik adresle gösterilmiş olan yerdeki veriyi register L'ye yükler.

<> Sonraki hafıza adresinin içeriğini de H register'ına yükler.

<> Örnek: **LHLD 2040H ;**  2040H'ı L'ye yükler,

2041H'ı da H'ye yükler.

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **STA** | **16 bit adres** | **Akümülatörü doğrudan yükle** |

<> Operand ile belirtilmiş adrese akümülatörün içeriği kopyalanır.

<> Örnek: **STA 2500H** ; 2500H adresinin içeriğine A'yı yükle

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **STAX** | **Register Çifti** | **Akümülatörü dolaylı olarak depola** |

<> Akümülatörün içeriği, register çifti tarafından belirtilen adrese kopyalanır.

<> Örnek: **STAX B** ; BC çiftinin gösterdiği adrese A'yı kopyala

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **SHLD** | **16 bit adres** | **HL Register’larını doğrudan depola** |

<> L register'ının içeriği 16 bitlik adres ile belirtilen hafıza konumuna depolanır.

<> H register'ının içeriği de sonraki hafıza konumuna depolanır.

<> Örnek: **SHLD 2550H** ; 2550H adresine L register'ını, sonraki adrese de H register'ını kopyala

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **XCHG** | **YOK** | **HL’yi DE’ile değiştir** |

<> H register'inin içeriği D' nin ki ile değiştirilir.

<> L register'ının içeriği E'nin ki ile değiştirilir.

<> Örnek: **XCHG**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **SPHL** | **YOK** | **HL register çiftinin içeriğini Stack Pointer (SP)’ye kopyala** |

<> Bu komut HL register çiftinin içeriğini SP'ye yükler.

<> Örnek: **SPHL**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **XTHL** | **YOK** | **HL’yi stack’in en üstü ile değiştir** |

<> L register'ının içeriği SP'ın gösterdiği yerin içeriği ile değiştirilir.

<> H register'ının içeriği bir sonraki yerin (SP+1) içeriği ile değiştirilir.

<> Örnek: **XTHL**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **PCHL** | **YOK** | **Program sayıcının içeriğine HL’yi ata** |

<> H ve L registerlarmın içeriği program sayıcının içerisine kopyalanır.

<> H'nin içeriği yüksek byte'ına, L'nin içeriği düşük byte'ına yerleştirilir.

<> Örnek: **PCHL**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **PUSH** | **Register Çifti** | **Register çiftinin içeriğini stack’e koy** |

<> Register çiftinin içeriği stack'e (yığına) kopyalanır.

<>SP 1 azaltılır, yüksek seviye register 'ların (B, D, H, A) içeriği stack'e kopyalanır.

<> SP tekrar 1 azaltılır, ve düşük seviye register'ların içeriği (C, E, L, flags) stack'e kopyalanır.

<> Örnek: **PUSH B**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **POP** | **Register Çifti** | **Stack’ten register çiftine aktarır** |

<> Stack'in en üstteki içeriği register çiftine kopyalanır.

<> SP tarafından gösterilen yerin içeriği düşük seviye register'a kopyalanır (C, E, L, Flags).

<> SP 1 arttırılır, ve içerik yüksek seviye register'a (B, D, H, A) kopyalanır.

<> Örnek: **POP H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **OUT** | **8 bitlik port**  **adresi** | **Akümülatördeki veriyi 8 bitlik adresi olan porta kopyala** |

<> Akümülatörün içeriği I/O portuna kopyalanır.

<> Örnek: **OUT 78H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **IN** | **8 bitlik port**  **adresi** | **Porttan veriyi akümülatöre kopyalar** |

<> I/O portunun içeriği akümülatöre kopyalanır.

