

# BM-311 Bilgisayar Mimarisi

---

Hazırlayan: M.Ali Akcayol  
Gazi Üniversitesi  
Bilgisayar Mühendisliği Bölümü



## Konular

---

- Makine komutu karakteristikleri
- Operand türleri
- İşlem türleri
- Assembly dili

## Makine komutu karakteristikleri

- İşlemcinin yapacağı iş makine komutlarıyla belirlenir.
- İşlemcinin tüm komutlarına **komut kümesi (instruction set)** denir.

### Makine komutunun bileşenleri

- **Operation code:** Kısaca **opcode** olarak adlandırılır ve gerçekleştirilecek işlemi belirler.
- **Source operand(s):** Gerçekleştirilecek işlem için girişleri belirler.
- **Result operand(s):** İşlem sonucu oluşan çıkışları belirler.
- **Next instruction:** Şu anda kullanılan instruction'dan sonra hangi instruction'a geçileceğini belirler.

## Makine komutu karakteristikleri

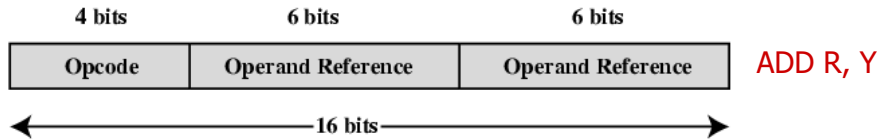
Source ve result operand'lar aşağıdakilerden birisi olabilir:

- **Memory:**  
Hafızada bir adres referans gösterilebilir.
- **CPU register:**  
CPU içindeki register'lar referans gösterilebilir.
- **I/O device:**  
Bir I/O cihazı referans olarak gösterilebilir.

## Makine komutu karakteristikleri

### Komut gösterimi

- Bir komut birden çok farklı türdeki alandan oluşabilir.



- Bir çok komut kümesi birden fazla farklı formata sahiptir.
- Kısaltmalar kullanılarak opcode'ların ifade edilmesi **mnemonics** olarak adlandırılır.

ADD	Add
SUB	Subtract
MPY	Multiply
DIV	Divide
LOAD	Load data from memory
STOR	Store data to memory

## Makine komutu karakteristikleri

### Komut türleri

- **Data processing:** Aritmetik ve mantık komutlar
- **Data storage:** Hafıza işlemleri
- **Data movement:** I/O komutları
- **Kontrol:** Test ve atlama komutları

## Makine komutu karakteristikleri

### Adres sayısı

- Aritmetik ve mantık komutlar daha çok adrese ihtiyaç duyar.
- **0, 1, 2 ve 3 operandlı komutlar** kullanılabilir.
- Aşağıda farklı operand sayıları için işlemler görülmektedir.

Adres sayısı	Sembolik gösterim	İşlem
3	OP A, B, C	$A \leftarrow B \text{ OP } C$
2	OP A, B	$A \leftarrow A \text{ OP } B$
1	OP A	$AC \leftarrow AC \text{ OP } A$
0	OP	$T \leftarrow (T - 1) \text{ OP } T$

## Makine komutu karakteristikleri

### Adres sayısı - devam

- Operand sayısına göre işlemlerdeki sıralama ve esneklik değişir.
- **$Y = (A - B) / (C + (D \times E))$**  işlemi **3, 2, 1 ve 0 operand** kullanılarak gerçekleştirilebilir.

### 3 operand

Komut	Açıklama
SUB Y, A, B	$Y \leftarrow A - B$
MPY T, D, E	$T \leftarrow D \times E$
ADD T, T, C	$T \leftarrow T + C$
DIV Y, Y, T	$Y \leftarrow Y / T$

#### Kısıtlar:

-İşlem sonucunda eşitliğin sağındaki operand'ların değeri değişmeyecek.  
-İşlem sonucu eşitliğin solundaki operand'a aktarılacak.

Interrupt kaç kez kontrol edilir?

## Makine komutu karakteristikleri

Adres sayısı - devam

$$Y = (A - B) / (C + (D \times E))$$

2 operand

Komut	Açıklama
MOVE Y, A	$Y \leftarrow A$
SUB Y, B	$Y \leftarrow Y - B$
MOVE T, D	$T \leftarrow D$
MPY T, E	$T \leftarrow T \times E$
ADD T, C	$T \leftarrow T + C$
DIV Y, T	$Y \leftarrow Y / T$

Interrupt kontrol sayısı?

