Bölüm 6
Temel Bilgisayarın Programlanması

- Bir program yazmak,doğrudan veya dolaylı olarak bir dizi makine buyruğunu belirlemek demektir.
- Bir kullanıcının yazdığı program, bilgisayardan bağımlı veya bağımsız olabilir.
- Örneğin fortran programlama ile yazılmış bir program bilgisayar bağımsız olarak çalışır. Çünkü her makine için bu programı, bu programın çalıştırılacağı bilgisayarın buyruklarına çeviren programlar vardır.
- Fakat bu çevirici programlar makine bağımlıdır.
- Bu bölümde, bazı ilkel programlama kavramlarını ve bunların donanımın oluşturduğu buyruklarla bağlantıları ele alınacaktır.

- Bölüm 5 te verilen temel bilgisayarın 25 buyrukluk kümesi, program yazımında kullanılan tekniklerin anlatılmasında kullanılacaktır.Bu sayede programla, programı çalıştıran donanım arasında ilinti kurulabilecektir.

Çizelge 6.1 Bilgisayar buyrukları

Sembol	Onaltılık kod	Tanımlama
AND	0 veya 8	*M* yi *AC* ile VE le
ADD	1 veya 9	*M* yi *AC* ye ekle, elde yi *E* ye aktar
LDA	2 veya A	Bellekten *AC* ye yükle
STA	3 veya B	*AC* yi belleğe sakla
BUN	4 veya C	*m* e şartsız dallan
BSA	5 veya D	Geri dönüş adresini bellekte *m* e sakla ve *m* + 1 e dallan
ISZ	6 veya E	*M* yi 1 arttır eğer 0 ise atla
CLA	7800	*AC* yi sil
CLE	7400	*E* yi sil
CMA	7200	*AC* yi tümle
CME	7100	*E* yi tümle
CIR	7080	*AC* ve *E* yi dairesel sağa kaydır
CIL	7040	*AC* ve *E* yi dairesel sola kaydır
INC	7020	*AC* yi 1 arttır
SPA	7010	*AC* pozitif ise bir sonraki buyruğu atla
SNA	7008	*AC* negatif ise bir sonraki buyruğu atla
SZA	7004	*AC* sıfır ise bir sonraki buyruğu atla
SZE	7002	*E* sıfır ise bir sonraki buyruğu atla
HLT	7001	Bilgisayarı durdur
INP	F800	Bilgiyi al ve bayrağı temizle
OUT	F400	Bilgiyi yolla ve bayrağı temizle
SKI	F200	Eğer giriş bayrağı varsa sonraki buyruğu atla
SKO	F100	Eğer çıkış bayrağı varsa sonraki buyruğu atla
ION	F080	Kesme yi çalıştır
IOF	F040	Kesme yi kapat

Makine Dili (Machine language)

- Bir program,istenen bir veri işleme işlemlerini icra etmesi için bilgisayarı komuta eden buyruklar listesidir.
- Kullanıcı birçok farklı üst seviye dilleri ile programlar yazabilir.Ancak bir bilgisayar,içindeki program binary kodda ise bunu icra edebilir. Diğer dillerde yazılmış programlar, icradan önce binary forma çevrilmelidir.
- Bilgisayar için yazılan programlar aşağıdaki sınıflardan birinde olabilir.

1- Binary kod: Buyruk ve verilerin binary listesidir.(makine bundan anlar-makine dili)

2- Octal veya hexadecimal kod: Binary kodların 8 veya 16'lık sistemde veriliş şekli.

3-Sembolik kod: Kullanıcı alfanümerik semboller kullanarak işlem kısmını yazar. Bu durumda her bir sembolik buyruk bir binary buyruğa dönüştürülmelidir. Bunu yapan programa birleştirici dil (**assembly language-Çevirici dil**) denir.

4-Yüksek seviyeli programlama dilleri: Donanımdan bağımsız,kullanıcı dostu bir programlama dilidir. Bu şekilde yazılan programları Binary koda dönüştüren programlara da derleyici (Compiler)denir.

Makine Dili-2

- Esasta binary kodda yazılmış bir program, makine diliyle yazılmış demektir (1.sınıftaki gibidir).
- 2. sınıfta yazılmış kodlar makine diliyle yazılmış sayılır, çünkü octal, hexa, binary dönüşümü kolaylıkla yapılır.
- Yukarıdaki tarifle, birebir aktarma yapılabilmesinden dolayı (sembollerin binary karşılığı vardır), birleştirici (assembler language-çevirici dil) dile makine seviyesinde dil denilebilir.
- Buradan yapılan çalışmalar, donanım ile yazılım arasındaki bağıntı olduğundan ve makine dilindeki programlarla ilgilenildiğinden makine dili seviyesinde incelenecektir.
- Örnek olması için, bir program parçasını alıp, makine dilinden yukarılara doğru çıkarılmasını anlatabiliriz.

