MCS-6502マイクロコンピュータの逆アセンブラー

八田　信*・高浪五男**・井上克司**

Disassembler for 6502 Microcomputer

Makoto HATTA, Itsuo TAKANAMI and Katsushi INOUE

Abstract

we design a disassembler for the MCS-6502 microcomputer which uses 904 byte memories under the condition that we can use the monitor program TIM. A list of the disassembler is given.

1. はじめに

ストアードプログラム方式をとる現代の電子計算機においては，実行すべき命令ルーチンが2進数の形でメモリーにストアされている．これを機械語といい，機械はこの“言葉”を理解して命令の実行を行なう．しかし，我々人間にとって，2進数で書かれた符号としての機械語はきわめて理解しにくく，理解しようとすると大変な時間と労力を要する．そのために我々人間は理解しやすい高級言語（FORTRAN等）やアセンブリ言語を用いてプログラムを行ない，これを計算機により機械語に変換しているのが普通である（コンパイラーやアセンブラ）．一方，既成の計算機システムに改良や拡張を行なうには，機械語で書かれたルーチンを解読する必要が生じる．機械語をアセンブリ言語に変換するルーチンがあれば，この解読作業の手間がかなり少なくなる．このルーチンを逆アセンブラーという．ここでは，MCS 6502をCPUとするマイクロコンピュータシステムの逆アセンブラーの構成手順とそのプログラムを与える．これの所要メモリー量は904バイトである．

2. 逆アセンブラの機能と構成

2.1 機　能

まず構成しようとする逆アセンブラーの機能の設定を行なう．

(1) マシンコードからニーモニックコードとアドレッシングモードを解読し，オペランドと共にアセンブラー様式で表示する．

(2) ブランチ命令では，オペランドの相対番地からブランチ先の絶対番地を計算して表示する．

(3) 6502で使用されていないコードが命令コードとしてプログラム中に現われた場合は‘?’を印字してこれを1バイト命令とみなし，次へ進む．

(4) ラベル名，サブルーチン名，変数名は使えないのでジャンプ先やサブルーチン名は16進数による絶対番地で表示する．

そのために，プログラムの区切れがわかりにくくなるので，その対策として，本プログラムでは，プログラムの節となるところに適当な空行を設け，プログラム表示をよりわかり易いものにする．

本プログラムは，モニタープログラムDEMON（またはTIM）が使用できることを前提としている．モニタープログラムに関しては3を参照．

2.2 アドレスマップ

プログラムの構成をアドレスマップとして図1に示す．プログラムサイズは，テーブルサイズ222バイトを含め904バイトで，ワーキングエリアとして$0000～$000Fの16バイトを必要とする．

2.3 使用方法

① $1AE6番地からプログラムをスタートさせる．CPUはCRLFを出力して入力待ちとなる．

* 大学院電気工学専攻
** 電子工学科

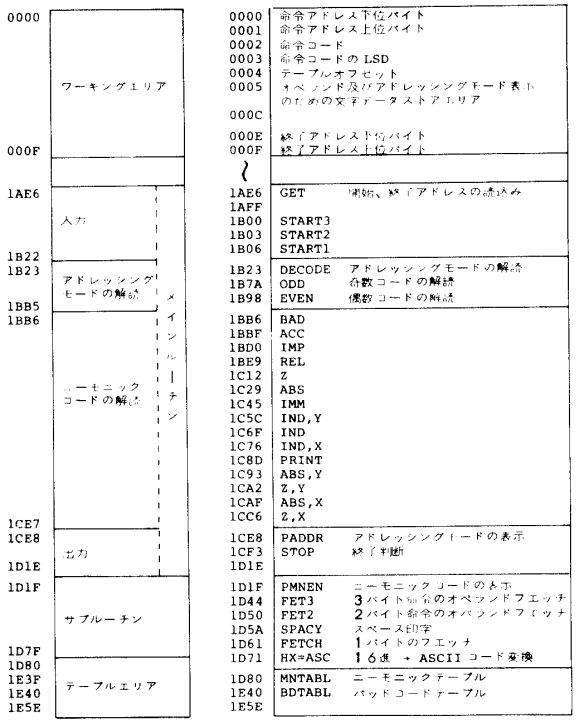

Fig. 1　Address Map for Disassembler

キーボードから，逆アセンブル開始番地と終了番地を入力する．
CPUは指定された番地から逆アセンブルを開始し，終了番地に達するとBRKによりコントロールはモニターに移る．

2.4　使用例

逆アセンブラーの使用例を図2に示す．これは，$1200番地から$121C番地の機械語（下線部）を逆アセンブルしたものである．プログラムとしては全く無意味なものであるが，13種類のアドレッシングモードと定義されていないコードの表示例を示している．使用方法は2.3に従って波線部をキーインしてやれば良い．コマンドについては3，アドレッシングモードについては4.1を参照．

3.　モニタープログラムの概略

モニタープログラムとは，システムの初期設定や入出力コントロール，内部レジスタの表示や変更，プログラムの実行等を行なうためのプログラムで，数種のコマンドが用意されている．

