

BİLGİSAYAR ORGANİZASYONU ve TASARIMI

DR. FATİH KELEŞ

Temel Bilgisayarın Programlanması

- ▶ Giriş
- ▶ Makine Dili
- ▶ Sembolik (Assembly) Dil
- ▶ Assemblerler
- ▶ Program Döngüleri
- ▶ Aritmetik ve Lojik İşlemlerin Programlanması
- ▶ Altprogramlar
- ▶ Giriş–Çıkış Programlama
- ▶ Kesme Servis Rutin Programı

Giriş

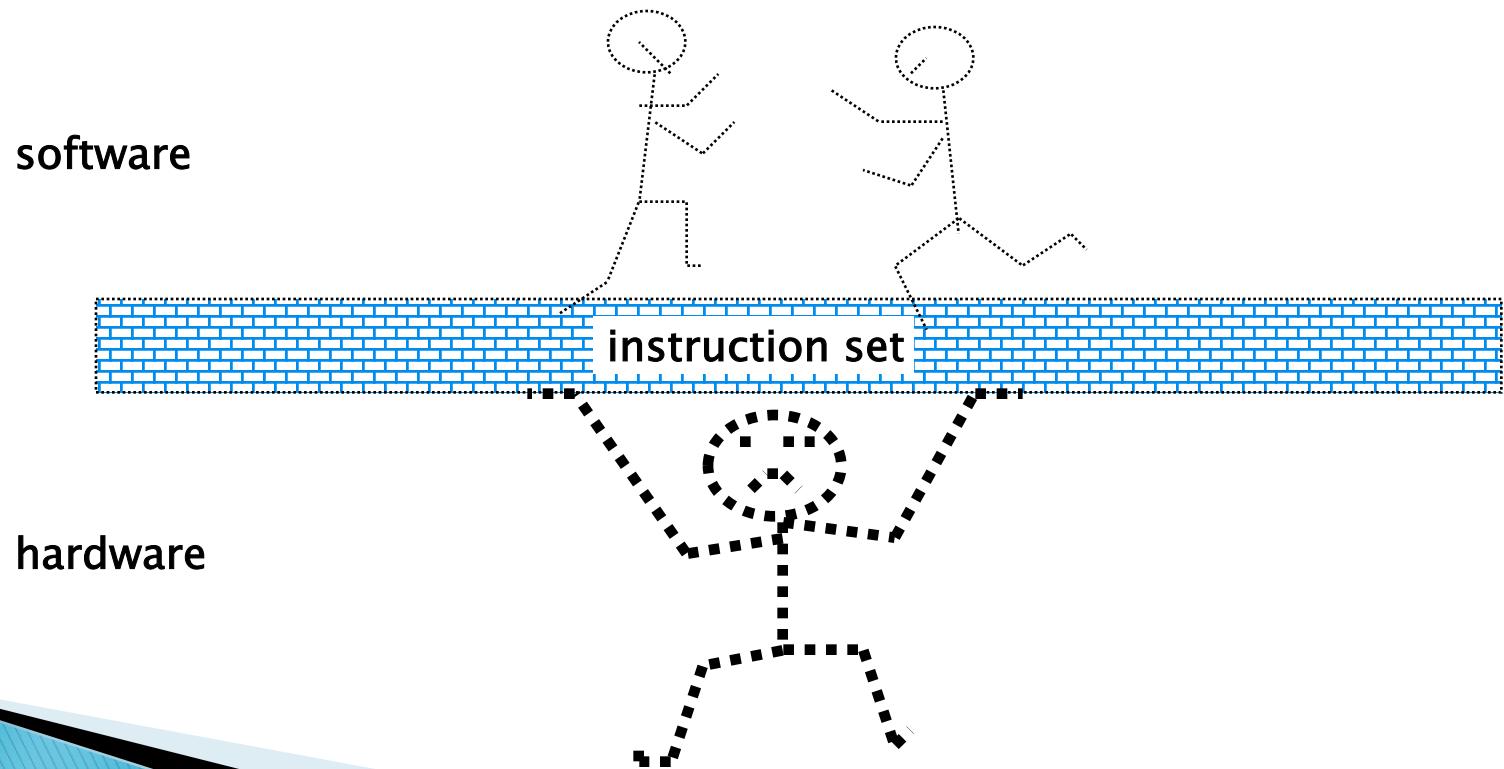
- ▶ Bilgisayar mimarisi hem donanım hem yazılım bilgisini içermelidir. Bilindiği gibi her iki alan da birbirlerini etkiler.
- ▶ Bu bölümde, temel programlama kavramları ve donanımsal gerçeklemeleri ile ilişkileri verilmektedir.
- ▶ Kullanıcı sembolik programı ikili kod karşılıklarına dönüştüren program (assembler) yapısı incelenecaktır.
- ▶ Makine Dili Programlama yapısı ve komutları TB organizasyonu açısından ele alınacaktır.

