3.4 言語プロセッサ

ミニコン用のプログラミング言語には，一般の計算機と同様に，基礎的なものとしてアセンブラ語が，コンパイラ言語として FORTRAN 語などが使われる．これらの言語のプロセッサ（処理系）も，一般の計算機システムに類似している．しかし，大型の計算機で行われているバッチ処理の手順——翻訳，編集結合…——をミニコンに適用するには，高速で，大容量の補助記憶装置（ドラム，ディスクなど）を用いないと非能率である．また，ミニコンの主記憶装置の容量は，通常，少ないので，ただ単に大型機の小型版としてミニコンを用いることには限界がある．ミニコンで働かせるプログラムは比較的小さいものが多く，このようなプログラムを処理するのに，大型機で行われている方法は，（特に，補助記憶装置に恵まれていない）ミニコンには不適切である．ミニコンの処理能率を高めるには，ミニコンの特質を生かした処理系が必要である．ここでは，この観点から，16 ビット 4K 語程度の主記憶と，入出力タイプライタ，紙テープ等の入力機器だけで構成されるミニコンでも，能率的に使うことができるように設計されている言語処理系について述べることにする．より豊富な記憶装置が備わっているシステムでは，一般の計算機システムの小型版と考えられる処理系もある．しかし，窮屈なミニコンで使われるように設計された処理系は，むしろ大型機に移行しても優れたものであることが多いからである．

3.4.1 アセンブラ[4]

アセンブラ語は，機械命令に対応づけられている言語で，ソフトウェア作成の基本となるものであり，記憶量に厳しい制限のあるミニコンのプログラムを作成するうえで，特によく用いられる重要なものである．アセンブラ語は，ハードウェアの構成に密接に関係しているので，一般的に扱うのは困難である．処理系（アセンブラ）には，アセンブラ語で書かれたプログラムを読みながら，主記憶装置に機械語命令を作ってゆくもの，もとのプログラムを数回読み込むもの，途中まで処理された目的プログラムを補助記憶に出して，もう一度それを読み込んで機械語命令の列を作り出すものなどがある．アセンブラは，機械語命令に対応する命令を，（人が）憶えやすいように記号で書けるようにした命令の部分の処理と，プログラム上の位置を名札で指定することができるようにされている部分の処理，それに，アセンブラへの制御指令(例えば，プログラムの終りを処理系に知らせる)の処理を行う．これらの処理を能率的に行うように設計するのはもちろんであるが，処理の方式はアセンブラ語に反映されることが多いので，使いやすい（書きやすい）良いアセンブラ語であるためには，これらの処理を系統的に行う処理系が望まれる．

また，特に機械語命令の名称は，利用者が憶えやすいものであることが必須であり，この名称を自由に変更できる機能も備わっているのがよいであろう．ミニコンにふさわしい

機能を備え，しかも小型であるアセンブラとしては，SPA[4]，NASS などがある．

3.4.2 FORTRAN[5~8]

FORTRAN 語は，主に，科学技術計算によく用いられる言語であり，多くのミニコンにはその処理系が備わっている．処理系には，一般の計算機の処理系の多くがそうであるように，プログラム単位毎に翻訳を行い，編集結合して実行可能な目的プログラムを作り出すものと，プログラム全体をひとまとめにして翻訳し直接に実行できる形にするものがある．さらに，処理系が（実行の際に必要なルーチンも含めて）主記憶装置に常駐する，いわゆる常駐型処理系もある．主記憶装置の容量が少ないミニコンに常駐する処理系を用いると，プログラムの大きさはより厳しい制限のもとにおかれることになるが，比較的小さいプログラムのデバッグや，それらを数多く処理する教育実習用としては，重要なものである．ミニコンの FORTRAN 処理系の比較も報告されている[5]が，常駐型の処理系の重要性，必要性が認識されて，E-FORTRAN[7]，ELF[8] などが作られている．どちらも，言語の水準はほぼ JIS 3000 に準拠しているが，ミニコンの事情を考慮して，使いやすいように工夫されている．常駐型の処理系は，もちろん小型である必要があり，多くは，コンパイラ-インタプリタの方式をとっている．すなわち，翻訳した結果の目的プログラムは，機械語で書かれたものではなくて，“架空の機械（語）”を想定して，その命令の列を作り出し，実行時には，解釈実行ルーチン——インタプリタ——によって“架空の機械”が実行するのである．この方式の利点は，処理系が比較的小型に作れること，目的プログラムを小さくできること，それに，常駐している処理系を安全に保護し，実行時の誤りの検出も容易であることなどである．一方，この方式では実行時の能率が極度に悪く，機械語命令で実行する場合に比べて数 100 倍の時間がかかることもある．FORTRAN の処理系も，目的に合わせて選択し，全体として能率のよい利用法を心掛けるべきである．作成されたプログラムが頻繁に使われるもので，変更されることのないものや，計算の量が非常に多いプログラムは，翻訳の際に時間がかかっても，実行時に能率のよい目的プログラムが作られるコンパイラを用いるのがよい．ミニコンで，機械語命令の目的プログラムを作り出すコンパイラは，通常はいくつかのパスに分けて，途中の結果を補助記憶に出力するものが多い．

3.4.3 PLAN[9]

ミニコン用のコンパイラ言語の処理系は，大型機に備わっている言語処理系をできるだけ言語仕様を変えないで，小型版にしたものが多い（例えば，FORTRAN）．しかし，ミニコンの特徴を生かし，ミニコンの制限にふさわしい言語で能率よく処理を行うことが要求されるとき，大型機で用いられている言語の仕様をそのまま持ち込むのは，ミニコンにとって荷が重い．そこで，ミニコンにふさわしいコンパイラ言語が必要になる．PLAN は，主として，ミニコン用に設計された言語であり，ALGOL の特徴を生かしたものであ

