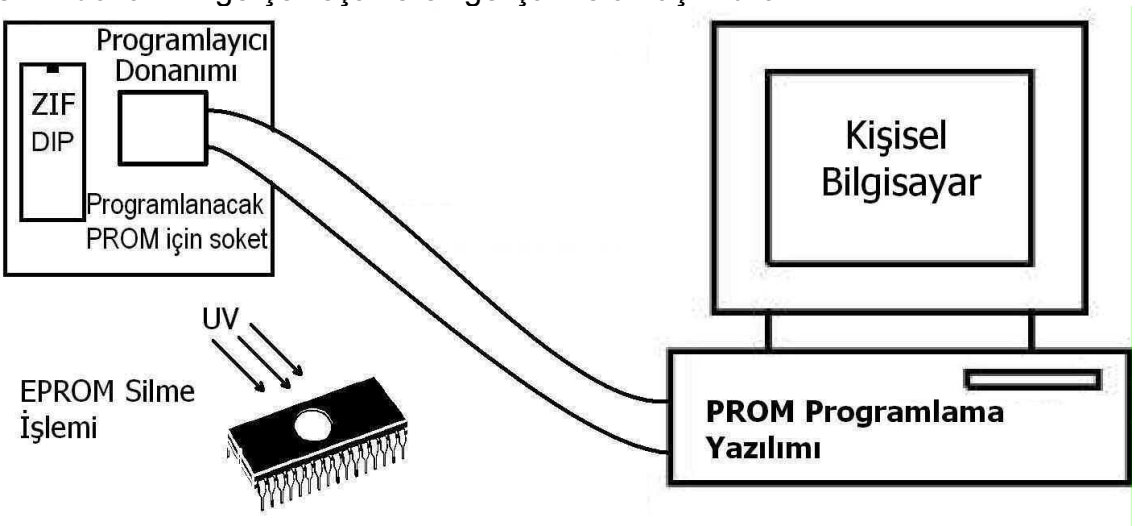
13. MİKROİŞLEMCİLİ SİSTEM DONANIMI VE YAZILIMI GELİŞTİRME ARAÇLARI

Mikroişlemcili sistemler geliştirilirken, tasarlanan mikroişlemci temelli sistemin donanımını ve yazılımını gerçekleştirmek, fiziksel dünyaya aktarmak amacıyla tasarlanmış donanım ve yazılımlar kullanılır.

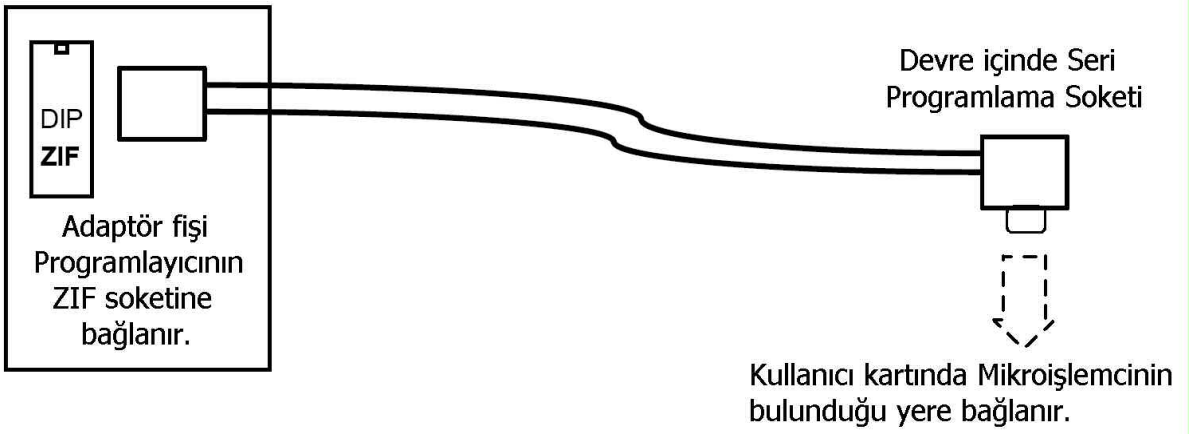
13.1. Mikroişlemcili Sistem Donanımı Geliştirme Araçları

Mikroişlemcili sistem donanımı geliştirilirken genellikle kullanıcı tarafından tasarlanan mikroişlemci temelli sistemin donanımı gerçekleştirilerek geliştirme amaçlı kullanılır.