Komut Seti Mimarisi (ISA)

- ▶ ISA makine dili programcısının doğru program yapabilmesi için anlamak zorunda olduğu bilgisayarın yapısıdır.
- ▶ ISA ayrıca donanım tasarımcısının da doğru gerçekleme yapabilmesi için bilmek zorunda olduğu makine tanımlamalarıdır.

Komut Seti Mimarisi

- ▶ ISA yazılım ve donanım arasında bir arayüz olarak çalışır.



Makine Dili

- Program

Gerekli data işlem görevini yürütmek için, bilgisayarı yönlendiren ifadeler ve komutlar listesi

- Çeşitli Türlerde Programlama Dilleri

- Programlama Dilleri Hiyerarşik Düzeni

- Makine Dili (Bilgisayar donanımının tanıdığı dil)

- İkili Kod

- Sekizli (Octal) veya Onaltılı (hexadecimal) Kod

- Sembolik Dil (Assembler)

- Sembolik Kod (harfler, sayılar ve alfanümerik kar.)

Her sembolik komut ikili kod karşılığına dönüştürülür (assembly)

- Yüksek Seviyeli Dil (Compiler-Derleyici)

Problem çözüm odaklı komutlar. (Fortran, Pascal, C vs...)

Derleyici bu komutları ikili kod karşılıklarına dönüştürür.

Temel Bilgisayarın Komut Kümesi

m: Efektif adres

25 farklı komut

7 MRI komut,
(bellek ref.)

18 non-MRI komut
(saklayıcı veya I/O ref.)

M: Bellek kelimesi (operand)

Sembol	Hex Kodu	Açıklama
AND	0 or 8	AND M to AC
ADD	1 or 9	Add M to AC, carry to E
LDA	2 or A	Load AC from M
STA	3 or B	Store AC in M
BUN	4 or C	Branch unconditionally to m
BSA	5 or D	Save return address in m and branch to m+1
ISZ	6 or E	Increment M and skip if zero
CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate right E and AC
CIL	7040	Circulate left E and AC
INC	7020	Increment AC, carry to E
SPA	7010	Skip if AC is positive
SNA	7008	Skip if AC is negative
SZA	7004	Skip if AC is zero
SZE	7002	Skip if E is zero
HLT	7001	Halt computer
INP	F800	Input information and clear flag
OUT	F400	Output information and clear flag
SKI	F200	Skip if input flag is on
SKO	F100	Skip if output flag is on
ION	F080	Turn interrupt on
IOF	F040	Turn interrupt off

Programlama Dillerinin Karşılaştırılması

- İki Sayıyı Toplamak için İkili Program

Adres	Komut Kodları
0	0010 0000 0000 0100
1	0001 0000 0000 0101
10	0011 0000 0000 0110
11	0111 0000 0000 0001
100	0000 0000 0101 0011
101	1111 1111 1110 1001
110	0000 0000 0000 0000

- Hex program

Adres	Komutlar
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

- Sembolik OP-kodlu Program

Adres	Komutlar	Açıklamalar
000	LDA 004	Load 1st operand into AC
001	ADD 005	Add 2nd operand to AC
002	STA 006	Store sum in location 006
003	HLT	Halt computer
004	0053	1st operand
005	FFE9	2nd operand (negative)
006	0000	Store sum here



- Sembolik Dil (Assembler) Program

```
ORG 0      /Origin of program is location 0
LDA A      /Load operand from location A
ADD B      /Add operand from location B
STA C      /Store sum in location C
HLT        /Halt computer
A, DEC 83  /Decimal operand
B, DEC -23 /Decimal operand
C, DEC 0   /Sum stored in location C
END        /End of symbolic program
```

- Fortran Program

```
INTEGER A, B, C
DATA A,83 / B,-23
C = A + B
END
```

Assembly Programı

Sayı1=80, Sayı2= 32

Toplam = Sayı1 + Sayı2

hesaplayan Assembly Programını yazınız.

Yer	İçerik	Assembly Programı
100	2104	ORG 100
101	1105	LDA Sy1
102	3106	ADD Sy2
103	7001	STA Top
		HLT
104	0050	Sy1, HEX 50 
105	0020	Sy2, HEX 20
106	0000	Top, HEX 0
		END

Assembly Dili

TB assembly dilinin özellikleri

Assembly dili prog. satırı alan (field) diye adlandırılan 3 bölümden oluşur.

