マイクロコンピュータ用プログラム作成のための代数的表記法

山　下　陸　夫

(受理　昭和58年5月31日)

AN ALGEBRAIC NOTATION FOR MICROCOMPUTER PROGRAMS

Mutsuo YAMASHITA

An algebraic notation is proposed so that even beginners can easily write the program. This notation that does not depend on mnemonics is familiar to the beginner, because it is certainly clearer and more intelligible than that of the Intel's notation. By using this notation together with an extended instruction set defined by each programmer, structured-programming techniques are permitted. In the application of this concept to microprocessors of different constitution, a source-program compatibility is discussed mathematically using the set theory. The notation has been put into practice by the students in Electronics Exercise and the advantage of this has been shown as compared with the Intel's one.

1.まえがき

マイクロコンピュータのためのシステムプログラムはほとんどアセンブリ言語で記述され，0,1の機械語の表現形式でROMに固定されている．このアセンブリ言語はマイクロプロセッサチップ（CPU）の製作者が作成したものでCPUの種類によって異なっている．アセンブリ言語はニモニックと呼ばれる命令を表わすコードで特長づけられている．このニモニックは大型計算機やミニコンの言語の影響を受け，動作を示す英単語列の頭字を集めて作られている（例 MVI r =Move immediate register）．したがってニモニックから命令の動作を示す英単語列を想起して命令の意味を理解せねばならず，初心者には煩雑である．既にいくつかの改善法が提案されているが[1)~3)]，不十分な点がある．本文では初心者にも分かり易いBASIC風の代入を主体とした新しい表記法を提案する．この表記法は

1) 命令表記に代入を主に用いるので命令の動作を理解し易い．
2) 命令展開の概念を導入し，命令拡張が容易である（拡張命令）．
3) 拡張命令を使用して記述することによりソースプログラムを短かくでき，分かり易い．
4) トップダウン式考え方でプログラムを記述できる．
5) 8080 CPUを対象にしているが基本命令コード表を変更すれば他のCPUにも適用できる．

などの特長がある．

この表記法を学生実験・電子工学演習Ⅰで使用したところ学生達にも好評であった．又一定の条件を満足する拡張命令の集合の存在の有無によって，異なるCPU間でソースプログラムの互換性の有無を検討できることも示した．

2.提案する新表記法

2.1 基本方針

新表記法は初心者でも少ない基礎知識でCPUの命令を理解できることを中心に

1) 現在マイクロ（パーソナル）コンピュータではBASICが広く普及している．このBASICの基礎知識を命令の理解に活用する．

2) BASICの命令と同じ働きの命令には同じ単語を使用し，その他は分かり易さ，拡張性を考慮する．
3) アセンブラ制御命令にはインテル表記のORG，EQU，DW，DB，DS，ENDを使用する．
4) 使用できる命令を基本命令，拡張命令，アセンブラ制御命令の3つとし，CPUの持っている基本命令で定義（展開）できる拡張命令を導入する．
5) 今回は処理系を作らずに手作業でソースプログラムを機械語に変換すること（ハンドアセンブル）を前提とし，プログラミング上の制限をできるだけ少なくする．

などを基本方針とした．

2.2 新表記法の概要

対象CPUは8080でプログラミングに必要な内部レジスタ群を図1に示す．プログラミングのとき内部のレジスタ類を変数として使用する．

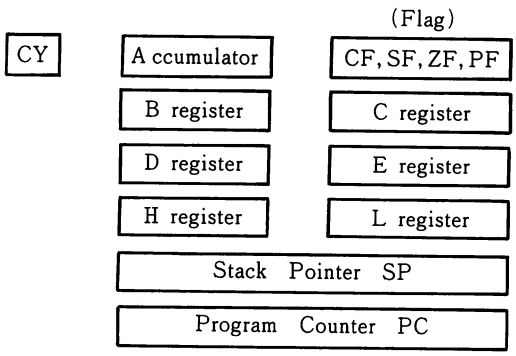

図1　8080 CPUの内部レジスタ群

基本命令とはCPUがハード的に持っている固有の命令で対応する機械語がある．拡張命令とは基本命令を2個以上組み合わせてまとまりのある動作を行なうもので，プログラマが必要に応じて定義できる命令である．