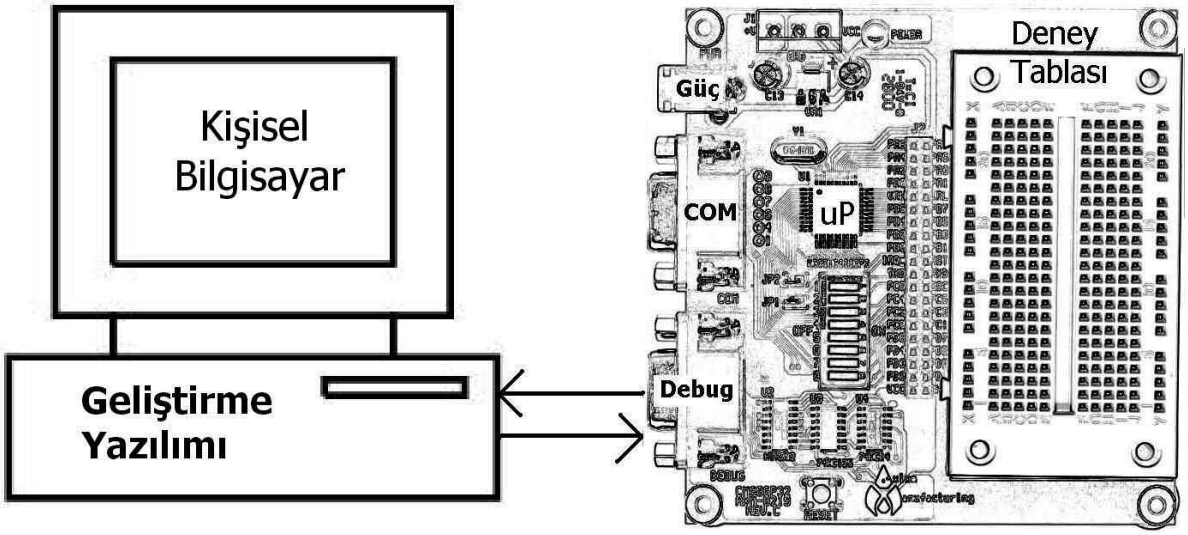
Şekil 13-1 EPROM tümleşik devrelerinin programlanması

Günümüzde yaygın olarak kullanılan mikroişlemcili sistemler, flaş tipinde program belleğini tümleşik olarak içinde bulunduran mikrodenetleyicilerdir. Veri akışı seri olduğu için bu tip programlama devrelerine “devre içi seri programlayıcı” (In Circuit Serial Programming, ICSP) adı verilir.



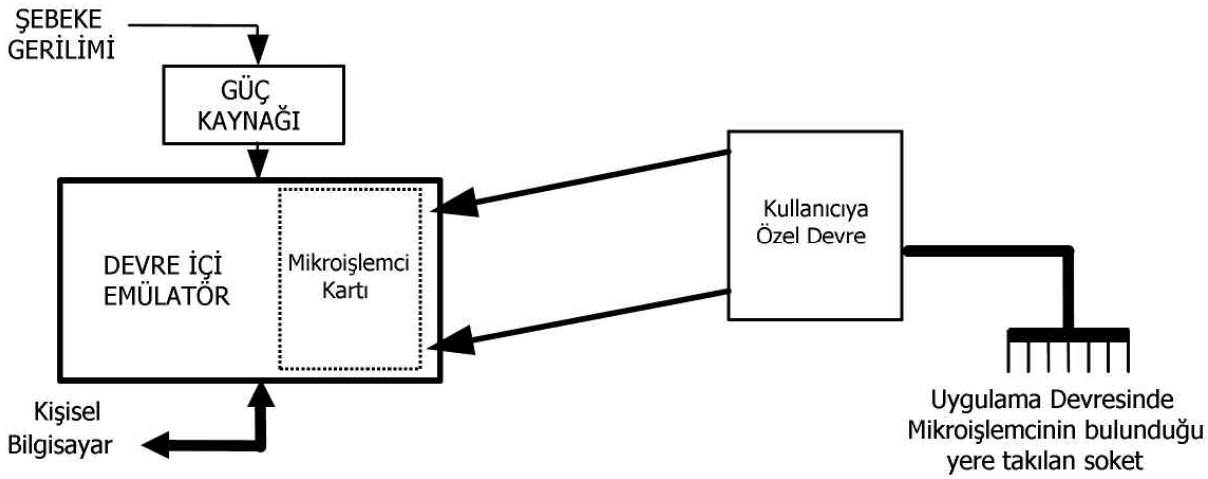
Şekil 13-2 Devre içi seri programlama

Eğitim amaçlı veya ekonomik mikroişlemcili veya mikrodenetleyicili ürün geliştirmelerinde ise geliştirmesi yapılacak mikroişlemciye özel genel amaçlı çalışabilen mikroişlemci temelli donanımlar kullanılır. Genel amaçlı mikroişlemci sistem donanımı geliştirme araçlarına “Mikroişlemci Geliştirme Sistemi”, (Microprocessor Development System) adı verilir.