6502のモニタープログラムDEMON（またはTIM）はテレタイプを入出力装置としたモニタープログラムであり，コマンドとして次の様なものが用意されている．

・R　内部レジスタの内容表示コマンド．表示様式は
　　PC　PS　ACC　IX　IY　SP
　　ここで
　　PC　：プログラムカウンタ
　　PS　：プロセッサステータスレジスタ
　　ACC：アキュームレーター

```
 *  7Ø52 3Ø Ø6 FF Ø1 FF
.WH 12ØØ 121C
;1812ØØ69122D3412C5120ACAA1129112D6125D3412BE3412FØF26CØ8F6
;Ø51218341296126301 8Ø
.R  7Ø52 3Ø Ø6 FF Ø1 FF
.:  1AE6
.G
12ØØ 121c

12ØØ  69 12          ADC #$12        :A+$12+C → A              Immediate
12Ø2  2D 3412        AND  $1234      :A∧($1234)* → A           Absolute
12Ø5  C5 12          CMP  $12        :A-($12)                  Zeropage
12Ø7  ØA             ASL A           :C ← b7←…b0 ← 0           Accumulator
1208  CA             DEX             :IX-1 → IX        **      Implied
12Ø9  A1 12          LDA ($12,X)     :($($12+X+1)($12+X)) → A; Indexed Indirect
12ØB  91 12          STA ($12),Y     :($($13)($12)+Y) → A;     Indirect Indexed
12ØD  D6 12          DEC  $12,X      : ($12+X)-1 → $12+X;  X Indexed Zeropage
12ØF  5D 3412        EOR  $1234,X    :A∨($1234+X) → A;     X Indexed Absolute
1212  BE 3412        LDX  $1234,Y    :($1234+Y) → X;       Y Indexed Absolute
1215  FØ F2          BEQ  $1209      :Branch on Z=1 to $(PC+$F2); Relative

1217  6C 3412        JMP ($1234)     :Jump to $($1234)($1235);   Indirect

121A  96 12          STX  $12,Y      :$(12+Y) → X;         Y Indexed Zeropage
121C  63             ?               :Undefined operation code
 *  1CFF BØ FF ØØ ØØ FF
```

* アドレスに（ ）をつけたものは そのアドレスの内容を意味する．

** $(アドレスA)（アドレスB）はアドレスAの内容を上位バイトとし、アドレスBの内容を下位バイトとするアドレスを意味する．

Fig. 2　An example of disassembled List

IX ：インデクスレジスタX
IY ：インデクスレジスタY
SP ：スタックポインタ下位バイト

・M メモリーの内容表示コマンド．アドレスを指定すると，その番地から8バイトが表示される．

・： 変更コマンド．Rコマンドの次に来れば内部レジスタの内容変更となり，Mコマンドの次に来れば，メモリの内容変更となる．

・LH 16進コードコマンド．入力様式は，
ccaaaadd……ddssss
ここで
cc ：入力するデータのバイト数
aaaa ：データの先頭番地
dd…d ：16進データ
ssss ：チェックサム

・WH 16進コードの出力コマンド．開始アドレスと終了アドレスを与えると，その間のアドレスの内容が出力される．出力様式は，LH コマンドに同じ．

・↘ ラインスピードコントロールコマンド．ここで↘はキャリッジリターンラインフィードである．

・G プログラム実行コマンド．PC のアドレスからプログラムを実行する．

DEMON では，これらのコマンドの他に，入出力関係を主としたいくつかの有用なサブルーチンが用意されている．

CRLF（$728A)；キャリッジリターン・ラインフィードを出力

WROB（$72B1)；ACC の内容を2桁の16進数として印字

WRT（$72C6)；ACC の内容を ASCII コードとして印字

RDT（$72E9)；キーボードから ASCII コードを入力して ACC に格納する．

ASCII（$7358)；ACC の LSD を ASCII コードに変換して ACC に格納する．

SPAC2（$7374)；スペースを2つ出力

SPACE（$7377)；スペースを1つ出力

RDOB（$73B3)；キーボードから16進数のキャラクタコードを2文字入力してそれを2桁の16進数に変換し，ACC に格納する．

4. 逆アセンブラのプログラム

全体の大まかなフローチャートを図3に示す．プログラムは内容的には大きく解読部と表示部の2つに分けられる．

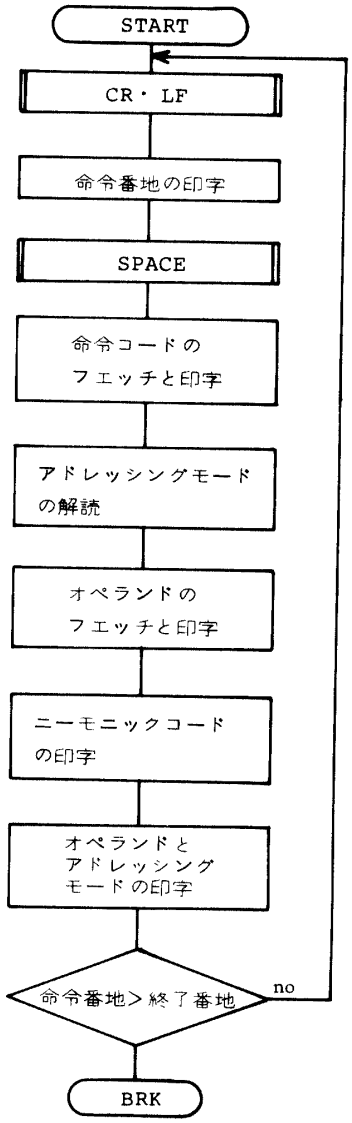

Fig. 3 A Flow Chart of the disassembler

4.1 アドレッシングモードの種類と解読

MCS 6502 では他のプロセッサ（6800 や 8080）に比べアドレッシングモードが多いのがひとつの特徴で，このアドレッシングモードをいかに取扱うかが，解読を効率よく行なう上で重要になってくる．