<> Örnek: **IN 8CH** ; 8C portundaki veriyi A'ya kopyalar

**Veri transferi/aktarım komutları**

**MVI B, 4FH** ; 8 bitlik 4f sayısını B registerına yükle

**MOV A, B** ; B registerından A registerına kopyala

**LXI H, 2050H**  ; 16 bitlik 2050H sayısını HL çiftine yükle

**MOV M,B** ; B registerını 2050H hafıza adresine kopyala

**OUT 01H** ; Akümülatörün içeriğini 01H portuna kopyala

**IN 07H** ; 07H portunu akümülatöre kopyala

**HLT** ; Programı Sonlandır

## Aritmetik işlem Komutları - Toplama

<> Herhangi 8 bitlik bir sayı veya register'ın içeriği veya herhangi bir hafıza yerinin içeriği

akümülatörün içeriğine eklenebilir.

<> Sonuç akümülatöre kaydedilir.

<> Diğer iki 8 bitlik register'lar direkt olarak toplanamazlar.

<> Örneğin; B register’ının içeriği C'nin içeriğine eklenemez.

## Aritmetik işlem Komutları - Çıkarma

<> Herhangi 8 bitlik bir sayı veya register'ın içeriği veya herhangi bir hafıza yerinin içeriği

akümülatörün içeriğinden çıkarılabilir.

<> Sonuç akümülatöre kaydedilir.

<> Çıkarma işlemi 2'nin tümleyeni şeklinde yapılır.

<> Sonuç negatif ise sonuç ikinin tümleyeni şeklinde kaydedilir.

<> Diğer iki 8 bitlik register'lar direkt olarak çıkarılamazlar.

## Arttırma / Azaltma

<> 8 bit register'ın içeriği veya hafiza yerinin içeriği 1 arttırılıp azaltılabilir.

<> 16 bit register çiftinin içeriği 1 arttırılıp azaltılabilir.

<> Arttırma ve azaltma bütün register'lar ve hafiza konumları için uygulanabilir.

# Aritmetik Komutlar

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **ADD** | **R**  **M** | **Akümülatöre register’ı veya hafızayı ekle** |

<> Register'ın veya hafızanın içeriği akümülatöre eklenir.

<> Sonuç akümülatöre yüklenir.

<> Operand hafıza yeri ise adresi H-L çiftinde belirtilir.

<> Bütün bayraklar sonucu yansıtacak şekilde değiştirilir.

<> Örnek: **ADD B** veya **ADD M**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **ADC** | **R**  **M** | **Akümülatöre, carry ile birlikte registerı veya hafızayı ekle.** |

<> Register'ın veya hafızanın içeriği ile CY, akümülatörün içeriğine eklenir.

<> Sonuç akümülatöre yüklenir.

<> Eğer operand hafıza konumu ise adresi HL çiftinde gösterilir.

<> Bütün bayraklar toplama sonucunu yansıtacak şekilde değiştirilir.

<> Örnek: **ADC B** veya **ADC M**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **ADI** | **8 bitlik veri** | **Hemen akümülatöre ekle** |

<> 8 bitlik veri akümülatörün içeriğine eklenir.

<> Sonuç akümülatöre kayıt edilir.

<> Bütün bayraklar toplama sonucunu yansıtacak şekilde değiştirilir.

<> Örnek: **ADI 45H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **ACI** | **8 bitlik veri** | **Elde ile hemen akümülatöre ekle** |

<> 8 bitlik veri ile CY, akümülatörün içeriğine eklenir.

<> Sonuç akümülatöre yüklenir.

<> Bütün bayraklar toplama sonucunu yansıtacak şekilde değiştirilir.

<> Örnek: **ACI 45H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **DAD** | **Register çifti** | **Register çiftini HL çiftine ekle** |

<>16 bitlik register çiftinin içeriği HL çiftinin içeriğine eklenir.

<> Sonuç HL çiftine yazılır.

<> Eğer sonuç 16 bitten fazla ise CY bir yapılır.

<> Başka bir bayrak değiştirilmez.

<> Örnek: **DAD B**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **SUB** | **R**  **M** | **Reigster’ı veya hafızayı akümülatörden çıkar** |

<> Register'ın veya hafızanın içeriği akümülatörden çıkarılır.

