

YILDIZ TEKNİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

6800 Aile, İçin Asse, Dış, Geliş.

Yüksek Lisans Tezi

Tamer Sabak

1987

Ref
EHM
230
1987

152
72

Elektronik. Hab

YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

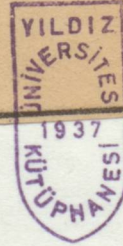
6800 AİLESİ İÇİN
ASSEMBLER/DISASSEMBLER
GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ
TAMER SABAK
ELEKTRONİK VE HABERLEŞME MÜH.

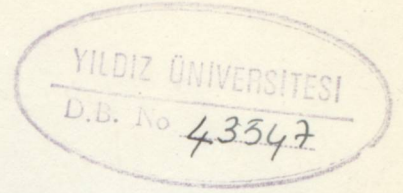
İSTANBUL 1987

YILDIZ UNİVERSİTESİ
GENEL KİTAPLIĞI

Kot : R 152
Alındığı Yer : Fen Bil. Ens. 72
Tarih : 9.12.1988
Fatura :
Fiatı : 3000 TL.
Ayniyat No : 1/21
Kayıt No : 45766
UDC : 378.242
Ek : 001.64



YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ



6800 AİLESİ İÇİN
ASSEMBLER/DISASSEMBLER
GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

TAMER SABAK

ELEKTRONİK VE HABERLEŞME MÜH.

İSTANBUL 1987

ÖNSÖZ

Tez çalışmalarım sırasında gösterdiği anlayış ve destek nedeniyle, Bölüm Müdürüm Halil Tuğal'a , Program algoritmasındaki desteği nedeniyle, Yük. Müh. Neşe Aktüre'ye ve tüm lisans üstü öğrenimim boyunca öğrettikleri ve tez çalışmalarımındaki yol göstericiliği nedeniyle Doç. Dr. Sezgin Alsan'a Teşekkür Ederim.

Tamer Sabak

Haziran - 1987

İÇİNDEKİLER

1. 6800 MİKROİŞLEMCİSİ
 - 1.1 MİKROİŞLEMCİ YAZMAÇLARI
 - 1.2 MİKROİŞLEMCİ KOMUT SETİ
 - Bellek ve birikeç işlemleri
 - Program kontrol işlemleri
 - Durum yazmacı işlemleri
2. ADRESLEME TÜRLERİ
 - 2.1 BİRİKEÇ ADRESLEME MODU
 - 2.2 ANINDA ADRESLEME MODU
 - 2.3 SIFIRINCI SAYFA VE DOĞRUDAN ADRESLEME MODU
 - 2.4 BAĞIL ADRESLEME MODU
 - 2.5 SIRA SAKLAYICISI İLE ADRESLEME MODU
3. " ASSEMBLER" PROGRAMI KULLANIMI
 - Komutlar
4. "DISASSEMBLER" PROGRAMI KULLANIMI
 - Komutlar
5. PROGRAMLAR
6. ÖRNEKLER
7. KAYNAK

ÖZET

Yüksek Lisans Tezi olarak seçilen, 6800 Ailesi İçin "Assembler - Disassembler" programı iki ayrı program olarak hazırlanmıştır.

"Assembler" Çevirici programı , programlayıcının anlayacağı komutlardan mikroişlemcinin anlayacağı makina diline çeviren programdır. Bu program iki geçişlidir. Bu nedenle programlama sırasında etiket kullanılabilir. Buda doğrudan makina dilinde yada tek geçişli çevirici (Line Assembler) bir program ile yazılırken yapılabilecek adresleme ve kodlama hatalarını ortadan kaldırır.

"Disassembler" ters çevirici programı onaltılı tabandaki sayıları programcının anlayacağı kodlara (operator) çevirir. Böylece bellekte bulunan bir program daha rahat anlaşılabilir hale getirilir.

Bu program aracılığı ile belli bir bellek bölgesindeki veriler onaltılı tabanda ve ASCII kodlanmış karakter olarak görülebilir ve değiştirilebilir.

Her iki program da birbirini yükleyebileceğinden ve aynı adres bölgesini kullanıcıya bıraktığından ayrı program olmaları bir sakınca oluşturmaz.

Özellikle "Assembler" çevirici programı, Amstrad bilgisayarında bulunan " Z-80 " çevirici programına benzetilmeye çalışılmıştır. Daha önce Z-80 programını kullananlar bu programı kullanırken yabancılik duymayacaklardır.

SUMMARY

The Assembler - Disassembler program chosen for master thesis is made up from two separate sections.

One of these is assembler which converts user program to machine language . Because it is two - pass assembler , labels can be used in programming. This provides the no-fault in machine language program.

The other is disassembler which converts hexal numbers to assembler programming language. Thus program in the memory can be easily understood

By the help of this program , datas in a specified memory area can be seen as hexal numbers and as ASCII characters and they can be chanded.

To run two separate program does not make any problem because two of them can run each other and let the same adress to user.

This program is made for Amstrad microcomputer and it can not be run in any other microcomputer without any change.

Assembler section of this program is similar to Amsoft DEVPAC for CPC

464 - 6128

GİRİŞ

Mikroişlemci işletim yazılımlarının ve hızlı çalışması istenen programların makina dilinde yazılması gereklidir. Doğrudan makina dilinde bir programın yazılması , programcı için çok zor ve hata yapmaya çok açık bir yoldur. Programcının anlayabileceği kodlarla (mnemonic) yazılmış doğrudan makina diline çeviren , makina dilindeki programları anlaşılabilir kodlara (mnemonic-operand) dönüştüren programlara ihtiyaç duyulur. Bu tür programlar "Assembler - Disassembler" programı olarak isimlendirilir.

Bu programlar, yine yine kendisi gibi programlar aracılığı ile yada ileri seviyeli (Basic, Fortran.....) yazılmış olabilirler.

Bu çalışmada da " Assembler - Disassembler " programı "Basic" dili ile yazılmıştır. Bu yazılım şekli program hızında yavaşlama getirmekle beraber zaman sınırı olmadığından sorun yaratmaz.

1. 6800 MİKROİŞLEMCİSİ

MC 6800, 8 bitlik bir mikro işlemci olup Motorola M6800 ailesi için merkezi kontrol amacı ile kullanılır. 16 adres teli ile 64 Kbyte'lık bir belleği adresliyebilir. 8 bitlik veri hattı çift yönlü iletişim sağlamaktadır. 72 komutu ve 7 adreleme modu bulunmaktadır:

- Sıfıncı sayfa adreslemesi (Direct)

- Bağlı adresleme (Relative)

- Anında adresleme (Immediate)

- Sıra saklayıcısı ile adresleme (Indexed)

- Doğrudan adresleme (Extendet)

- İmalı adresleme (Implied)

- Birikeç adreslemesi (Accumulator)

MC 6800 mikro işlemcisinin altı saklayıcısı vardır.

- İki birikeç (A ve B)

- Sıra saklayıcısı

- Program sayıcısı

- Durum saklayıcısı

- Yığın göstericisi

Saat frekansı 2 MHz'e kadar çıkar.

MC 6800 işlemcisinin bacaklarının fonksiyonları aşağıdaki gibidir.

- Saat fazı 1 ve 2 ($\phi 1$ ve $\phi 2$): Sistemin çalışabilmesi için gerekli

saat işaretleri bu uçlardan uygulanır.

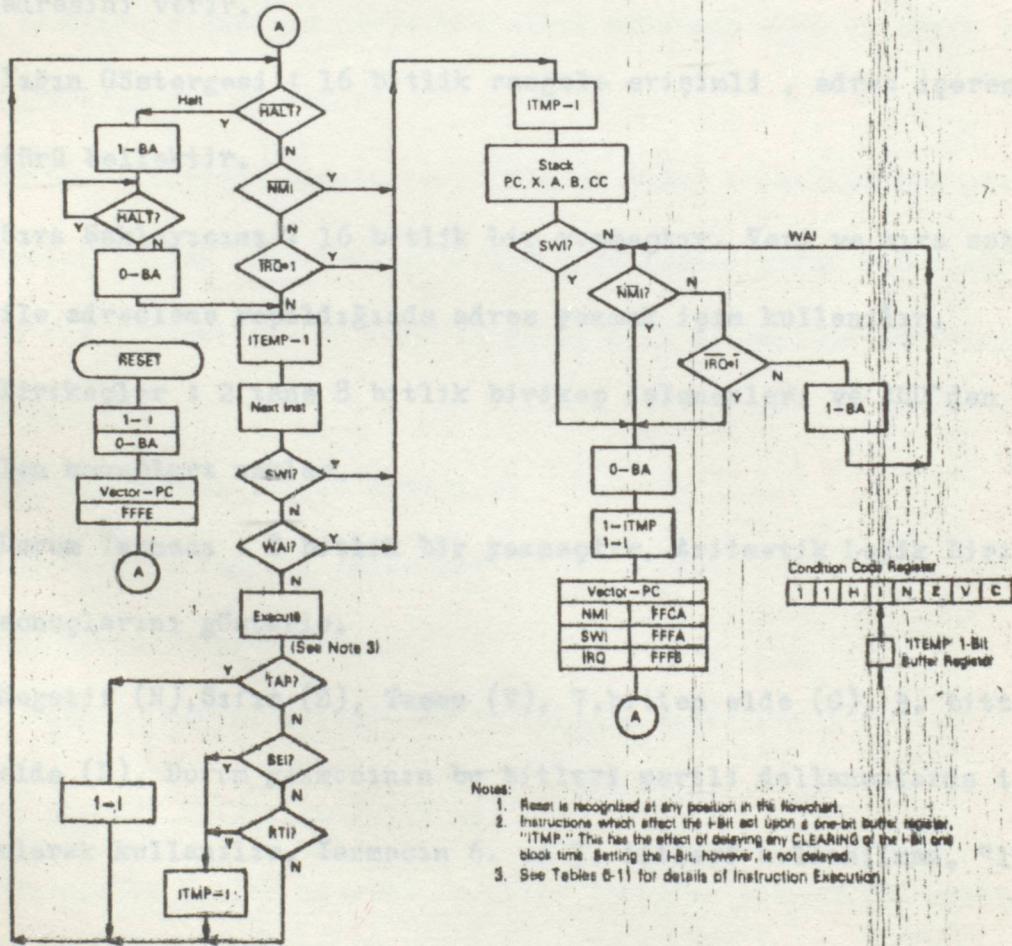
- Adres hattı (A0-A15): Bu hat 64 Kbyte'lık bir belleği adreslemek için kullanılır. Tek yönlüdür.
- Veri hattı (D0-D7): Bu hat verilerin bellek ve çevre birimleri arasındaki iletişimi için kullanılır. Çift yönlüdür.
- Veri hattı izni (DBE): Buç veri hattı üzerindeki bilginin iletiminin kontrolü için kullanılır.
- Hat geçerli ucu (BA): Aktif seviyesi "1" dir. Bu uç adres hattındaki bilginin geçerli olduğunu belirtmek için kullanılır.
- Oku/Yaz (R/W): Bu uç belleğe yada çevre birimlerine bilgi iletmeye yada onlardan bilgi alma kontrolü için kullanılır.
- Reset : Aktif "0" dir. Bu uç yardımıyla mikroislemcinin reset vektöründe yazılı olan adresten tekrar çalışmaya başlaması sağlanır.
- Kesinti isteği ($\overline{IRO}$): Sistemden bir kesinti isteği geldiğinde bu uç aktif konuma geçmeden istek geldiği an yaptığı akışın bitmesini bekler.
- Engellenemeyen Kesinti ($\overline{NMI}$) ve Kesinti Bekletme (WAIT): Sistemden NMI türü bir geldiğinde mikroislemci o an yürütmekte olduğu programı durdurur, NMI vektörünün bulunduğu yerdeki adrese yerleştirilmiş olan programı yürütür. WAIT komutu kesinti için hazırlık amacı için kullanılır. Bu mikroislemcinin kesintiye vereceği yanıtı hızlandırır.
- Üç konum kontrolü (TSC): Hatallı yazma ve okumaları engellemek için kullanılır. Aktif seviyesi "1" dir.

1.1. MİKROİŞLEMCİ YAZMAÇLARI

MC 6800 mikroişlemcisi programcı için 3 tane 16 bitlik, 3 tane de 8 bitlik yazmaç vardır.

Program Sayısı : 16 bitlik bir yazmaçtır. Yürütümdeki olan programın

adresini verir.



Notes:

1. Reset is recognized at any position in the flowchart.
2. Instructions which affect the I-Bit act upon a one-bit buffer register, "ITEMP." This has the effect of delaying any CLEARING of the I-Bit one block time. Setting the I-Bit, however, is not delayed.
3. See Tables 6-11 for details of Instruction Execution.

Şekil 1 - 2 Mikroişlemci Akış Şeması

1.1. MİKROİŞLEMCİ YAZMAÇLARI

MC 6800 mikroişlemcisinde programcının kullanacağı 3 tane 16 bitlik, 3 tane de 8 bitlik yazmaç vardır.

Program Sayacı : 16 bitlik bir yazmaçtır. Yürütülmekte olan programın adresini verir.

Yığın Göstergesi : 16 bitlik rasgele erişimli , adres içeren, oku/yaz türü bellektir.

Sıra Saklayıcısı : 16 bitlik bir yazmaçtır. Veri ve sıra saklayıcısı ile adresleme yapıldığında adres yazmak için kullanılır.

Birikeçler : 2 tane 8 bitlik birikeç işlenenleri ve ALU'dan elde edilen sonuçları saklar.

Durum Yazmacı : 8 bitlik bir yazmaçtır, Aritmetik Lojik Birimin işlem sonuçlarını gösterir.

Negatif (N), Sıfır (Z), Taşma (V), 7.bitten elde (C), 3. bitten yarım elde (H), Durum yazmacının bu bitleri şartlı dallanmalarda test biti olarak kullanılır. Yazmacın 6. ve 7. bitleri kullanılmaz, "1" dir.

1.2. MİKROİŞLEMCİNİN KOMUT SETİ

MC 6800 mikroişlemcisinin 72 farklı kaynak komutu vardır. Bunlar iki-
li, onlu aritmetik, lojik işlem, kaydırma, döndürme, kesinti, yığın
üzerinde işlemlerdir.

Komutların uzunluğu adresleme moduna göre 1-3 byte arasında değişir.

Mikroişlemci komutları genellikle 3 sınıfta toplanır.

(1) Bellek referanslı : Belli bir bellek üzerinde işlem yapar.

(2) Bellek referanssız : Belli bir belleğe gerek yoktur.

(3) Mikroişlemci ve çevre birimleri arasında veri aktarmak için I/O (giriş/çıkış) komutları.