4.1.1 アドレッシングモードの種類（図2参照）

◦イミディエイト・アドレッシング *〔IMM〕
2バイト命令で，オペランドそのものがデータとなる．

*〔 〕内は各アドレッシグモードの略号．以下同じ．

◦アブソリュート　アドレッシング〔ABS〕
3バイト命令で，2バイト目が実効アドレスの下位バイト，3バイト目が実効アドレスの上位バイトとなる．

◦ゼロページ・アドレッシング〔Z〕
2バイト命令で，2バイト目が実効アドレスの下位バイト，上位アドレスは$00となる．

◦アキュームレータ・アドレッシング〔ACC〕
1バイト命令で，アキュームレータを操作の対象としている．

◦インプライド・アドレッシング〔IMP〕
1バイト命令で，操作の対象は，命令コードによって示されている．

◦インデクスト・インダイレクト・アドレッシング

Table 1　Operation Code Table for MCS-6502

LSD / MSD	0	1	2	3	4	5	6	7
0	BRK. IX	ORA. IX			o	ORA. Z	ASL. Z	
1	BPL. R	ORA. IY			o	ORA. ZX	ASL. ZX	
2	JSR. !	AND. IX			BIT. Z	AND. Z	ROL. Z	
3	BMI. R	AND. IY			o	AND. ZX	ROL. ZX	
4	RTI	EOR. IX			o	EOR. Z	LSR. Z	
5	BVC. R	EOR. IY			o	EOR. ZX	LSR. ZX	
6	RTS	ADC. IX			o	ADC. Z	o	
7	BVS. R	ADC. IY			o	ADC. ZX	o	
8	o	STA. IX			STY. Z	STA. Z	STX. Z	
9	BCC. R	STA. IY			STY. ZX	STA. ZX	STX. ZY	
A	LDY. #	LDA. IX	LDX. #		LDY. Z	LDA. Z	LDX. Z	
B	BCS. R	LDA. IY			LDY. ZX	LDA. ZX	LDX. ZY	
C	CPY. #	CMP. IX			CPY. Z	CMP. Z	DEC. Z	
D	BNE. R	CMP. IY			o	CMP. ZX	DEC. ZX	
E	CPX. #	SBC. IX			CPX. Z	SBC. Z	INC. Z	
F	BEQ. R	SBC. IY			o	SBC. ZX	INC. ZX	

LSD / MSD	8	9	A	B	C	D	E	F
0	PHP	ORA. #	ASL. A		o	ORA. !	ASL. !	
1	CLC	ORA. !Y	o		o	ORA. !X	ASL. !X	
2	PLP	AND. #	ROL. A		BIT. !	AND. !	ROL. !	
3	SEC	AND. !Y	o		o	AND. !X	ROL. !X	
4	PHA	EOR. #	LSR. A		JMP. !	EOR. !	LSR. !	
5	CLI	EOR. !Y	o		o	EOR. !X	LSR. !X	
6	PLA	ADC. #	o		JMP. (1)	ADC. !	o	
7	SEI	ADC. !Y	o		o	ADC. !X	o	
8	DEY	o	TXA		STY. !	STA. !	STX. !	
9	TYA	STA. !Y	TXS		o	STA. !X	o	
A	TAY	LDA. #	TAX		LDY. !	LDA. !	LDX. !	
B	CLV	LDA. !Y	TSX		LDY. !X	LDA. !X	LDX. !Y	
C	INY	CMP. #	DEX		CPY. !	CMP. !	DEC. !	
D	CLD	CMP. !Y	o		o	CMP. !X	DEC. !X	
E	INX	SBC. #	NOP		CPX. !	SBC. !	INC. !	
F	SED	SBC. !Y	o		o	SBC. !X	INC. !X	

# …IMM.	! …ABS.	Z …Zero Page	A …ACC.	…IMP.
IX …(IND, X)	IY …(IND), Y	ZX…Z, X	!X…ABS, X	!Y…ABS, Y
R …REL	(1)…(IND)	ZY…Z, Y	o …BAD	

Table 2 Mnemonic Table

MSD＼LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	?	1	?	O	R	A	P	H	P	B	R	K	—	—	—	—
1	B	P	L	A	S	L	C	L	C	?	4	?	—	—	—	—
2	B	I	T	A	N	D	P	L	P	J	S	R	—	—	—	—
3	B	M	I	R	O	L	S	E	C	J	M	P	—	—	—	—
4	J	M	P	E	O	R	P	H	A	R	T	I	—	—	—	—
5	B	V	C	L	S	R	C	L	I	?	5	?	—	—	—	—
6	?	2	?	A	D	C	P	L	A	R	T	S	—	—	—	—
7	B	V	S	?	3	?	S	E	I	?	6	?	—	—	—	—
8	S	T	Y	S	T	A	D	E	Y	T	X	A	—	—	—	—
9	B	C	C	S	T	X	T	Y	A	T	X	S	—	—	—	—
A	L	D	Y	L	D	A	T	A	Y	T	A	X	—	—	—	—
B	B	C	S	L	D	X	C	L	V	T	S	X	—	—	—	—
C	C	P	Y	C	M	P	I	N	Y	D	E	Y	—	—	—	—
D	B	N	E	D	E	C	C	L	D	?	7	?	—	—	—	—
E	C	P	X	S	B	C	I	N	X	N	O	P	—	—	—	—
F	B	E	Q	I	N	C	S	E	D	?	8	?	—	—	—	—

〔(IND·X)〕

2バイト命令で，2バイト目とXレジスタの内容を加え，それが示すゼロページ上のアドレスの内容が実効アドレスの下位バイトとなり，その次のアドレスの内容が実効アドレスの上位バイトとなる．

◦インダイレクト・インデクスト・アドレッシング〔(IND), Y〕

2バイト命令で，2バイト目で示されるゼロページ上のアドレスの内容を下位バイトとし，その次のアドレスの内容を上位バイトとする2バイトの2進数にYレジスタの内容を加えたものが実効アドレスとなる．

◦Xインデクスト・ゼロページ・アドレッシング〔*Z*, X〕

2バイト命令で，2バイト目にXレジスタの内容を加えたものが実効アドレスの下位バイトとなり，上位バイトは$00となる．キャリーは無視される．

◦Xインデクスト・アブソリュート・アドレッシング〔ABS, X〕

3バイト命令で，実効アドレスは2,3バイト目によって示されるアドレスにXレジスタの内容を加えたものになる．

◦Yインデクスト・アブソリュート・アドレッシング〔ABS, Y〕

ABS, XのXレジスタをYレジスタに置き換えたものに等しい．

◦リラティブ・アドレッシング〔REL〕

2バイト命令．ブランチ命令で分岐条件が成立したとき，(このブランチ命令のある番地)＋2に2バイト目の内容を加えた番地に分岐する．分岐できる範囲は＋127～－128である．

◦インダイレクト・アドレッシング〔(IND)〕

3バイト命令で，2,3バイト目によって示されるアドレスの内容を実効アドレスの下位バイトとし，その次のアドレスを上位バイトとする．

◦Yインデクスト・ゼロページ・アドレッシング〔*Z*, Y〕

Z, XのXレジスタをYレジスタに置き換えたものに等しい．

4.1.2 アドレッシングモードの解読

表1に6502の命令コード表を示す．この表で，命令コードの値とアドレッシングモードとの間にある種の規則性があることに気づく．(例えば，LSDが1でMSDが偶数ならば(IND, X)である) この規則性を利用して，命令コードがどのアドレッシングモードであるかを解読していく．