<> Sonuç akümülatöre yüklenir.

<> Operand hafıza yeri ise adresi H-L çiftinde belirtilir.

<> Bütün bayraklar sonucu yansıtacak şekilde değiştirilir.

<> Örnek: **SUB B**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **SBB** | **R**  **M** | **Register veya hafızayı ödünç ile birlikte akümülatörden çıkar** |

<> Register'ın veya hafızanın içeriği ile ödünç bayrağı, akümülatörün içeriğinden çıkarılır.

<> Sonuç akümülatöre yüklenir.

<> Eğer operand hafıza konumu ise adresi HL çiftinde gösterilir.

<> Bütün bayraklar toplama sonucunu yansıtacak şekilde değiştirilir.

<> Örnek: **SBB B** veya **SBB M**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **SUI** | **8 bit data** | **Hemen akümülatörden çıkar** |

<> 8 bitlik veri akümülatörün içeriğinden çıkarılır.

<> Sonuç akümülatöre kayıt edilir.

<> Bütün bayraklar çıkarma sonucunu yansıtacak şekilde değiştirilir.

<> Örnek: **SUI 45H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **SBI** | **8 bit veri** | **Akümülatörden ödünç ile birlikte çıkar** |

<> 8 bitlik veri ile elde bayrağının içeriği, akümülatörün içeriğinden çıkarılır.

<> Sonuç akümülatöre yüklenir.

<> Bütün bayraklar çıkarma sonucunu yansıtacak şekilde değiştirilir.

<> Örnek: **SBI 45H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **INR** | **R**  **M** | **Register’ı veya hafızayı 1 arttırır** |

<> Register'ın veya hafızanın içeriği bir arttırıldı.

<> Sonuç aynı yere yüklenir.

<> Eğer oprerand hafıza yeri ise adresi HL çifti ile gösterilir.

<> Örnek: **INR B** veya **INR M**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **INX** | **R** | **Register çiftini bir arttır** |

<> Register çiftinin içeriği 1 artar.

<> Sonuç aynı yere yüklenir.

<> Örnek: **INX H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **DCR** | **R**  **M** | **Register’ı veya hafızayı 1 azalt** |

<> Register'ın veya hafızanın içeriği bir azaltılır.

<> Sonuç aynı yere yüklenir.

<> Eğer oprerand hafıza yeri ise adresi HL çifti ile gösterilir.

<> Örnek: **DCR B**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **DCX** | **R** | **Register çiftini 1 azalt** |

<> Register çiftinin içeriği 1 azalır.

<> Sonuç aynı yere yüklenir.

<> Örnek: **DCX H**

**Aritmetik komutlar**

**ADI 32H** ; 8 bitlik 32H sayısını akümülatöre ekle

**ADD B** ; B registerını akümülatöre ekle

**SUI 32H** ; 8 bitlik 32H sayısını akümülatörden çıkar

**SUB C** ; C registerının içeriğini akümülatörden çıkar

**INR D** ; D registerını 1 arttır

**DCR E** ; E registerını 1 azalt

# Mantıksal Komutlar

<>Bu komutlar register'larda, hafızada ve durum bayraklarında yüklenmiş olan veriler üzerinde mantıksal işlemleri gerçekleştirirler.

<> Mantıksal işlemler şöyledir:

<>AND

<> OR

<> XOR

<>Rotate

<> Compare

<>Complement

**AND, OR, XOR**

<>Herhangi bir 8 bitlik veri, veya hafızanın veya register'lerın içeriği, akümülatörün içeriğiyle birlikte

<>AND

<>OR

<>XOR

mantıksal işlemlerine maruz kalabilir.

<> Sonuç akümülatöre yüklenir.

## Rotate

<> Akümülatördeki her bir bit ya sağa ya da sola bir sonraki pozisyonuna kaydırılabilir.