MC 6800, arabirim adaptörleri (PIA ve ACIA) aracılığı ile çevre birimlerini bellek bölgeleri gibi, I/O komutlarına gerek kalmadan kullanır. Bu özellikleri ile MC 6800 ün komut seti şu şekilde sınıflandırılabilir.

(1) Birikeç ve bellek işlemleri

(2) Program kontrol işlemleri

(3) Durum yazmacı işlemleri

(Tabl-1 Makina dilinin 16'lı tabanda değerleri)

1) Birikeç Ve Bellek İşlemleri

Birikeç ve bellek üzerinde yapılan işlemler tabl-2`de özetlenmiştir.

2) Program Kontrol İşlemleri

Program kontrol işlemleri iki katogarıde toplana bilir.

a) Sıra saklayıcı/Yığın gösterge komutları

b) Atlama ve dallanma işlemleri

Sıra Saklayıcı/ Yığın Gösterge İşlemleri :

Mikroişlemcinin sırasaklayıcısı ve yığın göstergesi üzerinde gerçekleştirilen işlemler tab.1-3`de özetlenmiştir.

ADDRESSING MODES										BINARY/ARITHMETIC OPERATION										COND. CODE REG.				
OPERATIONS	Mnemonic	IMMED		DIRECT		INDEX		EXTND		IMPLIED		(All register labels refer to contents)	H					I						
		OP	~	OP	~	OP	~	OP	~	OP	~		OP	~	OP	~	OP	~	OP	~				
Add	ADDA	88	2 2	98	3 2	A8	5 2	B8	4 3			A + M - A	1	0	1	1	1	1	1					
	ADDB	C8	2 2	D8	3 2	E8	5 2	F8	4 3			B + M - B	1	0	1	1	1	1						
Add Accumulators	ABA									18	2 1	A + B - A	1	0	1	1	1	1	1					
Add with Carry	ADCA	89	2 2	99	3 2	A9	5 2	B9	4 3			A + M + C - A	1	0	1	1	1	1	1					
	ADCB	C9	2 2	D9	3 2	E9	5 2	F9	4 3			B + M + C - B	1	0	1	1	1	1	1					
And	ANDA	84	2 2	94	3 2	A4	5 2	B4	4 3			A - M - A	1	0	1	1	1	1	1					
	ANDB	C4	2 2	D4	3 2	E4	5 2	F4	4 3			B - M - B	1	0	1	1	1	1	1					
Bit Test	BITA	85	2 2	95	3 2	A5	5 2	B5	4 3			A - M	1	0	1	1	1	1	1					
	BITB	C5	2 2	D5	3 2	E5	5 2	F5	4 3			B - M	1	0	1	1	1	1	1					
Clear	CLR											00 - M	1	0	1	1	1	1	1					
	CLRA									4F	2 1	00 - A	1	0	1	1	1	1	1					
	CLRB									5F	2 1	00 - B	1	0	1	1	1	1	1					
Compare	CMPA	81	2 2	91	3 2	A1	5 2	B1	4 3			A - M	1	0	1	1	1	1	1					
	CMPS	C1	2 2	D1	3 2	E1	5 2	F1	4 3			B - M	1	0	1	1	1	1	1					
Compare Accumulators	CBA									11	2 1	A - B	1	0	1	1	1	1	1					
Complement, 1's	COM											M - M	1	0	1	1	1	1	1					
	COMA									43	2 1	A - A	1	0	1	1	1	1	1					
	COMB									53	2 1	B - B	1	0	1	1	1	1	1					
Complement, 2's (logical)	NEG											00 - M + M	1	0	1	1	1	1	1					
	NEGA									40	2 1	00 - A + A	1	0	1	1	1	1	1					
	NEGB									50	2 1	00 - B + B	1	0	1	1	1	1	1					
Decimal Adjust, A	DAA									18	2 1	Converts Binary Add of BCD Characters into BCD format	1	0	1	1	1	1	1					
Decrement	DEC											M - 1 - M	1	0	1	1	1	1	1					
	DECA									4A	2 1	A - 1 - A	1	0	1	1	1	1	1					
	DECB									5A	2 1	B - 1 - B	1	0	1	1	1	1	1					
Exclusive OR	EORA	80	2 2	90	3 2	A0	5 2	B0	4 3			A ⊕ M - A	1	0	1	1	1	1	1					
	EORB	C0	2 2	D0	3 2	E0	5 2	F0	4 3			B ⊕ M - B	1	0	1	1	1	1	1					
Increment	INC											M + 1 - M	1	0	1	1	1	1	1					
	INCA									4C	2 1	A + 1 - A	1	0	1	1	1	1	1					
	INCB									5C	2 1	B + 1 - B	1	0	1	1	1	1	1					
Load Accumulator	LDAA	86	2 2	96	3 2	A6	5 2	B6	4 3			M - A	1	0	1	1	1	1	1					
	LDAB	C6	2 2	D6	3 2	E6	5 2	F6	4 3			M - B	1	0	1	1	1	1	1					
Or, Inclusive	ORAA	8A	2 2	9A	3 2	AA	5 2	BA	4 3			A + M - A	1	0	1	1	1	1	1					
	ORAB	CA	2 2	DA	3 2	EA	5 2	FA	4 3			B + M - B	1	0	1	1	1	1	1					
Push Data	PSHA									36	4 1	A - Mgr, SP - 1 - SP	1	0	1	1	1	1	1					
	PSHB									37	4 1	B - Mgr, SP - 1 - SP	1	0	1	1	1	1	1					
Pop Data	PULA									32	4 1	SP - 1 - SP, Mgr - A	1	0	1	1	1	1	1					
	PULB									33	4 1	SP - 1 - SP, Mgr - B	1	0	1	1	1	1	1					
Rotate Left	ROL											M	1	0	1	1	1	1	1					
	ROLA									49	2 1	A	1	0	1	1	1	1	1					
	ROLB									59	2 1	B	1	0	1	1	1	1	1					
Rotate Right	ROR											M	1	0	1	1	1	1	1					
	RORA									48	2 1	A	1	0	1	1	1	1	1					
	RORB									58	2 1	B	1	0	1	1	1	1	1					
Shift Left, Arithmetic	ASL											M	1	0	1	1	1	1	1					
	ASLA									48	2 1	A	1	0	1	1	1	1	1					
	ASLB									58	2 1	B	1	0	1	1	1	1	1					
Shift Right, Arithmetic	ASR											M	1	0	1	1	1	1	1					
	ASRA									47	2 1	A	1	0	1	1	1	1	1					
	ASRB									57	2 1	B	1	0	1	1	1	1	1					
Shift Right, Logical	LSR											M	1	0	1	1	1	1	1					
	LSRA									46	2 1	A	1	0	1	1	1	1	1					
	LSRB									56	2 1	B	1	0	1	1	1	1	1					
Store Accumulator	STAA											A - M	1	0	1	1	1	1	1					
	STAB											B - M	1	0	1	1	1	1	1					
Subtract	SUBA	8D	2 2	9D	3 2	AD	5 2	BD	4 3			A - M + A	1	0	1	1	1	1	1					
	SUBB	CD	2 2	DD	3 2	ED	5 2	FD	4 3			B - M + B	1	0	1	1	1	1	1					
Subtract Accumulators	SBA									10	2 1	A - B - A	1	0	1	1	1	1	1					
Subst. with Carry	SBCA	8C	2 2	9C	3 2	AC	5 2	BC	4 3			A - M + C - A	1	0	1	1	1	1	1					
	SBCB	CC	2 2	DC	3 2	EC	5 2	FC	4 3			B - M + C - B	1	0	1	1	1	1	1					
Transfer Accumulator	TAA									18	2 1	A - B	1	0	1	1	1	1	1					
	TBA									17	2 1	B - A	1	0	1	1	1	1	1					
Test, Zero or Minus	TST											M - 00	1	0	1	1	1	1	1					
	TSTA									4D	2 1	A - 00	1	0	1	1	1	1	1					
	TSTB									5D	2 1	B - 00	1	0	1	1	1	1	1					

LEGEND:

- OP Operation Code (Hexadecimal)
- ~ Number of MPU Cycles
- 8 Number of Program Bytes
- 9 Arithmetic Flag
- Arithmetic Minus
- Arithmetic AND
- Mgr Contents of memory location pointed to by Bus/Port
- Bit/Byte Indicator OR
- ⊕ Exclusive-OR
- 8 Complement of 8
- Transfer Minus
- 8 Bit - Zero
- 00 Bit - Zero

CONDITION CODE SYMBOLS:

- H Half-carry from bit 3
- I Interrupt mask
- N Negative (sign bit)
- Z Zero (byte)
- V Overflow, 2's complement
- C Carry from bit 7
- B Borrow Always
- S Bit Always
- T Test and set if true, cleared otherwise
- U Not Affected

CONDITION CODE REGISTER NOTES:

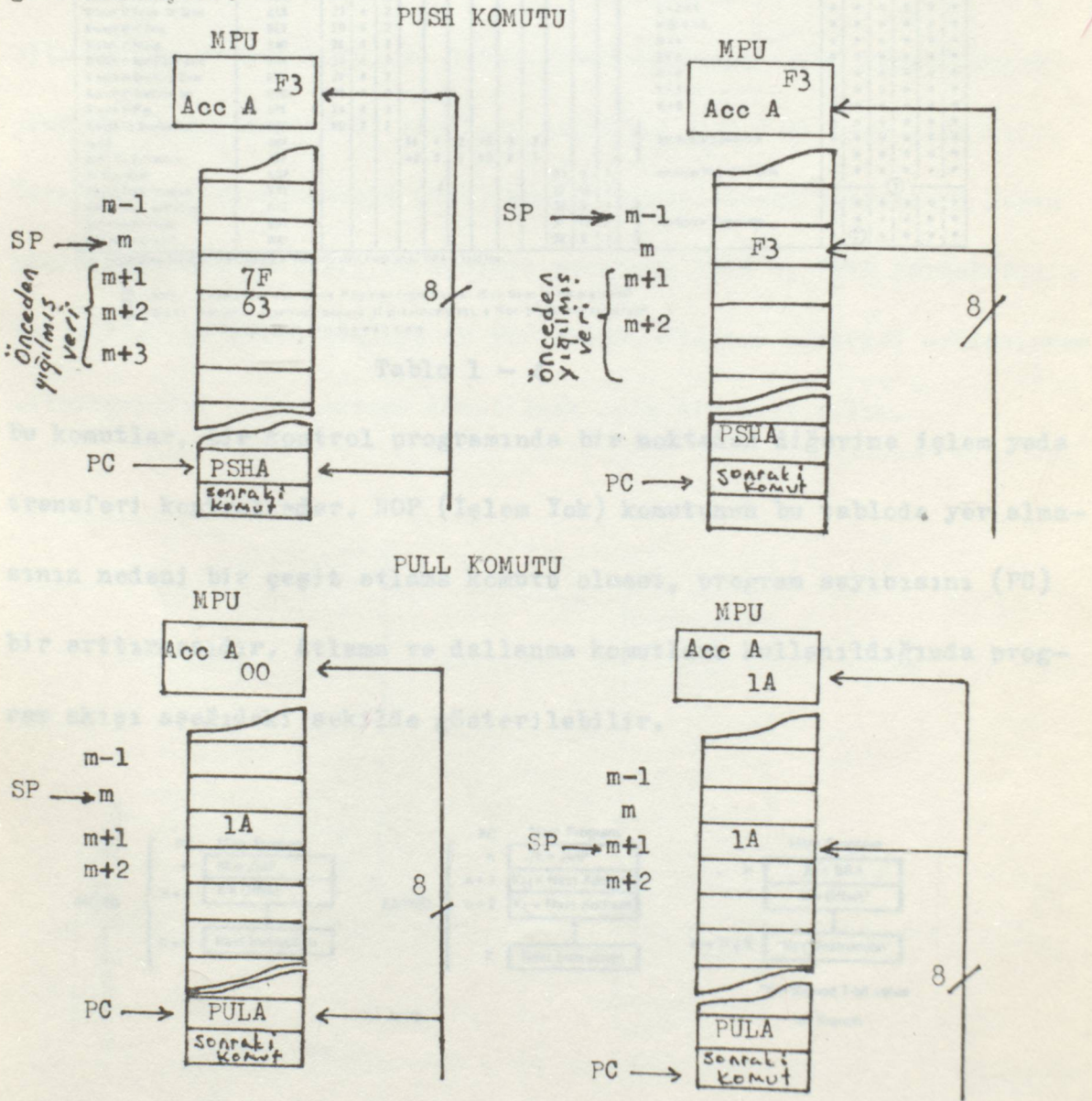
(Bit set if left is true and cleared otherwise)

- 1 (Bit V) Test: Result = 10000000
- 2 (Bit C) Test: Result = 00000000
- 3 (Bit Z) Test: Decimal value of most significant BCD Character greater than nine (Not cleared if previously set.)
- 4 (Bit V) Test: Operand = 10000000 prior to execution?
- 5 (Bit V) Test: Operand = 01111111 prior to execution?
- 6 (Bit V) Test: Set equal to result of R @ C after shift has occurred.

Note: Accumulator/PSW status register instructions are included in the 18000000 to 18000000 address range.

Tablo 1 - 3

Yığın, mikroişlemcinin oku/yaz belleğinde saklı verinin ardışıl listesi-
tesidir. Yığın göstergesi 16 bitlik bellek adresi içerir, bu adres
ile (LIFO) son giren ilk çıkar kuralına göre çalışan listenin bir u-
cuna erişilir. Yığın uzunluğu uygun bellek miktarı ile sınırlandırılır.
Yığın göstergesinin PUSH ve PULL komutlarıyla çalışma şekli 1-4
gösterilmiştir.



PC: Program sayacı SP: Yığın göstergesi

Şekil 1 - 4

Atlama Ve Dallanma İşlemleri :

Mikroişlemci ile yapılan atlama ve dallanma işlemleri tabl-4 de özetlenmiştir.