また，表の中でニーモニックコードが書かれてない部分には，命令コードは存在しないが，もしそのようなコード(BAD·CODE)が現われたときは，それを検出して，エラー表示ができなければならない．この場合もできれば規則性を利用するが，規則的でないものについては，バッドテーブルを用意し，それと比較して検出する．○印をつけたものがバッドテーブルに

よって検出されるBAD･CODEである．

解読部のフローチャートを図4bに示す．まず，バッドテーブルと比較して，表1における○印のBAD･CODEをチェックし，次にアドレッシングモードの不規則なコード（$20, $A2, $96, $B6, $6C, $BE）をチェックする．あとは，表1からこれらのコードを除いた残りについて考えれば良いので，解読は楽になる．

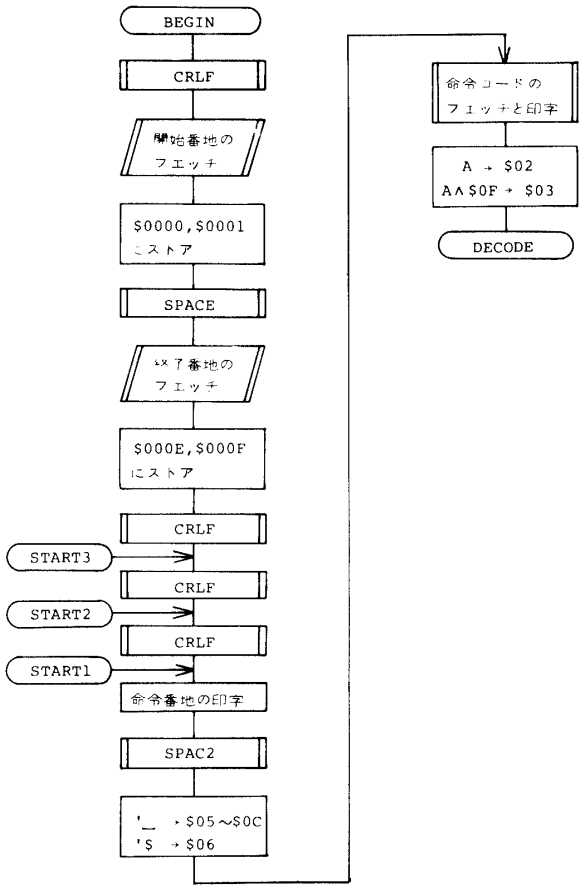

Fig. 4-a

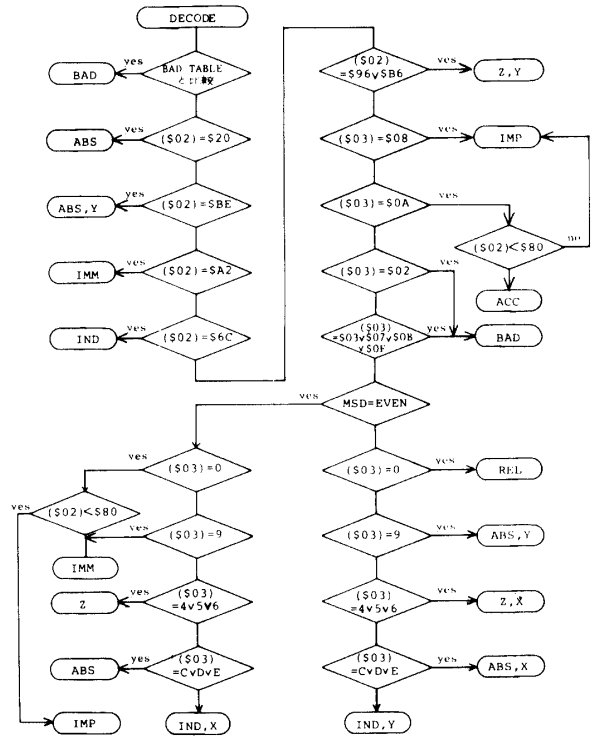

Fig. 4-b

4.2 表　示

表示例として図2の他に，自分自身を逆アセンブルしたプログラムリストを図5に掲載している．

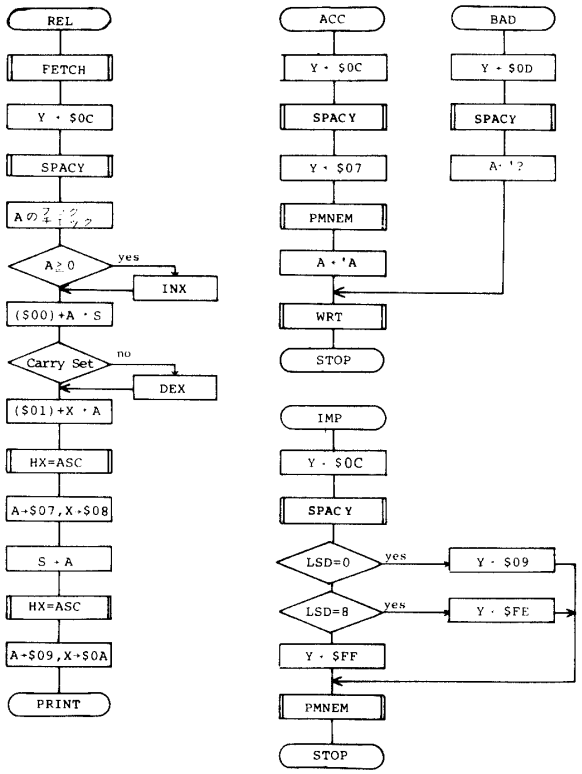

Fig. 4-c

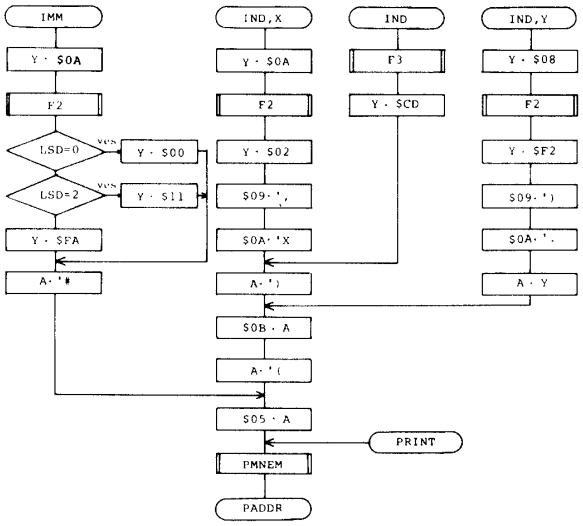

Fig. 4-d

4.2.1 ニーモニックコードの表示

ニーモニックコードは3文字のアルファベットからなっている．これの印字には，まず各命令のニーモニックコードを3バイトのASCIIコード（文字コード）としてメモリー内に格納しておく（表2のニーモニックテーブル）．命令コードが与えられると，そのニーモニックコードに対応する文字コードが格納されているアドレスを求め，その内容をプリントする．表2は表1をもとにして構成したもので，表1の縦列を左右に平行移動して作られている．従って命令コードが与えられると，その値に平行移動で動かした値（オフセット値）を加えると，それが求める文字コードの1文字目のアドレスとなる．2文字目，3文字目はアドレスを順にインクリメントしてフェッチすれば良い．例えば，アドレッシングモードが（IND, X）ならば，オフセット値として$02を命令コードに加えれば，1文字目のアドレスが求まる．すなわち，ORAの場合$01＋$02＝$03となり，アドレス$03，$04，$05にそれぞれ$4F（O），$52（R），$41（A）がストアされている．ただし，アドレッシングモードが同じでも，オフセット値は必ずしも同じ値にはならないことがあるので，それらは特別にオフセット値を考えなければならない．

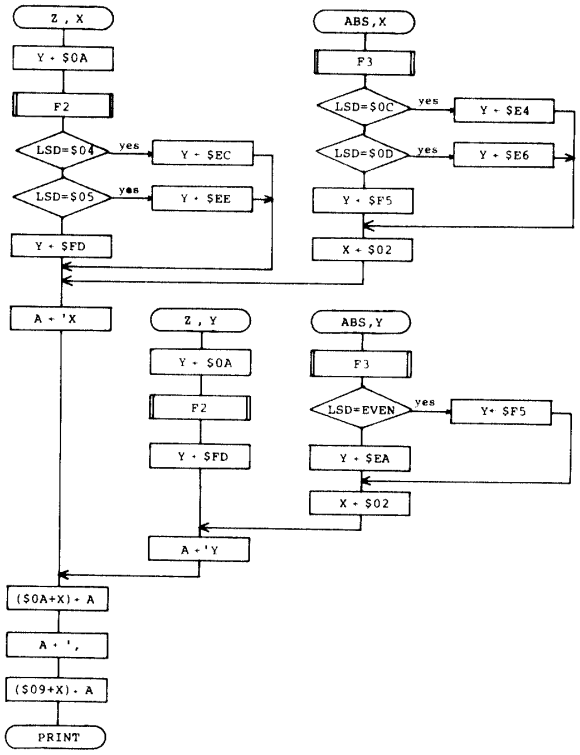

Fig. 4-e

さて，表2のテーブル構成ではLSDがC～Fとなるものはなく，テーブルに連続した番地がとれないので，メモリーが遊ぶことになる．そのため，実際にメモリーにストアするときは，LSDとMSDとを入れ替えている．