命令の表記に使用されるBASICの単語は

GOTO，GOSUB，RET(URN)，IF … THEN

である．以下の説明や例での（　）のついたものはその内容を示し，16進数は最後に“H”をつける．

例　(8250H)：16進数8250番地のメモリの内容
　　(HL)　　：HLレジスタで表わされるメモリの内容

1) 基本命令の表記法

基本命令の表記法は大きく

イ）代入表記のもの　ロ）BASICの単語で表記するもの　ハ）イ，ロ以外で表記を変更したもの　ニ）インテル表記のまま変更しないもの

の4つに分けられる．

イ）代入表記命令（183個）

この表記法はBASICのLETを省略した形と考えてよく，全基本命令の75％がこの代入表記で表わされる．したがって初心者でも容易に大部分の命令の動作を理解できるのが本記法の特長である．命令の表記例を例1に示す．

例1　B：=C，C=：B

いずれもBレジスタにCレジスタの内容を転送する動作を示している．“：”を省略するときは例2のように必ずBを左辺に書く必要がある．

例2　B=C　　例3　C=B

例3ではCレジスタにBレジスタの内容を転送することを表わしている．以下動作別に命令を述べる．

①データ転送命令
　例 A=B(A：=Bと同じ)，C=3，(8250H)=A，
　　BC=8250H，DE：=：HL［DE，HLレジスタの内容を交換する命令］

②レジスタ増減命令
　例．D=D+1，SP=SP−1，(HL)=(HL)+1

③算術演算命令（加減算）
　例．A=A+B，A=A+5+CY，
　　HL=HL+DE，A=A−D−CY

④論理演算命令（ビット毎）
　例．A=A AND B，A=A OR 15，
　　A=A XOR (HL)

⑤入出命令
　例．A=P(BEH)，P(BEH)=A［P(BEH)はデバイス番号BEHのポートを示す．］

⑥その他（アキュームレータA，キャリィCY）
　例．A=$\overline{A}$，CY=1，CY=$\overline{CY}$［$\overline{\ }$は否定を示す．］

注　②，③，④で等号（=）及び等号の左のレジスタ名を省略しても意味が明確なとき省略できる．
　例　D+1，A AND B，A−D−CY

ロ）BASICの単語で表記する命令

分岐命令及びサブルーチン命令で，無条件命令と条件付命令とがある．条件付命令はIF … THEN

～と無条件命令を組合せて表記する．

①無条件命令（3個）

例　GOTO，GOSUB，RET(URN)

②条件付命令（24個）

条件は4個のフラグ（ZF，CF，PF，SF）で表わされ，セットされると“1”，それ以外は“0”である．

例　IF ZF=0 THEN GOTO LOOP［LOOP：ラベル］，IF SF=1 THEN GOSUB 8300H

ハ）上記以外で表記を変更した命令（13個）

①アキュームレータAと比較後フラグのみ変化

例．F(A−B)，F(A−8)

②シフト命令　［ザイログ表記へ変更］

例　RLA，RRA，RLCA，RRCA

ニ）表記を変更しない命令（21個）

①スタック操作命令［レジスタ表記をBC，DE，HL，AFと変更する］

例　PUSH HL，POP AF

②割込み，その他の命令

例　EI，DI，DAA，NOP，HLT，RSTn(n=0 ～ 7)

以上の基本命令の一覧表を付録に示す．又類似の表記法にSMAL/80[1]，F. G. Duncanの表記法[2]，BASE−80[3]，ソニーのANN記法[4]などがあるが，理解しにくい部分がある．本表記法では特に拡張命令の概念及びその表記を基本命令と同一にした点が他の表記法よりすぐれている．

2）拡張命令の表記法と展開

拡張命令は前に述べたようにプログラマが必要に応じて定義できる命令であり，その表記法は基本命令と同様に表記する。拡張命令を基本命令の集まりに変換することを「展開」といい，ソースプログラムを機械語に変換する前に必ず展開しておく必要がある。