Etiket Alanı

En fazla 3 karakterli (1. harf, 2–3 harf veya rakam) sembolik bir adres veya boş – virgül ile sonlandırılır

Komut Alanı

- Makine veya Sahte (Pseudo) komut belirler.
- Komut

1. Bellek Referanslı komut (MRI)

- MRI boşluklarla ayrılmış 2 veya 3 sembol içerir
ADD OPR (direk adresli komut MRI)
ADD PTR I (dolaylı adresli komut MRI)

2. Saklayıcı Referanslı veya giriş–çıkış komutları .

- Non-MRI bir adres alanı yoktur. CLA gibi...
- Sahte komutlar bir operanda sahip olabilir
- Komut alanında kullanılan sembolik adresler bir etiket olarak tanımlıdır.

Açıklama Alanı

Boş olabilir veya bir açıklama içerebilir.

/ işaretini asembler için açıklama alanı belirler.

Sahte (Pseudo) Komutlar

ORG N

N hex sayısı aşağıdaki satırda listelenen komut veya operand için bir bellek adresidir. Programın hangi bellek adresinden başlayacağını veya operandın alınacağı adresi belirler.

END Sembolik programın sonunu belirler

DEC N İşaretli onlu N sayısı ikili eşdeğerine dönüştürülür.

HEX N Hex N sayısı ikili eşdeğerine dönüştürülür.

Örnek: İki sayıyı çıkarmak için bir assembler programı

ORG 100	/ Origin of program is location 100
LDA SUB	/ Load subtrahend to AC
CMA	/ Complement AC
INC	/ Increment AC
ADD MIN	/ Add minuend to AC
STA DIF	/ Store difference
HLT	/ Halt computer
MIN,	DEC 83 / Minuend
SUB,	DEC -23 / Subtrahend
DIF,	HEX 0 / Difference stored here
	END / End of symbolic program

İkili Değere Dönüşüm (Assembling)



Hex Kod		Sembolik Program	
Adres	İçerik		
100	2107		ORG 100
101	7200		LDA SUB
102	7020		CMA
103	1106		INC
104	3108		ADD MIN
105	7001		STA DIF
106	0053	MIN, SUB, DIF,	HLT
107	FFE9		DEC 83
108	0000		DEC -23
			HEX 0
			END

Assembly Dilinin Derleme Prensibi

Assembly Dili iki inceleme ile makine diline, diğer bir ifade ile binary koduna çevrilmektedir:

- **Birinci İnceleme (First Pass):** Değişken isimleri ve hangi satırda geçtikleri ilk değerlerini aldıkları tespit edilir. (a,b,c gibi)
- **İkinci İnceleme (Second Pass):** Komutlar 0-1 karşılıklarına çevrilir.

Assembly Dilinin Derleme Prensibi

Assemblerler

Kaynak Program – Sembolik Assembly Dili Program
Obje Program – İkili Makine Dili Program