Şekil 13-3 Mikroişlemci Geliştirme Sisteminin Blok Diyagramı

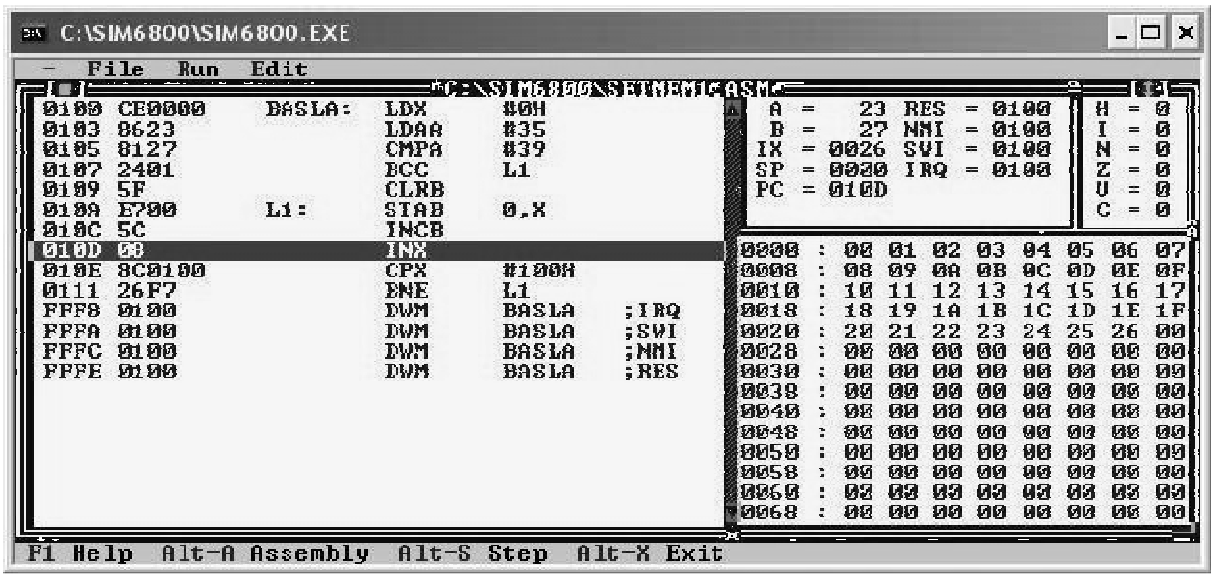
Araştırma, geliştirme gerektiren mikroişlemcili veya mikrodenetleyicili ürün geliştirmelerinde ise geliştirmesi yapılacak mikroişlemciye özel çalışabilen mikroişlemci temelli donanımlar kullanılır. Bu tür özel amaçlı mikroişlemci sistem donanımı geliştirme araçlarına “Devre İçi Donanım Benzetici, Emülatör”, (In Circuit Emulator, ICE) adı verilir.



Şekil 13-4 Devre İçi Donanım Benzetici sisteminin blok diyagramı.