例1　B=A+C　を展開すると　A=A+C，B=A　となる．

例2　DE：=：BC　を展開すると　PUSH DE，PUSH BC，POP DE，POP BC　となる．

拡張命令を使用する利点を次にあげる．

(1) ソースプログラムを短かくできる．

(2) 基本命令より大きな概念の動作を表わすことができ，プログラム作成の能率が向上する。

(3) 過去に作成したプログラム（一部分でも可）を拡張命令として定義しておけば以後のプログラム作成に利用可能である．

(4) 構造化された分かり易いプログラムを書ける．

3）本表記法によるプログラム例

次の例題のプログラムを本表記法とインテル表記法とで図2に示す．

例題　1バイトの正の数DATA 1，DATA 2がある．

DATA 1÷DATA 2　を計算し商をSYO，余りをAMAに入れるプログラムを作りなさい。

DATA 1=17，DATA 2=3　として例題のプログラムを図2(a)～(c)に示す．参考のためインテル表記法のものを(d)に示す．例題より直接(c)又は(d)のプログラムを作ることは困難であるが，途中(a)，

```
        ORG   8200H
        SYO=DATA1／DATA2      *
        AMA=DATA1  MOD  DATA2 *
        HLT

DATA1   DC    17
DATA2   DC    3
SYO     DS    1
AMA     DS    1
        END
```

(a) ステップ1（＊拡張命令）

```
        ORG   8200H
        C=DATA2               *
        A=DATA1
        B=A／C                *
        A=A  MOD  C           *
        AMA=A
        SYO=B                 *
        HLT

DATA1   DC    17
DATA2   DC    3
SYO     DS    1
AMA     DS    1
        END
```

(b) ステップ2（＊拡張命令）

```
        ORG   8200H
        A=DATA2
        C=A
        A=DATA1
        B=0
LOOP    B=B+1
        A=A−C
        IF  A>=0  THEN GOTO LOOP
        A=A+C
        B=B−1
        AMA=A
        A=B
        SYO=A
        HLT

DATA1   DC    17
DATA2   DC    3
SYO     DS    1
AMA     DS    1
        END
```

(c) ステップ3　基本命令のみ

```
        ORG   8200H
        LDA   DATA2
        MOV   C, A
        LDA   DATA1
        MVI   B, 0
LOOP    INR   B
        SUB   C
        JP    LOOP
        ADD   C
        DCR   B
        STA   AMA
        MOV   A, B
        STA   SYO
        HLT

DATA1   DC    17
DATA2   DC    3
SYO     DS    1
AMA     DS    1
        END
```

(d) インテル形式

図2　例題のプログラム

(b)の段階を経ると作成は容易である．プログラマの能力に応じて途中の段階で分析を止めて拡張命令で記述しておけば，分かり易く，短かいソースプログラムが作成できる．

3．機械語と命令との関係

機械語とアセンブリ言語のニモニックとは一対一に対応をしているが，本表記法では命令の動作を表わしているために必ずしも一対一の対応が成立しない．

例 A=0 ⇒ MVIA,0，SUB A，XRA A

まず1種類のCPUについて機械語，ニモニック，本表記法での命令セットの対応関係について検討し，次に2種類以上のCPU間でのソースプログラムの互換性について考える．

3．1　1種類のCPUでの命令の関係

任意のCPUにおいて，機械語の集合を**M**，アセンブリ言語のニモニックの集合を**N**，本表記法での基本命令の集合を**F**，拡張命令の集合を**E**とし，各集合の要素の数を$|\mathbf{M}|$，$|\mathbf{N}|$，$|\mathbf{F}|$，$|\mathbf{E}|$とする．

定義1　基本命令とはCPUがハード的に持っている命令で，CPUによって動作，個数が定まっている。

定義2　拡張命令とはCPUがハード的に持っておらず，基本命令をソフト的に組み合わせて一定の動作を行うように定義された命令で，動作，個数は定まっていない．プログラマが任意に定義できるものである．