4.2.2 オペランドとアドレッシングモードの表示

オペランドとアドレッシングモードは，ニーモニッ

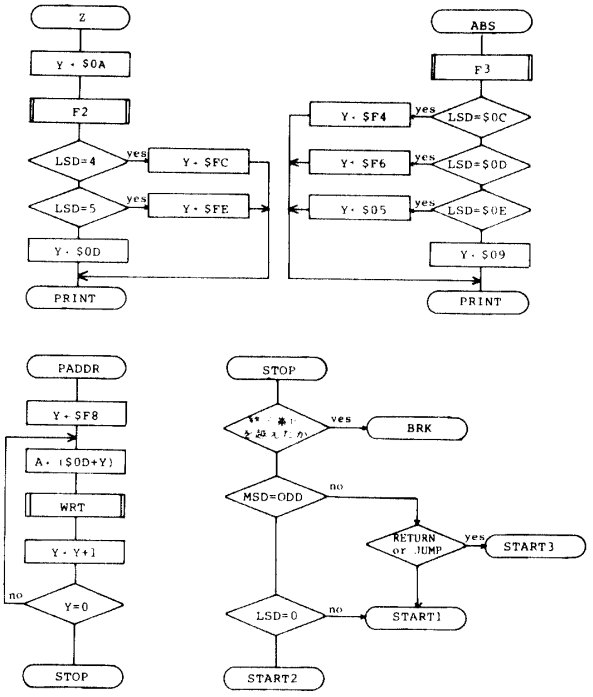

Fig. 4-f

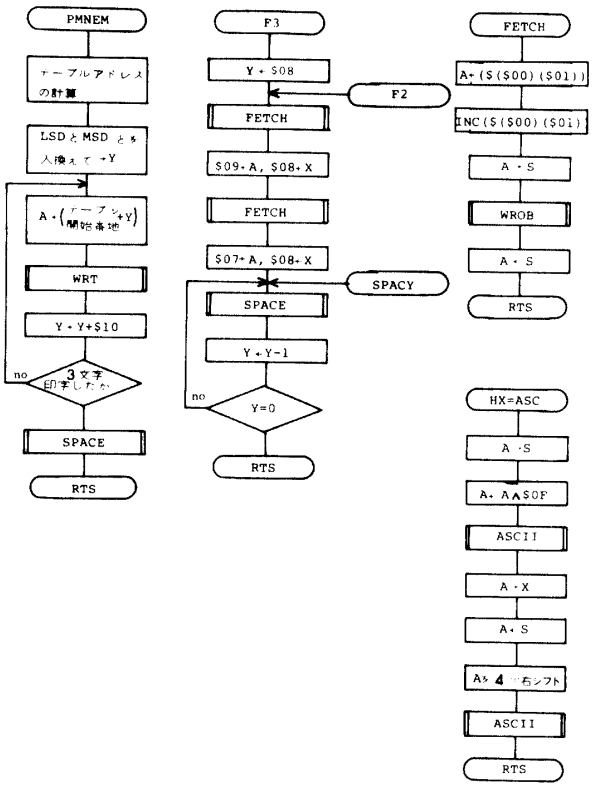

Fig. 4-g

```
 *  7052 30 21 FF 01 FF
.:  1AE6
.G
1AE6 1D80

1AE6  20 8A72       JSR  $728A
1AE9  20 B373       JSR  $73B3
1AEC  85 01         STA  $01
1AEE  20 B373       JSR  $73B3
1AF1  85 00         STA  $00
1AF3  20 7773       JSR  $7377
1AF6  20 B373       JSR  $73B3
1AF9  85 0F         STA  $0F
1AFB  20 B373       JSR  $73B3
1AFE  85 0E         STA  $0E
1B00  20 8A72       JSR  $728A
1B03  20 8A72       JSR  $728A
1B06  20 8A72       JSR  $728A
1B09  A5 01         LDA  $01
1B0B  20 B172       JSR  $72B1
1B0E  A5 00         LDA  $00
1B10  20 B172       JSR  $72B1
1B13  20 7473       JSR  $7374
1B16  A9 20         LDA #$20
1B18  A2 F8         LDX #$F8
1B1A  95 0D         STA  $0D,X
1B1C  E8            INX
1B1D  D0 FB         BNE  $1B1A

1B1F  A9 24         LDA #$24
1B21  85 06         STA  $06
1B23  20 611D       JSR  $1D61
1B26  48            PHA
1B27  29 0F         AND #$0F
1B29  85 03         STA  $03
1B2B  68            PLA
1B2C  85 02         STA  $02
1B2E  20 7773       JSR  $7377
1B31  A0 1F         LDY #$1F
1B33  D9 3F1E       CMP  $1E3F,Y
1B36  F0 3A         BEQ  $1B72

1B38  88            DEY
1B39  D0 F8         BNE  $1B33

1B3B  C9 20         CMP #$20
1B3D  F0 72         BEQ  $1BB1

1B3F  C9 BE         CMP #$BE
1B41  F0 3F         BEQ  $1B82

1B43  C9 A2         CMP #$A2
1B45  F0 5B         BEQ  $1BA2

1B47  C9 6C         CMP #$6C
1B49  D0 03         BNE  $1B4E

1B4B  4C 6E1C       JMP  $1C6E
```

Fig. 5a Program List for disaserbler

```
1B4E  29 DF         AND #$DF
1B50  C9 96         CMP #$96
1B52  D0 03         BNE  $1B57

1B54  4C A21C       JMP  $1CA2


1B57  A5 03         LDA  $03
1B59  C9 08         CMP #$08
1B5B  F0 0A         BEQ  $1B67

1B5D  C9 0A         CMP #$0A
1B5F  D0 09         BNE  $1B6A

1B61  A5 02         LDA  $02
1B63  C9 80         CMP #$80
1B65  30 58         BMI  $1BBF

1B67  4C D01B       JMP  $1BD0


1B6A  C9 22         CMP #$02