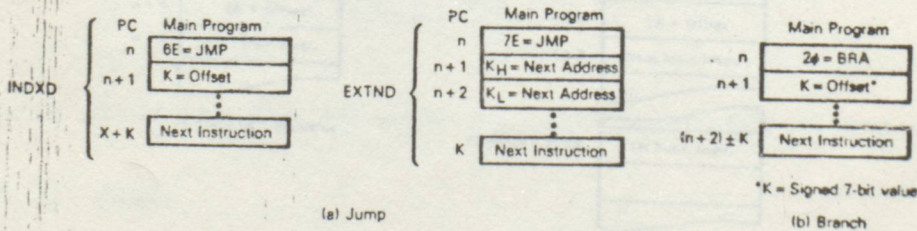
OPERATIONS	Mnemonic	RELATIVE				INDEX				EXTND				IMPLIED				BRANCH TEST	COND. CODE REG.					
		OP	~	≠	OP	~	≠	OP	~	≠	OP	~	≠	OP	~	≠	OP		S	O	N	Z	V	C
		OP	~	≠	OP	~	≠	OP	~	≠	OP	~	≠	OP	~	≠	OP		H	T	N	Z	V	C
Branch Always	BRA	20	4	2														None	*	*	*	*	*	*
Branch If Carry Clear	BCC	24	4	2														C = 0	*	*	*	*	*	*
Branch If Carry Set	BCS	25	4	2														C = 1	*	*	*	*	*	*
Branch If = Zero	BEC	27	4	2														Z = 1	*	*	*	*	*	*
Branch If > Zero	BGE	28	4	2														N ⊕ V = 0	*	*	*	*	*	*
Branch If > Zero	BGT	29	4	2														Z = 1 & N ⊕ V = 0	*	*	*	*	*	*
Branch If Higher	BH	22	4	2														C = Z = 0	*	*	*	*	*	*
Branch If < Zero	BLE	2F	4	2														Z = 1 & N ⊕ V = 1	*	*	*	*	*	*
Branch If Lower Or Same	BLS	23	4	2														C = Z = 1	*	*	*	*	*	*
Branch If < Zero	BLT	2D	4	2														N ⊕ V = 1	*	*	*	*	*	*
Branch If Minus	EMI	2E	4	2														N = 1	*	*	*	*	*	*
Branch If Not Equal Zero	ENE	2F	4	2														Z = 0	*	*	*	*	*	*
Branch If Overflow Clear	EVC	28	4	2														V = 0	*	*	*	*	*	*
Branch If Overflow Set	EVS	29	4	2														V = 1	*	*	*	*	*	*
Branch If Plus	EPL	2A	4	2														N = 0	*	*	*	*	*	*
Branch To Subroutine	BSR	8C	8	2																				
Jump	JMP				EE	4	2	7E	5	3								See Special Operations						
Jump To Subroutine	JSP				AD	8	2	8D	8	3														
No Operation	NOP													01	2	1		Advances Prog. Contr. Only						
Return From Interrupt	RTI													3B	10	1								
Return From Subroutine	RTS													3B	5	1								
Software Interrupt	SWI													3F	12	1		See Special Operations						
Wait for Interrupt*	WAI													3E	9	1								

*WAI, push Address Bus, A.M. and Data Bus in the three state mode when WAI is held low.

- ① JAI) Load Condition Code Register from Stack. (See Special Operations)
 ② Bit 1) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.

Tablo 1 - 4

Bu komutlar, bir kontrol programında bir noktadan diğerine işlem yada transferi kontrol eder. NOP (İşlem Yok) komutunun bu tabloda yer almasının nedeni bir çeşit atlama komutu olması, program sayıcısını (PC) bir arttırmasıdır. Atlama ve dallanma komutları kullanıldığında program akışı aşağıdaki şekilde gösterilebilir.



Şekil 1 - 3

Mikroişlemci atlama komutu gördüğünde sıra saklayıcısındaki değere artımı ekler. Ve bulduğu adresi bir sonra işlenecek komutun adresi gibi görür. Doğrudan adresleme modunda atlama komutundan sonra gelen iki byte bir sonra işlenecek komutun adresini doğrudan verir.

Dallanma komutu atlama komutuna benzer fakat dallanma sınırı -125, +127 byte ile sınırlanmıştır.

Altprograma atlama (JSR) ve Altprograma dallanma (BSR) komutlarının program akışı üzerindeki etkileri şekil 1-4,5,6 da gösterilmiştir.

Yazılımda istenen kesinti (SWI) sonucu mikroişlemci içerikleri yığın saklayıcısında saklanır ve kesinti programı FFFA ve FFFB adreslerinde belirtilen adresten başlayarak yürütülür. Program sayıcısı saklanırken altprogramdan doğru adrese dönebilmek için bir arttırılır.

Dallanma komutu, yığın saklayıcısı gibi diğer programlara göre farklı olarak çalışır. Diğer programların K, B, V, C bitleri dallanma şartları olarak kullanılır.

3) Dura

6. Bili

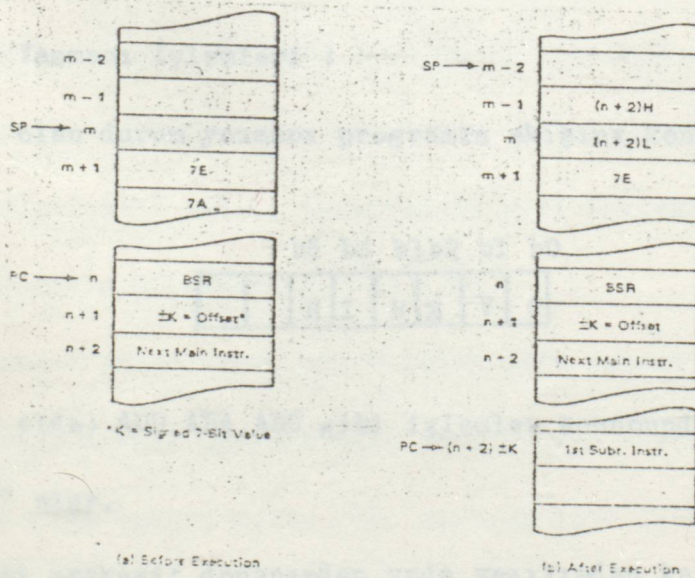
İkinci

11. Dura

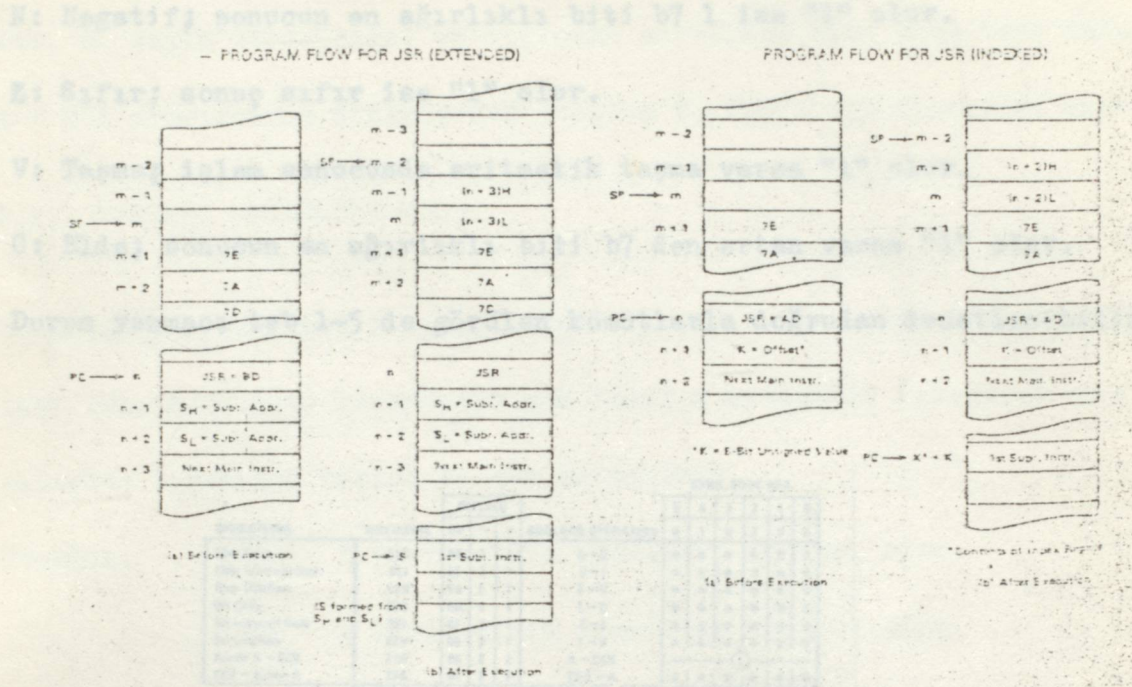
12. Dura

13. Dura

14. Dura



Şekil 1 - 4

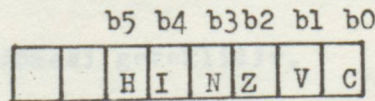


Şekil 1 - 5,6

Dallanma doğrudan yapılabildiği gibi durum yazmacına göre şartlı olarakta yapılabilir. Durum yazmacının N,Z,V,C bitleri dallanma şartlarını oluşturur.

3) Durum Yazmacı İşlemleri :

6 bitlik olan durum yazmacı programın akışını kontrol etmek için kullanılır.



H: Yarım elde; ADD,ABA,ADC gibi işlemler sonucunda b3 ten b4 e elde varsa "1" olur.

I: Kesinti maskesi; donanımdan yada yazılımdan kaynaklanan bir kesinti isteği veya SEI komutu ile "1" olur.

N: Negatif; sonucun en ağırlıklı biti b7 1 ise "1" olur.

Z: Sıfır; sonuç sıfır ise "1" olur.

V: Taşma; işlem sonucunda aritmetik taşma varsa "1" olur.

C: Elde; sonucun en ağırlıklı biti b7 den artan varsa "1" olur.

Durum yazmacı tab 1-5 de görülen komutlarla doğrudan denetlenebilir.

OPERATIONS	MNEMONIC	IMPLIED		BOOLEAN OPERATION	COND. CODE REG.					
		OP	~		S	A	3	2	1	0
Clear Carry	CLC	0C	2	1	0	0	0	0	0	R
Clear Interrupt Mask	CLI	0E	2	1	0	0	0	0	0	0
Clear Overflow	CLV	0A	2	1	0	0	0	0	0	0
Set Carry	SEC	0D	2	1	1	0	0	0	0	S
Set Interrupt Mask	SEI	0F	2	1	1	0	0	0	0	S
Set Overflow	SEV	0B	2	1	1	0	0	0	0	S
Accum A - CCR	TAP	06	2	1	1	0	0	0	0	S
CCR - Accum A	TPA	07	2	1	1	0	0	0	0	S

R = Reset

S = Set

0 = Not affected

① (ALL) Set according to the contents of Accumulator A.

Tablo 1 - 5

2. ADRESLEME TÜRLERİ

Mikroişlemci veri hattındaki 8 bitlik sayıları işler. Bu 8 bitlik sayı veri yada komut olabilir. MC 6800 ün 72 temel komut sayısı birden fazla adresleme türü olduğundan artmaktadır. Mikroişlemci yazmaçlarını ve dış bellek bölgelerini rahat, istenildiği gibi kullanabilmek için bu adresleme türlerinin bilinmesi gereklidir.

Program yazıldıktan sonra uygun işlem kodu seçimi adresleme türüne bağlıdır. İşlem koduna çevirme elle yapılıyorsa adresleme modu işlem kodu açısından önem taşır. İşlem kodu üretmek için dönüştürücü program kullanılıyorsa istenen adresleme türü bazı işaretlerle belirtilir. Örneğin " " işaretini gördüğünde dönüştürücü program anında adresleme türünü ta-

nır. 0. sayfa adreslemesi ve doğrudan adresleme eğer ilen veri değeri 0 - 255 arasındaysa dönüştürücü program 0. sayfa adreslemesini seçer.

2.1. BİRİKEÇ ADRESLEME MODU

Bu adresleme modu iki birikeç için kullanılır. Bu modda A ve B birikeçleri işlenecek veri yerindedir ama yazılım sırasında işlenecek veri yazılırken bırakılan boşluk bu kez bırakılmaz.

Örneğin, TSTB B birikecinin içeriğini kontrol eder.
 TSTA A birikecinin içeriğini kontrol eder.
 INCB B birikecinin içeriğini arttırır.

2.2. ANINDA ADRESLEME MODU

Bu adresleme modunda önemli olan işlenecek veridir.

Örneğin, LDAA #25 ; 25 sayısını A birikecine yerleştir.

Burada adres gerekmeden işlenmek istenen sayı doğrudan verilir. Bu modun seçilebilmesi için işlenecek verinin önüne "#" işareti konur.

CPx (Sıra saklayıcısını karşılaştır), LDx (Sıra saklayıcısına yükle), LDS (Yığın göstergesini yükle) komutları haricinde bu adresleme türü 1 byte'lik işlenecek veri kabul eder. Yukarıda adı geçen komutların işlenecek verileri ise 16 bitliktir.

2.3. SIFIRINCI SAYFA VE DOĞRUDAN ADRESLEME MODU

Bu adresleme modlarında işlenecek veri yerinde adres bulunmaktadır. Bu iki mod arasındaki adresleyebildikleri bölge uzunluğudur. Sıfırıncı sayfada adreslemesinde adreslenebilen bölge 0 - 255 adresleri arasındadır. 256 - 65535 adreslerine erişebilmek için doğrudan adresleme moduna gerek duyulur. Bu programda, sıfırıncı sayfa için label kullanılacağı zaman başına \$ işareti konulmalıdır.

Alt Programına Atılma), RTS (Alt Program Dönüşü) kullanılır.

Örneği: Mikroöğrenci B201 konutuna girdiğinde (son işlenen konutun sonu-

0 işlenişi), durum yapısının sıfır bitini test eder. Bu bit "0" i-

se son işlenen konut sonu B201 konutudur. Bu durumda mikroöğrenci

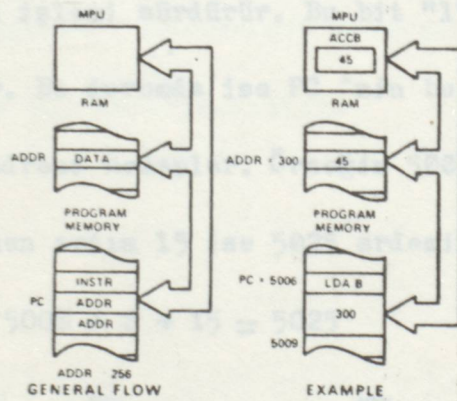
kalışık yerden ayrılır. Eğer bit "1" ise bir sonraki işleniş sonu

sıfır konutudur. Bu durumda adres artışı ekleye-

rek gideceği adresinde bu olay gerçekleş-

mektedir. Bu işleniş sonu B201 konutudur.