これらの集合には次の性質がある．

性質1．**M**，**N**，**F**の要素の数は等しい．

$$|\mathbf{M}|=|\mathbf{N}|=|\mathbf{F}| \qquad \text{———(1)}$$

性質2．**F**のある要素の動作は他の2個以上の要素の動作の組み合わせで表わされるものもある．

性質3．**F**のある要素の動作は他の要素の動作と部分的に同じ動作をするものもある．

性質4．**E**の要素を表わす**F**の要素の組み合わせ方は2つ以上あるものもある。

性質2，3より基本命令の中に，ある種の関係の存在を予想できるがその詳細は不明であり，今後の研究課題である．性質2～4によって同じ動作のプログラムでも数種類の異なったプログラムができることが予想される．

インテル表記法ではプログラムは基本命令のみで記述されるため，適当な方法により基本命令のレベルまで問題分析を行ない，その結果に対応した命令の選択を行ないプログラムを記述する．したがってソースプログラムから機械語への変換は一意的に定まる．本表記法では拡張命令の使用が許され，トップダウン方式でプログラム作成ができるのでソースプログラムは色々なレベルのプログラムが考えられる．したがって機械語への変換は一意に定まらず，機械語への変換時に使用する基本命令の決定を行なう必要がある．

3．2　2種類以上のCPU間の命令

2種類のCPU 1，2の基本命令の集合を$\mathbf{F}_1$，$\mathbf{F}_2$とし，それらの和集合を**F**とすると次式が成立する．

$$\mathbf{F}=\mathbf{F}_1\cup\mathbf{F}_2 \qquad \text{———(2)}$$

各々のCPUに対して

$$\mathbf{E}_1=\mathbf{F}-\mathbf{F}_1 \qquad \text{———(3)}$$

$$\mathbf{E}_2=\mathbf{F}-\mathbf{F}_2 \qquad \text{———(4)}$$

なる拡張命令の集合を考える．$\mathbf{F}_1$，$\mathbf{F}_2$は**F**の部分集合で$\mathbf{E}_1$，$\mathbf{E}_2$は**F**に対する$\mathbf{F}_1$，$\mathbf{F}_2$の補集合である．$\mathbf{E}_1$，$\mathbf{E}_2$の存在の有無によってソースプログラムがCPU 1，2間で互換性を持つか否かが定まる．定理1にこれを示す．

> 定理1　CPU 1について式(3)の$\mathbf{E}_1$が存在すればCPU 2で実行可能なプログラムはCPU 1でも実行可能である．又，CPU 2について式(4)の$\mathbf{E}_2$が存在すればCPU 1で実行可能なプログラムはCPU 2でも実行可能である．

〔証明〕$\mathbf{F}_1$，$\mathbf{F}_2$の積集合を$\mathbf{F}_A$とすると

$$\mathbf{F}_A=\mathbf{F}_1\cap\mathbf{F}_2 \qquad \text{———(5)}$$

が成立する．今式(3)が成立すれば

$$\mathbf{E}_1=\mathbf{F}-\mathbf{F}_1=\mathbf{F}_2-\mathbf{F}_A \qquad \text{———(6)}$$

$$\therefore\ \mathbf{F}_2=\mathbf{F}_A+\mathbf{E}_1 \qquad \text{———(7)}$$

となり$\mathbf{E}_1$が存在すれば$\mathbf{F}_2$も存在する．

又式(4)が成立すれば

$$\mathbf{E}_2=\mathbf{F}-\mathbf{F}_2=\mathbf{F}_1-\mathbf{F}_A \qquad \text{———(8)}$$

$$\therefore\ \mathbf{F}_1=\mathbf{F}_A+\mathbf{E}_2 \qquad \text{———(9)}$$

となり$\mathbf{E}_2$が存在すれば$\mathbf{F}_1$も存在する．

よって定理1は証明された．

定理1をk種類（$k\geq 3$）のCPUについて容易に拡張できる．これを定理2に示す．

定理2　k 種類の CPU 間で任意のソースプログラムが実行可能であるためには，各 CPU の基本命令の集合の和集合を全体集合とするとき，各 CPU の基本命令の補集合が拡張命令の集合に含まれることである。