## Compare

<>Herhangi bir 8 bitlik veri, veya hafızanın veya register'ların içeriği, akümülatörün içeriğiyle birlikte

<>Eşitlik

<>büyüklük

<>küçüklük

Açısında karşılaştırma mantıksal işlemlerine maruz kalabilir.

<>Sonuç durum bayraklarına yansıtılır.

## Complement

<> Akümülatörün içeriğinin complementi alınabilir.

<> Her bir 0,1 ile ve her bir 1 de 0 ile değiştirilir.

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **CMP** | **R**  **M** | **Register’ı veya hafızayı akümülator ile karşılaştır** |

<>Operand'ın içeriği, akümülatörün içeriği ile karşılaştırılır.

<>Tüm içerikler korunmuştur.

<> Karşılaştırmanın sonucunda durum bayrakları şöyle düzenlenir:

<> Eğer (A) < reg./mem => elde bayrağı (CF) 1 yapılır

<>Eğer (A) = reg./mem => sıfır bayrağı (ZF) 1 yapılır

<>Eğer (A) > reg./mem => elde ve sıfır bayrağı (CF ve ZF) 1 yapılır

<> Örnek: **CMP B**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **CPI** | **8 bitlik veri** | **Hemen akümülatör ile karşılaştır** |

<> 8 bitlik data akümülatörün içeriği ile karşılaştırılır.

<> Karşılaştırılan değer değiştirilmez.

<> Karşılaştırma= sonucunda durum bayrakları şöyle düzenlenir:

<> Eğer (A) < data=> elde bayrağı (CF) 1 yapılır

<> Eğer (A) = data => sıfır bayrağı (ZF) 1 yapılır

<> Eğer (A) > data => elde ve sıfır bayrağı (CF ve ZF) **0** yapılır

<> Örnek: **CPI 89H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **ANA** | **R**  **M** | **Register veya hafızayı, Akümülatör ile AND işlemine tabi tut** |

<> Akümülatörün içeriği, register veya hafızanın içeriği ile AND'Ienir.

<> Sonuç akümülatöre yazılır.

<> Eğer operand hafıza konumu ise bu HL çiftinde tutulur.

<> S, Z, P işlem sonucunu yansıtacak şekilde düzenlenir

<> CY 0, AC 1 olur.

<> Örnek: **ANA B**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **ANI** | **8 bit data** | **Akümülatör ile hemen AND işlemine tabi tut** |

<> Akümülatörün içeriği, 8 bitlik veri içeriği ile AND'lenir.

<> Sonuç akümülatöre yazılır.

<> S, Z, P işlem sonucunu yansıtacak şekilde düzenlenir

<> CY 0, AC 1 olur.

<> Örnek: **ANI 86H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **ORA** | **R**  **M** | **Register veya hafızayı, Akümülatör ile OR işlemine tabi tut** |

<> Akümülatörün içeriği, register veya hafızanın içeriği ile OR'lanır.

<> Sonuç akümülatöre yazılır.

<> Eğer operand hafıza konumu ise bu HL çiftinde tutulur.

<> S, Z, P işlem sonucunu yansıtacak şekilde düzenlenir

<> CY ve AC 0 olur.

<> Örnek: **ORA B**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **ORI** | **8 bit data** | **Akümülatör ile hemen OR işlemine tabi tut** |

<> Akümülatörün içeriği, 8 bitlik veri içeriği ile OR'lanır.

<> Sonuç akümülatöre yazılır.

<> S, Z, P işlem sonucunu yansıtacak şekilde düzenlenir

<> CY ve AC 0 olur.

<> Örnek: **ORI 86H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **XRA** | **R**  **M** | **Register veya hafızayı, Akümülatör ile XOR işlemine tabi tut** |

<> Akümülatörün içeriği, register veya hafizanın içeriği ile XOR'lanır.

<> Sonuç akümülatöre yazılır.

<> Eğer operand hafıza konumu ise bu HL çiftinde tutulur.

<> S, Z, P işlem sonucunu yansıtacak şekilde düzenlenir

<> CY ve AC 0 olur.