PC + 2 + 2 = 4



Şekil 2 - 1

2.4. BAĞIL ADRESLEME MODU

Bu adresleme modu dallanmalar için kullanılmaktadır. Her iki yöne de dallanma yapılabileceğinden artım 7 bit ile verilmekte , 8. bit işaret biti olarak kullanılmaktadır. (0: Pozitif yönde , 1: Negatif yönde) Bu

durumda ± 127 bellek gözüne dallanma yapılabilir. Bu adreslemede 2 byte'lık komut kullanıldığından bir sonraki komut PC+2 adresinde bulunacaktır. Bu durumda D dallanılacak yerin adresi ise dallanma sınırı,

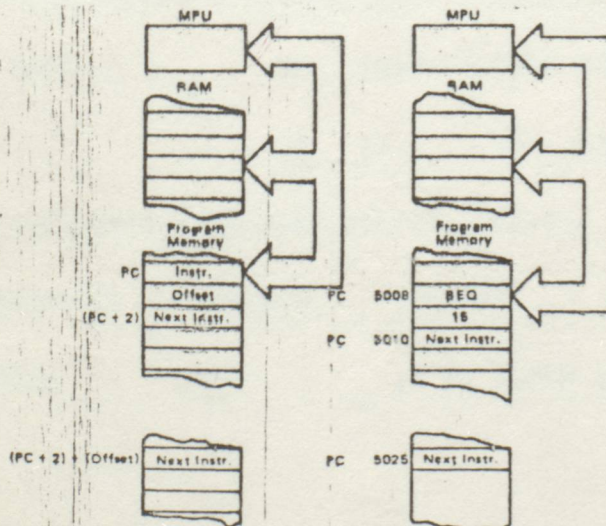
$$(PC + 2) - 127 \leq D \leq (PC + 2) + 127$$

$$PC - 127 \leq D \leq PC + 129$$

Bu sınırların dışına atlayabilmek için JMP (Şartsız Atlama), JSR (Alt Programa Atlama), RTS (Alt Program Dönüşü) kullanılır.

Örnek: Mikroişlemci BEQ komutunu gördüğünde (son işlenen komutun sonucu 0 ise dallan), durum yazmacının sıfır bitini test eder. Bu bit "0" ise son işlenen komut sonucu 0 değil demektir. Bu durumda mikroişlemci kaldığı yerden işlemi sürdürür. Bu bit "1" ise bir önceki işlem sonucu sıfır demektir. Bu durumda ise PC 'nin bulunduğu adrese artımı ekleyerek gideceği adresi hesaplar. Örneğin 500E adresinde bu olay gerçekleşmişse ve istenen artım 15 ise 5025 adresine dallanır.

$$PC + 2 + F = 500E + 2 + 15 = 5025$$



Şekil 2 - 2

2.5. SIRA SAKLAYICISI İLE ADRESLEME MODU

Bu adresleme modunda adres deęiřkendir ve sıra saklayıcısının içerięi-
ne baęlıdır.

Örneęin "STAA x " sıra saklayıcısının bölgesindeki A ya koy

Bu komut iřlenirken mikroiřlemci x ile gösterilen sıra saklayıcısının
içerięi ile belirlenen bellek gözündeki bilgiyi A birikecine yazar.

İřlenecek veri yerinde x'e eklenecek birde sayısal deęer bulunabilir.

Bu durumda mikroiřlemci istenen geręek adresi bulmak için sıra sakla-
yıcısının içindeki adrese bu deęeri ekler.

Ařaęıdaki řekil buna örnek olarak verilmiřtir.

Bu bilgiler ışığında " Assembler" programı (sayfa 27) ve "Disassembler"
programı (sayfa 33) hazırlanmıřtır.

3. "ASSEMBLER" VE KOMUTLARI

Assembler programı 6800 mikroişlemcisi için iki geçişli (two - pass) olarak tasarlanmıştır. Tüm standart 6800 komutlarını dönüştürür.

Bir program çevrilmek istendiğinde editör komutu olan " A " kullanılır.

Bu komut kullanıldığında " Option " sorulur. Geçerli seçenekler :

Option 1 : Assembly'nin 2. geçişinden sonra ekrana tüm programın listesini verir.

Option 2 : Assembly listesini üretmez, yalnızca hatalı satırların listesini verir.

Option 3 : Assembly'nin 2. geçişinden sonra yazıcıya tüm programın listesini verir.

Assembly programı iki geçişte tamamlanır. Birinci geçişte yazım hataları , komutların kodları bulunur. İkinci geçişte ise etiketler ile yazılmış satırlarda etiket değerleri yerleştirilir.

Bir assembler listenin normal şekli aşağıdaki gibidir.

7000 B7 E000 10 label : STAA E000 ; tanım

1 6 15 21 28 33. karakter olarak başlarlar.

Program yazılırken aşağıdaki şekilde yazılır :

LABEL : MNEMONIC OPERANDS COMMENT

Start : STAA E000 ;A yı E000H de sakla

Satır yazılırken yalnızca mnemonic ve operand arasında en az bir boşluk bırakılmasına özen gösterilmelidir.

Satırların incelenmesi şu şekilde yapılır. (DEL) tuşu ile silinir.

" : " buraya kadar olan kısım etikettir. Bu " : " dan sonraki boşluğa kadar olan kısım ise mnemonics'tir.

" ; " Bu işarete kadar olan kısım ise operanddır. Sonrası bilgi olarak yazılır. " ; " işareti yoksa satırın sonuna kadar olan kısım operand olarak kullanılır. İstenen satır girilir. (ENTER) tuşuna basılır.

Etiketler en çok 5 karakter olabilir. 0 - 9 ve A - F arasındaki karakterlerle başlayamaz. F - Z arasındaki karakterlerle başlayabilir. Etiketlerin başına da adresleme türlerini belirten işaretler konmalıdır.

Operand bölümünde yazılanlar etiketin dışında ondalık yada onaltılı düzende sayılar olabilir. Onaltılı düzendeki sayılar sonlarına konacakları " H " ile belirlenmelidir. programın ismi vermektedir. Tanımlanmış.

Assembler programının kendine ait iki komutu vardır. Bunlar:

ORG nl Programı yerleştirilmeye başlayacağı adresi belirlemek için kullanılır. 0 ve 2 sayıları dahil, bu satırlar arasında kalan

EQU nl Mutlaka başında bir etiket ile kullanılmalıdır. Etiketinin değeri nl ile belirlenir. 1. "a" belirtilmediğinde 1000 olarak kabul edilir.

6800 mikroişlemcisi dilinde program satırları; satır numarası, bir boşluk ve istenen satır şeklinde yazılabildiği gibi " I " komutu kul-

lanılarak da yazılabilir. Yazılan satır numarasından sonra (ENTER) tuşuna basıldığında numarası yazılan satır silinir. Satır girilirken yapılan yanlışlıklar o satırın başına kadar (DEL) tuşu ile düzeltilebilir.

- Komut : I n,m

Otomatik satır girmek için kullanılır. (n) programın yazılmaya başlanacağı satır numarasını (m) ise artımı verir. Satır numarası ekrana çıktıktan sonra yazılmak istenen satır girilir. (ENTER) tuşuna basılarak satır geçerli kılınır.

Yazılmakta olan bir satırdan vazgeçildiğinde yada yazma işlemi bittiğinde (CTRL) C ile "Insert moddan" çıkılır. Daha önce var olan bir satır numarası ile satır girildiğinde eski satır silinir , geçerli olan yenisidir.

Program 1000 satırlık bir assembler programa izin vermektedir. Yazılabilecek en büyük satır numarasıda 1000 dir.

- Komut : L n,m

Geçerli programın n ve m satırları dahil , bu satırlar arasında kalan kısmını ekranda gösterir.

"n" belirtilmediğinde 1, "m" belirtilmediğinde 1000 olarak kabul edilir.

- Komut : D n,m

"n" ve "m" satırları dahil , bu satırlar arasında kalan tüm satırları siler. "m" belirtilmediğinde yalnızca "n" ile belirtilen satır silinir. Tek satır silinmesi için istenen satır numarası yazılıp, (ENTER) tuşuna da basılabilir.

- Komut : E n

"n" nolu satırın düzeltilmesi için kullanılır. n ile başlayan satır yoksa hiç birşey olmaz. n. satır varsa o satır ekrana çıkar altına da satır no'su çıkar , satırın düzeltilmesi için bekler. Bu komutun alt komutları vardır.

" " (space) : Satır göstergesi bir arttırılır.Gösterilen yerdeki karakter ekrana basılır. Satırın sonundan daha ileriye (space) tuşu ile geçilemez.

(CTRL)I : Düzeltme modunda ekleme yapmak için kullanılır. Ekleme yapılmak istenen yere kadar (space) tuşu ile gelinir. Burada (CTRL)I kullanılarak ekleme moduna geçilir. Tekrar (CTRL)I ile bu moddan çıkılır. (space) tuşuna basılarak kalınan yerden devam edilir. satırın herhangi bir yerinde (enter) tuşuna basıldığında satırın geri kalanı ekrana basılır ve kabul edilir.

DEL : Düzeltme modunda iken (DEL) tuşu çalışır. (CTRL)I ile bu moddan çıkıldığında (DEL) tuşu çalışmaz.

(CTRL)D : Düzeltme modunda iken silinmek istenen karaktere kadar (space) tuşuna basılarak gelinir. Silinmek istenen karakter sayısı

kadar (CTRL)D ye basılır. Satırın geri kalanı (SPACE) yada (ENTER) tuşu ile tamamlanır.

(CTRL)Q : Satır düzeltme modundan çıkmak için (CTRL)Q kullanılabilir.

Bu durumda yapılan düzeltmeler dikkate alınmaz.

- Komut : P

Yazılmış bir programı saklamak için kullanılır. P tuşuna basıldığında dosya adı sorulur. Adı verilen dosyada yazılan program saklanır.

- Komut : G

Diskette yada teypte saklı olan programları okumak için kullanılır.

Bu komutta da dosya adı sorulur.

- Komut : A

Yazılan programı makina diline çevirmek için ve belleğe yerleştirmek için kullanılır. A komutu kullanıldığında "Option" sorulur. Seçenekleri:

Option 1 : Makina diline dönüştürülen tüm programı ekrana çıkarır.

Option 2 : Makina diline dönüştürülen programda yapılan hataları ekrana çıkartır.

Option 3 : Makina diline dönüştürülen tüm programı yazıcıya çıkarır.

- Komut : W

Ekranı 40 karakterden 80 karaktere dönüştürmek için kullanılır.

- Komut : J

Disassembler programına geçmek için kullanılır.

Ayrıca ,

- Anında adresleme modunda, operandın yada etiketin (label) başına

"#" konması gerektiği

- Sıfırıncı sayfa adresleme modunda yalnızca etiketin (label) başı-

na \$ konması gerektiği Unutulmamalıdır.

4. "DİSASSEMBLER" VE KOMUTLARI

"Disassembler" programı bir bellek gözünden veriyi okuyup, bunun kodunu (mnemonic) bulmayı amaçlayan bir programdır. Program,

- (Disassembler) Kodlara çevirme
- Onaltılı tabanda (Hex dump) ve ASCII karşılıkları listesi
- Editör
- Program saklama (Save)
- Program yükleme (Load)
- Yeniden adres tanımlama
- Yazıcıya çıkış
- "Assembler"e geçiş

bölümlerinden oluşur.

Program koşturaya başladığında, başlangıç adresini sorar. Bu adres hangi bölüm seçilirse (değiştirilmediği sürece) o bölümün başlangıç adresidir. Adres girildikten sonra tüm bölümlerin komutlarının bulunduđu "Help" programına geçer. Seçilen bölümden başka bir bölüme dönmek için "Help" programına dönülmelidir. Yalnızca kodlara çevirme (Disassembler) programından , Editör , "Hex dump", "Save" ve "Load" bölümlerine geçilebilir, yeni adres girilebilir.

"Disassembler" programı okuduđu ilk veriyi işlemci kodu (op code) kabul eder ve bunun operatörünü (mnemonic) bulur. Bunun operandının olup olmadığına karar verir ve varsa bir yada iki "byte"lık ayrımını yapar. Buna göre ardından gelen veriyi (yada verileri) "operand olarak yazar. Daha sonra gelen veriyi yine işlemci kodu kabul eder.

- Komut : A Programın çalıştırılması sırasında girilen yada programın ilerleyişi nedeniyle değişen adresi değiştirmek için kullanılır. Kullanıldığı bölüme döner.

- Komut : (CTRL)D

Başlangıç adresinden itibaren 20 satır olarak kodlara (disassemble) çevirir. Daha sonra her (space) tuşu ile 10 satır ilerler.

- Komut : L

Başlangıç adresinden itibaren (yada son kaldığı yerden) 80 "byte"lık bellek bölgesinin "Hex dump" ını alır ve bunların ASCII karşılıklarını yazar. (Space) tuşu ile 80 "byte" ilerler.

- Komut : E

Editör moduna geçer. Adres göstericisinin bulunduğu adresin içeriğini gösterir ve değiştirmeye olanak sağlar. Alt komutları vardır:

Z : Adres göstericisini değiştirmek için kullanılır. Adresin üzerinde ↓ çıkar.

X : Veri değiştirmek için kullanılır. Veri bölümünün üzerinde ↓ çıkar.

- : Program veri değiştirmededir. Adres azaltmak için kullanılır. Bir kez basılırsa bir önceki adrese ait veriye ulaşılır.

(ENTER): Adres arttırmak için kullanılır. Bir kez basılırsa bir sonrakine ait veriye ulaşılır.

Veri değiştirildikten sonra (ENTER) yada "-" tuşu ile adres değiştirilerek başka bir bellek gözüne ulaşılabilir, sayı yazıldığı anda belleğe yazılır.

- Komut : S Belli bir bellek bölgesini saklamak için kullanılır. Saklanacak bölgenin uzunluğunun ve başlangıç adresinin bilinmesi gerekir.

- Komut : L

Belleğe disket yada teypten daha önce saklanmış veriler yüklenir. Dosya ismi ve yerleştirilmeye başlanacak adres sorulur.

5. PROGRAMLAR

- Komut : P

Yalnız "Help" programında çalışır. Bu komut kullanıldığında, ekranın en alt satırında "Printer" yazar ve genişlik sorulur. Normalde 80 olarak verilmelidir. Bundan sora tekrar "Help" programına döner. "Disassembler" yada "Hex dump" programları kullanılırsa ekrana çıkanlar yazıcıya da çıkar. "Help" programında tekrar "P" tuşuna basılırsa yazıcı çıkışı kapanır.

- Komut : H

Herhangi bir program bölümünden "Help" programına dönmek için kullanılır.

- Komut : (CTRL) A

Assembler programını yüklemek için kullanılır.