13.2. Mikroişlemcili Sistem Yazılımı Geliştirme Araçları

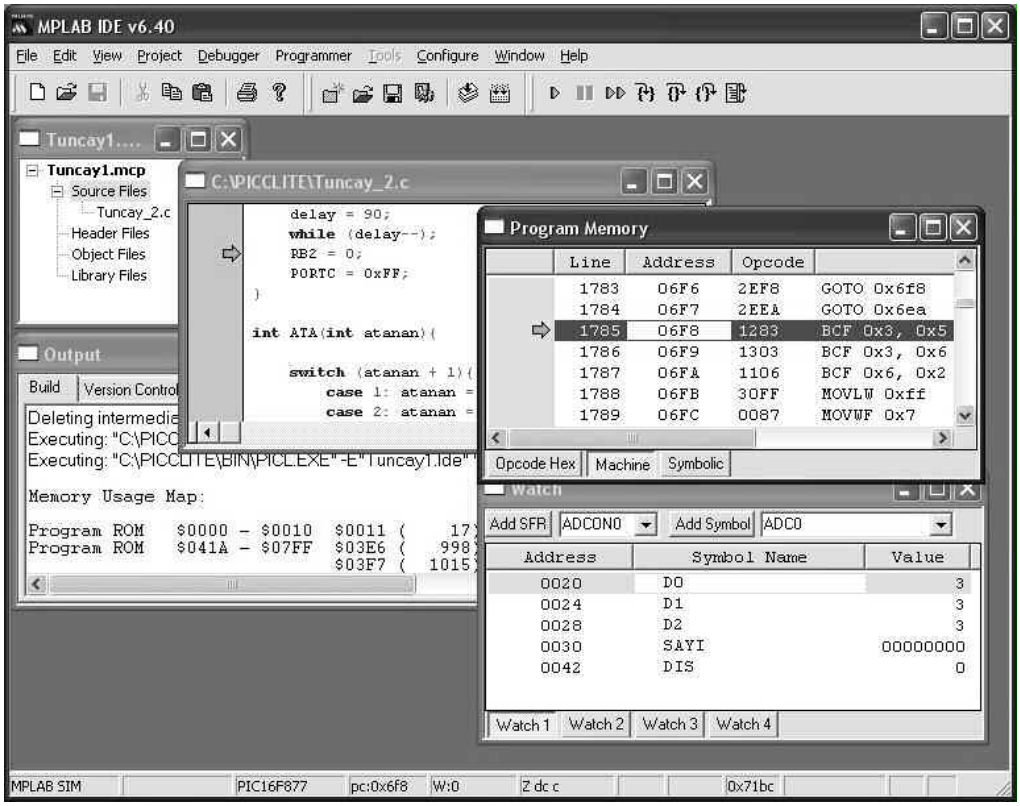
Mikroişlemcili sistem yazılımı geliştirilirken genellikle kullanıcı tarafından tasarlanan mikroişlemciye özel geliştirme amaçlı çevirici dilinden makine dilene çeviriciler, C ve BASIC gibi yüksek seviyeli dillerden makine diline derleyiciler (Compiler), gibi yazılımlar kullanılır. Bu yazılımlar genellikle tasarlanan programın bir kişisel bilgisayar ortamında çalışmasını, incelenmesini, hata ayıklanmasını (debug) sağlar.



Şekil 13-5 6800 Mikroişlemcisi için kullanılan bir simülatör yazılımı

Mikroişlemci temelli sistem programının kişisel bilgisayar ortamında çalışmasını sağlayan programlara "Benzetim, Simülatör" (Simulator) programı denir.

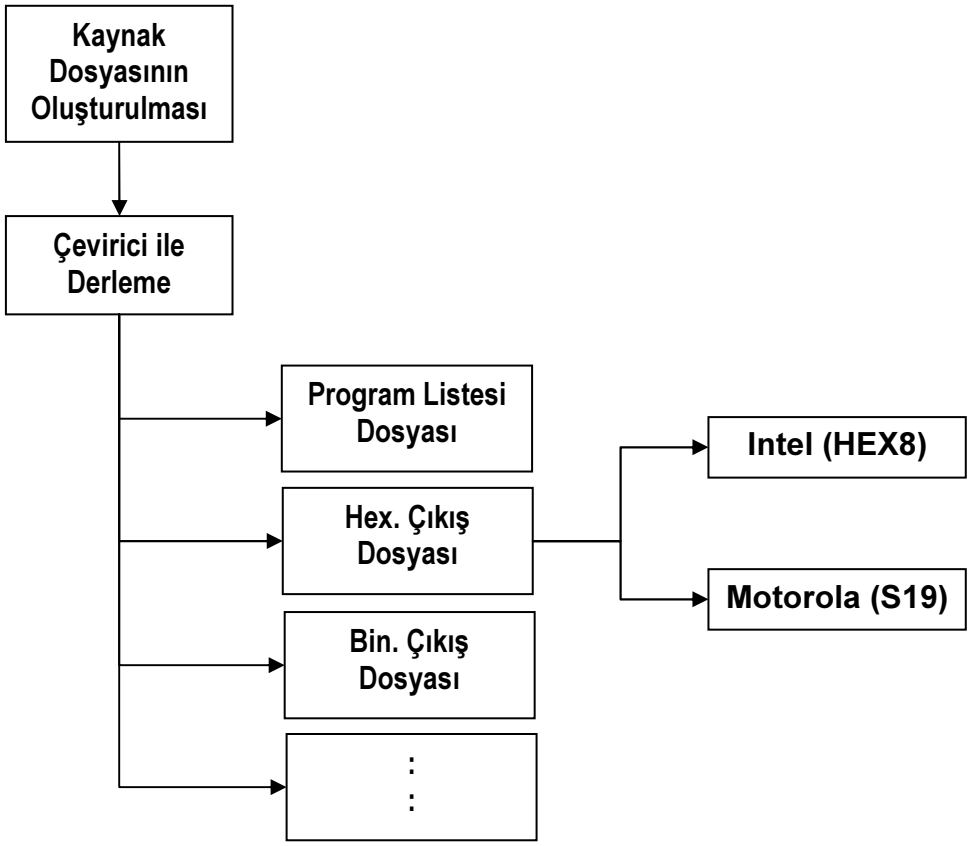
Günümüzde, en yaygın olarak kullanılan mikroişlemci yazılımı geliştirme araçları, tümleşik geliştirme ortamlarıdır (Integrated Development Environment, IDE).