ONALTILIK KOD

- Her buyruk için 16 bitlik sözcükler yazmak oldukça sıkıcı bir iştir.
- Bunun için bu 16 bitlik sözcüklerin hexa karşılıkları ile yazılması daha kolaydır. Bu durumda icra için bu kodlar bilgisayar tarafından binary'ye çevrilir.

• Binary program to add two numbers

Location	Instruction Code
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

• Hexa program

Location	Instruction
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

Sembolik İşlem kodları ve Assemler

Kullanıcı için biraz daha kolay kodlama hexa kodlar yerine,sembolik kodlarının kullanılmasıdır. Buyrukların adres kısımları ve veriler hexa olarak bırakılmıştır.

Kullanıcıyı daha da rahatlatan bir dil, (assemly language) birleştirici-çevirici dilidir. Bu dilde yazılmış programlar (assemly language program) çevirici dil programları ile makine koduna çevrilir.

Sonunda bir sayı bulunan ORG bir makine buyruğu değildir. Adresler semboliktir. Verinin 10 tabanlı olduğu DEC ile belirtilir. END programın sonudur.

ORG,END,DEC sözde buyruktur.

• Program with symbolic opcode

Location	Instruction		Comments
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

• Assembly language 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
```

- Hexa program

Location	Instruction
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

- Program with symbolic opcode

Location	Instruction		Comments
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

- Assembly language 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
```

Üst seviye dil

- Fortran program

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

- İki sayının toplamına ilişkin fortran programı (üst seviye dilde yazılmış) yanda görülmektedir.
- Burada A ve B değerleri INPUT ve DATA buyrukları ile verilir. Toplama, tek bir işlem ile olmaktadır.
- Bu programın çalışması için için 3 ayrı bellek alanı gerekir.
- Derleyici (compiler) vasıtasıyla bu program binary'e (makine koduna) dönüşür.

6.3 Birleştirici Dil (assemly language-çevirici dil)

- Bir programlama dili bir dizi kurallar ile tanımlanır, kullanıcı bu kurallara uygun olarak yazılımını yaptığında, program doğru bir şekilde binary koda (makinanın anlayacağı koda) dönüştürülür.
- Sembolik kodda, birleştirici (çevirici- assembler) dil kurallarına uygun olarak yazılmış bir program, eğer doğru yazılmış ise; kolaylıkla makinanın anlayacağı şekilde binary forma dönüştürülür.
- Birleştirici dilinde birim, komut hattıdır. Kurallar, kullanılacak sembolleri ve bu sembollerin, komut satırında nasıl biraraya getirileceğini belirler.

Birleştirici (assemly laguage) dilin kuralları

- Bu dilde, her satır alanı 3 sütundan oluşur.

1- Başlık alanı : Boş olabilir veya sembolik adres belirtebilir.

2- Buyruk Alanı: Bir makine buyruğu veya açıklaması vardır.

3- Açıklama alanı: Boştur veya açıklama vardır.

Sembolik adres: 1,2 veya 3 alfanümerik karakterden oluşabilir. İlk karakter harf olmalıdır. Diğerleri harf veya rakam olabilir. Sembol kullanıcı tarafından keyfi seçilebilir.

Başlık içindeki sembolik adres, bir virgülle ayırd edilmelidir. Birleştirici bunun başlık olduğunu bu şekilde anlar.

Birleştirici dilin kuralları-1

- Buyruk alanı esas birleştirici dil programının olduğu alandır. Buraya aşağıdaki buyruk tipleri yazılır.

1- Bellek Adreslemeli buyruk (MRI): İki veya üç sembolden oluşur. Boşluklarla biribirnden ayrılır. İlk sembol üç harfli olup MRI işlem kodunu (And, SUM v.b) belirtir. İkincisi sembolik adrestir. 3. sembol ise I'dır ve kullanılması zorunlu değildir. Yoksa doğrudan, varsa dolaylı adreslemelidir.

2- Yazaç Adreslemeli Buyruk (MRI olmayan): Adres kısmı olmayan buyruklardır (CLA,CLE,INC, INP,OUT V.B). Bunlar, yazaç adreslemeli ve G/Ç buyruklarıdır. Buyruk sembolleri örnekleri;

CLA			*MRI-değil*
ADD	*OPR*		*MRI doğrudan adresli*
ADD	*PTR*	*I*	*MRI dolaylı adresli*

Buyruk alanındaki adres sembolü, bir verinin bellek adresini belirtir.Bu adres programın herhangi bir yerinde ilk sütundaki başlık ile verilmelidir. Sembolik bir birleştirici programının binary'e dönüştürülmesi için, buyruk alanındaki her bir sembolik adresin başlık alanında mutlak surette bir daha görünmesi gereklidir.