## Makine komutu karakteristikleri

Adres sayısı - devam

$$Y = (A - B) / (C + (D \times E))$$

1 operand

Komut	Açıklama
LOAD D	$AC \leftarrow D$
MPY E	$AC \leftarrow AC \times E$
ADD C	$AC \leftarrow AC + C$
STOR Y	$Y \leftarrow AC$
LOAD A	$AC \leftarrow A$
SUB B	$AC \leftarrow AC - B$
DIV Y	$AC \leftarrow AC / Y$
STOR Y	$Y \leftarrow AC$

Interrupt kontrol sayısı?

## Makine komutu karakteristikleri

$$Y = (A - B) / (C + (D \times E))$$

0 operand

Komut	Açıklama
PUSH A	$T \leftarrow A$
PUSH B	$T \leftarrow B, (T - 1) \leftarrow A$
SUB	$(T) \leftarrow (T - 1) - T$
PUSH D	$T \leftarrow D$
PUSH E	$T \leftarrow E$
MPY	$(T) \leftarrow (T - 1) * T$
PUSH C	$T \leftarrow C$
ADD	$(T) \leftarrow (T - 1) + T$
DIV	$(T) \leftarrow (T - 1) / T$
POP Y	$Y \leftarrow T$

Interrupt kontrol sayısı?

## Makine komutu karakteristikleri

### Adres sayısı arttıkça

- Daha karmaşık ve güçlü komutlar
- Operand'lar register adresleme yapıyorsa daha çok register
- Programlarda daha az sayıda komut

### Adres sayısı azaldıkça

- Daha basit komutlar
- Programlarda daha çok sayıda komut
- Komutların daha hızlı fetch/execute yapılması

## Makine komutu karakteristikleri

### Komut kümesi tasarımı

- **Operation repertoire:** Kaç tane işlem olduğunu ve hangi işlemlerin yapılacağını belirler.
- **Data types:** Komutların işlem yapacağı veri türlerini belirler.
- **Instruction format:** Komut boyutu (bit), adres sayısı, field boyutunu belirler.
- **Registers:** Komutların kullanacağı CPU register'larını belirler.
- **Addressing:** Komutlardaki operand'ların adresleme modlarını belirler.

## Konular

- Makine komutu karakteristikleri
- **Operand türleri**
- İşlem türleri
- Assembly dili

## Operand türleri

- Temel operand türleri aşağıdaki gibidir:
  - **Adresler:** Adresleme modlarıyla operand'ların adresleri belirlenir.
  - **Sayılar:** Integer, fixed point, floating point, packet decimal (her digit 4-bit) olabilir.
  - **Karakterler:**
    - ASCII (American Standard Code for Information Interchange) (Her karakter 7-bit, 128 karakter, 8.bit parity bit),
    - EBCDIC (Extended Binary Coded Decimal Interchange Code) (8-bit kod kullanır)
  - **Mantıksal data:** 1-bit data 0 veya 1'ler ile ifade edilir.

## Konular

- Makine komutu karakteristikleri
- Operand türleri
- **İşlem türleri**
- Assembly dili




## İşlem türleri

- Temel işlem türleri aşağıdaki gibidir:
  - **Data transfer:** Veri aktarım komutları kullanılır (Load, Store).
  - **Arithmetic:** Aritmetik mantık komutlar kullanılır (Add, Sub).
  - **Logical:** Mantıksal işlem komutları kullanılır (And, Or, ...).
  - **Conversion:** Formatlar arasında dönüştürme yapan komutlar kullanılır (Translate, Convert, ...)
  - **Input/Output:** Giriş/çıkış cihazlarıyla veri alışverişi yapan komutlar kullanılır (Input, Output, ...).
  - **System control:** İşletim sistemi fonksiyonları kullanılır (user/supervisor mode, control/status register change, ...).
  - **Transfer of control:** Şartlı ve şartsız atlama komutları kullanılır (Jump, Branch, ...).