5. PROGRAMLAR

```
10 '***** MASTER OF SCIENCE THESIS *****
20 '*** STUDY ON Ass/Diss FOR 6800 FAMILY ***
30 '
40 REM ** DISASSAMBLER **
45 MEMORY &6FFF
50 DIM T$(15,15)
60 MODE 2
65 GOSUB 610
67 PRINT #2
70 GOSUB 1000
80 INPUT #1,"START ADRESS ";A$
90 IF LEN (A$)<5 THEN GOTO 110
100 PRINT #1, "HATALI ADRES":GOTO 80
110 IF ASC(A$)<&30 OR (ASC(A$)>&39 AND ASC(A$)<&41) OR ASC(A$)>&46 THEN 100
120 A=VAL("&" + A$)
130 IF A<0 THEN A=65536+A
135 GOTO 700
140 REM DISASSAM
145 IF K$="" THEN 700
150 BB=21:AA=A
160 FOR O=1 TO BB
165 S=1
170 D=PEEK(A):NE$=""
180 DA$=HEX$(D):IF D<&10 THEN DA$="0"+DA$
182 GOSUB 1500:REM 3BYTE
184 GOSUB 2000:REM 1BYTE
186 M$=MID$(DA$,1,1):F2$=""
187 IF M$="B" OR M$="C" THEN NE$="#"
188 IF D=&B3 OR D=&B7 OR D=&BF OR D=&C3 OR D=&C7 OR D=&CC OR D=&CD OR D=&CF THEN NE$=""
189 IF M$="A" OR M$="E" OR M$="6" THEN F2$=",X"
190 N$=MID$(DA$,1,1):M$=MID$(DA$,2,1):N=VAL("&" + N$):M=VAL("&" + M$)
192 IF T$(N,M)="" THEN T$(N,M)="NOP"
193 IF T$(N,M)="NOP" THEN S=0
194 IF S=1 THEN EK=PEEK(A+1):EK$=HEX$(EK):IF EK<&10 THEN EK$="0"+EK$
196 IF S=2 THEN EK1=PEEK(A+1):EK1$=HEX$(EK1):IF EK1<&10 THEN EK1$="0"+EK1$
197 IF S=2 THEN EK2=PEEK(A+2):EK2$=HEX$(EK2):IF EK2<&10 THEN EK2$="0"+EK2$
198 IF S=2 THEN EK$=EK1$+EK2$
199 IF S=0 THEN EK$=""
200 F$=HEX$(A)
202 IF D=&A3 OR D=&E3 OR D=&EC OR D=&ED THEN F1$="":F2$=""
210 A=A+S+1
220 PRINT #1,TAB(5) F$;:PRINT #1,TAB(14) DA$+" "+EK$;:PRINT
```



```
#1,TAB(35) T$(N,M);:PRINT #1,TAB(45) NE$+EK$+F2$
225 IF PP$<>"P" THEN 230
227 PRINT #8,TAB(5) F$;:PRINT #8,TAB(14) DA$+" "+EK$;:PRINT
#8,TAB(35) T$(N,M);:PRINT #8,TAB(45) NE$+EK$+F2$
230 NEXT O
240 K$=INKEY$: IF K$="" THEN BB=10:GOTO 240
250 IF K$=" " THEN BB=10:AA=A:GOTO 160
255 IF K$="H" THEN 700
257 IF K$="E" THEN 3000
260 IF K$="A" THEN 80
265 IF K$=CHR$(12) THEN 803
267 IF K$="S" THEN 500
269 IF K$="L" THEN 550
270 GOTO 240
499 END
500 REM SAVE BIN.DOSYA
501 CLS #1:LOCATE 1,14:INPUT "FILE NAME: ",F$
503 IF F$="H" THEN 525
505 INPUT "START ADRESS: ",ADR1$:ADR1$="&"+ADR1$
510 INPUT "LENGT: ",UZ$:UZ$="&"+UZ$
515 ADR1=VAL(ADR1$):UZ=VAL(UZ$)
520 SAVE F$,B,ADR1,UZ
525 CLS #1
545 GOTO 700
550 REM LOAD BIN. FILE
552 CLS #1
555 LOCATE 1,14:INPUT "FILE NAME: ",F$
557 IF F$="H" THEN 572
560 INPUT "DESTINATION: ",ADR2$
562 IF ADR2$="H" THEN 572
563 ADR2$="&"+ADR2$
565 ADR2=VAL(ADR2$)
567 F$=F$+".BIN"
570 LOAD F$,ADR2
572 CLS #1
575 GOTO 700
610 REM HELP
620 LOCATE 35,24:PRINT " [H] E L P "
630 WINDOW #1,1,80,1,21
640 RETURN
700 CLS #1
710 LOCATE 10,1:PRINT "[A] : NEW START ADRESS"
715 LOCATE 10,2:PRINT "[S] : SAVE"
720 LOCATE 10,3:PRINT "[L] : LOAD"
725 LOCATE 10,4:PRINT "[(CTRL)D] : LIST > DISSASEMB"
730 LOCATE 10,5:PRINT "[(CTRL)L] : DISSASEMB > LIST"
735 LOCATE 10,6:PRINT "[E] : EDITOR"
737 LOCATE 10,7:PRINT "[(CTRL)D] DISSASEMB"
```



```

740 LOCATE 10,8:PRINT "[P] : Printer ON/OFF"
745 LOCATE 10,10:PRINT "[ (CTRL)A] ASSEMBLER"
780 K$=INKEY$:IF K$="" THEN 780
781 IF K$=CHR$(1) THEN RUN "6800A.BAS"
782 IF K$="S" THEN 500
784 IF K$="L" THEN 550
785 IF K$="E" THEN 3000
787 IF K$=CHR$(4) THEN 150
789 IF K$=CHR$(12) THEN 803
790 IF K$="P" THEN GOSUB 2500
792 IF K$="A" THEN CLS #1:GOTO 80
799 GOTO 780
800 'CLS #1
803 FOR R=1 TO 10
805 FOR I=0 TO 7:BN=PEEK(A+I):BN$=HEX$(BN)
807 IF BN<&10 THEN BN$="0"+BN$
808 BN$=" "+BN$
810 IF BN<33 OR BN>127 THEN BM$=" .":GOTO 825
820 BM$=" "+CHR$(BN)
825 BMT$=BMT$+BM$:BNT$=BNT$+BN$
830 NEXT I
840 PRINT #1,TAB(5) HEX$(A);:PRINT #1,TAB(12) BNT$;:PRINT #1
,TAB(50) BMT$
842 IF PP$<>"P" THEN 845
844 PRINT #8,TAB(5) HEX$(A);:PRINT #8,TAB(12) BNT$;:PRINT #8
,TAB(50) BMT$
845 A=A+8:BNT$="":BMT$="":BM$="":BN$=""
847 NEXT R
890 K$=INKEY$:IF K$="" THEN 890
893 IF K$=" " THEN 800
895 IF K$=CHR$(4) THEN A=A-80:GOTO 150
896 IF K$="H" THEN 700
899 GOTO 890
1000 T$(0,6)="TAP":T$(0,7)="TPA":T$(0,8)="INX":T$(0,9)="DEX"
:T$(0,10)="CLV":T$(0,11)="SEV":T$(0,12)="CLC":T$(0,13)="SEC"
:T$(0,14)="CLI":T$(0,15)="SEI"
1010 T$(1,0)="SAB":T$(1,1)="CBA":T$(1,6)="TAB":T$(1,7)="TBA"
:T$(1,9)="DAA":T$(1,&B)="ABA":T$(2,0)="BRA":T$(2,2)="BHI":T$(
(2,3)="BLS":T$(2,4)="BCC":T$(2,5)="BCS":T$(2,6)="BNE":T$(2,7
)="BEQ":T$(2,8)="BVC":T$(2,9)="BVS":T$(2,&A)="BPL":T$(2,&B)=
"BMI"
1020 T$(2,&C)="BGE":T$(2,&D)="BLT":T$(2,&E)="BGT":T$(2,&F)="
BLE":T$(3,0)="TSX":T$(3,1)="INS":T$(3,2)="PULA":T$(3,3)="PUL
B":T$(3,4)="DES":T$(3,5)="TXS":T$(3,6)="PSHA":T$(3,7)="PSHB"
:T$(3,9)="RST":T$(3,&B)="RTI":T$(3,&E)="WAI":T$(3,&F)="SWI"
1030 T$(4,0)="NEGA":T$(4,3)="COMA":T$(4,4)="LSRA":T$(4,6)="R
ORA":T$(4,7)="ASRA":T$(4,8)="ASLA":T$(4,9)="ROLA":T$(4,&A)="
DECA":T$(4,&C)="INCA":T$(4,&D)="TSTA":T$(4,&F)="CLRA"

```



```

1040 T$(5,0)="NEGB":T$(5,3)="COMB":T$(5,4)="LSRB":T$(5,6)="R
ORB":T$(5,7)="ASRB":T$(5,8)="ASLB":T$(5,9)="ROLB":T$(5,&A)="
DECB":T$(5,&C)="INCB":T$(5,&D)="TSTB":T$(5,&F)="CLRB"
1050 T$(6,0)="NEG":T$(6,3)="COM":T$(6,4)="LSR":T$(6,6)="ROR"
:T$(6,7)="ASR":T$(6,8)="ASL":T$(6,9)="ROL":T$(6,&A)="DEC":T$
(6,&C)="INC":T$(6,&D)="TST":T$(6,&E)="JMP":T$(6,&F)="CLR"
1060 T$(7,0)="NEG":T$(7,3)="COM":T$(7,4)="LSR":T$(7,6)="ROR"
:T$(7,7)="ASR":T$(7,8)="ASL":T$(7,9)="ROL":T$(7,&A)="DEC":T$
(7,&C)="INC":T$(7,&D)="TST":T$(7,&E)="JMP":T$(7,&F)="CLR"
1070 T$(8,0)="SUBA":T$(8,1)="CMPA":T$(8,2)="SBCA":T$(8,4)="A
NDA":T$(8,5)="BITA":T$(8,6)="LDAA":T$(8,8)="EORA":T$(8,9)="A
DCA":T$(8,&A)="ORAA":T$(8,&B)="ADDA":T$(8,&C)="CPXA":T$(8,&D
)="BSR":T$(8,&E)="LDS"
1080 T$(9,0)="SUBA":T$(9,1)="CMPA":T$(9,2)="SBCA":T$(9,4)="A
NDA":T$(9,5)="BITA":T$(9,6)="LDAA":T$(9,7)="STAA":T$(9,8)="E
ORA":T$(9,9)="ADCA":T$(9,&A)="ORAA":T$(9,&B)="ADDA":T$(9,&C)
)="CPX":T$(9,&E)="LDS":T$(9,&F)="STS"
1090 T$(&A,0)="SUBA":T$(&A,1)="CMPA":T$(&A,2)="SBCA":T$(&A,4
)="ANDA":T$(&A,5)="BIT A":T$(&A,6)="LDAA":T$(&A,7)="STAA":T$
(&A,8)="EORA":T$(&A,9)="ADCA":T$(&A,&A)="ORAA":T$(&A,&B)="AD
DA":T$(&A,&C)="CPX":T$(&A,&D)="JSR":T$(&A,&E)="LDS"
1100 T$(&A,&F)="STS":T$(&B,0)="SUBA":T$(&B,1)="CMPA":T$(&B,2
)="SBCA":T$(&B,4)="ANDA":T$(&B,5)="BITA":T$(&B,6)="LDAA":T$
(&B,7)="STAA":T$(&B,8)="EORA":T$(&B,9)="ADCA":T$(&B,&A)="ORAA
":T$(&B,&B)="ADDA":T$(&B,&C)="CPX":T$(&B,&D)="JSR"
1110 T$(&B,&E)="LDS":T$(&B,&F)="STS":T$(&C,0)="SUBB":T$(&C,1
)="CMPB":T$(&C,2)="SBCB":T$(&C,4)="ANDB":T$(&C,5)="BITB":T$
(&C,6)="LDAB":T$(&C,8)="EORB":T$(&C,9)="ADCB":T$(&C,&A)="ORAB
":T$(&C,&B)="ADDB":T$(&C,&E)="LDX"
1120 T$(&D,0)="SUBB":T$(&D,1)="CMPB":T$(&D,2)="SBCB":T$(&D,4
)="ANDB":T$(&D,5)="BITB":T$(&D,6)="LDAB":T$(&D,7)="STAB":T$
(&D,8)="EORB":T$(&D,9)="ADCB":T$(&D,&A)="ORAB":T$(&D,&B)="ADD
B":T$(&D,&E)="LDX":T$(&D,&F)="STX"
1130 T$(&E,0)="SUBB":T$(&E,1)="CMPB":T$(&E,2)="SBCB":T$(&E,4
)="ANDB":T$(&E,5)="BITB":T$(&E,6)="LDAB":T$(&E,7)="STAB":T$
(&E,8)="EORB":T$(&E,9)="ADCB":T$(&E,&A)="ORAB":T$(&E,&B)="ADD
B":T$(&E,&E)="LDX":T$(&E,&F)="STX"
1140 T$(&F,0)="SUBB":T$(&F,1)="CMPB":T$(&F,2)="SBCB":T$(&F,4
)="ANDB":T$(&F,5)="BITB":T$(&F,6)="LDAB":T$(&F,7)="STAB":T$
(&F,8)="EORB":T$(&F,9)="ADCB":T$(&F,&A)="ORAB":T$(&F,&B)="ADD
B":T$(&F,&E)="LDX":T$(&F,&F)="STX"
1190 RETURN
1500 KK$=MID$(DA$,1,1):IF KK$<>"7" AND KK$<>"8" AND KK$<>"B"
AND KK$<>"F" AND KK$<>"C" THEN RETURN
1505 IF D=&BB OR D=&FB OR D=&B9 OR D=&F9 OR D=&B4 OR D=&F4 O
R D=&B5 OR D=&F5 OR D=&7F OR D=&B1 OR D=&F1 OR D=&73 OR D=&7
0 OR D=&7A OR D=&BB OR D=&FB OR D=&7C THEN S=2
1510 IF D=&B6 OR D=&F6 OR D=&BA OR D=&FA OR D=&79 OR D=&76 O