Şekil 13-6 Bir tümleşik geliştirme ortamının ekran görüntüsü

13.2.1. Çevirici Dili ve özellikleri

En temel mikroişlemci yazılımı geliştirme aracı olarak çevirici programlar kullanılır. Kısa komut adları kullanılarak oluşturulmuş mikroişlemci yazılımlarını, ilgili mikroişlemciye özel makine diline çeviren programlara çevirici programı (Assembler) denir.
Değişik mikroişlemciler için ayrı ayrı veya bir bütün içinde yönlendirmeyle seçilen mikroişlemci çevirici dilleri (Assembly Language) vardır. Bunlar temel özellikleri aynı olmak kaydıyla, birbirinde farklı ek özelliklere sahip olabilir. Çevirme şekline göre: tek geçişli, iki geçişli, makro gibi değişik ön adlar alabilirler.



13.2.2. Kaynak Dosyası Özellikleri

Bir kaynak dosyası her bir satırında sadece bir ifade bulunan birçok satırdan meydana gelmiş bir dosyadır. Word, WordPad, not defteri, vs. gibi kelime işlem yazılımları kullanılarak ASCII tipinde, farklı kaydetme “Düz Metin (*.txt)” seçimi ile oluşturulur. Bu kaynak dosya içindeki ifadelerin biçimleri izin verilen tanımların dışına çıkamaz ve çıkılması durumunda çıkış dosyalarında belirtilen hatalara neden olunur. Her satır geriye taşıma (CR) ve/veya satır besleme (LF) ile birbirinden ayrılır. <CR><LF> algılaması sisteme bağlıdır. Sonra satır sayacına bir eklenir. Bir çevirici satırı belirli sayıda karakterden (132, 255 gibi) büyük olamaz. Çevirici programları kaynak dosyası tip kısmı verilmediği zaman, genellikle ASM (*.ASM) kelimesini tip kodu olarak kabul eder. Kolaylık olması açısından yazılan çevirici kaynak programlarında tip kodu ASM seçilmelidir.

13.2.3. Çevirici Satırları Yazım Biçimi

Farklı firmaların ürettiği çevirici yazılımların kaynak dosya girişleri küçük farklar dışında benzerdir. Satırın ilk kısmı, satırdaki komut adresinin belirlenmesini sağlayan ve program yazımı sırasında çeviricinin daha sonra satır eklenmesi esnekliği getiren satırın devamından ":" özel karakteri ile ayrılan "Etiket" kısmıdır. İkinci kısımda ise komut kısmı kısa komut adı ile beraber adresleme şekli belirteci ve işlenenden oluşur. Üçüncü ve son kısım, ";" özel karakteri ile ayrılan satırın devamı, satırdaki komutun açıklama kısmıdır. Aşağıda bir çevirici yazılımı için düzenlenmiş kaynak dosyası satır yazım biçimi verilmiştir.

Etiket [:] Kısa Komut Adı [;] Açıklama

13.2.4. Çevirici Yönetim Komutları

ORG (adres)
Mikroişlemci yazılımının makine diline çevrileceği program belleği, veri belleği, vektörlerin bulunduğu bellek bölgesinin başlangıç adresinin belirlenmesi için kullanılır.

END
Mikroişlemci yazılımının makine diline çevriminin bittiği yerin belirlenmesi için kullanılır.

EQU
Değişmez değer tanımlamak için kullanılır.

DFB veya FCB
1 bayt veri tanımlamak için kullanılır.