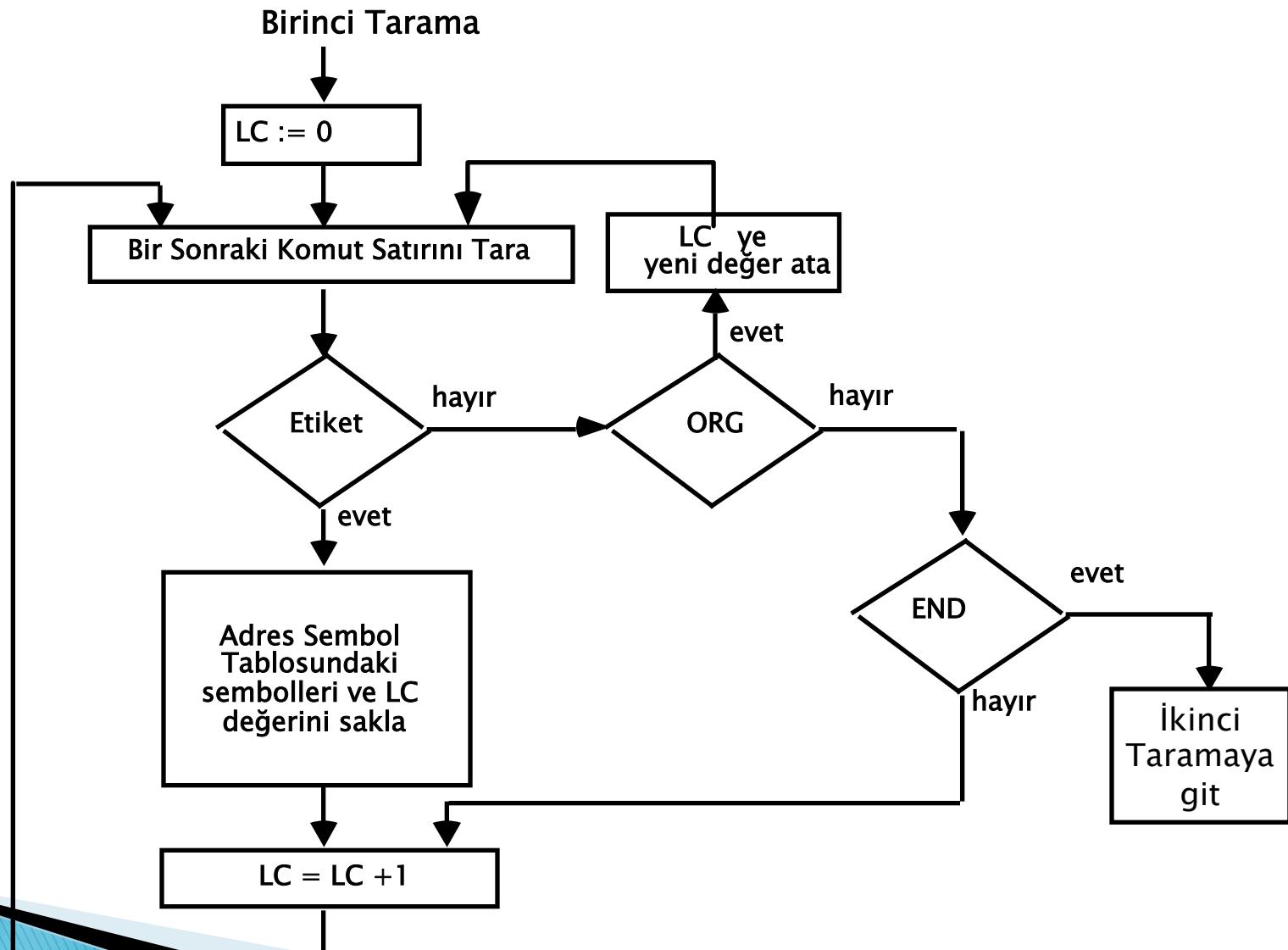
İki Taramalı Assemblerler

Birinci Tarama: Kullanıcının tanımladığı tüm semboller (adres) ikili eşdeğerleri ile ilişkilendiren bir tablo üretilir.