```



```

R D=&78 OR D=&77 OR D=&74 OR D=&B7 OR D=&F7 OR D=&B0 OR D=&F
0 OR D=&B2 OR D=&F2 OR D=&7D THEN S=2
1520 IF D=&BC OR D=&CE OR D=&BE OR D=&BC OR D=&FE OR D=&BE O
R D=&FF OR D=&BF OR D=&7E OR D=&BD THEN S=2 (ADR):GOTO 3250
1590 RETURN
2000 KK$=MID$(DA$,1,1):IF KK$<>"0" AND KK$<>"1" AND KK$<>"3"
AND KK$<>"4" AND KK$<>"5" THEN RETURN
2010 IF D=&1B OR D=&4F OR D=&5F OR D=&11 OR D=&43 OR D=&53 O
R D=&40 OR D=&50 OR D=&19 OR D=&4A OR D=&5A OR D=&4C OR D=&5
C OR D=&36 OR D=&37 OR D=&32 OR D=&33 THEN S=0
2020 IF D=&49 OR D=&59 OR D=&46 OR D=&56 OR D=&48 OR D=&58 O
R D=&47 OR D=&57 OR D=&44 OR D=&54 OR D=&10 OR D=&16 OR D=&1
7 OR D=&4D OR D=&5D THEN S=0
2030 IF D=&C OR D=8 OR D=&E OR D=9 OR D=&39 OR D=&A OR D=&D
OR D=&F OR D=&B OR D=6 OR D=7 THEN S=0
2100 RETURN
2500 REM PRINTER
2510 IF PP$="P" THEN PP$="": GOTO 2550
2515 LOCATE 10,25:PRINT "PRINTER"
2520 PP$=K$:LOCATE 10,15:INPUT "WIDTH ? ",W:WIDTH W
2530 LOCATE 10,15:PRINT " ":RETURN
2550 LOCATE 10,25:PRINT " ":RETURN
3000 CLS:LOCATE 1,25:PRINT"Z:Adres X:Data -:Address Decreme
nt RETURN:Address Increment H:Help"
3010 LOCATE 10,5:PRINT" Address Data"
3020 LOCATE 10,6:PRINT"-----"
3030 ADR$=HEX$(A):DAT$=HEX$(PEEK(A)):ADR=A
3040 LOCATE 11,7:PRINT"&";ADR$:LOCATE 25,7:PRINT"&";:IF LEN(
DAT$)<2 THEN PRINT"0";DAT$ ELSE PRINT DAT$
3050 K$=INKEY$:IF K$="" THEN GOTO 3050
3065 IF K$="H" THEN CLS #2:GOSUB 610:GOTO 700
3070 IF K$="Z" OR K$="z" THEN GOTO 3160: 'ADRES
3080 IF K$="X" OR K$="x" THEN GOTO 3320: 'DATA
3090 IF K$="-" AND ADR=0 THEN GOTO 3050
3100 IF K$="-" THEN ADR=ADR-1 :ADR$=HEX$(ADR) :GOTO 3120
3110 IF K$=CHR$(13) THEN ADR=ADR+1:ADR$=HEX$(ADR) ELSE GOTO
3150
3120 IF LEN(ADR$)<4 THEN ADR$="0"+ADR$:GOTO 3120
3130 LOCATE 12,7:PRINT ADR$:LOCATE 26,7:IF LEN(HEX$(PEEK(ADR
)))<2 THEN PRINT "0";HEX$(PEEK(ADR)) ELSE PRINT HEX$(PEEK(AD
R))
3140 GOTO 3320: 'DATA
3150 GOTO 3050
3160 LOCATE 13,4:PRINT CHR$(241)
3170 K$=INKEY$:IF K$="" THEN GOTO 3170
3185 IF K$="H" THEN CLS #2:GOSUB 610:GOTO 700
3190 IF K$="Z" OR K$="z" THEN GOTO 3170
3200 IF K$="X" OR K$="x" THEN LOCATE 13,4:PRINT " ":GOTO 332

```



```

0: 'DATA
3210 IF K$="-" AND ADR=0 THEN GOTO 3170
3220 IF K$="-" THEN ADR=ADR-1 :ADR$=HEX$(ADR) :GOTO 3250
3230 IF K$=CHR$(13) THEN ADR=ADR+1:ADR$=HEX$(ADR):GOTO 3250
3240 GOTO 3290
3250 IF LEN(ADR$)<4 THEN ADR$="0"+ADR$:GOTO 3250
3260 DAT$=HEX$(PEEK(ADR+&6000))
3270 LOCATE 12,7:PRINT ADR$:LOCATE 26,7:IF LEN(DAT$)<2 THEN
PRINT "0";DAT$ ELSE PRINT DAT$
3280 LOCATE 13,4:PRINT " ":GOTO 3320: 'DATA
3290 IF ASC(K$)<&30 OR (ASC(K$)>&39 AND ASC(K$)<&41) OR ASC(
K$)>&46 THEN 3170
3300 ADR$=RIGHT$((ADR$+K$),4):ADR=VAL("&"+ADR$):LOCATE 12,7:
PRINT ADR$:LOCATE 26,7:IF LEN(HEX$(PEEK(ADR)))<2 THEN PRINT
"0";HEX$(PEEK(ADR)) ELSE PRINT HEX$(PEEK(ADR))
3310 GOTO 3170
3320 LOCATE 26,4:PRINT CHR$(241)
3330 K$=INKEY$:IF K$="" THEN GOTO 3330
3345 IF K$="H" THEN CLS #2:GOSUB 610:GOTO 700
3350 IF K$="Z" OR K$="z" THEN LOCATE 26,4:PRINT " ":GOTO 316
0
3360 IF K$="X" OR K$="x" THEN 3330
3370 IF K$="-" AND ADR=0 THEN GOTO 3330
3380 IF K$="-" THEN ADR=ADR-1 :ADR$=HEX$(ADR) :GOTO 3400
3390 IF K$=CHR$(13) THEN POKE ADR,VAL("&"+DAT$):ADR=ADR+1:AD
R$=HEX$(ADR) ELSE GOTO 3440
3400 DAT$=HEX$(PEEK(ADR))
3410 IF LEN(ADR$)<4 THEN ADR$="0"+ADR$:GOTO 3410
3420 LOCATE 12,7:PRINT ADR$:LOCATE 26,7:IF LEN(DAT$)<2 THEN
PRINT "0";DAT$ ELSE PRINT DAT$
3430 GOTO 3330
3440 IF ASC(K$)<&30 OR (ASC(K$)>&39 AND ASC(K$)<&41) OR ASC(
K$)>&46 THEN 3330
3450 DAT$=RIGHT$((DAT$+K$),2):LOCATE 26,7:PRINT DAT$:POKE (A
DR),VAL("&"+DAT$):GOTO 3330
3499 RETURN

```



```
10 '***** MASTER OF SCIENCE THESIS *****
20 '**** STUDY ON Ass/Diss FOR 6800 FAMILY ****
30 '
40 '** ASSEMBLER **
50 N11=1:DIM T$(15,15),LN$(1000),LNB$(1000)
60 MEMORY &6FFF:GOSUB 10000:REM TABLE
70 MODE 1:LOCATE 1,7
80 PRINT TAB(5);"Assemble ";SPC(5);"Insert"
90 PRINT TAB(5) "Bye ";SPC(5);"Jump to Diss."
100 PRINT TAB(5) "Delete ";SPC(5);"List"
110 PRINT TAB(5) "Edit ";SPC(5);"reNumber"
120 PRINT TAB(5) "Get text ";SPC(5);"Put text"
130 PRINT TAB(5) "Help ";SPC(5);"Width":PRINT:PRINT
140 PRINT CHR$(197) " ";:LINE INPUT "COM$:C$=MID$(COM$,1,1)
150 IF C$="A" OR C$="a" THEN GOSUB 1000
160 IF C$="B" OR C$="b" THEN END
170 IF C$="D" OR C$="d" THEN GOSUB 2000
180 IF C$="E" OR C$="e" THEN GOSUB 3000
190 IF C$="G" OR C$="g" THEN GOSUB 4000
200 IF C$="H" OR C$="h" THEN GOTO 80
210 IF C$="I" OR C$="i" THEN GOSUB 5000
220 IF C$="J" OR C$="j" THEN RUN "6800D"
230 IF C$="L" OR C$="l" THEN GOSUB 6000
240 IF C$="N" OR C$="n" THEN GOSUB 7000
250 IF C$="P" OR C$="p" THEN GOSUB 8000
260 IF C$="W" OR C$="w" THEN MODE 2
270 IF VAL(COM$)<>0 THEN GOSUB 2500
280 GOTO 140
1000 'A S S E M B L E R
1005 SE1=0
1007 FOR I=1 TO N11:LNB$(I)="":NEXT I
1010 FOR I=1 TO N11
1015 LN$="":LNB$=""
1025 IF LN$(I)="" THEN 1115
1030 LN$=LN$(I)
1035 S1=INSTR(1,LN$,";"):IF S1=0 THEN S1=1:S11=S1 ELSE S11=S1+1
1040 S2=INSTR(S11,LN$," ")
1041 IF S2=0 AND S1=0 AND LEN(LN$)>4 THEN GOTO 1110
1042 IF S2=0 AND LEN(MID$(LN$,S1+1))>4 THEN GOTO 1110
1043 IF S2=0 THEN S2=LEN(LN$)+1:S22=S2:S3=S2:S33=S2:GOTO 1070
1045 IF S2=S11 THEN S11=S11+1:GOTO 1040
1050 S22=S2+1
1055 S3=INSTR(S22,LN$,";"):IF S3=0 THEN S33=S22:GOTO 1070
1060 IF S3=S22 THEN S22=S22+1:GOTO 1055
1065 S33=S3+1
1070 S4=INSTR(S33,LN$,";"):IF S4=0 AND S3=0 THEN S3=LEN(LN$)+1:S4=S3:GOTO 1085
```



```

1075 IF S3=0 THEN S3=S4:S33=S4:GOTO 1085
1080 IF S4=0 THEN S4=S3
1085 IF S1<>1 THEN LNB#=LNB#+MID$(LN$,1,S1-1)+SPACE$(5-(S1-1))+" " ELSE LNB#=LNB#+SPACE$(7)
1090 LNB#=LNB#+MID$(LN$,S11,S2-S11)+SPACE$(4-(S2-S11))+" "
1095 LNB#=LNB#+MID$(LN$,S22,S3-S22)+SPACE$(6-(S3-S22))+" "
1100 LNB#=LNB#+MID$(LN$,S4)
1105 LN$(I)=LNB$:GOTO 1115
1110 LNB$(I)="SYNTAX ERROR":SE1=SE1+1
1115 NEXT I
1125 SE2=0:ADR=&7000:ADR$=HEX$(ADR):S=0
1127 FOR T=0 TO 10:DAT$(T)="":DAT(T)=0:NEXT T
1130 FOR I=1 TO N11
1135 ' O R G
1140 LN$="":LNB$="":L=1
1145 IF LN$(I)="" OR LNB$(I)="SYNTAX ERROR" THEN 1505
1150 IF MID$(LN$(I),8,4)="ORG " THEN ADR$=MID$(LN$(I),14,6)
ELSE GOTO 1185
1155 IF RIGHT$(ADR$,L)=SPACE$(L) THEN L=L+1:GOTO 1155
1160 IF RIGHT$(ADR$,L)="H"+SPACE$(L-1) THEN ADR$=HEX$(VAL("&"+MID$(ADR$,1,6-L))) ELSE ADR$=HEX$(VAL(MID$(ADR$,1,7-L)))
1165 IF VAL("&"+ADR$)<&7000 THEN LNB$(I)="WRONG START ADDRESS":SE2=SE2+1:GOTO 1505
1170 IF LEN(ADR$)<4 THEN ADR$="0"+ADR$:GOTO 1170
1175 OPC$="":OP$="":AD$=ADR$:GOTO 1500
1180 ' E Q U
1185 LN$="":LNB$="":L=1
1190 IF MID$(LN$(I),8,4)="EQU " AND MID$(LN$(I),1,7)="" THEN LNB$(I)="SYNTAX ERROR":SE2=SE2+1:GOTO 1505
1195 IF MID$(LN$(I),8,4)="EQU " THEN DAT$=MID$(LN$(I),14,6)
ELSE GOTO 1225
1200 IF RIGHT$(DAT$,L)=SPACE$(L) THEN L=L+1:IF L=7 THEN LNB$(I)="SYNTAX ERROR":SE2=SE2+1:GOTO 1505 ELSE GOTO 1200
1205 IF RIGHT$(DAT$,L)="H"+SPACE$(L-1) THEN DAT$="&"+MID$(DAT$,1,6-L):DAT=VAL(DAT$):GOTO 1215
1210 DAT$="&"+HEX$(VAL(MID$(DAT$,1,7-L))):DAT=VAL(DAT$)
1215 DAT$(S)=MID$(LN$(I),1,INSTR(1,LN$(I),":")-1):DAT(S)=DAT:S=S+1
1220 OPC$="":OP$="":AD$=ADR$:GOTO 1500
1225 ' O T H E R S
1230 MNE$=MID$(LN$(I),8,4):IF RIGHT$(MNE$,1)=" " THEN MNE$=MID$(MNE$,1,3)
1235 LM=LEN(MNE$):OP$=MID$(LN$(I),14,6):IF OP$<>" " THEN N GOTO 1295
1240 ' MNE.S WITHOUT OPER.
1245 IF LM=3 THEN W=&0:GOSUB 1800:IF CTRL=1 THEN GOTO 1290
1250 IF LM=3 THEN W=&1:GOSUB 1800:IF CTRL=1 THEN GOTO 1290
1255 IF LM=3 THEN W=&3:GOSUB 1800:IF CTRL=1 THEN GOTO 1290 E

```



```

1405 L=1
1410 IF RIGHT$(OP$,L)=SPACE$(L) THEN L=L+1:GOTO 1410
1415 IF RIGHT$(OP$,L)="H"+SPACE$(L-1) THEN OP$=MID$(OP$,1,6-
L):GOTO 1422
1420 OP$=HEX$(VAL(MID$(OP$,1,7-L)))
1422 IF LEN(OP$)<2 THEN OP$="0"+OP$:GOTO 1422
1425 IF VAL("&"+OP$)=0 AND OP$<>STRING$(LEN(OP$),"0") THEN M
ID$(LN$(I),7,1)="*":GOTO 1505
1430 OPC$=HEX$(W)+HEX$(Q):AD$=ADR$:ADR$=HEX$(VAL("&"+ADR$)+2
):GOTO 1500
1435 '      DIRECT AND EXTEND
1440 L=1
1445 IF RIGHT$(OP$,L)=SPACE$(L) THEN L=L+1:GOTO 1445
1450 OP$=MID$(OP$,1,7-L)
1455 IF LEN(OP$)=6 THEN MID$(LN$(I),7,1)="*":GOTO 1505
1460 IF RIGHT$(OP$,1)="H" THEN OP$=MID$(OP$,1,LEN(OP$)-1) EL
SE   OP$=HEX$(VAL(MID$(OP$,1,LEN(OP$))))
1465 IF LEFT$(OP$,1)="$" THEN L=2 ELSE L=4
1466 IF L=2 THEN OP$=MID$(OP$,2,LEN(OP$)-1)
1467 IF LEN(OP$)<L THEN OP$="0"+OP$:GOTO 1467
1470 IF VAL("&"+OP$)=0 AND OP$<>STRING$(LEN(OP$),"0") THEN M
ID$(LN$(I),7,1)="*":GOTO 1505
1475 IF L=2 THEN W=&9:GOSUB 1800:IF CTRL=1 GOTO 1497
1480 IF L=2 THEN W=&D:GOSUB 1800:IF CTRL=1 GOTO 1497 ELSE LN
B$(I)="SYNTAX ERROR":SE2=SE2+1:GOTO 1505
1485 IF L=4 THEN W=&7:GOSUB 1800:IF CTRL=1 GOTO 1497
1490 IF L=4 THEN W=&B:GOSUB 1800:IF CTRL=1 GOTO 1497
1495 IF L=4 THEN W=&F:GOSUB 1800:IF CTRL=1 GOTO 1497 ELSE LN
B$(I)="SYNTAX ERROR":SE2=SE2+1:GOTO 1505
1497 OPC$=HEX$(W)+HEX$(Q):AD$=ADR$:IF L=2 THEN ADR$=HEX$(VAL
("&"+ADR$)+2) ELSE IF L=4 THEN ADR$=HEX$(VAL("&"+ADR$)+3)
1500 LNB$(I)=AD$+" "+OPC$+" "+OP$
1505 NEXT I
1506 END
1508 SE3=0
1510 FOR I=1 TO N11
1512 IF LN$(I)="" THEN 1605
1515 IF MID$(LN$(I),7,1)<>"*" THEN 1605
1516 OP$=MID$(LN$(I),14,6):L=1
1517 IF RIGHT$(OP$,L)=SPACE$(L) THEN L=L+1:GOTO 1517
1518 OP$=MID$(OP$,1,7-L)
1520 FOR T=0 TO 10
1525 IF OP$<>DAT$(T) THEN 1550
1530 DAT$=HEX$(DAT(T))
1535 IF LEN(DAT$)>2 AND LEN(DAT$)<4 THEN DAT$="0"+DAT$:GOTO
1535
1545 LNB$(I)=LNB$(I)+" "+DAT$:MID$(LN$(I),7,1)=" ":GOTO 1605
1550 NEXT T