<> Örnek: **XRA B**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **XRI** | **8 bit data** | **Akümülatör ile hemen XOR işlemine tabi tut** |

<> Akümülatörün içeriği, 8 bitlik veri içeriği ile XOR'lanır.

<> Sonuç akümülatöre yazılır.

<> S, Z, P işlem sonucunu yansıtacak şekilde düzenlenir

<> CY ve AC 0 olur

<> Örnek: **XRI 86H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **RAL** | **Yok** | **Akümülatörü carry (elde) üzerinden sola doğru döndür** |

<>Akümülatörün her bir bitini elde biti üzerinden sola doğru bir pozisyon döndürür.

<> D7 biti elde bitine yazılır, elde biti de DO yazılır.

<> CY, D7'ye göre değişir.

<> S, Z, P, AC etkilenmez.

<> Örnek: **RAL**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **RAR** | **Yok** | **Akümülatörü carry (elde) üzerinden sağa doğru döndür** |

<> Akümülatörün her bir bitini elde biti üzerinden sağa doğru bir pozisyon döndürür.

<> DO biti elde bitine yazılır, elde biti de D7 yazılır.

<> CY, D0'ye göre değişir.

<> S, Z, P, AC etkilenmez.

<> Örnek: **RAR**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **RRC** | **Yok** | **Akümülatörü sağa doğru döndür** |

<> Akümülatörün her bir bitini sağa doğru bir pozisyon döndürür.

<> D0 biti elde bitine yazılır, aynı zamanda D7 yazılır.

<> CY, D0'a göre değişir.

<> S, Z, P, AC etkilenmez.

<> Örnek: **RRC**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **CMA** | **Yok** | **Akümülatörün tümleyeni** |

<> Akümülatörün tümleyeni alınır.

<> Bayraklar etkilenmez.

<> Örnek: **CMA**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **CMC** | **Yok** | **Eldenin tümleyeni** |

<> Elde bitinin tümleyeni alınır.

<> Diğer bayraklar etkilenmez.

<> Örnek: **CMC**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **STC** | **Yok** | **Eldeyi 1 yap** |

<> Elde bayrağı 1 yapılır.

<> Diğer bayraklar etkilenmez.

<> Örnek: **STC**

**Mantıksal komutlar**

**ANA H** ; H registerını akümülatörle AND'le

**ORA L** ; L registerını akümülatörle OR'la

**XRA B** ; B registerını akümülatörle XOR'la

**CMP C** ; C registerını akümülatörle karşılaştır

**CMA**  ; Akümülatörün tümleyenini al

**RAL** ; Akümülatörü sola rotate et

# Program Akış Komutları

<> Program akış komutları programın normal gidişatnı değiştirirler.

<> Bu komutlar koşullu yada koşulsuz değiştirme işlemini yaparlar.

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **JMP** | **16 bitlik adres** | **Koşulsuz atla** |

<> Program, operand ile verilen 16 bitlik hafızadaki adrese gider.

<> Örnek: **JMP 2034H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **JZ (pdfte yanlış)** | **16 bitlik adres** | **Koşullu atla** |

<> Program, operand ile verilen 16 bitlik hafızadaki adrese belirtilen bayrağın durumuna göre gider.

<> Örnek: **JZ 2034H**

**Program Akış Komutları**

**Opcode Açıklama Durum bayrakları**

JC Elde 1 ise atla CY= 1

JNC Elde 0 ise atla CY=0

JP Pozitif ise atla S=0

JM Negatif ise atla S=1

JZ Sıfırsa atla Z=1

JNZ Sıfır değilse atla Z=0

JPE Parity çift ise atla P=1

JPO Parity tek ise atla P=0

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **CALL** | **16 bitlik adres** | **Call koşulsuz** |

<> Program, operand ile verilen 16 bitlik hafızadaki adrese gider.

<> Gitmeden önce, Call komutundan sonraki komutun adresi stack'e konur.