Birleştirici dilin kuralları-2

3- Sözde buyruk (verili veya verisiz): Makinanın

işleyeceği bir buyruk değildir. Daha çok derleyiciye bilgi veren bir buyruktur. Ve dönüşümün bir aşaması için bilgi verir.Örneğin ORG sembolü, onu takip eden satırdaki buyruk veya verinin ORG'un yanındaki adreste olduğunu gösterir. Bir program içinde ORG'u birden fazla kullanmak mümkündür.

Bazı sözde buyruklar

ORG N	**N(hex) sayısı, buyruk veya veri listelerinin bellekteki başlangıç yeri.**
END	**Sembolik programın bittiğini belirtir.**
DEC N	**Binary'e çevrilecek işaretli on tabanlı N sayısı.**
HEX N	**İkiliye çevrilecek Hexa N sayısı.**

END sembolü, programın sonuna yazılarak derleyiciye, programın bittiğini haber verir.

Programın 3.alanı ise, açıklamalara ayrılmıştır. Olabilir veya olmayabilir. Açıklama varsa derleyicinin anlaması için / işareti ile başlamalıdır.

Bir Örnek

Çizelge 6.8 2 sayının çıkarılması için birleştirici dili program

	ORG 100	/ Program 100 adresinden başlamakta
	LDA SUB	/ Çıkarılanı *AC* ye yükle
	CMA	/ *AC* nin tümleyenini al
	INC	/ *AC* yi 1 arttırır
	ADD MIN	/ Çıkanı *AC* ye topla
	STA DIF	/ Farkı depola
	HLT	/ Programı durdur
MIN,	DEC 83	/ Çıkan
SUB,	DEC –23	/ Çıkarılan
DIF,	HEX 0	/ Fark buraya saklanacak
	END	/ Sembolik programın bitişi

- Üstteki sembolik birleştirici programı örneğinde, ORG sözde buyruğundan , programın başlangıcının 100 (hexa) bellek adresinden başlayacağı anlaşılır.
- Bundan sonraki 6 satır, makine buyruklarını, son 4 satır ise 4 sözde buyruğu sembollemektedir. Üç tane sembolik adres (MIN,SUB,DIF) kullanılmıştır. Bunlar 1.sutundaki başlık kısmında belirtilmiştir.Son buyruk ise programın sonunu gösterir.
- Program ikili koda dönüştürülüp makine tarafından icra edilirse, 83+(-23) = 106 sunucunu bulur. (Tamamlayıcı toplama yoluyla çıkarma yaparak)

Binary'e Çevrilme

- Birleştirici (Çevirici-assembler): Sembolik programın binary koda dönüştürülmesi birlrştirici (çevirici) adı verilen programla sağlanır. Çizelge 6.8'deki sembolik programın eşdeğer koda dönüştürülmesi; programın taranması ve sembollerin eşdeğeri olan ikili makine kodlarıyla değiştirilmesi yoluyla yapılır.
- Bu işe ilk satırdan başlayarak ORG sözde buyruğundan, bellekteki 100 (hexa) adresinden başlanacağı anlaşılır.

Çizelge 6.9 Çizelge 6.8 deki programın dönüştürülmüş listesi

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

Binary'e Çevirme-2

- 2.satırdaki iki sembol var. Bu satır 100 adresine yerleştirilecek MRI buyruğudur. I harfi olmadığından buyruk kodunun ilk biti 0 olmalıdır.
- İşlemin sembolik harfi LDA'dır. Çizelge 6.1'e göre buyruğun ilk hexa basamağı 2 olmalıdır.
- Adresin binary değeri ise adres sembolü SUB'dan elde edilmelidir. Başlık sütununa bakıldığında, bu sembol 9.satırda görülür. Bu satırda belleğin 107 adresi vardır.
- Buyruğun parçaları birleştirilince 2107 hexa kodu ortaya çıkar.
- Sembolik (assemler) programda iki satır ondalık verileri DEC sözde buyruğu ile vermektedir.
- 3.satırda ise sonucun hex olarak yazılması istenmekte.
- 83 sayısı ise binary çevrilip 106 adresine yazılacaktır.
- 107 adresinde de -23'ün 2'ye tümleyeni alınıp hexa olarak değeri görülmektedir.
- END sembolü de programın bitimidir.

Adres Sembol Tablosu

- Sembolik program (birleştirici dilde yazılımış program), iki defa taranarak, çevirme işlemi(Binary koda-makine diline) gerçekleştirilir.
- Birinci taramada sadece her buyruk ve veri için bir bellek adresi ve yeri ayrılır.
- Çizelge 6.9da 100 adresi, ORG'dan sonraki ilk satırdaki buyruk için ayrılır. Satırlardaki ORG ve END birer buyruk veya adres göstermediklerinden adres verilmez.
- Birinci taramadan sonra herbir başlık sembolü için karşılık gelen adres çizelgeye yazılır.