1B6C  F0 04         BEQ  $1B72

1B6E  29 03         AND #$03
1B70  C9 03         CMP #$03
1B72  F0 42         BEQ  $1BB6

1B74  A5 02         LDA  $02
1B76  29 10         AND #$10
1B78  F0 1E         BEQ  $1B98

1B7A  A5 03         LDA  $03
1B7C  F0 6B         BEQ  $1BE9

1B7E  C9 09         CMP #$09
1B80  D0 03         BNE  $1B85

1B82  4C 931C       JMP  $1C93


1B85  29 0C         AND #$0C
1B87  C9 04         CMP #$04
1B89  D0 03         BNE  $1B8E

1B8B  4C C61C       JMP  $1CC6


1B8E  C9 0C         CMP #$0C
1B90  D0 03         BNE  $1B95

1B92  4C AF1C       JMP  $1CAF


1B95  4C 5C1C       JMP  $1C5C
```

Fig. 5b Program List for disassembler

クコードの右（Col. 26～Col. 33）に表示する．Col. 26～Col. 33に表示すべきキャラクターはメモリーの $0005～$000CにASCIIコードとして格納しておき，PRADDルーチンによってプリントする．

4.3 サブルーチン

◦ HX＝ASC

ACCの値の上位4ビット，下位4ビットをそれぞれ16進数として表示するためにコード変換を行なう．

```
1B98  A5 03     LDA  $03
1B9A  D0 09     BNE  $1BA5

1B9C  A5 02     LDA  $02
1B9E  C9 80     CMP #$80
1BA0  30 C5     BMI  $1B67

1BA2  4C 451C   JMP  $1C45


1BA5  C9 09     CMP #$09
1BA7  F0 F9     BEQ  $1BA2

1BA9  29 0C     AND #$0C
1BAB  C9 04     CMP #$04
1BAD  F0 63     BEQ  $1C12

1BAF  C9 0C     CMP #$0C
1BB1  F0 76     BEQ  $1C29

1BB3  4C 761C   JMP  $1C76


1BB6  A0 0D     LDY #$0D
1BB8  20 5A1D   JSR  $1D5A
1BBB  A9 3F     LDA #$3F
1BBD  10 0C     BPL  $1BCB

1BBF  A0 0C     LDY #$0C
1BC1  20 5A1D   JSR  $1D5A
1BC4  A0 09     LDY #$09
1BC6  20 1F1D   JSR  $1D1F
1BC9  A9 41     LDA #$41
1BCB  20 C672   JSR  $72C6
1BCE  F0 16     BEQ  $1BE6

1BD0  A0 0C     LDY #$0C
1BD2  20 5A1D   JSR  $1D5A
1BD5  A0 09     LDY #$09
1BD7  A5 03     LDA  $03
1BD9  F0 08     BEQ  $1BE3

1BDB  A0 FE     LDY #$FE
1BDD  C9 08     CMP #$08
1BDF  F0 02     BEQ  $1BE3

1BE1  A0 FF     LDY #$FF
1BE3  20 1F1D   JSR  $1D1F
1BE6  4C F31C   JMP  $1CF3


1BE9  20 611D   JSR  $1D61
1BEC  A0 0A     LDY #$0A
1BEE  20 5A1D   JSR  $1D5A
1BF1  48        PHA
1BF2  68        PLA
1BF3  30 01     BMI  $1BF6

1BF5  E8        INX
1BF6  18        CLC
1BF7  65 00     ADC  $00
1BF9  48        PHA
1BFA  B0 01     BCS  $1BFD
```