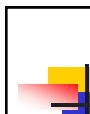
## İşlem türleri

Type	Operation Name	Description
Data Transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
Arithmetic	Pop	Transfer word from top of stack to destination
	Add	Compute sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand



## İşlem türleri

Type	Operation Name	Description
Logical	AND	Perform logical AND
	OR	Perform logical OR
	NOT (complement)	Perform logical NOT
	Exclusive-OR	Perform logical XOR
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class of instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
	Rotate	Left (right) shift operand, with wraparound end



## İşlem türleri

Transfer of Control	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition
	Jump to Subroutine	Place current program control information in known location; jump to specified address
	Return	Replace contents of PC and other register from known location
	Execute	Fetch operand from specified location and execute as instruction; do not modify PC
	Skip	Increment PC to skip next instruction
	Skip Conditional	Test specified condition; either skip or do nothing based on condition
	Halt	Stop program execution
	Wait (hold)	Stop program execution; test specified condition repeatedly; resume execution when condition is satisfied
	No operation	No operation is performed, but program execution is continued

İşlem türleri		
Input/Output	Input (read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register)
	Output (write)	Transfer data from specified source to I/O port or device
	Start I/O	Transfer instructions to I/O processor to initiate I/O operation
	Test I/O	Transfer status information from I/O system to specified destination
Conversion	Translate	Translate values in a section of memory based on a table of correspondences
	Convert	Convert the contents of a word from one form to another (e.g., packed decimal to binary)

İşlem türleri	
■ Data transfer	<ul style="list-style-type: none"> <li>• Data transfer komutları <b>kaynak ve hedef operand'ları</b> belirler.</li> <li>• Aktarılacak <b>verinin boyutu</b> belirlenir.</li> <li>• Operand'ların <b>adresleme modları</b> belirlenir.</li> <li>• Kaynak ve operand'lar register ise CPU register'lar arasında veri aktarımı yapar.</li> <li>• Eğer operand'lar hafızada ise <b>adresleme moduna göre adres hesaplanır.</b></li> <li>• Hesaplanan adres değerine göre <b>önce cache belleğe bakılır</b>, yoksa hafızadan alınır.</li> </ul>

## İşlem türleri

### ■ Arithmetic

- Çoğu mikroişlemci temel aritmetik işlemleri sağlar (**add, subtract, multiply, divide**).
- Diğer aritmetik işlemler ise aşağıdakiler olabilir:
  - **Absolute:** Operand'ın mutlak değeri alınır.
  - **Negate:** Operand'ın işareti terslenir.
  - **Increment:** Operand'a 1 eklenir.
  - **Decrement:** Operand'dan 1 çıkartılır.

## İşlem türleri

### ■ Logical

- Boolean operatörlere göre operand'lara mantıksal işlemler (**OR, AND, XOR, NOT**) uygulanır.
- Aşağıda AND ve XOR işlemleri verilmiştir.  
(R1) = 10100101  
(R2) = 00001111  
(R1) **AND** (R2) = 00000101  
  
(R1) = 10100101  
(R2) = 11111111  
(R1) **XOR** (R2) = 01011010

## İşlem türleri

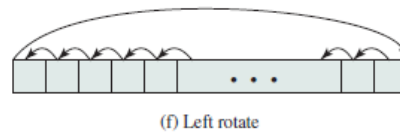
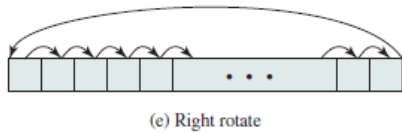
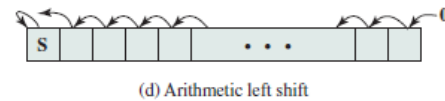
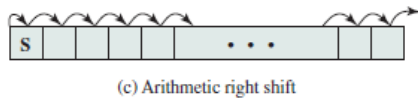
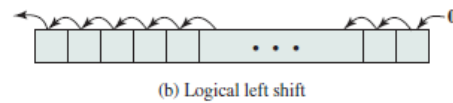
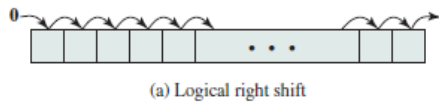
### ■ Logical - devam

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P=Q
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

## İşlem türleri

### ■ Logical - devam

- Bitwise logical işlemlerin yanında bir çok mikroişlemcide shift ve rotate işlemleri de yapılabilir.