Adres sembolü	*onaltılık Adres*
MIN	*106*
SUB	*107*
DIF	*108*

İkinci taramada MRI buyrukları için adres bulmakta, adres sembol çizelgesine başvurulur. SUB'ın hexa değeri ise yukarıdaki adres-sembol çizelgesinden alınarak oluşturulur.

Sembollerden binary'e geçişte derleyici kullanılıyorsa birinci taramaya *ilk geçiş*, ikinciye *ikinci geçiş* denir.

6.4. Derleyici

- Derleyici, sembolik programı alır ve bunu birleştirici dil binary eşdeğerine çevirir.
- Sembolik programa kaynak program, sonuç binary programa amaç program denir.
- Birleştirici karakterler üzerinde işlem yapar ve Binary karşılıklarını bulur.

Sembolik Programın Bellekteki Görünüşü

- Birleştirme işlemi başlamadan önce sembolik programın bellekte yerleştirilmiş olması gerek. Bu işlem klavyeden olacağı gibi,bir yükleyici program da kullanılabilir.
- Sembolik progamın bellekteki görünümü, alfasayısal karakter kodları (ASCII) formatında olur.

ASCII kodlar

Çizelge 6.10 Onaltılık karakter kodu

Karakter	Kod	Karakter	Kod	Karakter	Kod	
A	41	Q	51	6	36	
B	42	R	52	7	37	
C	43	S	53	8	38	
D	44	T	54	9	39	
E	45	U	55	boşluk	20	
F	46	V	56	(	28	
G	47	W	57	)	29	
H	48	X	58	*	2A	
I	49	Y	59	+	2B	
J	4A	Z	5A	,	2C	
K	4B	0	30	-	2D	
L	4C	1	31	.	2E	
M	4D	2	32	/	2F	
N	4E	3	33	=	3 D	
O	4F	4	34	CR	0D	(yeni satır başı)
P	50	5	35			

- CR kodu, enter tuşuna basıldığında elde edilir. Birleştirici CR kodunu bir kod satırının sonu olarak algılar
- Kod satırı ardışık adreslere yerleştirilir. Bir adres 16 bitlik olduğundan her adres sembolleri boşluk ile biter.Satır sonu CR ile bellidir.

- Örneğin bir satır kodu;

  PL3, LDA SUB I

  Ardışık 7 bellek adresine yerleştirilir (Çizelge 6.11). PL3 etiketi iki kelime içerir ve virgülle sona erer. Kod satırındaki buyruk alanı bir veya daha fazla sembol içerebilir.
- Her sembol boşluk (20) ile sona erer. Sadece son sembol CR (0D) ile sona erer.
- Komut satırında açıklama var ise birleştirici bunu /(2F) kodundan tanır.
- Birleştirici Açıklama alanındaki bütün satırları atlayıp CR kodunu arar.Bu kod bulununca son sembolden sonra yazılan boşluk CR ile değiştirilir. CR kodu satırın son karakteri olarak yazılır.

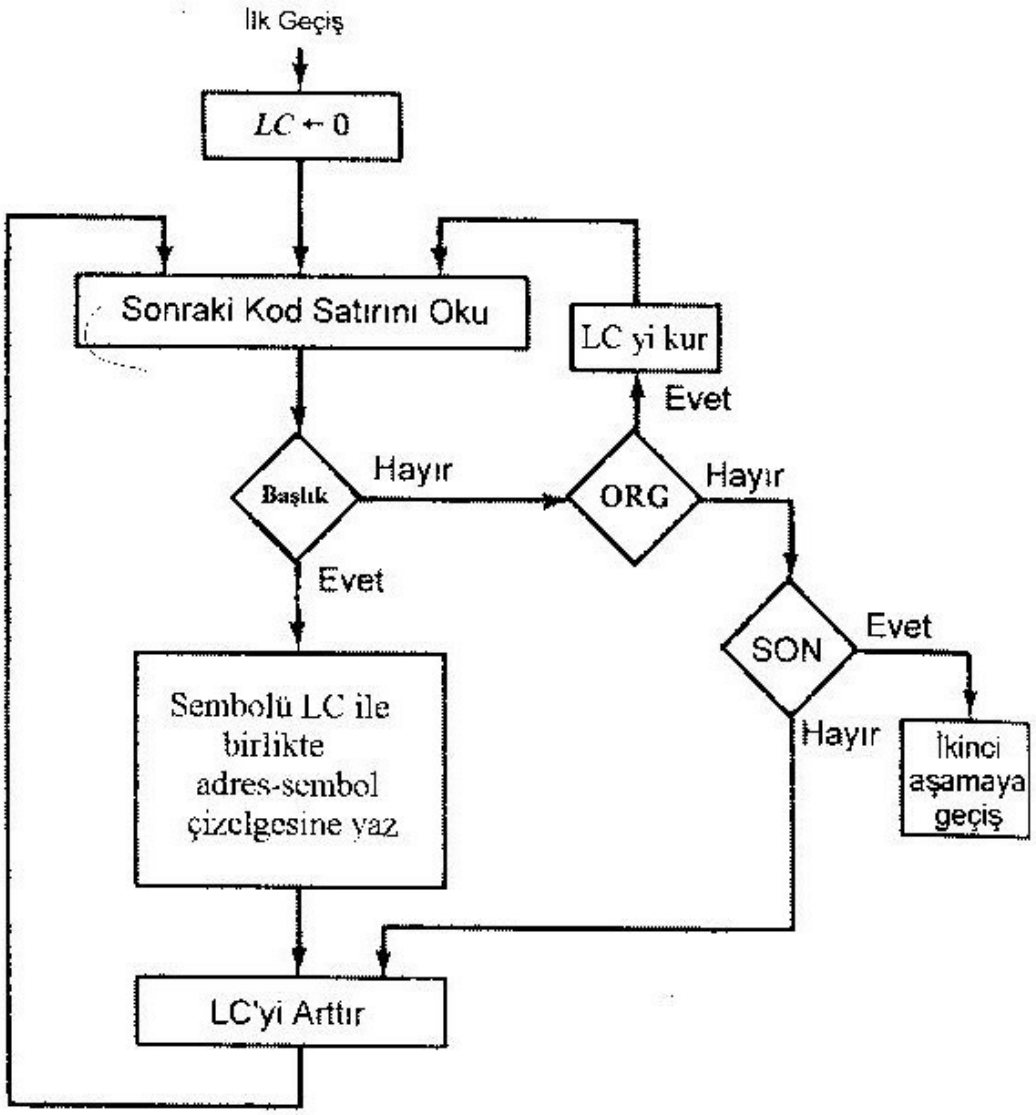
Çizelge 6.1 PL3, LDA SUB I nın bilgisayarda gösterimi