```



```
1555 IF LEN(OP$)<5 THEN OP$=OP$+" ":GOTO 1555
1560 FOR T=1 TO N11
1565 IF LN$(T)="" THEN 1590
1570 LB$=MID$(LN$(T),1,5)
1575 IF LB$="" THEN 1590
1580 IF LB$<>OP$ THEN 1590
1585 LNB$(I)=LNB$(I)+" "+MID$(LNB$(T),1,4):MID$(LN$(I),7,1)=
" "
1590 NEXT T
1600 LNB$(I)="SYNTAX ERROR":SE3=SE3+1
1605 NEXT I
1700 ' LIST OF ASS.
1705 INPUT "Option: ",O
1707 PRINT"Pass 1 Errors: ";SE1+SE2
1710 IF O<1 OR O>3 THEN 1705
1715 IF O=2 THEN 1750
1717 IF O=1 THEN O=0 ELSE O=8
1720 FOR I=1 TO N11
1725 IF LN$(I)="" THEN 1745
1730 IF LNB$(I)="SYNTAX ERROR" THEN PRINT #0,TAB(35)I,LN$(I)
:PRINT #0,TAB(30)"* Syntax Error *" ELSE 1740
1735 IF INKEY$="" THEN GOTO 1735
1740 PRINT #0,TAB(5)LNB$(I)TAB(25)I" "LN$(I)
1745 NEXT I
1750 IF O<>2 THEN 1790
1755 FOR I=1 TO N11
1760 IF LN$(I)="" THEN 1770
1765 IF LNB$(I)="SYNTAX ERROR" THEN PRINT TAB(35)LN$(I):PRIN
T TAB(30)"* Syntax Error *"
1770 NEXT I
1790 PRINT"Pass 2 Errors: ";SE1+SE2+SE3
1795 RETURN
1800 ' SEARCH FOR MNE.
1805 FOR Q=0 TO &F:IF MNE$=T$(W,Q) THEN CTRL=1:RETURN ELSE N
EXT Q
1810 CTRL=0:RETURN
2000 ' D E L E T E
2010 IF LEN(COM$)=1 THEN RETURN
2020 N=INSTR(1,COM$,"")
2030 IF N=0 THEN DN=VAL(MID$(COM$,2,LEN(COM$)-1)):LN$(DN)=""
:RETURN
2040 DN1=VAL(MID$(COM$,2,N-2)):DN2=VAL(MID$(COM$,N+1,LEN(COM
$)-N))
2050 IF DN1=0 OR DN2=0 THEN RETURN
2060 FOR I=DN1 TO DN2:LN$(I)="":NEXT I
2070 RETURN
2500 ' L N D E L E T E      O R      L N I N S E R T
2510 DL$=STR$(VAL(COM$)):DL$=MID$(DL$,2,LEN(DL$)-1)
2520 IF DL$=COM$ THEN LN$(VAL(DL$))="":GOTO 2560
```



```
LSE LNB$(I)="SYNTAX ERROR":SE2=SE2+1:GOTO 1505
1260 IF LM=4 THEN W=&4:GOSUB 1800:IF CTRL=1 THEN GOTO 1290
1265 IF LM=4 THEN W=&5:GOSUB 1800:IF CTRL=1 THEN GOTO 1290
1270 IF LM=4 AND MNE$=T$(3,2) THEN W=3:Q=2:GOTO 1290
1275 IF LM=4 AND MNE$=T$(3,3) THEN W=3:Q=3:GOTO 1290
1280 IF LM=4 AND MNE$=T$(3,6) THEN W=3:Q=6:GOTO 1290
1285 IF LM=4 AND MNE$=T$(3,7) THEN W=3:Q=7:GOTO 1290 ELSE LN
B$(I)="SYNTAX ERROR":SE2=SE2+1:GOTO 1505
1290 OPC$=HEX$(W)+HEX$(Q):OP$="":AD$=ADR$:ADR$=HEX$(VAL("&"+
ADR$)+1):GOTO 1500
1295 ' IMMEDIATE
1300 IF MID$(OP$,1,1)<>"#" THEN GOTO 1340
1305 W=&8:GOSUB 1800:IF CTRL=1 THEN GOTO 1315
1310 W=&C:GOSUB 1800:IF CTRL=1 THEN GOTO 1315 ELSE LNB$(I)="
SYNTAX ERROR":SE2=SE2+1:GOTO 1505
1315 L=1
1320 IF RIGHT$(OP$,L)=SPACE$(L) THEN L=L+1:GOTO 1320
1325 IF RIGHT$(OP$,L)="H"+SPACE$(L-1) THEN OP$=MID$(OP$,2,5-
L) ELSE OP$=HEX$(VAL(MID$(OP$,2,7-L)))
1327 IF (MNE$<>"CPXA" AND MNE$<>"LDX" AND MNE$<>"LDS") AND L
EN(OP$)>3 THEN LNB$(I)="SYNTAX ERROR":SE2=SE2+1:GOTO 1505
1330 IF VAL("&"+OP$)=0 AND OP$<>STRING$(LEN(OP$),"0") THEN M
ID$(LN$(I),7,1)="*":GOTO 1505
1332 IF (MNE$="CPXA" OR MNE$="LDX" OR MNE$="LDS") AND LEN(OP
$)<4 THEN OP$="0"+OP$:GOTO 1332
1334 IF LEN(OP$)<2 THEN OP$="0"+OP$:GOTO 1334
1335 OPC$=HEX$(W)+HEX$(Q):AD$=ADR$:IF MNE$="CPXA" OR MNE$="L
DX" OR MNE$="LDS" THEN ADR$=HEX$(VAL("&"+ADR$)+3) ELSE ADR$=
HEX$(VAL("&"+ADR$)+2)
1337 GOTO 1500
1340 ' INDEXED
1345 IF INSTR(1,OP$,"X")=0 THEN GOTO 1385
1350 W=&6:GOSUB 1800:IF CTRL=1 THEN GOTO 1365
1355 W=&A:GOSUB 1800:IF CTRL=1 THEN GOTO 1365
1360 W=&E:GOSUB 1800:IF CTRL=1 THEN GOTO 1365 ELSE LNB$(I)="
SYNTAX ERROR":SE2=SE2+1:GOTO 1505
1365 OP$=MID$(OP$,1,INSTR(1,OP$,"X")-1)
1370 IF RIGHT$(OP$,1)="H" THEN OP$=MID$(OP$,1,LEN(OP$)-1) EL
SE OP$=HEX$(VAL(MID$(OP$,1,LEN(OP$))))
1372 IF LEN(OP$)<2 THEN OP$="0"+OP$:GOTO 1372
1375 IF VAL("&"+OP$)=0 AND OP$<>STRING$(LEN(OP$),"0") THEN M
ID$(LN$(I),7,1)="*":GOTO 1505
1380 GOTO 1430
1385 ' RELATIVE
1390 IF LM<3 OR LEFT$(MNE$,1)<>"B" THEN GOTO 1435
1395 W=&2:GOSUB 1800:IF CTRL=1 THEN GOTO 1405
1400 IF MNE$=T$(8,&D) THEN W=&B:Q=&D ELSE LNB$(I)="SYNTAX ER
ROR":SE2=SE2+1:GOTO 1505
```



```
2530 NB=INSTR(1,COM$," "):IF NB=0 THEN PRINT"** SYNTAX ERROR  
**":RETURN  
2540 LN$(VAL(DL$))=MID$(COM$,NB+1,LEN(COM$)-NB)  
2550 IF N11<VAL(DL$) THEN N11=VAL(DL$)  
2560 RETURN  
3000 ' E D I T  
3010 IF LEN(COM$)=1 THEN RETURN  
3020 N=INSTR(1,COM$,""):IF N<>0 THEN RETURN  
3030 EL=VAL(MID$(COM$,2,LEN(COM$)-1)):E$="":E=1  
3040 PRINT TAB(5) EL;" ";LN$(EL):PRINT TAB(5) EL;" ";  
3050 FOR I=1 TO LEN(LN$(EL))  
3060 EL$=INKEY$:IF EL$="" THEN 3060  
3070 IF EL$=" " THEN E=-E:GOSUB 3180  
3075 IF EL$="" THEN PRINT:RETURN  
3080 IF EL$=CHR$(13) THEN PRINT MID$(LN$(EL),I,LEN(LN$(EL))-  
I+1):E$=E$+MID$(LN$(EL),I,LEN(LN$(EL))-I+1):GOTO 3160  
3090 IF EL$=" " THEN PRINT MID$(LN$(EL),I,1):E$=E$+MID$(LN$(  
EL),I,1):GOTO 3110  
3100 IF EL$<>"" THEN 3060  
3110 NEXT I  
3120 EL$=INKEY$:IF EL$="" THEN 3120  
3130 IF EL$<>" " THEN GOTO 3150  
3140 GOSUB 3180  
3150 PRINT  
3160 LN$(EL)=E$  
3170 RETURN  
3180 EL$=INKEY$:IF EL$="" THEN 3180  
3190 IF EL$=" " AND E=-1 THEN E=-E:RETURN  
3200 IF EL$=CHR$(13) THEN RETURN  
3202 IF EL$=CHR$(127) AND E$="" THEN GOTO 3180  
3203 IF EL$=CHR$(127) THEN E$=MID$(E$,1,LEN(E$)-1):PRINT CHR  
$(&D)SPACE$(40)CHR$(&D);TAB(5)N1;SPC(1);E$;:GOTO 3180  
3205 IF EL$="" THEN RETURN  
3210 PRINT EL$;:E$=E$+EL$:GOTO 3180  
4000 REM GET TEXT  
4010 CLS:LOCATE 10,10:INPUT "FILE NAME : ",GT$  
4020 OPENIN GT$  
4030 INPUT #9,N11  
4040 FOR NE=1 TO N11  
4050 INPUT #9,LN$(NE)  
4060 NEXT NE  
4070 CLOSEIN  
4080 LOCATE 10,10:PRINT"  
4099 RETURN  
5000 ' I N S E R T  
5010 IF LEN(COM$)=1 THEN N1=10:N2=N1:GOTO 5060  
5020 N=INSTR(1,COM$,"")  
5030 IF N=0 THEN N1=VAL(MID$(COM$,2,LEN(COM$)-1)):N2=10:GOTO
```



```

5050
5040 N1=VAL(MID$(COM$,2,N-2)):N2=VAL(MID$(COM$,N+1,LEN(COM$)-N))
5050 IF N1=0 THEN RETURN
5060 LN$="":PRINT TAB(5) N1;SPC(1);
5070 IF LN$(N1)<>" " THEN PRINT"*";
5080 L$=INKEY$:IF L$="" THEN 5080
5090 IF L$="" THEN PRINT:IF N11<N1 THEN N11=N1:RETURN ELSE RETURN
5100 IF L$=CHR$(13) AND LN$(N1)<>" " AND LN$="" THEN N1=N1+N2:GOTO 5060
5110 IF L$=CHR$(127) AND LN$="" THEN GOTO 5080
5120 IF L$=CHR$(127) THEN LN$=MID$(LN$,1,LEN(LN$)-1):PRINT CHR$(&D)SPACE$(40)CHR$(&D);TAB(5)N1;SPC(1);LN$;GOTO 5080
5130 IF L$=CHR$(13) THEN 5180
5140 IF L$=" " AND LN$="" THEN PRINT SPC(7);LN$=LN$+SPACE$(7):GOTO 5080
5150 IF L$=" " AND LEN(LN$)<7 THEN PRINT SPC(7-LEN(LN$));LN$=LN$+SPACE$(7-LEN(LN$)):GOTO 5080
5160 IF L$=" " AND (LEN(LN$)>8 AND LEN(LN$)<11) THEN PRINT SPC(12-LEN(LN$));LN$=LN$+SPACE$(12-LEN(LN$)):GOTO 5080
5170 PRINT L$;LN$=LN$+L$:GOTO 5080
5180 LN$(N1)=UPPER$(LN$):N1=N1+N2:GOTO 5060
6000 'LIST IF ONLY "L"
6010 IF LEN(COM$)<>1 THEN 6030
6020 NL1=1:NL2=N11-N2:GOTO 6150
6030 'LIST IF L,n1
6040 IF MID$(COM$,2,1)<>" " THEN 6070
6050 NL1=1:NL2=VAL(MID$(COM$,3,LEN(COM$)-2))
6060 IF NL2=0 THEN RETURN ELSE GOTO 6150
6070 'LIST IF Ln1,
6080 N=INSTR(1,COM$,""):IF N=0 THEN RETURN ELSE IF N<>LEN(COM$) THEN 6110
6090 NL1=VAL(MID$(COM$,2,N-2)):NL2=N11
6100 IF NL1=0 THEN RETURN ELSE GOTO 6150
6110 'LIST IF Ln1,n2
6120 N=INSTR(1,COM$,""):IF N=0 THEN RETURN
6130 NL1=VAL(MID$(COM$,2,N-2)):NL2=VAL(MID$(COM$,N+1,LEN(COM$)-N))
6140 IF NL1=0 OR NL2=0 THEN RETURN ELSE GOTO 6150
6150 'LIST
6160 FOR I=NL1 TO NL2
6170 IF LN$(I)="" THEN GOTO 6210
6180 I$=STR$(I)
6190 IF LEN(I$)<4 THEN I$=" "+I$:GOTO 6190
6200 PRINT TAB(4);I$;" ";LN$(I)
6210 NEXT I
6220 RETURN