この証明は定理1の拡張で自明であるので省略する．この定理2によって数種類の CPU 上で実行可能なプログラムの作成方法が次のように示される．

1）各 CPU の持つ基本命令の集合の和集合 **F** を定める．

2）CPUj の拡張命令の集合 $\mathbf{E}_j$ を定義する．

$$\mathbf{E}_j=\mathbf{F}-\mathbf{F}_j \qquad \text{——(10)}$$

ここで $\mathbf{E}_j$ が定義できなければ CPUj では互換性のあるプログラムを実行させることはできない．

3）集合 $\mathbf{F}(=\mathbf{F}_j\cup\mathbf{E}_j)$ の要素でソースプログラムを記述する．

各 CPU に対して式(10)を満足する拡張命令の集合が存在するか否かを調べれば他の CPU 上でのプログラムを実行できるか否かが分かる．

4．提案した表記法の有効性

提案した表記法の有効性を命令やプログラムの分かり易さと拡張命令導入の効果の点から検討する．

4．1　命令やプログラムの分かり易さ

本表記法の使用経験が未だ浅く，調査数も少ないが現在迄に得られた結果を述べる．調査対象は電子工学演習Ⅰの受講者で測定方法は小テスト及びアンケートで行なった．

1）命令の表記法

8080 CPU の持つ基本命令を選び，インテル表記法及び提案の表記法にて提示した命令の動作を説明させ，分かり易い表記法を選択させた．たゞし命令は両表記法とも全く同一ではなかった．同一調査紙にて2回の調査を行なった結果を表1に示す．命令の正答率，表記法の選択の結果から提案の表記法が分かり易いと言える．

表1　命令の分かり易さ（%）

項目 ＼ 表記法	命令の正答率		分かり易い表記法	
	1回目	2回目	1回目	2回目
インテル表記法	14.1	27.0	7.4	0
提案の表記法	44.1	75.7	81.5	91.3
どちらでもよい	——	——	11.1	8.7

2）プログラムの分かり易さ

簡単な6つのプログラムを2つの表記法で作成し，分かり易いものに“○”，分かりにくいものに“×”をつけさせて調査した．表2には“○”をつけた割合を示した．ほとんどの学生が提案の表記法を分かり易いプログラムと判断している．

表2　プログラムの分かり易さ（%）

プログラム番号	1	2	3	4	5	6
インテル表記	34.8	13.0	17.4	17.4	26.1	4.3
提案の表記	100	95.6	95.6	91.3	100	100

4．2　拡張命令の効果

拡張命令の導入により異種 CPU 間でのプログラムの互換性を検討できることは既に述べた．拡張命令は過去に作成したプログラムの一部又は全部を以後のプログラム作成に利用することも可能としている．そのためには拡張命令についての色々な情報（入出力のデータ）などを明確に定義し，再利用可能となるように整理しておかなくてはならない．この再利用によってプログラム作成の生産性が向上する．

現在のところ，命令の拡張機能を実際の処理系でどのようにして実現するか検討していないが，データベースと同様な構成法となることが予想できる．

5．あとがき

マイクロコンピュータ用アセンブリ言語の代数的表記法及びプログラマ自身が定義できる拡張命令の使用を提案した．これによってインテル表記法に比較して分かり易く，容易に，能率良くプログラムを書くことができる．プログラム記述の命令を基本命令と拡張命令に区分して考察することで，異なる CPU 間で任意のプログラムが実行可能となる条件を得ることができた．たゞしこれには現在のところメモリ空間やハードウェア上の制限は考慮していない．

本研究に関連する今後の研究課題は

1）8080 CPU での命令の相互関係

2）本表記法の他の CPU への適用

3）本表記法による処理系の作成(基本命令中心)

4）命令の拡張性を考慮した処理系の設計と試作

5）プログラムの互換性を考慮した新 CPU のための

命令セットの決定

などがある.

最後に御指導頂く武石泰亮教授，山下義信助教授はじめ，諸先生方に深く感謝の意を表します.