İkinci Tarama: ikili dönüşüm

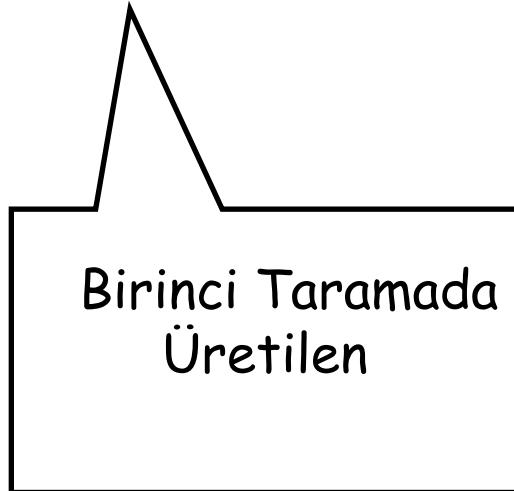
Assembler

Birinci Tarama



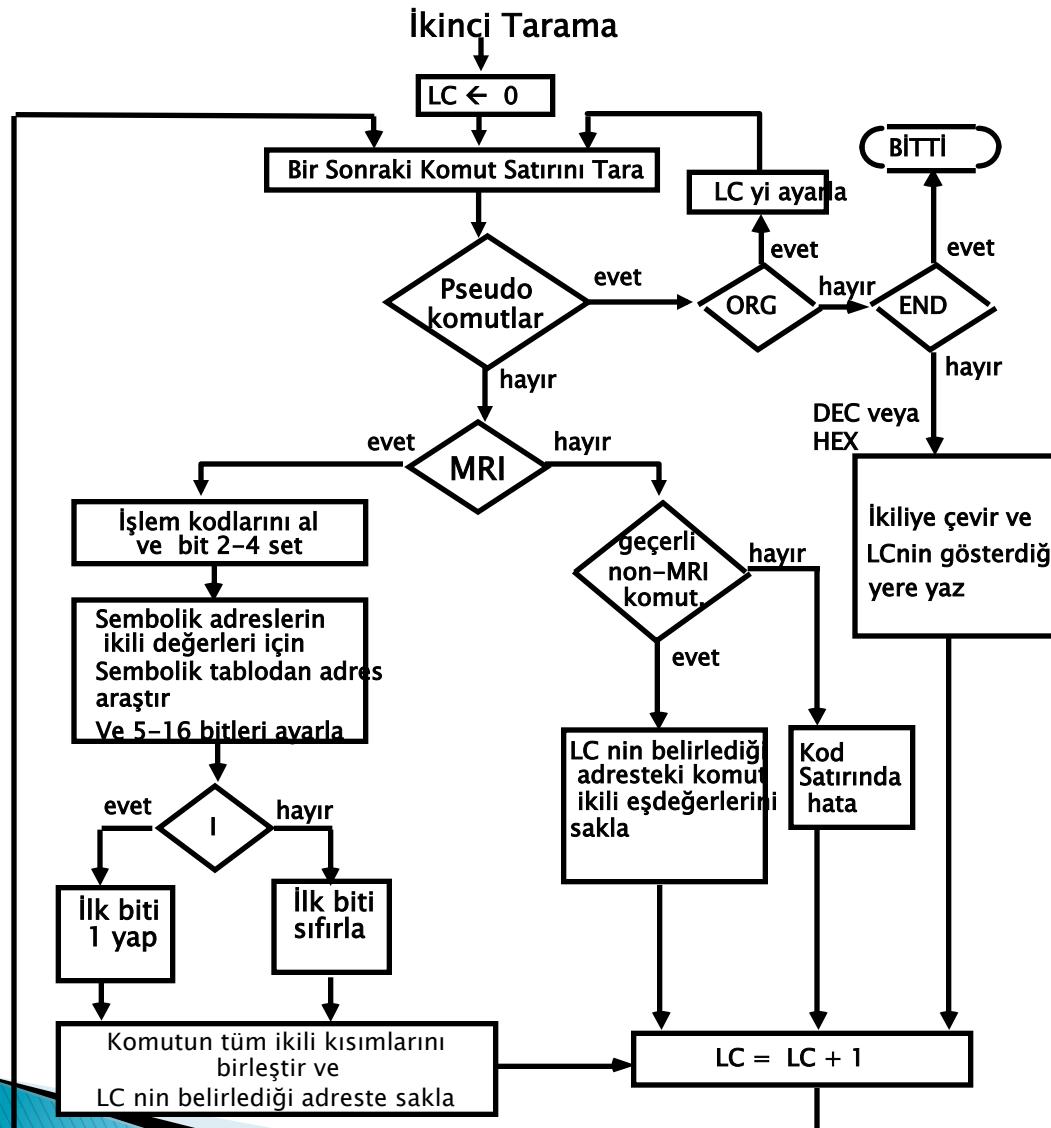
Makine Komutları LUT (look-up table) yöntemiyle dönüştürülür:

1. Sahte (Pseudo) Komut Tablosu,
2. MRI Tablosu (bellek ref. komutlar)
3. Non-MRI Tablosu (saklayıcı veya I/O ref.)
4. Adres Sembol Tablosu



Assembler

İkinci Tarama



Program Döngüleri

Döngü: Farklı bir data seti için, aynı programın pek çok kez icra edilmesi işlemi

100 sayısını toplayan bir FORTRAN programı:

```
DIMENSION A(100)
INTEGER SUM, A
SUM = 0
DO 3 J = 1, 100
3 SUM = SUM + A(J)
```

Program Döngüleri

100 sayısını toplamak için Assembly-dili program

	ORG 100	/ Origin of program is HEX 100
	LDA ADS	/ Load first address of operand
	STA PTR	/ Store in pointer
	LDA NBR	/ Load -100
	STA CTR	/ Store in counter
	CLA	/ Clear AC
LOP,	ADD PTR I	/ Add an operand to AC
	ISZ PTR	/ Increment pointer
	ISZ CTR	/ Increment counter
	BUN LOP	/ Repeat loop again
	STA SUM	/ Store sum
	HLT	/ Halt
ADS,	HEX 150	/ First address of operands
PTR,	HEX 0	/ Reserved for a pointer
NBR,	DEC -100	/ Initial value for a counter
CTR,	HEX 0	/ Reserved for a counter
SUM,	HEX 0	/ Sum is stored here
	ORG 150	/ Origin of operands is HEX 150
	DEC 75	/ First operand
	.	
	DEC 23	/ Last operand
	END	/ End of symbolic program