```



```

7000 ' R E N U M B E R
7010 IF LEN(COM$)=1 THEN RETURN
7020 N=INSTR(1,COM$,"")
7030 NR1=VAL(MID$(COM$,2,LEN(COM$)-1))
7040 IF N=0 THEN S=10 ELSE S=VAL(MID$(COM$,N+1,LEN(COM$)-N))
7050 FOR I=1 TO N11
7060 IF LN$(I)="" THEN 7080
7070 LNB$(NR1)=LN$(I):NR1=NR1+S
7080 NEXT I
7090 FOR I=1 TO 1000
7100 LN$(I)=LNB$(I):LNB$(I)=""
7110 NEXT I
7120 IF N11<NR1-S THEN N11=NR1-S
7130 RETURN
7140 END

8000 REM PUT TEXT
8010 CLS:LOCATE 10,10:INPUT " FILE NAME : ",PTX$
8020 OPENOUT PTX$
8030 WRITE #9,N11
8040 FOR NE=1 TO N11
8050 WRITE #9,LN$(NE)
8060 NEXT NE
8070 CLOSEOUT
8080 LOCATE 10,10:PRINT"
8099 RETURN

10000 'TABLE FOR MACHINE CODES
10010 '
10020 FOR I=0 TO 15
10030 FOR J=0 TO 15
10040 READ T$(I,J)
10050 NEXT J,I
10060 RETURN
10070 DATA ,NOP,,,,,TAP,TPA,INX,DEX,CLV,SEV,CLC,SEC,CLI,SEI,
SBA,CBA,,,,,TAB,TBA,,DAA,,ABA,,,,,BRA,,BHI,BLS,BCC,BCS,BNE,B
EQ,BVC,BVS,BPL,BMI,BGE,BLT,BGT,BLE,TSX,INS,PULA,PULB,DES,TXS
,PSHA,PSHB,,RTS,,RTI,,,WAI,SWI,NEGA,,COMA,LSRA,,RORA,ASRA,A
SLA,ROLA,DECA
10080 DATA ,INCA,TSTA,,CLRA,NEGB,,,COMB,LSRB,,RORB,ASRB,ASLB
,ROLB,DECB,,INCB,TESTB,,CLRB,NEG,,,COM,LSR,,ROR,ASR,ASL,ROL
,DEC,,INC,TST,JMP,CLR,NEG,,,COM,LSR,,ROR,ASR,ASL,ROL,DEC,,INC
,TST,JMP,CLR,SUBA,CMPA,SBCA,,ANDA,BITA,LDAA,,EORA,ADCA,ORAA,
ADDA,CPXA,BSR
10090 DATA LDS,,SUBA,CMPA,SBCA,,ANDA,BITA,LDAA,STAA,EORA,ADC
A,ORAA,ADDA,CPX,,LDS,STS,SUBA,CMPA,SBCA,,ANDA,BITA,LDAA,STAA
,EORA,ADCA,ORAA,ADDA,CPX,JSR,LDS,STS,SUBA,CMPA,SBCA,,ANDA,BI
TA,LDAA,STAA,EORA,ADCA,ORAA,ADDA,CPX,JSR,LDS,STS,SUBB,CMPB,S
BCB,,ANDB,BITB
10100 DATA LDAB,,EORB,ADCB,ORAB,ADDB,,,LDX,,SUBB,CMPB,SBCB,,

```


ANDB,BITB,LDAB,STAB,EORB,ADCB,ORAB,ADDB,,LDX,STX,SUBB,CMPB,
SBCB,,ANDB,BITB,LDAB,STAB,EORB,ADCB,ORAB,ADDB,,LDX,STX,SUBB
CMPB,SBCB,,ANDB,BITB,LDAB,STAB,EORB,ADCB,ORAB,ADDB,,LDX,ST
X

6. ÖRNEKLER

DEVİRİCİ PROGRAM İLE BİR BELLEK BÖLGESİ OKUNURSA,

7A09	36	B6	B0	1B	B7	F0	01	32	6	.	.	.	.	.	.	2
7A11	B7	F0	00	39	CE	FC	0D	B6	.	.	.	9	.	.	.	.
7A19	77	36	B6	7C	7E	CA	A4	CE	w	6	.	i	~	.	.	.
7A21	FC	0F	B6	77	36	B6	79	7E	.	.	.	w	6	.	y	~
7A29	6A	A4	CE	FC	11	B6	6D	36	j	.	.	.	.	.	m	6
7A31	B6	7C	7E	6A	A4	CE	FC	13	.	i	~	j	.	.	.	.
7A39	B6	6D	36	B6	79	7E	CA	A4	.	m	6	.	y	~	.	.
7A41	CE	FC	15	B6	5E	36	B6	7C	.	.	.	.	^	6	.	i
7A49	7E	6A	A4	CE	FC	17	B6	77	~	j	.	.	.	.	.	w
7A51	36	B6	6D	7E	6A	A4	B6	7C	6	.	m	~	j	.	.	i
7A59	37	20	4B	CE	FC	19	B6	77	7	.	H	.	.	.	.	w
7A61	36	B6	0E	7E	6A	A4	CE	FC	6	.	.	~	j	.	.	.
7A69	1B	B6	7C	36	B6	5E	36	96	.	.	i	6	.	^	6	.
7A71	00	36	C6	40	BD	69	6A	32	.	6	.	e	.	i	j	2
7A79	BD	69	6A	32	BD	69	6A	32	.	i	j	2	.	i	j	2
7A81	BD	69	6A	B6	4B	BD	69	6A	.	i	j	.	H	.	i	j
7A89	B6	FC	28	49	25	02	6F	00	.	.	(	I	%	.	o	.
7A91	C6	02	BD	69	DC	02	02	C6	.	.	.	i	.	.	.	.
7A99	7C	F7	FC	27	C6	02	F7	FC	i	.	.	.	.	.	.	.
7AA1	26	20	2D	C6	40	BD	69	6A	&	.	-	.	e	.	i	j
7AA9	32	BD	69	6A	B6	4B	BD	69	2	.	i	j	.	H	.	i
7AB1	6A	B6	FC	28	49	25	04	6F	j	.	.	i	I	%	.	o
7AB9	00	6F	01	C6	0B	F7	FC	26	.	o	.	.	.	.	.	&
7AC1	C6	0B	BD	69	DC	0B	BD	69	.	.	.	i	.	.	.	i
7AC9	DC	02	09	C6	70	F7	FC	27	.	.	.	.	p	.	.	'
7AD1	B6	FC	23	BA	10	B7	FC	23	.	.	#	.	.	.	.	#
7AD9	BD	67	B6	39	CE	6B	09	36	.	g	.	9	.	k	.	6
7AE1	36	44	44	BD	67	A1	32	84	6	D	D	.	g	.	2	.
7AE9	03	E6	00	4C	4A	27	04	54	.	.	.	L	J	.	.	T
7AF1	54	20	F9	C4	03	32	B1	7E	T	.	.	.	.	2	.	~
7AF9	26	03	B6	02	39	B4	F0	B1	&	.	.	.	9	.	.	.
7B01	20	26	03	B6	01	39	4F	39	.	&	.	.	.	9	0	9
7B09	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.
7B11	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.
7B19	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.
7B21	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.
7B29	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.
7B31	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.
7B39	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.
7B41	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.

BU BELLEK BÖLGESİ DÖNÜŞTÜRÜCÜ PROGRAM İLE KODLARA DÖNÜŞTÜRÜLİRSE,

7A09	36	PSHA	
7A0A	86 B0	LDAA	#B0
7A0C	1B	ABA	
7A0D	B7 F001	STAA	F001
7A10	32	PULA	
7A11	B7 F000	STAA	F000
7A14	39	RST	
7A15	CE FC0D	LDX	#FC0D
7A1B	86 77	LDAA	#77
7A1A	36	PSHA	
7A1B	86 7C	LDAA	#7C
7A1D	7E CAA4	JMP	CAA4
7A20	CE FC0F	LDX	#FC0F
7A23	86 77	LDAA	#77
7A25	36	PSHA	
7A26	86 79	LDAA	#79
7A28	7E 6AA4	JMP	6AA4
7A2B	CE FC11	LDX	#FC11
7A2E	86 6D	LDAA	#6D
7A30	36	PSHA	
7A31	86 7C	LDAA	#7C
7A33	7E 6AA4	JMP	6AA4
7A36	CE FC13	LDX	#FC13
7A39	86 6D	LDAA	#6D
7A3B	36	PSHA	
7A3C	86 79	LDAA	#79
7A3E	7E CAA4	JMP	CAA4
7A41	CE FC15	LDX	#FC15
7A44	86 5E	LDAA	#5E
7A46	36	PSHA	
7A47	86 7C	LDAA	#7C
7A49	7E 6AA4	JMP	6AA4
7A4C	CE FC17	LDX	#FC17
7A4F	86 77	LDAA	#77
7A51	36	PSHA	
7A52	86 6D	LDAA	#6D
7A54	7E 6AA4	JMP	6AA4
7A57	86 7C	LDAA	#7C
7A59	37	PSHB	
7A5A	20 4B	BRA	4B
7A5C	CE FC19	LDX	#FC19
7A5F	86 77	LDAA	#77
7A61	36	PSHA	
7A62	86 0E	LDAA	#0E
7A64	7E 6AA4	JMP	6AA4
7A67	CE FC1B	LDX	#FC1B

7A6A	B6 7C	LDAA	#7C
7A6C	36	PSHA	
7A6D	B6 5E	LDAA	#5E
7A6F	36	PSHA	
7A70	96 00	LDAA	00
7A72	36	PSHA	
7A73	C6 40	LDAB	#40
7A75	BD 696A	JSR	696A
7A7B	32	PULA	
7A79	BD 696A	JSR	696A
7A7C	32	PULA	
7A7D	BD 696A	JSR	696A
7A80	32	PULA	
7A81	BD 696A	JSR	696A
7A84	B6 4B	LDAA	#4B
7A86	BD 696A	JSR	696A
7A89	B6 FC2B	LDAA	FC2B
7A8C	49	RDLA	
7A8D	25 02	BCS	02
7A8F	6F 00	CLR	00,X
7A91	C6 02	LDAB	#02
7A93	BD 69DC	JSR	69DC
7A96	02	NOP	
7A97	02	NOP	
7A9B	C6 7C	LDAB	#7C
7A9A	F7 FC27	STAB	FC27
7A9D	C6 02	LDAB	#02
7A9F	F7 FC26	STAB	FC26
7AA2	20 2D	BRA	2D
7AA4	C6 40	LDAB	#40
7AA6	BD 696A	JSR	696A
7AA9	32	PULA	
7AAA	BD 696A	JSR	696A
7AAD	B6 4B	LDAA	#4B
7AAF	BD 696A	JSR	696A
7AB2	B6 FC2B	LDAA	FC2B
7AB5	49	RDLA	
7AB6	25 04	BCS	04
7ABB	6F 00	CLR	00,X
7ABA	6F 01	CLR	01,X
7ABC	C6 0B	LDAB	#0B
7ABE	F7 FC26	STAB	FC26
7AC1	C6 0B	LDAB	#0B
7AC3	BD 69DC	JSR	69DC
7AC6	0B	INX	
7AC7	BD 69DC	JSR	69DC
7ACA	02	NOP	
7ACB	09	DEX	
7ACC	C6 70	LDA	

7ACC	C6 70.	LDAB	#70
7ACE	F7 FC27	STAB	FC27
7AD1	B6 FC23	LDAA	FC23
7AD4	BA 10	ORAA	#10
7AD6	B7 FC23	STAA	FC23
7AD9	BD 67B6	JSR	67B6
7ADC	39	RST	
7ADD	CE 6B09	LDX	#6B09
7AE0	36	PSHA	
7AE1	36	PSHA	
7AE2	44	LSRA	
7AE3	44	LSRA	
7AE4	BD 67A1	JSR	67A1
7AE7	32	PULA	
7AEB	B4 03	ANDA	#03
7AEA	E6 00	LDAB	00,X
7AEC	4C	INCA	
7AED	4A	DECA	
7AEE	27 04	BEQ	04
7AF0	54	LSRB	
7AF1	54	LSRB	
7AF2	20 F9	BRA	F9
7AF4	C4 03	ANDB	#03
7AF6	32	PULA	
7AF7	B1 7E	CMPA	#7E
7AF9	26 03	BNE	03
7AFB	B6 02	LDAA	#02
7AFD	39	RST	
7AFE	B4 F0	ANDA	#F0
7B00	B1 20	CMPA	#20
7B02	26 03	BNE	03
7B04	B6 01	LDAA	#01
7B06	39	RST	
7B07	4F	NOP	
7B08	39	RST	
7B09	00	NOP	
7B0A	00	NOP	
7B0B	00	NOP	
7B0C	00	NOP	
7B0D	00	NOP	
7B0E	00	NOP	

ŞEKİLİNDE BİR PROGRAM ELDE EDİLİR.

Örnek Programlar : (Assembler için)

```
7100          ORG 7100H ;CIKISLARI BELIRLE
7100 4F      CLRA      ;KONTROL KAYT. SIFIRLA
7101 B7 E001 STAA E001H ;VERI YDN KAYT.SEC
7104 B7 E003 STAA E003H
7107 B7 E000 STAA E000H ;A YI GIRIS SEC
710A 43      COMA
710B B7 E002 STAA E002H ;B YI CIKIS SEC
710E B6 04   LDAA #04H ;VERI KAYT.SEC
7110 B7 E001 STAA E001H
7113 B7 E003 STAA E003H
7116          ;R/W ISLEMI YAPILABILIR
7116 B6 E000 LDAA E000H
7119 B7 E002 STAA E002H
711C 3F      SWI
```

```
7200          ORG 7200H
7200 B6 03   LDAA #03H ;MASTER RESET
7202 B7 EC00 STAA EC00H
7205 B6 11   LDAA #11H ;B-BIT VERI
7207 B7 EC00 STAA EC00H ;16 CLOCK
720A 3F      SWI
```

```
7300          ORG 7300H
7300 B6 01   LDAA #01H ;CR20 YE 1 YUKLE
7302 B7 EB01 STAA EB01H
7305 B6 B2   LDAA B2H  ;#1 SAVICI CIKISI SERBEST
7307 B7 EB00 STAA EB00H
730A B6 50   LDAA #50H ;MSB YE 50H
730C B7 EB02 STAA EB02H
730F B6 05   LDAA #05H ;LSB YE 05H YUKLE
7311 B7 EB03 STAA EB03H
7314 3F      SWI
```

7. KAYNAK

Motorola , 8 bit microcomputer data book - 1986