文　献

1) F. G. Duncan, "Level-Independent Notation for Microcomputer Programs," IEEE MICRO, Vol. 1, No. 2, May 1981, pp. 47 - 52

2) Allan Mosak, "Structured Programming Can Be Applied to Microprocessors-Even by Novices," IEEE MICRO, Vol. 2, No. 1, Feb. 1982, pp. 63 - 71

3) 佐々木，山下："BASE - 80", I/O, Vol. 6, No. 4（昭 56 - 5）

4) アスキー編："続報　SONY SMC - 70", ASCII, Vol. 6, No. 7（昭 57 - 6）

5) インテル　ジャパン："8080/8085 アセンブリ言語（プログラミング　マニュアル）", 1978

6) シャープ："マイクロコンピュータ Z80　ハンドブック[I]", 1979

付録　基本命令一覧表（16進数は機械語を示す）

付表1．8ビット代入表記命令（レジスタ）

表記 \ r, B2	A	B	C	D	E	H	L	M HL	B2	備考
A=r	7F	78	79	7A	7B	7C	7D	7E	3E	
B=r	47	40	41	42	43	44	45	46	06	
C=r	4F	48	49	4A	4B	4C	4D	4E	0E	
D=r	57	50	51	52	53	54	55	56	16	データ転送
E=r	5F	58	59	5A	5B	5C	5D	5E	1E	
H=r	67	60	61	62	63	64	65	66	26	
L=r	6F	68	69	6A	6B	6C	6D	6E	2E	
(HL)=r	77	70	71	72	73	74	75	—	36	
r=r+1	3C	04	0C	14	1C	24	2C	34	—	増加
r=r+1	3D	05	0D	15	1D	25	2D	35	—	減少
A=A+r	87	80	81	82	83	84	85	86	C6	加算
A=A+r+CY	8F	88	89	8A	8B	8C	8D	8E	CE	
A=A−r	97	90	91	92	93	94	95	96	D6	減算
A=A−r−CY	9F	98	99	9A	9B	9C	9D	9E	DE	
A=A AND r	A7	A0	A1	A2	A3	A4	A5	A6	E6	
A=A or r	B7	B0	B1	B2	B3	B4	B5	B6	F6	論理演算
A=A XOR r	AF	A8	A9	AA	AB	AC	AD	AE	EE	
F(A−r)	BF	B8	B9	BA	BB	BC	BD	BE	FE	フラグセット

※　B2：　2バイト目のデータ（1バイト）

付表2．ペアレジスタ（PR）に関する命令

表記 \ PR	BC	DE	HL	SP	AF	備　考
PR=B3B2	01	11	21	31	—	16ビットデータセット
HL=HL+PR	09	19	29	39	—	16ビット加算
PR=PR+1	03	13	23	33	—	PR増加
PR=PR−1	0B	1B	2B	3B	—	PR減少
PUSH　PR	C5	D5	E5	—	F5	スタック操作
POP　PR	C1	D1	E1	—	F1	

付表3　分岐，サブルーチン命令

表記 \ 条件	無条件	条件付 ZF=0	ZF=1	CF=0	CF=1	PF=0	PF=1	SF=0	SF=1	備　考
GOTO B3B2	C3	C2	CA	D2	DA	E2	EA	F2	FA	ジャンプ
GOSUB B3B2	CD	C4	CC	D4	DC	E4	EC	F4	FC	サブルーチンコール
RET	C9	C0	C8	D0	D8	E0	E8	F0	F8	

※　B3B2：　3，2バイト目のデータ（2バイト）

付表4　その他雑命令

表　記	機械語	表　記	機械語	備　考
(BC)=A	02	A=(BC)	0A	Accとのデータ転送
(DE)=A	12	A=(DE)	1A	
(B3B2)=A	32	A=(B3B2)	3A	
(B3B2)=HL	22	HL=(B3B2)	2A	HLレジスタとのデータ転送
(SP):=:HL	E3	DE:=:HL	EB	
PC=HL	E9	SP=HL	F9	
P(B2)=A	D3	A=P(B2)	DB	入出力
CY=1	37	CY=$\overline{CY}$	3F	キャリ　操作
A=$\overline{A}$	2F	DAA	27	Acc
RLCA	07	RRCA	0F	回転(シフト)
RLA	17	RRA	1F	
DI	F3	EI	FB	割込制御
HLT	27	NOP	00	
RST 0	C7	RST 1	CF	リスタート命令
RST 2	D7	RST 3	DF	
RST 4	E7	RST 5	EF	
RST 6	F7	RST 7	FF	