Aritmetik ve Lojik İşlemlerin Programlanması

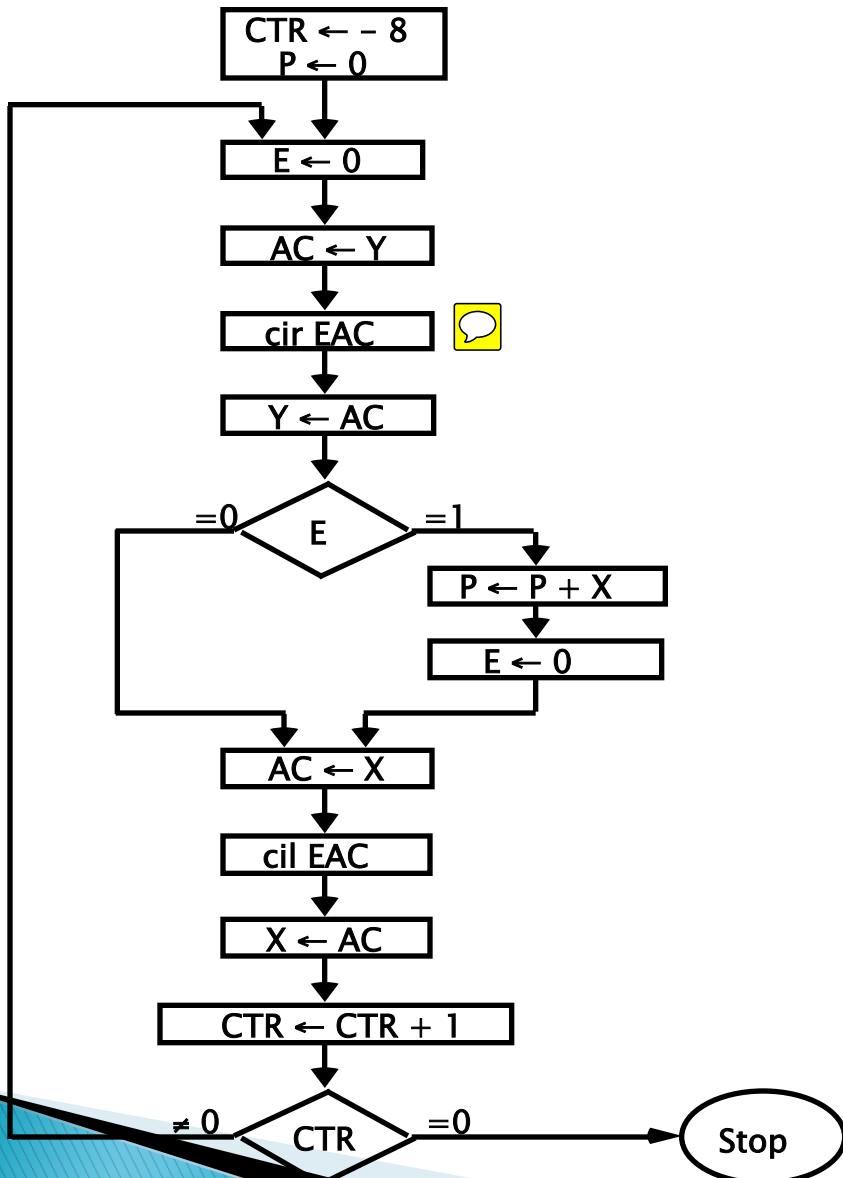
Aritmetik ve Lojik İşlemlerin Gerçeklenmesi

- **Yazılımsal Gerçekleme**
 - Makine Komut Kümesini kullanarak bir program ile bir işlemin gerçekleşmesi
 - Genellikle işlem komut kümesinde yer almıyorsa yazılımsal olarak gerçekleşir.
- **Donanımsal Gerçekleme**
 - Bir makine komutu olarak bilgisayarda bir işlemin gerçekleşmesidir.

Yazılımsal Gerçekleme Örneği:

- **Çarpma**
 - Basitlik açısından sayılar işaretsiz pozitif olsunlar.
 - 8-bit sayılar → 16-bit çarpım sonucu