## İşlem türleri

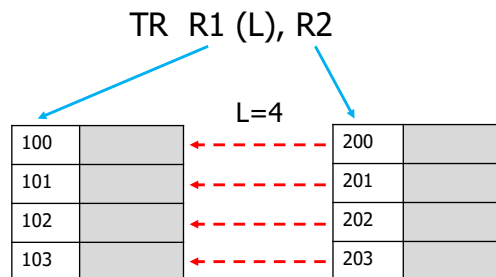
### ■ Logical - Örnek

Input	Operation	Result
10100110	Logical right shift (3 bits)	00010100
10100110	Logical left shift (3 bits)	00110000
10100110	Arithmetic right shift (3 bits)	11110100
10100110	Arithmetic left shift (3 bits)	10110000
10100110	Right rotate (3 bits)	11010100
10100110	Left rotate (3 bits)	00110101

## İşlem türleri

### ■ Conversion

- **Bir formattan diğer formata dönüştürme işlemi yapılır.**
- IBM EAS/390 makinesindeki aşağıdaki komut, R2 adresindeki tabloya göre, R1 adresinden başlayıp L adet dönüştürme yapar.
- R1 adresinden başlar, L adet adresin içeriğini, R2 adresinden başlayarak değiştirir.



## İşlem türleri

- Input / Output
  - I/O cihazlarından **veri almak** veya I/O cihazlarına **veri yazmak için kullanılan komutlardır.**
  - Okunan veri hafızaya veya register'a yazılabilir.
- System Control
  - **İşletim sistemi tarafından kullanılan komutlardır.**
  - Örn: Multiprogramming sistemlerde PCB (Process Control Block) içeriğine erişilir.
  - Yapılan işlemler, **global olarak memory cache aktif/pasif, global olarak write-back/write-through aktif/pasif, ...**

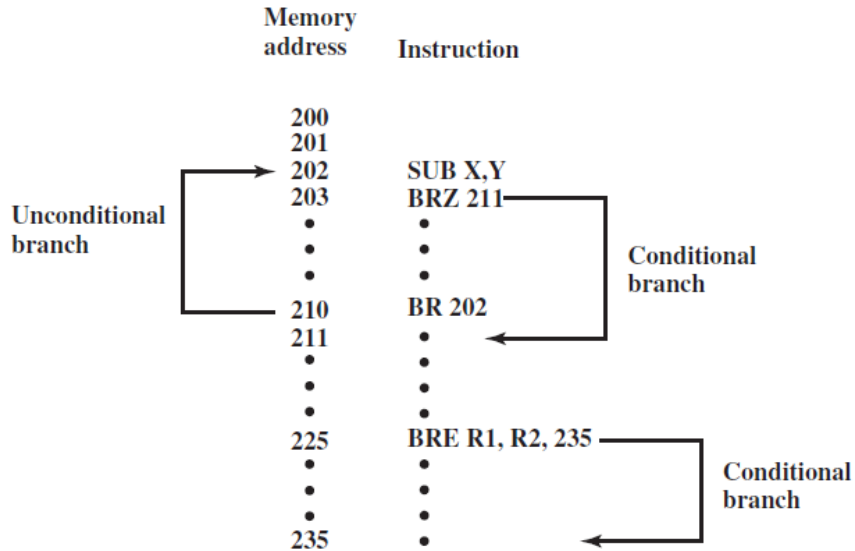
## İşlem türleri

- Transfer of Control
  - Programlar çalışırken her zaman komutlar sıralı çalışmazlar.
  - Programların çalışma akışını değiştiren komutlardır.
  - Bir çok makinede **branch, skip** ve **procedure call** komutları bulunmaktadır.
  - **Branch komutları, şartlı (conditional) veya şartsız (unconditional)** atlama yapar.
  - Son yapılan işlem (add, sub, div, ..) sonucuna göre **ilgili bayrak bitine göre** aşağıdaki atlamalar yapılabilir:

BRP X	; Sonuç pozitifse X adresine atla
BRN X	; Sonuç negatifse X adresine atla
BRZ X	; Sonuç sıfırsa X adresine atla
BRO X	; Overflow oluşmuşsa X adresine atla
BRE R1, R2, X	; R1 = R2 ise X adresine atla

## İşlem türleri

### Transfer of Control - Branch



## İşlem türleri

### Transfer of Control - Skip

- Skip komutları PC değerini 1 artırır ve sonraki komut yerine ondan sonraki komutu çalıştırır.

```

301
:
309 ISZ R1
310 BR 301
311
    
```

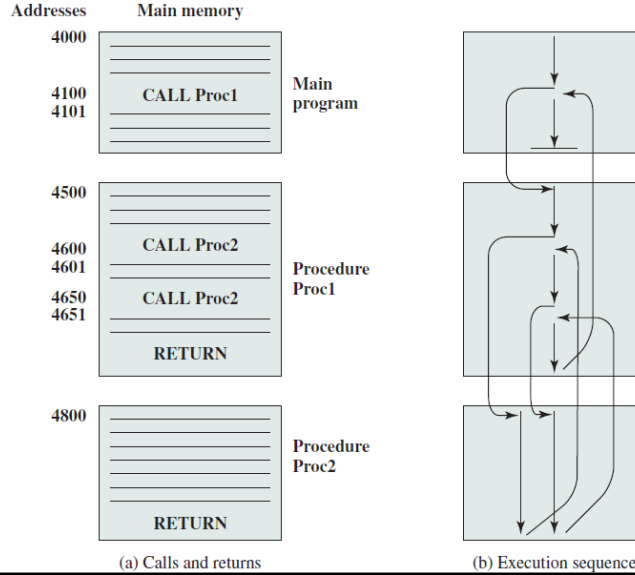
- **ISZ** (Increment and skip if zero), **Z biti 1 ise sonraki komutu atlar.**
- R1 negatif bir değerden başlar 0 olana kadar artırılır.
- R1 = 0 olduğunda BR 301 komutu atlanır ve 311 adresine geçilir. Böylece döngüden çıkmış olur.



## İşlem türleri

### Transfer of Control - Procedure call

- Fonksiyonlar programlarda modülerlik sağlar.



## İşlem türleri

### Transfer of Control - procedure call

- Prosedür çağrılarında dönüş adresiyle birlikte giriş/çıkış parametrelerinin de gönderilip alınması gerekir.
- Register, start of procedure** ve **top of stack** yöntemleri kullanılır.

#### Register

$$RN \leftarrow PC + \Delta$$

$$PC \leftarrow X$$

- Burada,  $\Delta$  komut uzunluğuna eşittir.
- RN, dönüş adresinin saklandığı register'dır.
- X, atlanacak adrestir.
- İç içe çağrılarda tek register ile sorun oluşur daha fazla register gereklidir.**

## İşlem türleri

### Transfer of Control - procedure call

#### Start of procedure

$$X \leftarrow PC + \Delta$$

$$PC \leftarrow X + 1$$

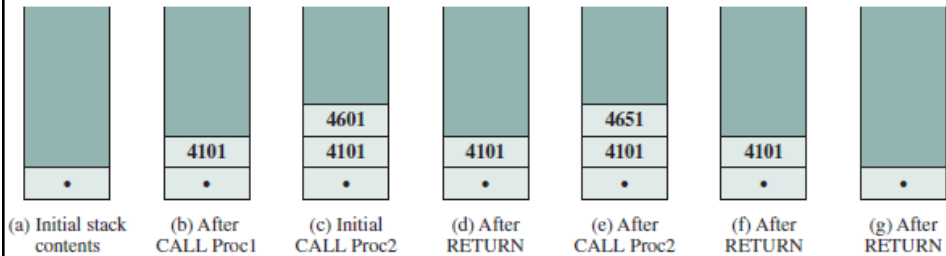
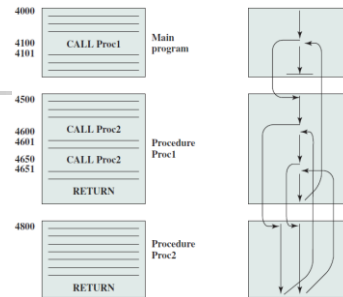
- Burada,  $\Delta$  komut uzunluğuna eşittir.
- X, procedure başlangıç adresidir.
- X + 1 atlanacak adrestir.
- Recursive çağırımlarda sorun oluşur.**

## İşlem türleri

### Transfer of Control - procedure call

#### Top of stack

- Hafızada oluşturulan bir alanda stack yapısı oluşturulur.