Bellek kelimesi	Sembol	Onaltılık kod		İkili gösterimi			
1	PL	50	4C	0101	0000	0100	1100
2	3 ,	33	2C	0011	0011	0010	1100
3	L D	4C	44	0100	1100	0100	0100
4	A	41	20	0100	0001	0010	0000
5	S U	53	55	0100	0011	0101	0101
6	B	42	20	0100	0010	0010	0000
7	I CR	49	0D	0100	1001	0000	1101

- Birleştirici program için giriş, kullanıcının sembolik programıdır.Bu Program ASCII kodundadır. Birleştirici tarafından iki kez taranarak binary program elde edilir. Dönüştürme işlemi süresince birleştirici dil ne yapar?

İlk Geçiş

- İlk geçişte, çevirici program bir adres çizelgesi oluşturup, bu çizelgeye kullanıcının tanımladığı tüm adres sembollerinin binary eşdeğerini yazar.
- Binary dönüşüm 2.geçişte yapılır.
- Buyrukların adreslerinin sırasını takip edebilmek için, birleştirici (çevirici) bir bellek kelimesi kullanır. Bu adrese satır sayıcı (LC) adı verilir.Her bir satır yerleşiminden sonra LC 1 arttırılır.
- Eğer sembolik programda ORG yoksa, LC herzaman 0'dan başlar.
- Birleştirici tarafından ilk geçiş sürecinde yapılan işlerin akış şeması Şek.6.1'de verilmiştir.



Şekil 6.1 Derleyicinin birinci aşaması için akış şeması

Çizelge 6.12 Çizelge 6.8 deki program için adres sembol çizelgesi

Bellek kelimesi	sembol veya LC	Onaltılık kod	İkili gösterimi
1	M I	4D 49	0100 1101 0100 1001
2	N ,	4E 2C	0100 1110 0010 1100
3	(LC)	01 06	0000 0001 0000 0110
4	S U	53 55	0101 0011 0101 0101
5	B ,	42 2C	0100 0010 0010 1100
6	(LC)	01 07	0000 0001 0000 0111
7	D 1	44 49	0100 0100 0100 1001
8	F ,	46 2C	0100 0110 0010 1100
9	(LC)	01 08	0000 0001 0000 1000

- Çizelgeden görüldüğü gibi, her bir başlık sembolü iki bellek adresine yerleştirilir ve bir virgülle sonuçlanmalıdır.
- Satır işlem gördüğünde LC içinde bulunan değer, bir sonraki bellek adresine yerleştirilir.

İKİNCİ GEÇİŞ

- Makine buyrukları 2.geçiş sırasında binary'e dönüştürülürler. Dönüşümde 4 tane çizelgeye bakılarak karar verilir.Programdaki herşey bu 4 çizelgenin içinde bulunmalıdır.

1- Sözde buyruk çizelgesi: 4 tane girişi, ORG,END,DEC ve HEX'dir.