<> Örnek: **CALL 2034H**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **RET** | **yok** | **Return koşulsuz** |

<> Program, çağrılan alt programdan döner .

<> Stack'in en üstteki iki byte'ı program sayıcıya kopyalanır, program bu yeni adresten devam eder.

<> Örnek: **RET**

Program Akış Komutları

**JC 2080H** ; eğer CF=1 ise 16 bitlik 2080H adresine atla

**JMP 2050H**  ; 2050H adresine atla

CALL **3050H** ; 3050H adresindeki alt programı çağır

**RET**  ; alt programdan dön

**CNC 3050H**  ; eğer CF=0 ise 16 bitlik 3050H

; adresindeki alt programı çağır

**RZ** ; eğer ZF=1 ise dön

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **NOP** | **yok** | **İşlem Yapma** |

<> İşlem yapılmaz

<> Komut getirilir ve çözülür ama işlem yapılmaz.

<> Örnek: **NOP**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **HLT** | **yok** | **Halt** |

<> CPU bu komutu görünce işlemleri durdurur.

<> Halt'tan çıkmak için kesme veya reset gereklidir.

<> Örnek: **HLT**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **DI** | **yok** | **Kesmeleri devre dışı et** |

<> TRAP dışındaki bütün kesmeler devre dışı bırakılır.

<> Bayraklar etkilenmez

<> Örnek: **DI**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Operand** | **Açıklama** |
| **EI** | **yok** | **Kesmeleri aktif et** |

<> Bütün kesmeler aktif edilir.

<> Bayraklar etkilenmez

<> Örnek: **EI**

# Adresleme Modları

<> Adresleme modu, assebmly dilinde kullanılan işlem yöntemidir. Bir komutun gerektirdiği işlemin yürütülmesi sırasında verilen bir bellek yerinden yani bir adresten alınması veya bir adrese yüklenmesi ya da akümülatör ve kaydediciler arasında değiştirilmesi için değişik yöntemler uygulanmaktadır.

<> Neden adresleme modlarına ihtiyaç duyulmuştur?

<> Değişik adresleme modları, kullanılma amacına göre, bilgisayardan daha hızlı ve daha randımanlı şekilde yararlanılmasını sağlamaktadır.

<> Daha çok ADRESLEME MODU geliştirilerek ihtiyaç duyulan Komut Sayısı azaltılmıştır

<> Adresleme Modlarıyla belleğin daha ekonomik kullanılması sağlanmıştır.

**ADRESLEME YÖNTEMLERİ**

<> 8085 mikroişlemci yapısında temel olarak 5 adresleme yöntemi vardır. Bunlar şu şekildedir:

1) Doğrudan Adresleme Yöntemi (Direct addressing mode)

2) İvedi Adresleme Yöntemi (Immediate addressing mode)

3) Kaydedici Adresleme Yöntemi (Register addressing mode)

4) Dolaylı Adresleme Yöntemi (Indirect addressing mode)

5) İmali Adresleme Yöntemi (Implied addressing mode)

1) Doğrudan Adresleme Yöntemi (Direct addressing mode)

Bu adresleme yönteminde yerinin bulunduğu bellek adresi direkt olarak verilerek yapılır.

Örnek olarak:

<> **LDA 3000H** ; Akümülatöre 3000H bellek adresindeki veriyi yükler.

<> **STA 2000H** ; Akümülatör içinde bulunan veriyi 2000H adresindeki bellek gözüne kopyalar.

2) İvedi Adresleme Yöntemi (Immediate addressing mode)

Bu adresleme yönteminde veri direkt olarak kullanılır. Yani veri ile direkt işlem yapılır. Bu yöntemi komutlar üzerinde ayırt etmenin en önemli yolu komutların sonlarında büyük "I" olup olmadığına bakılmalıdır. Büyük "I" bulunduran komutlar

ivedi adresleme yöntemi kullanıyor demektir. Bu adresleme yöntemine birkaç örnek kod parçası:

<> **MVI A,50H** ;50H direkt veridir.