Fig. 5c Program List for disassembler

```
1BFC  CA        DEX
1BFD  8A        TXA
1BFE  18        CLC
1BFF  65 01     ADC  $01
1C01  20 711D   JSR  $1D71
1C04  85 07     STA  $07
1C06  86 08     STX  $08
1C08  68        PLA
1C09  20 711D   JSR  $1D71
1C0C  85 09     STA  $09
1C0E  86 0A     STX  $0A
1C10  10 30     BPL  $1C42

1C12  A0 0A     LDY #$0A
1C14  20 501D   JSR  $1D50
1C17  A5 03     LDA  $03
1C19  A0 FC     LDY #$FC
1C1B  C9 04     CMP #$04
1C1D  F0 08     BEQ  $1C27

1C1F  A0 FE     LDY #$FE
1C21  C9 05     CMP #$05
1C23  F0 02     BEQ  $1C27

1C25  A0 0D     LDY #$0D
1C27  10 19     BPL  $1C42

1C29  20 441D   JSR  $1D44
1C2C  A5 03     LDA  $03
1C2E  A0 F4     LDY #$F4
1C30  C9 0C     CMP #$0C
1C32  F0 0E     BEQ  $1C42

1C34  A0 F6     LDY #$F6
1C36  C9 0D     CMP #$0D
1C38  F0 08     BEQ  $1C42

1C3A  A0 05     LDY #$05
1C3C  C9 0E     CMP #$0E
1C3E  F0 02     BEQ  $1C42

1C40  A0 09     LDY #$09
1C42  4C 8D1C   JMP  $1C8D


1C45  A0 0A     LDY #$0A
1C47  20 501D   JSR  $1D50
1C4A  A0 00     LDY #$00
1C4C  A5 03     LDA  $03
1C4E  F0 08     BEQ  $1C58

1C50  A0 11     LDY #$11
1C52  C9 02     CMP #$02
1C54  F0 02     BEQ  $1C58

1C56  A0 FA     LDY #$FA
1C58  A9 23     LDA #$23
1C5A  10 2F     BPL  $1C8B
```

Fig. 5d Program List for disassembler

上位4ビットに対応する文字コードはACCに，下位4ビットに対応する文字コードはＩＸにストアする．

```
1C5C  A0 0A     LDY #$0A
1C5E  20 501D   JSR  $1D50
1C61  A0 F2     LDY #$F2
1C63  A9 29     LDA #$29
1C65  85 09     STA  $09
1C67  A9 2C     LDA #$2C
1C69  85 0A     STA  $0A
1C6B  A9 59     LDA #$59
1C6D  10 18     BPL  $1C87

1C6F  20 441D   JSR  $1D44
1C72  A0 CD     LDY #$CD
1C74  D0 0F     BNE  $1C85

1C76  A0 0A     LDY #$0A
1C78  20 501D   JSR  $1D50
1C7B  A0 02     LDY #$02
1C7D  A9 2C     LDA #$2C
1C7F  85 09     STA  $09
1C81  A9 58     LDA #$58
1C83  85 0A     STA  $0A
1C85  A9 29     LDA #$29
1C87  85 0B     STA  $0B
1C89  A9 28     LDA #$28
1C8B  85 05     STA  $05
1C8D  20 1F1D   JSR  $1D1F
1C90  4C E81C   JMP  $1CE8


1C93  20 441D   JSR  $1D44
1C96  A0 F5     LDY #$F5
1C98  46 03     LSR  $03
1C9A  90 02     BCC  $1C9E

1C9C  A0 EA     LDY #$EA
1C9E  A2 02     LDX #$02
1CA0  D0 09     BNE  $1CAB

1CA2  A0 0A     LDY #$0A
1CA4  20 501D   JSR  $1D50
1CA7  A0 FD     LDY #$FD
1CA9  A2 00     LDX #$00
1CAB  A9 59     LDA #$59
1CAD  10 30     BPL  $1CDF

1CAF  20 441D   JSR  $1D44
1CB2  A5 03     LDA  $03
1CB4  A0 E4     LDY #$E4
1CB6  C9 0C     CMP #$0C
1CB8  F0 08     BEQ  $1CC2

1CBA  A0 E6     LDY #$E6
1CBC  C9 0D     CMP #$0D
1CBE  F0 02     BEQ  $1CC2

1CC0  A0 F5     LDY #$F5
1CC2  A2 02     LDX #$02
1CC4  10 17     BPL  $1CDD

1CC6  A0 0A     LDY #$0A
1CC8  20 501D   JSR  $1D50
1CCB  A5 03     LDA  $03
1CCD  A0 EC     LDY #$EC
1CCF  C9 04     CMP #$04
1CD1  F0 08     BEQ  $1CDB
```

Fig. 5e Program List for disassembler

```
1CD3  A0 EE     LDY #$EE
1CD5  C9 05     CMP #$05
1CD7  F0 02     BEQ  $1CDB

1CD9  A0 FD     LDY #$FD
1CDB  A2 00     LDX #$00
1CDD  A9 58     LDA #$58
1CDF  95 0A     STA  $0A,X
1CE1  A9 2C     LDA #$2C
1CE3  95 09     STA  $09,X
1CE5  4C 8D1C   JMP  $1C8D


1CE8  A0 F8     LDY #$F8
1CEA  B6 0D     LDX  $0D,Y
1CEC  8A        TXA
1CED  20 C672   JSR  $72C6
1CF0  C8        INY
1CF1  D0 F7     BNE  $1CEA

1CF3  38        SEC
1CF4  A5 0E     LDA  $0E
1CF6  E5 00     SBC  $00
1CF8  A5 0F     LDA  $0F
1CFA  E5 01     SBC  $01
1CFC  B0 01     BCS  $1CFF

1CFE  00        BRK
1CFF  A5 02     LDA  $02
1D01  48        PHA
1D02  29 10     AND #$10
1D04  D0 11     BNE  $1D17

1D06  68        PLA
1D07  29 DF     AND #$DF
1D09  C9 40     CMP #$40
1D0B  F0 07     BEQ  $1D14

1D0D  C9 4C     CMP #$4C
1D0F  F0 03     BEQ  $1D14

1D11  4C 061B   JMP  $1B06


1D14  4C 001B   JMP  $1B00


1D17  68        PLA
1D18  29 0F     AND #$0F
1D1A  D0 F5     BNE  $1D11

1D1C  4C 031B   JMP  $1B03


1D1F  98        TYA
1D20  A2 03     LDX #$03
1D22  86 04     STX  $04
1D24  18        CLC
1D25  65 02     ADC  $02
1D27  A8        TAY
1D28  38        SEC
1D29  30 01     BMI  $1D2C
```