DWM veya FDB
Motorola biçimli 2 bayt veri tanımlamak için kullanılır.
16-bit değerin yüksek ağırlıklı baytı küçük değerli adreste, düşük ağırlıklı baytı büyük değerli adreste bulunur.

Örnek 13-1

```
SAYI1:  EQU   28H
        ORG   0H
VERI:   DFB   17H,32H,0F6H,SAYI1,0FEH,0DCH
SONUC:  DWM   6278H
```

Bu tanımlamalardan sonra bellek içeriği aşağıda verilen şekilde olur.

Adres	Veri
0000H	17H
0001H	32H
0002H	F6H
0003H	28H
0004H	FEH
0005H	DCH
0006H	62H
0007H	78H

13.2.5. Kaynak Dosya Örnekleri

Farklı firmaların ürettiği çevirici yazılımların kaynak dosya girişleri küçük farklar dışında benzerdir. Burada iki değişik çevirici yazılımı için düzenlenmiş kaynak dosyası örneği verilmiştir. İlk örnekte çevirici değişik mikroişlemcileri desteklediği için önce “CPU” komutu ile mikroişlemci tanımı yapılması gerekir. İkinci örnek ise tümleşik geliştirme ortamı içinde yer alan çeviricide mikroişlemci tipi program ayarları ile belirtilir. İki örnek incelenirse yukarıda da açıklandığı gibi bazı yönetim komutları dışında aynı oldukları görülür.

Örnek 13-2

```
;           0000H->00FFH ADRESLERİ ARASINDAKİ
;           BELLEK ALANININ 00H->FFH iLE DOLDURULMASI.
            CPU    "6800.TBL"   ; mikroişlemci tipinin tanımlanması
            HOF    "MOT8"       ; on altılık çıkış dosyasının tipi
            ORG    100H         ; programın başlangıç adresi
BASLA:      LDX    #0H          ; X dizin yazmacına başlangıç adresinin yükle
            CLRA                ; A akümülatörüne sıfır ilk değerinin yüklenmesi
L1:         STAA   0,X          ; A akümülatörünü (X+0) bellek adresinde sakla
            INCA                ; A akümülatöründeki değeri bir artır
            INX                 ; X dizin yazmacındaki adresi bir artır
            CPX    #100H        ; son adrese gelindi mi?
            BNE    L1           ; gelinmediyse L1 etiketli adrese giderek devam et
            NOP                 ; yoksa programı bitir.
;           VEKTÖR ADRESLERİ
            ORG    0FFFEH
            DWM    BASLA        ; yeniden başlatma vektör adresi
            END
```

Örnek 13-3

```
;           0000H->00FFH ADRESLERİ ARASINDAKİ
;           BELLEK ALANININ 00H->FFH iLE DOLDURULMASI.
            ORG    100H         ; programın başlangıç adresi
BASLA:      LDX    #0H          ; X dizin yazmacına başlangıç adresini yükle
            CLRA                ; A akümülatörüne sıfır ilk değerini yükle
L1:         STAA   0,X          ; A akümülatörünü (X+0) bellek adresinde sakla
            INCA                ; A akümülatöründeki değeri bir artır.
            INX                 ; X dizin yazmacındaki adresi bir artır.
            CPX    #100H        ; son adrese gelindi mi?
            BNE    L1           ; gelinmediyse L1 etiketli adrese giderek devam et
            NOP                 ; yoksa programı bitir.
;           VEKTÖR ADRESLERİ
            ORG    0FFFEH
            FDB    BASLA        ; yeniden başlatma vektör adresi
            END
```

13.2.6. Program Listesi Dosyası Örnekleri

Yukarıda verilen iki değişik kaynak dosyasının çevirici çıkışında oluşan program listesi dosyası örnekleri aşağıda verilmiştir. İki örnek incelenirse bunların da biçimleri dışında özünde aynı özellikte oldukları görülür. İkinci örneğin birinci örnekten farkı her satırın başına satır numarası ve komutların çevrim sayıları eklenmesidir.
Kullanıcı tarafından doğrudan görülebilen program listesi çıkış dosyası da kaynak dosyası gibi ASCII yapıda dosyadır. Program listesi dosyası kaynak dosyasında yazılan satırların çevirici yanıtlarını verir. Ayrıca varsa yazım hatalarını da belirtir ve genellikle LST (*.LST) kelimesini tip kodu olarak kabul eder.