<> **ADI 50h** ;Akümülatör ile direkt veriyi toplar, sonucu akümülatöre atar.

3) Kaydedici Adresleme Yöntemi (Register addressing mode)

Kaydediciler arasındaki işlemlerin yapıldığı yöntemdir. Hızlı çalışan bir adresleme yöntemidir. Örnek olarak:

<> **MOV A,B** ; B kaydedicisindeki veriyi Akümülatöre kopyalar.

<> **INR B** ; B kaydedicisini bir arttırır.

4) Dolaylı Adresleme Yöntemi (Indirect addressing mode)

Dolaylı adresleme yöntemini kullanan komutlarda önce bir kaydedici çifti veya bellek bölgesi okunur ve okunan değerin belirttiği bellek bölgesindeki veri üzerinde işlem yapılır. Örnek olarak:

<> **MOV A, M** ; H-L çiftinin işaret ettiği bellek yerinin içeriğini akümülatöre taşınır

5) İmalı Adresleme Yöntemi (Implied addressing mode)

imal' adresleme yönteminde üzerinde işlem yapılacak olan kaydedici ima edilir. Örnek olarak:

**CMA ;** Bu komut akümülatörün içeriğinin tersi (tümleyenini) alır.

## Veri Transfer işlemleri (kopyalama)

ı. D500 adresinin içeriğini Akümülatöre kopyala 1. **LDA D500H**

2. Akümülatör içeriğini 95FF adresine kopyala 2. **STA 95FFH**

3. 8-bitlik 3E sayısını register B'ye **Yükle** 3. **MVI B, 3EH**

4. Register B den Register A ya **Kopyala** 4. **MOV A,B**

s. 16-bitlik 2040 sayısını HL Register çiftine **Yükle** 5. **LXI H, 2040H**

6. Register B den Memory Adres 2040'a **Kopyala**  6. **MOV M,B**

7. Input/Output Portları ve Akümülator arası **Kopyala** 7. **OUT 01H**

LDA : Hafızadaki içerik Akümülatöre - **LDA D500H**

STA : Akümülatördeki içerik Hafızaya - **STA 95FFH**

## Aritmetik işlemler

ı. 8-bitlik 32H sayısını Akumulator'e **EKLE** ı**. ADI 32H**

2. B Register içeriğini Akumulator'e **EKLE**  2. **ADD B**

3. 8-bitlik 32H sayısını Akumulator'den **ÇIKAR**  3. **SUI 32H**

4. C Register içeriğini Akumulator'den **ÇIKAR**  4. **SUB C**

s. D Register içeriğini 1 **Artır**  5. **INR D**

6. E Register içeriğini 1 **AZALT**  6**. DCR E**

## Mantıksal işlemler

1. H Registerını Akümülatör ile AND' le 1. **ANA H**

2. L Registerını Akumulator ile OR'la 2. **ORA L**

3. B Registerını Akumulator ile XOR'la 3. **XRA B**

4. C Register içeriğini Akumulator ile karşılaştır 4. **CMP C**

s. Accumulatorün tümleyenini al 5. **CMA**

## Dallanma işlemleri

1. Carry flag değeri SET (Elde değeri 1) ise 16bitlik 2080H adresine **DALLAN**(atla) 1**. JC 2080H**

2. Koşulsuz Dallanma (Unconditional jump) 2. **JMP 2050H**

3. 16-bit Adreslik altyordamı çağır (CALL) 3. **CALL 3050H**

4. ÇAĞRI'dan (Call) geri dön 4. **RET**

5. Carry flag değeri RESET(0) ise 16-bit Adreslik altyordamı çağır 5. **CNC 3050H**

6. Zero flag değeri SET(1) ise geri dön 6. **RZ**

|  |  |
| --- | --- |
| Elde bayrak biti (Carry flag - CY): Aritmetik bir işlem sonucunda elde oluşması durumunda `CY' bayrağı ‘1’ yapılırken, elde oluşmazsa '0' yapılır. | Sıfır (0) bayrak biti (Zero flag - Z): Yürütülen bir komut sonunda, işlenenin bulunduğu akümülatördeki veya bir kaydedicideki sayı 0 olursa, ` 1 ' yapılır. |