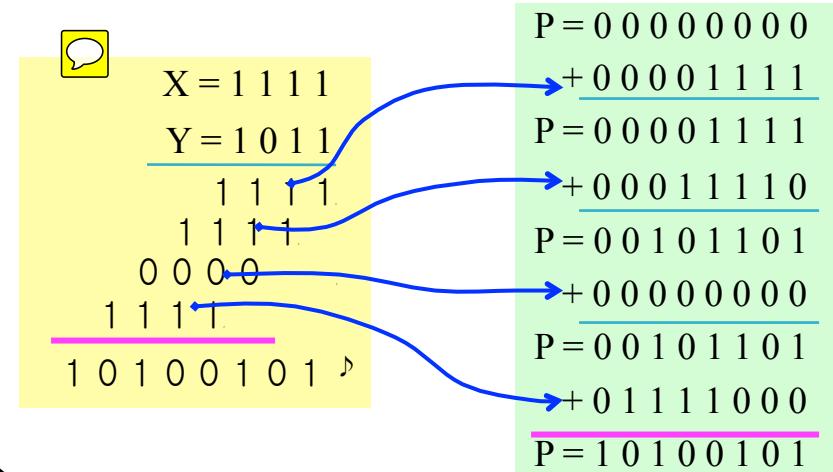
Çarpma Programı Akış Şeması



X çarpılanı tutar
 Y çarpanı tutar
 P kısmi çarpım terimlerini tutar

4 anlamlı bit taşıyan X ve Y sayılarının Çarpımı

$$\begin{array}{r}
 X = 0000\ 1111 \\
 Y = 0000\ 1011 \\
 \hline
 & P \\
 & 0000\ 0000 \\
 & 0000\ 1111 \\
 & 0001\ 1110 \\
 & 0000\ 0000 \\
 & 0111\ 1000 \\
 \hline
 & 1010\ 0101
 \end{array}$$



Assembly Dili Programı

Çarpma

	ORG 100	
LOP,	CLE	/ Clear E
	LDA Y	/ Load multiplier
	CIR	/ Transfer multiplier bit to E
	STA Y	/ Store shifted multiplier
	SZE	/ Check if bit is zero
	BUN ONE	/ Bit is one; goto ONE
	BUN ZRO	/ Bit is zero; goto ZRO
ONE,	LDA X	/ Load multiplicand
	ADD P	/ Add to partial product
	STA P	/ Store partial product
	CLE	/ Clear E
ZRO,	LDA X	/ Load multiplicand
	CIL	/ Shift left
	STA X	/ Store shifted multiplicand
	ISZ CTR	/ Increment counter
	BUN LOP	/ Counter not zero; repeat loop
	HLT	/ Counter is zero; halt
CTR,	DEC -8	/ This location serves as a counter
X,	HEX 000F	/ Multiplicand stored here
Y,	HEX 000B	/ Multiplier stored here
P,	HEX 0	/ Product formed here
	END	



Assembly Dili Programı

Lojik ve Öteleme İşlemleri

• Lojik İşlemler

- Temel Bilg. komutları : AND, CMA, CLA
- OR işlemi için program

LDA	A	/ Load 1st operand
CMA		/ Complement to get A'
STA	TMP	/ Store in a temporary location
LDA	B	/ Load 2nd operand B
CMA		/ Complement to get B'
AND	TMP	/ AND with A' to get A' AND B'
CMA		/ Complement again to get A OR B

De Morgan theorem

• Öteleme İşlemleri- TB. Komutu: CIR, CIL

- Lojik Sağa Ötelemeişlemi

CLE
CIR

- Lojik Sola Öteleme işlemi

CLE
CIL

- Aritmetik sağa öteleme işlemi

CLE	/ Clear E to 0
SPA	/ Skip if AC is positive
CME	/ AC is negative
CIR	/ Circulate E and AC

Altprogram

Altprogram

- Bir programda pek çok kez kullanılabilen ortak komut kümesi.
- Altprogram Bağlantısı: Altprograma dallanma ve ana programa geri dönme prosedürü

Örnek:

Loc.			
100		ORG 100	/ Main program
101		LDA X	/ Load X
102		BSA SH4	/ Branch to subroutine
103		STA X	/ Store shifted number
104		LDA Y	/ Load Y
105		BSA SH4	/ Branch to subroutine again
106		STA Y	/ Store shifted number
107	X,	HLT	
108	Y,	HEX 1234	
109	SH4,	HEX 0	/ Subroutine to shift left 4 times
10A		CIL	/ Store return address here
10B		CIL	/ Circulate left once
10C		CIL	
10D		CIL	/ Circulate left fourth time
10E		AND MSK	/ Mask lower 4 bit
10F		BUN SH4 I	/ Return to main program
110	MSK,	HEX FFF0	/ Mask operand
		END	

Altprogram Parametreleri ve Data Bağlantıları

Ana program ve altprogram arasındaki data ve parametre bağlantıları
– Saklayıcılar
– Bellek Kelimeleri
üzerinden kurulur.