Örnek 13-4

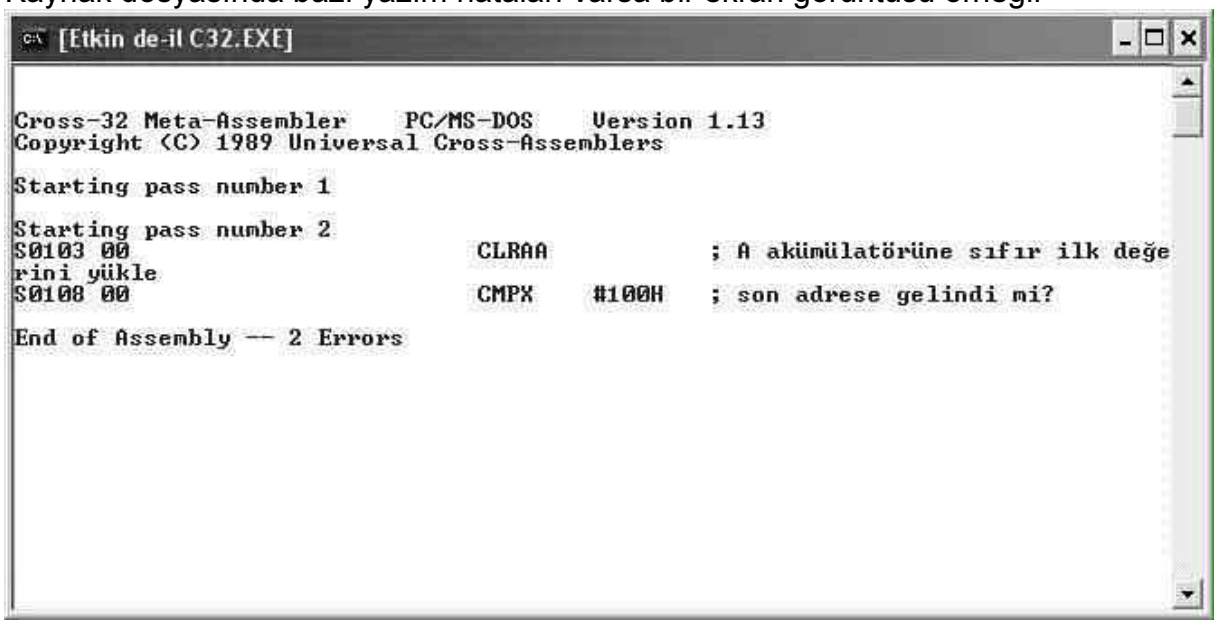
```
                ;       0000H->00FFH ADRESLERİ ARASINDAKİ
                ;       BELLEK ALANININ 00H->FFH iLE DOLDURULMASI.
0000                    CPU     "6800.TBL"  ; mikroişlemci tipinin tanımlanması
0000                    HOF     "MOT8"      ; on altılık çıkış dosyasının tipi
0100                    ORG     100H        ; programın başlangıç adresi
0100 CE0000  BASLA:     LDX     #0H         ; X dizin yazmacına başlangıç adresinin yükle
0103 4F                 CLRA                ; A akümülatörüne sıfır ilk değerinin yüklenmesi
0104 A700    L1:        STAA    0,X         ; A akümülatörünü (X+0) bellek adresinde sakla
0106 4C                 INCA                ; A akümülatöründeki değeri bir artır
0107 08                 INX                 ; X dizin yazmacındaki adresi bir artır
0108 8C0100             CPX     #100H       ; son adrese gelindi mi?
010B 26F7               BNE     L1          ; gelinmediyse L1 etiketli adrese giderek devam et
010D 01                 NOP                 ; yoksa programı bitir.
                ;       VEKTÖR ADRESLERİ
FFFE                    ORG     0FFFEH
FFFE 0100               DWM     BASLA       ; yeniden başlatma vektör adresi
0000                    END
0100  BASLA       0104  L1
```