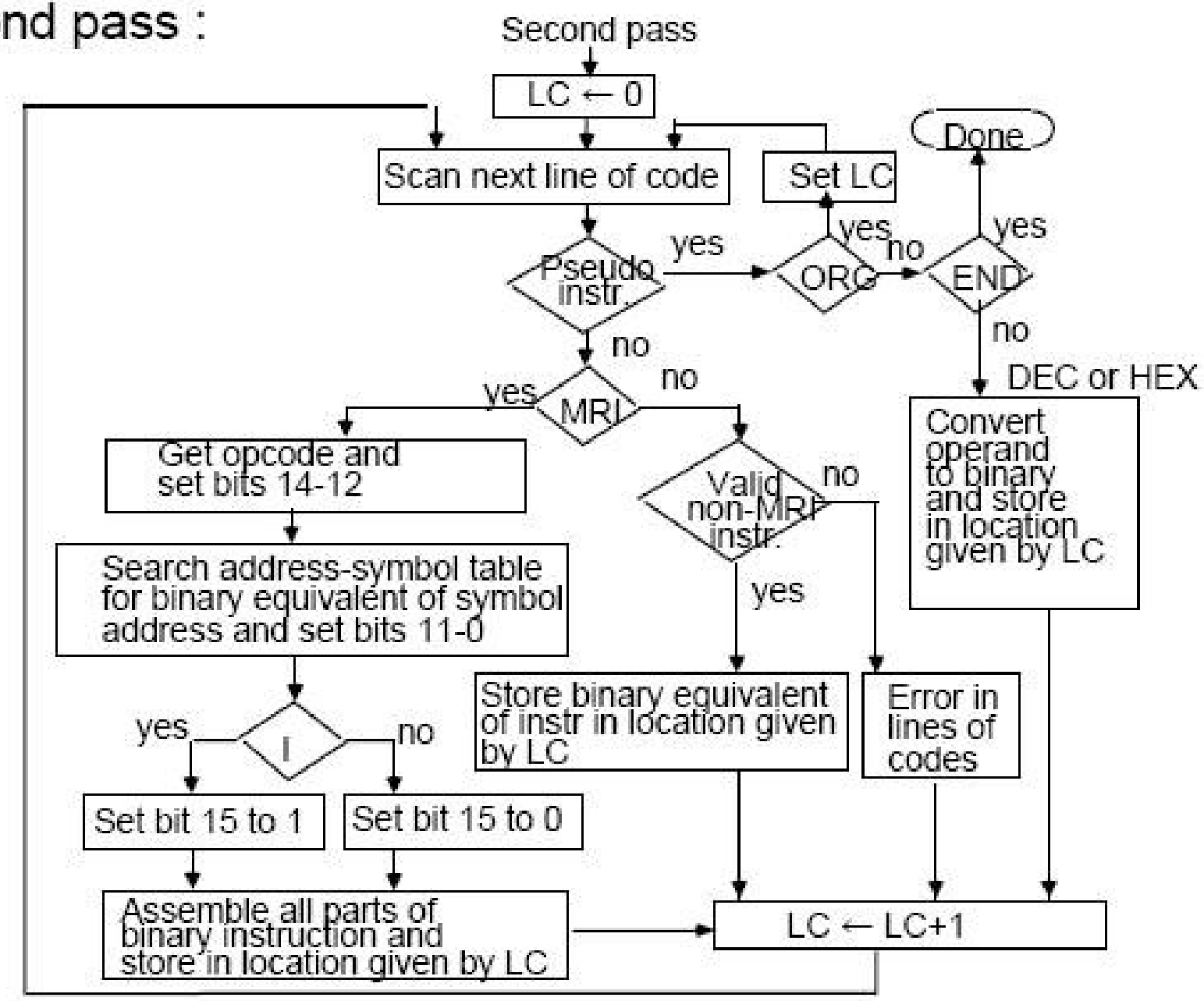
2- MRI buyrukların çizelgesi: 7 adet MRI işlem kodunu içerir.(Bellek adreslemeli buyruk)

3-MRI olmayan buyrukların çizelgesi:18 adet yazaç adreslemeli buyruğu ve, G/Ç buyruklarını içerir.

4- Adres-sembol çizelgesi: 1.geçişte kullanılır.

İkinci geçişteki işler şekil 6.2'de verilmiştir.