Örnek: Lojik OR işlemi altprogram bağlantısı 2 parametre ile sağlanması

Loc.		
200	ORG 200	
201	LDA X	/ Load 1st operand into AC
202	BSA OR	/ Branch to subroutine OR
203	HEX 3AF6	/ 2nd operand stored here
204	STA Y	/ Subroutine returns here
205	HLT	
206	X,	HEX 7B95 / 1st operand stored here
207	Y,	HEX 0 / Result stored here
208	OR,	HEX 0 / Subroutine OR
209	CMA	/ Complement 1st operand
20A	STA TMP	/ Store in temporary location
20B	LDA OR I	/ Load 2nd operand
20C	CMA	/ Complement 2nd operand
20D	AND TMP	/ AND complemented 1st operand
20E	CMA	/ Complement again to get OR
20F	ISZ OR	/ Increment return address
210	BUN OR I	/ Return to main program
	TMP,	HEX 0 / Temporary storage
		END

Altprogram

Data Bloğunun Transferi

	BSA MVE	/ Main program
	HEX 100	/ Branch to subroutine
	HEX 200	/ 1st address of source data
	DEC -16	/ 1st address of destination data
	HLT	/ Number of items to move
MVE,	HEX 0	/ Subroutine MVE
	LDA MVE I	/ Bring address of source
	STA PT1	/ Store in 1st pointer
	ISZ MVE	/ Increment return address
	LDA MVE I	/ Bring address of destination
	STA PT2	/ Store in 2nd pointer
	ISZ MVE	/ Increment return address
	LDA MVE I	/ Bring number of items
	STA CTR	/ Store in counter
	ISZ MVE	/ Increment return address
LOP,	LDA PT1 I	/ Load source item
	STA PT2 I	/ Store in destination
	ISZ PT1	/ Increment source pointer
	ISZ PT2	/ Increment destination pointer
	ISZ CTR	/ Increment counter
	BUN LOP	/ Repeat 16 times
	BUN MVE I	/ Return to main program
PT1,	--	
PT2,	--	
CTR,	--	

- Fortran altprogramı

```
SUBROUTINE MVE (SOURCE, DEST, N)
DIMENSION SOURCE(N), DEST(N)
DO 20 I = 1, N
20 DEST(I) = SOURCE(I)
      RETURN
      END
```

Giriş-Çıkış Programlama



Bir Karakter (Byte) Girme Programı:

CIF,	SKI	/ Check input flag
	BUN CIF	/ Flag=0, branch to check again
	INP	/ Flag=1, input character
	OUT	/ Display to ensure correctness
	STA CHR	/ Store character
	HLT	
CHR,	--	/ Store character here

Bir Karakter (Byte) Çıkarma Programı:

COF,	LDA CHR	/ Load character into AC
	SKO	/ Check output flag
	BUN COF	/ Flag=0, branch to check again
	OUT	/ Flag=1, output character
	HLT	
CHR,	HEX 0057	/ Character is "W"

Karakter İşleme

İki karakter girme ve bir kelime haline getirme altprogramı

```
IN2, --          / Subroutine entry
FST, SKI
BUN FST
INP           / Input 1st character
OUT
BSA SH4       / Logical Shift left 4 bits
BSA SH4       / 4 more bits
SCD, SKI
BUN SCD
INP           / Input 2nd character
OUT
BUN IN2 I    / Return
```

Program Kesme İşlemi

Kesme Servis Rutin Görevleri:

1. İşlemci saklayıcılarının içeriklerinin saklanması (AC, E).
2. Hangi bayrağın aktiflendiğinin bulunması.
3. Kesme gelen eleman için servis rutin programının yürütülmesi.
4. İşlemci saklayıcıların içeriklerinin tekrar yerlerine yazılması.
5. Kesme izninin tekrar açılması (ION).
6. Ana programa geri dönülmesi.

Kesme Servis Rutin Programı

<i>Loc.</i>			
0	ZRO,	-	/ Return address stored here
1		BUN SRV	/ Branch to service routine
100		CLA	/ Portion of running program
101		ION	/ Turn on interrupt facility
102		LDA X	
103		ADD Y	/ Interrupt occurs here
104		STA Z	/ Program returns here after interrupt
200	SRV,	STA SAC	/ <i>Interrupt service routine</i>
		CIR	/ Store content of AC
		STA SE	/ Move E into AC(15)
		SKI	/ Store content of E
		BUN NXT	/ Check input flag
		INP	/ Flag is off, check next flag
		OUT	/ Flag is on, input character
		STA PT1 I	/ Print character
		ISZ PT1	/ Store it in input buffer
		SKO	/ Increment input pointer
		BUN EXT	/ Check output flag
		LDA PT2 I	/ Flag is off, exit
		OUT	/ Flag is on, load character from output buffer
		ISZ PT2	/ Output character
		LDA SE	/ Increment output pointer
		CIL	/ Restore value of AC(15)
		LDA SAC	/ Shift it to E
		ION	/ Restore content of AC
		BUN ZRO I	/ Turn interrupt on
	SAC,	-	/ Return to running program
	SE,	-	/ AC is stored here
	PT1,	-	/ E is stored here
	PT2,	-	/ Pointer of input buffer
			/ Pointer of output buffer