## Makine Kontrol işlemleri

1. Programın çalışmasını sonlandır (durdur) 1. **HLT**

2. Herhangi bir işlem yapma 2. **NOP**

# Örnek Programlar

Örnek 1. D ve E registerlarındaki iki sayıyı toplayarak sonucu C kaydedicisine aktarınız.



Çözüm **MOV A, D** ; A=D

**ADD E** ; A=A+E

**MOV C, A** ; C=A (D+E)

**HLT**  ; Sonlandır

|  |
| --- |
| MVI 4 8 bitlik veri hedef register'a veya hafızaya kaydedilir. Operand hafıza bölgesi ise, bu bölgenin adresi HL register'ları tarafından gösterilir. |

**Örnek 2**. 32H verisini 6000H hafıza adresine yükleyiniz.

**ÇÖZÜM 1;**

<> **MVI A, 32H** : 32H verisini akümülatöre yükle

<> **STA 6000H** : Akümülatör içeriğini 6000H adresine kopyala

<> **HLT**  : Program yürütmesini sonlandır

|  |
| --- |
| LXI -> register çiftine yükler .  B,D,H yazılır bu da BC, DE, HL demektir. |

**Çözüm 2.**

<>**LXI H, 6000H** : HL'ye 6000H yaz

<> **MVI M, 32H** : HL register çifti tarafından gösterilen hafıza (6000H) konumuna

32H verisini yükle (M: hafızaya-RAM'e)

<> **HLT** : Program yürütmesini sonlandır

**Örnek 3.** 3000H ve 5000H hafıza bölgelerinin içeriklerini yer değiştiriniz.

Çözüm 1.

<> **LDA 3000H** : 3000H bellek konumunun içeriğini akümülatör içine al

<> **MOV B, A** : içeriği B registerına kaydet

<> **LDA** **5000H** : 5000H hafıza konumunun içeriğini akümülatör içine al

<> **STA** **3000H** : Akümülatörün içeriğini 3000H adresine yükle

<> **MOV A, B** : Kayıtlı içeriği tekrar A akümülatörüne al

<> **STA 5000H** : Akümülatörün içeriğini 5000H adresine yükle

<> **HLT**  : Program yürütmesini sonlandır.

**Örnek 3.** 3000H ve 5000H hafıza bölgelerinin içeriklerini yer değiştiriniz.

Çözüm2.

<> **LXI H 3000H** : HL çiftine 300011'1 yükler, yani HL register çiftini 3000H hafıza konumuna işaretçi olarak atar.

<> **LXI D 5000H** : DE çiftine 5000H't yükler, yani...

<> **MOV B, M** : 3000H hafıza bölgesinin içeriğini B registerına al (M->H)

<> **LDAX D** : 5000H hafıza bölgesinin içeriğini A registerınaa al.

<> **MOV M, A** : A register içeriğini 3000H hafıza konumuna kaydet.

<> **MOV A, B** : B registerının içeriğini akümülatöre kopyala.

<> **STAX D** :A registerının içeriğini 5000H hafıza konumuna kaydet.

<> **HLT**  : Program yürütmesini sonlandır.

|  |
| --- |
| **LXI ->**  register çiftine yükler. B,D,H yazılır bu da **BC, DE, HL** demektir |
| **LDAX ->**  Operand kısmında register çifti hafıza bölgesini gösterir. Bu komut belirtilen 16 bitlik hafıza adresinin içeriği akümülatöre kopyalar. Kaynağın içeriği değiştirilmez. |
| **STAX- >** Akümülatörün içeriği, register çifti tarafından belirtilen adrese kopyalanır |