Fig. 5f Program List for disassembler

```
1D2B  18          CLC
1D2C  2A          ROL A
1D2D  CA          DEX
1D2E  10 F7       BPL  $1D27

1D30  A8          TAY
1D31  B9 801D     LDA  $1D80,Y
1D34  20 C672     JSR  $72C6
1D37  98          TYA
1D38  18          CLC
1D39  69 10       ADC  #$10
1D3B  C6 04       DEC  $04
1D3D  D0 F1       BNE  $1D30

1D3F  20 7773     JSR  $7377
1D42  EA          NOP
1D43  60          RTS


1D44  A0 08       LDY  #$08
1D46  20 611D     JSR  $1D61
1D49  20 711D     JSR  $1D71
1D4C  85 09       STA  $09
1D4E  86 0A       STX  $0A
1D50  20 611D     JSR  $1D61
1D53  20 711D     JSR  $1D71
1D56  85 07       STA  $07
1D58  86 08       STX  $08
1D5A  20 7773     JSR  $7377
1D5D  88          DEY
1D5E  D0 FA       BNE  $1D5A

1D60  60          RTS


1D61  A2 00       LDX  #$00
1D63  A1 00       LDA  ($00,X)
1D65  48          PHA
1D66  E6 00       INC  $00
1D68  D0 02       BNE  $1D6C

1D6A  E6 01       INC  $01
1D6C  20 B172     JSR  $72B1
1D6F  68          PLA
1D70  60          RTS


1D71  48          PHA
1D72  29 0F       AND  #$0F
1D74  20 5873     JSR  $7358
1D77  AA          TAX
1D78  68          PLA
1D79  4A          LSR A
1D7A  4A          LSR A
1D7B  4A          LSR A
1D7C  4A          LSR A
1D7D  20 5873     JSR  $7358
1D80  60          RTS
 *  1CFF B0 FF 00 B6 FF
```

Fig. 5g Program List for disassembler

◦FETCH

$0000と$0001の2バイトによって示される番地の内容をACCにロードし，その値を2桁の16進数として印字する．このとき$0000と$0001に格納されている番地はインクリメントされる．

◦F3

3バイト命令のオペランドのフェッチと印字を行なう．

◦F2

2バイト命令のオペランドのフェッチと印字を行なう．

◦SPACY

IYにストアされている値だけスペースを印字する．

◦PRADD

$0005～$000Cに格納されているASCIIコードを順に印字する．

◦STOP

$0000と$0001によって示される番地が$000Eと$000Fによって示される番地（逆アセンブル終了番地）より大きくなったとき，BRK命令により，モニターに戻る．それ以外の場合はその時解読した命令の種類により次の3つのようになる．

① 命令がRTS, RTI, JMPであった場合には，START 3にジャンプする．

② 命令がブランチ命令であった場合には，START 2にジャンプする．

③ その他の命令の場合には，START 1にジャンプする．

この結果①の命令のあとには空行が2行，②の命令のあとには空行が1行，③の命令のあとは空行なしとなり，プログラムの区切れを明確にすることができる．

①の命令では，プログラムが次へ続いていくことはなく，局部的にみると，プログラムの終点となっている．また②の命令は，プログラムの流れの中で節点となっている．従ってこれらをプログラムの区切りとみなすことができる．

4.4 ドットプリンター用への改修

出力コントロールルーチンをドットプリンタ用に変えてやれば良い．DEMONの出力ルーチンに対応するドットプリンタ用のルーチンは文献〔3〕の3・4・3で示したので，これらで置き換える．文献〔3〕のドットプリンタのモニタープログラムリストは，これによって逆アセンブルしたものである．

ドットプリンタのプリント速度はTTYに比べ格段に速いので，実際には，こちらのプログラムの方が実用的である．

5. 逆アセンブラの実行手順

図4に逆アセンブラの実行手段の詳細をフローチャートによって示す．実際のプログラムリストは，図5の通りである．これは，自分自身の機械語を逆アセンブルしてプリントしたものである．

6. むすび

最後に本プログラムを，6502を用いた他のシステムに活用するためリロケートする場合について補足する．本プログラムでは，メモリーの都合上$1AE6から始めているが，リロケートする場合には，JMP命令や，モニター以外へのJSR命令のオペランドは移動させるバイト数だけ加減する必要がある．多少めんどうであるが，ミスさえなければきちんと動作するはずである．

以上，逆アセンブラーの説明をしてきたが，特に複雑な考え方はしていないので，容易に理解できるであろう．このプログラムの完成により，オペレーティングシステムの解読や，新たなプログラムの作製が容易になり，今後のシステムの発展が期待できる．

参考文献

1) MCS 6500 Microcomputer Family Programming Manual, MOS TECHNOLOGY, INC. 1976
2) 松本吉彦：わかるマイクロコンピュータ〔第1回〕～〔第7回〕：トランジスタ技術　1976. 8～1977. 2
3) 八田，高浪，井上：MCS-6502をCPUとするマイクロコンピュータシステム—高速ドットプリンタPA-6651-Aとのインタフェース—：山口大学工学部研究報告, **29**, **1** **03** (1978)
4) ゴールドスターリングス，6800逆アセンブラ，ASCII, 1977. 8

（昭和53年4月12日　受理）