- İç içe ve recursive çağırımlarda sorun olmaz.**
- Stack içerisinde her çağırma için bir frame oluşturularak parametrelerde aktarılabilir.**



## Konular

---

- Makine komutu karakteristikleri
- Operand türleri
- İşlem türleri
- **Assembly dili**

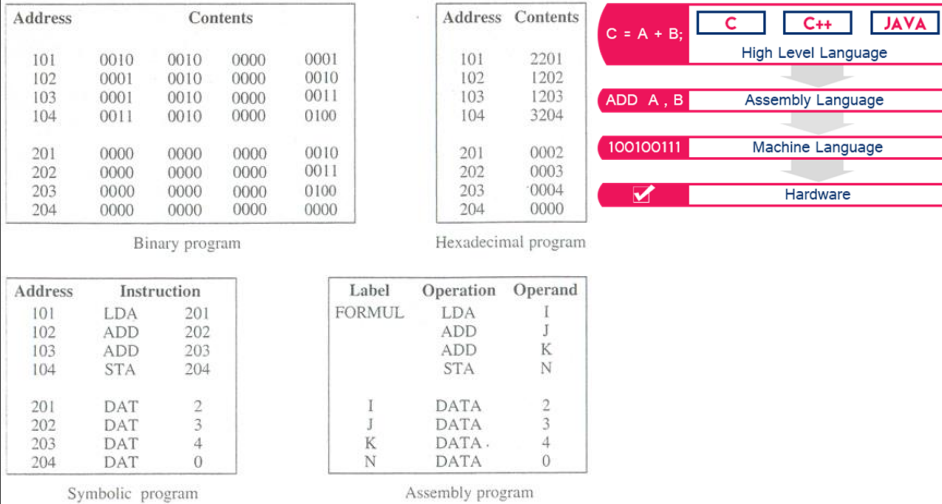


## Assembly dili

---

- CPU makine komutlarını anlar ve çalıştırır.
- Makine komutları binary ifade edilir.
- **Makine komutlarıyla program yazmak ve hata kontrolü çok zordur.**
- Binary sayılar yerine hexadecimal sayılar kullanılabilir.
- Hexadecimal sayılarda program yazmak yine de zordur.
- Hexadecimal komutlar yerine kısaltılmış ve daha kolay anlaşılabilen sembolik ifadeler (**mnemonics**) kullanılabilir (**Sembolik Programlama**).
- Sembolik programlamada program hafızaya tekrar yüklendiğinde tüm adreslerin update edilmesi gerekir.
- **Assembly programlama**, adresleri etiketlerle göstererek programın tekrar yüklenmesinde veya güncelleştirilmesinde kolaylık sağlamıştır.

## Assembly dili



## Ödev

- Her satırı 16 bit olan bir hafızada 1000-1999 adresleri arasında yer alan 1000 adet sırasız tamsayı dizisini kabarcık sıralaması (bubble sort) algoritmasıyla yerinde sıralayan bir programı aşağıdaki komut kümesini kullanarak yazınız.

Komut kümesi			
<b>ADD</b>	topla	<b>LDA</b>	register'a yükle
<b>SUB</b>	çıkart	<b>STA</b>	hafızaya aktar
<b>MUL</b>	çarp	<b>PSH</b>	stack'e push
<b>DIV</b>	böl	<b>POP</b>	stack'ten pop
<b>MOV</b>	hafızada bir adresten diğerine kopyala (MOV K, L //L'deki değeri M'ye kopyalar.)		
<b>JZ</b>	sıfır ise atla	<b>JP</b>	pozitif ise atla
<b>JN</b>	negatif ise atla	<b>JMP</b>	şartsız atla

Not: Aritmetik işlemlerde iki operand bulunmaktadır.