Second pass :


Second pass
LC ← 0
Scan next line of code
Set LC
Done
Pseudo instr.
yes
ORG
yes
no
END
yes
no
MRI
yes
no
DEC or HEX
Convert operand to binary and store in location given by LC
Get opcode and set bits 14-12
Valid non-MRI instr.
no
yes
Search address-symbol table for binary equivalent of symbol address and set bits 11-0
I
yes
no
Store binary equivalent of instr in location given by LC
Error in lines of codes
Set bit 15 to 1
Set bit 15 to 0
Assemble all parts of binary instruction and store in location given by LC
LC ← LC+1

HATA BULMA

- Çevirici derleyicisini önemli bir görevi sembolik programdaki hataları bulmaktır.
- Yanlış yazılmış bir sembol,
- MRI veya MRI olmayan çizelgesinde bulunan bie sembol
- Adresin sembolünün verilmeyişi

6.5 Program Döngüleri

- Br program döngüsü birçok kez icra edilen buyruklar sırasıdır.Her bir çalışmada verileri farklı olabilir. Fortrandaki Do deyimi ile tanımlanan bir döngü örneğinde, 3 satırı 100 kez A(J) farklı değerleri için tekrarlanmaktadır.

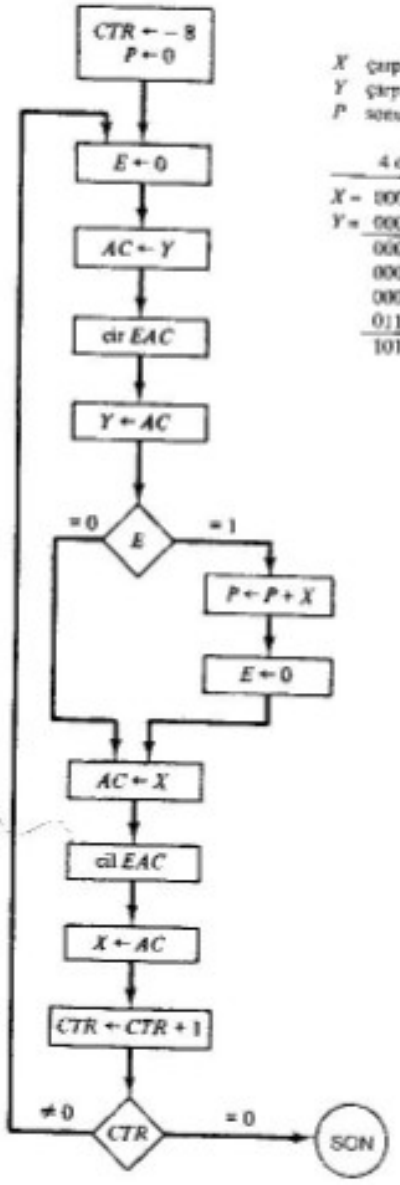
```
      DIMENSION A(100)
      INTEGER SUM, A
      SUM = 0
      DO 3 J = 1, 100
    3 SUM = SUM + A(J)
```

6.5 Program Döngüleri-1

- Bir önceki slaytta görüldüğü gibi, yüksek seviyeli bir dille yazılmış programı makine koduna çeviren programlara derleyici (Compiler) denir. Çalışmaları Çeviricilere göre (sembolik programı- assemler'ı makine diline çeviren) göre oldukça karmaşıktır.
- Derleyiciler, doğrudan yüksek seviye dilden, binary koda çevirebildikleri gibi, çevirici (assemler) diline çevirip, oradan makine diline de çevirebilirler.
- Burada yüksek seviyeden sembolik dile çevrilme işlemleri açıklanacaktır. Ş,çizelge 6.13 bunu gösterir.
- Yüksek seviye dildekiilk iki terim işlenemez komuttur. Bunlar derleyicide sözde buyruk gibi algılanır.

Çizelge 6.13 100 sayının toplamını yapan sembolik program

Sıra			
1		ORG 100	/Program bellekte 100 üncü adreste bulunmakta
2		LDA ADS	/İşlenenin ilk adresini yükle
3		STA PTR	/Göstergeye aktar
4		LDA NBR	/ –100 ü yükle
5		STA CTR	/Sayıcıya aktar
6		CLA	/*AC* yi sil
7	LOP,	ADD PTR I	/Bir ileneni *AC* ile topla
8		ISZ PTR	/Göstergeyi 1 arttır
9		ISZ CTR	/Sayıcıyı 1 arttır
10		BUN LOP	/LOP a şartsız dallan
11		STA SUM	/TOPLAM a yükle
12		HLT	/Programı durdur
13	ADS,	HEX 150	/Verilerin ilk adresi
14	PTR,	HEX 0	/Bu adres gösterge için ayrılmıştır
15	NBR,	DEC -100	/İlkleştirilmiş sayaç için sabit
16	CTR,	HEX 0	/Bu adres sayaç için ayrılmıştır
17	SUM,	HEX 0	/Toplam buraya yüklenecektir
18		ORG 150	/Verilerin bellekteki başlangıç adresi
19		DEC 75	/İlk veri
•			
•			
•			
118		DEC 23	/Son veri
119		END	/Sembolik programın sonu



X çarpılanı tutar
Y çarpanı tutar
P sonucu tutar

4 önemli bit için örnek

			P
X =	0000	1111	0000 0000
Y =	0000	1011	0000 1111
	0000	1111	0010 1101
	0001	1110	0010 1101
	0000	0000	1010 0101
	0111	1000	
	1010	0101	