Örnek 13-5

```
00001                           ;       0000H->00FFH ADRESLERİ ARASINDAKİ
00002                           ;       BELLEK ALANININ 00H->FFH iLE DOLDURULMASI.
00003                                   ORG     $D000   ; programın başlangıç adresi
00004   [3]  D000 CE 00 00      BASLA:  LDX     #0H     ; X dizin yazmacına başlangıç adresinin yükl
00005   [2]  D003 4F                    CLRA            ; A akümülatörüne sıfır ilk değerinin yüklenm
00006   [4]  D004 A7 00          L1:    STAA    0,X     ; A akümülatörünü (X+0) bellek adresinde sa
00007   [2]  D006 4C                    INCA            ; A akümülatöründeki değeri bir artır
00008   [3]  D007 08                    INX             ; X dizin yazmacındaki adresi bir artır
00009   [4]  D008 8C 01 00              CPX     #100H   ; son adrese gelindi mi?
00010   [3]  D00B 26 F7                 BNE     L1      ; gelinmediyse L1 etiketli adrese giderek dev
00011   [2]  D00D 01                    NOP             ; yoksa programı bitir.
00012                           ;       VEKTÖR ADRESLERİ
00013                                   ORG     0FFFEH
00014        FFFE D0 00                 FDB     BASLA   ; yeniden başlatma vektör adresi
00015                                   END

-------- SYMBOL Table --------
L1          A $D004    BASLA        A $D000
```

Kaynak dosyasında bazı yazım hataları varsa bir ekran görüntüsü örneği:



Aynı Kaynak dosyasında bazı yazım hataları varsa bir çıkış dosyası örneği:

```
                ;       0000H->00FFH ADRESLERİ ARASINDAKİ
                ;       BELLEK ALANININ 00H->FFH iLE DOLDURULMASI.
 0000                   CPU     "6800.TBL"  ; mikroişlemci tipinin tanımlanması
 0000                   HOF     "MOT8"      ; on altılık çıkış dosyasının tipi
 0100                   ORG     100H        ; programın başlangıç adresi
 0100 CE0000  BASLA:    LDX     #0H         ; X dizin yazmacına başlangıç adresinin yükle
S0103 00                CLRX                ; A akümülatörüne sıfır ilk değerinin yüklenmesi
 0104 A700      L1:     STAA    0,X         ; A akümülatörünü (X+0) bellek adresinde sakla
 0106 4C                INCA                ; A akümülatöründeki değeri bir artır
 0107 08                INX                 ; X dizin yazmacındaki adresi bir artır
S0108 00                CMPX    #100H       ; son adrese gelindi mi?
 0109 26F7              BNE     L1          ; gelinmediyse L1 etiketli adrese giderek devam et
 010B 01                NOP                 ; yoksa programı bitir.
                ;       VEKTÖR ADRESLERİ
 FFFE                   ORG     0FFFEH
 FFFE 0100              DWM     BASLA       ; yeniden başlatma vektör adresi
 0000                   END

0100  BASLA       0104  L1
```

Yukarıdaki çıkış dosyasında “S” karakteri ile hatanın bulunduğu satırın yeri belirtilmiştir.

13.2.7. Onaltılık Çıkış Dosyası Örnekleri

Onaltılık çıkış dosyası program listesi dosyasında bulunan verileri ve makine dilindeki değerleri saklar. Programlayıcı cihazı, tümleşik geliştirme ortamı, simülatör programı vs. tarafından program belleğinin programlanması için kullanılır ve genellikle HEX (*.HEX) kelimesini tip kodu olarak kabul eder.

Motorola tipindeki 8-bit onaltılık çıkış dosyasındaki her satırın sonunda yer alan kontrol toplamı (checksum) aşağıda verilen şekilde hesaplanır.
(veri değerlerinin toplamı + adres bayt değerleri + uzunluk değeri) = sonucun 1'e tümleyeni

Örnek 13-6 Motorola 8-bit onaltılık çıkış dosyasının biçimi

S00600004844521B
S111D000CE00004FA7004C088C010026F7015B
S105FFFED0002D
S9030000FC

S00600004844521B

		Adres	"H" "D" "R"	Kontrol Toplamı (checksum)
S0	06	00 00	48 44 52	1B=FF-(06+00+00+48+44+52)
	Uzunluk →	1 2	3 4 5	6

S1 11 D000 CE 00 00 4F A7 00 4C 08 8C 01 00 26 F7 01 5B

		Adres	Veriler	Kontrol Toplamı
S1	11	D0 00	CE00004FA7004C088C010026F701	5B=FF-(A4)
	Uzunluk →	1 2	3 4 5 6 7 8 9 10 11 12 13 14 15 16	17

S1 05 FFFE D0 00 2D

		Adres	Veriler	Kontrol Toplamı
S1	05	FF FE	D0 00	2D=FF-(05+FF+FE+D0+00)=FF – (2D2)
	Uzunluk →	1 2	3 4	5

S9 03 0000 FC

		Adres	Kontrol Toplamı
S9	03	00 00	FC=FF-(03+00+00)
	Uzunluk →	1 2	3