Şekil 6.3 çarpma programı akış şeması

Çizelge 6.14 İki pozitif sayının çarpımı ile ilgili program

```
        ORG  100
LOP,    CLE          /E yi temizle
        LDA  Y       /Çarpanı yükle
        CIR          /Çarpan biti E ye aktar
        STA  Y       /Kaydırılmış çarpanı sakla
        SZE          /E = 0 mı diye bak
        BUN  ONE     /bit = 1 ise ONE a git
        BUN  ZRO     /bit = 0 ise ZRO ya git
ONE,    LDA  X       /çarpılanı yükle
        ADD  P       /Kısmi çarpıma ekle
        STA  P       /Kısmi çarpımı sakla
        CLE          /E yi temizle
ZRO,    LDA  X       /Çarpılanı yükle
        CIL          /Sola kaydır
        STA  X       /Kaymış çarpılanı sakla
        ISZ  CTR     /Sayacı arttır
        BUN  LOP     /Sayaç sıfır değil; döngüyü tekrarla
        HLT          /Sayaç sıfır; durdur
CTR,    DEC  -8      /Bu adres sayaç olarak çalışır
X,      HEX  000F    /Çarpılan buraya saklanır
Y,      HEX  000B    /Çarpan buraya saklanır
P,      HEX  0       /Sonuç buraya saklanır
        END
```

Çizelge 6.16 Alt program kullanan bir program gösterimi

Adres		Program		Açıklama
		ORG	100	/Ana program
100		LDA	X	/X i AC ye yükle
101		BSA	SH4	/Alt programa git
102		STA	X	/AC yi X e aktar
103		LDA	Y	/AC ye Y yi aktar
104		BSA	SH4	/Alt programa git
105		STA	Y	/AC yi Y ye aktar
106		HLT		/Dur.
107	X,	HEX	1234	
108	Y,	HEX	4321	
				/4 kez sola kaydıran alt program
109	SH4,	HEX	0	/Buraya geri dönüş adresini yaz
10A		CIL		/Bir kez sola kaydır
10B		CIL		
10C		CIL		
10D		CIL		
10E		AND	MSK	/AC nin 13–16 bitlerini sıfırla
10F		BUN	SH4	/Ana programa dön
110	MSK,	HEX	FFF0	/Maskeleme işleneni
		END		

Çizelge 6.17 Parametre aktarımı ile ilgili program

Location				
		ORG	200	
200		LDA	X	/İlk veriyi AC ye yükle
201		BSA	OR	/OR alt programa gir
202		EX	3AF6	/İkinci veri burada saklanmakta
203		STA	Y	/AC dekini Y ye aktar
204		HLT		/Programı durdur
205	X,	HEX	7B95	/İlk veri burada saklanmakta
206	Y,	HEX	0	/Sonuç buraya saklanacak
207	OR,	HEX	0	/OR alt programı
CMA				/İlk verinin tümleyenini al
STA	TMP			/AC yi geçici yazaca yaz
20A		LDA	OR I	/AC ye ikinci veriyi yükle
20B		CMA		/İkinci verinin tümleyenini al
20C		AND	TMP	/İlk veri ile VE le
20D		CMA		/VEYA işlemini elde etmek için tümleyenini al
20E		ISZ	OR	/Geri dönüş adresini 1 arttır
20F		BUN	OR I	/Ana programa dön
TMP,	HEX	0		/geçici sonuç değişkeni
END				

Çizelge 6.18 Veri bloğunun taşınması alt programı

```
                        /Ana program
BSA     MVE             /Alt programa dallanma
        HEX  100        /Kaynak verinin başlangıç adresi
        HEX  200        /Hedef verinin başlangıç adresi
        DEC  -16        /Aktarılacakların sayısı
        HLT
MVE,    HEX  0          /MVE alt program
        LDA  MVE I      /kaynak verinin başlangıç adresini AC ye aktar
        STA  PT1        /AC yi PT1 göstergesine yükle
        ISZ  MVE        /Dönüş adresini arttırır.
        LDA  MVE I      /Hedefin başlangıç adresini AC ye getir
        STA  PT2        /AC yi PT2 ye yükle
        ISZ  MVE        /(MVE) Dönüş adresini 1 arttır
LOP,    LDA  PT1  I     /PT1 in gösterdiği adresin içeriğini AC ye aktar
        STA  PT2  I     /AC nin içeriğini PT2 nin gösterdiği adrese aktar
        ISZ  PT1        /PT1 in değerini 1 arttır
        ISZ  PT2        /PT2 nin değerini 1 arttır
        ISZ  CTR        /CTR nin değerini 1 artır
        BUN  LOP        /Döngüyü 16 Kez tekrarla
        BUN  MVE  I     /Ana programa dön
PT1,    —
PT2,    —
CTR,    —
```