る．処理系は，16 ビット 2K 語以内におさまり，常駐型コンパイラとして利用でき，しかも実行時の効率を考慮して，機械語命令が目的プログラムとして作り出されるように設計された言語である．データの型は1つ（整数型）で，データの構造は（1次元の）配列だけである．プログラムの構成要素としての名前には，変数名，配列名，名札名，および手続き名の区別があり，これらはすべて宣言してから用いなければならない．また，これらの名前の有効範囲は，手続きを単位として決まる．動作を記述する文には，代入文，構造をもった文（条件判定，反復などを表す），飛越し文，手続き呼出し文，入出力文などがあり，いくつかの文をくくって1つの文にすることもできる．手続きは再帰的に呼び出してもよい．PLAN は，計算手順（算法）の開発と験証に役立つように設計されたものであるが，アセンブラ語でプログラミングを行う作業の労力の軽減にもなる．記憶容量等の制約から，ミニコン用のコンパイラ言語では，限られた機能しか持たせられないが，PLAN ではプログラムの構成要素として必要かつ有用なものを備えている．データ構造の表現機能に乏しいが，ミニコンで，記憶場所，時間の両面で，複雑な構造をもつデータを能率よく処理するには，機械に固有の命令を有利に使えるように言語を設計するのが望ましい．

【例】 ハノイの塔の問題[14] (PLAN)

```
{TOWERS OF HANOI}
proc HANOI;
def HANOI(from,via,to,N)
    proc MOVE;
    def MOVE(from,to,N)
        begin put(eol,!MOVE PIECE !,N+"0,
                      ! FROM !,from,! TO !,to,!.!)
        end{MOVE};
    begin if N>0 do
          begin HANOI(from,to,via,N-1);
                MOVE(from,to,N);
                HANOI(via,from,to,N-1)
          end
    end{HANOI};
begin {MAIN}
      put(eol,! *** TOWERS OF HANOI ***!,eol);
      HANOI("A,"B,"C, 4)
end{TOWERS OF HANOI}.
```

図 3.14

```
*** TOWERS OF HANOI ***

MOVE PIECE 1 FROM A TO B.
MOVE PIECE 2 FROM A TO C.
MOVE PIECE 1 FROM B TO C.
MOVE PIECE 3 FROM A TO B.
MOVE PIECE 1 FROM C TO A.
MOVE PIECE 2 FROM C TO B.
MOVE PIECE 1 FROM A TO B.
MOVE PIECE 4 FROM A TO C.
MOVE PIECE 1 FROM B TO C.
MOVE PIECE 2 FROM B TO A.
MOVE PIECE 1 FROM C TO A.
MOVE PIECE 3 FROM B TO C.
MOVE PIECE 1 FROM A TO B.
MOVE PIECE 2 FROM A TO C.
MOVE PIECE 1 FROM B TO C.
```

図 3.14 (つづき)

3.4.4 FOL[10]

FORTRAN 語を基本として，ALGOL の長所であるブロック構造などを取り入れ，プログラムを書く際の表現法が強化された言語である．表現の多様性が複雑な感を与えるが，問題に応じた記述を行える利点がある．手続きの再帰的な呼出しは許されていない．

3.4.5 BASIC[11]

BASIC は，時分割処理 (TSS) 用言語として設計されたものであり，ミニコンを，1つの端末装置のように対話的に使用する際には，便利な言語である．プログラムやデータの編集機能をある程度まで備えているので，少量のデータ処理に適しているといえる．プログラムテキストや，データを貯える必要があるので，高速の補助記憶装置を備えていることが望ましい．

3.4.6 SIMPLE[12]

ミニコン向けに，FORTRAN から機能を抜粋して設計された言語で，初心者の負担が少ないように工夫されている．

3.4.7 リスト処理，文字処理用言語[13]

通常，作業用領域を多く必要とする LISP などのリスト処理言語の処理系は，ミニコンには向いていないといえるが，これらの処理系の研究用として作られている．

文字処理用の言語は，ミニコンにおいても利用価値が高い．適当なマクロを設定してプログラムを書き，これらの言語処理系で，マクロの展開を行う場合に便利である．マクロ展開などの処理にふさわしい言語としては，GPM，TRAC などがある．これらの処理系はいずれも小さく，ミニコンに適しているといえる．

【例】 ハノイの塔の問題[14] (TRAC)

```
#(PS,(

 *** TOWERS OF HANOI ***
))#(DS,HANOI,
(#(GT,n,0,
(#(HANOI,from,to,via,##(AD,n,-1))
#(PS,(
)MOVE PIECE n FROM from TO to.)
#(HANOI,via,from,to,##(AD,n,-1))))))
#(SS,HANOI,from,via,to,n)
#(HANOI,A,B,C,4)
```

図 3.15

3.4.8 プログラム例

本節にあげたいくつかのミニコン用のプログラミング言語で,「データの整列化 (並べかえ) を行う」プログラム (サブルーチン) を書いたものを示す.

```
SORT: ; "return address save area"
      ASK/A-0/; "clear Acc"
      S/N;      "subtract N, i.e. -N"
      T/1;      "index register 1 --- i"
SORT1:L/1;      "load i"
      T/2;      "index register 2 --- j"
      T/3;      "index register 3 --- k"
SORT2:IMS/3;    "increment k & skip if k=0"
      J/SORT3;  "jump to SORT3"
      L/I/A1#1; "A[i]"
      T/W;      "store in W"
      L/I/A1#2; "A[j]"
      T/I/A1#1; "A[i]"
      L/W;
      T/I/A1#2; "A[j]"
      IMS/1;
      J/SORT1;  "next i"
      J/I/SORT; "return"
SORT3:L/I/A1#2; "A[j]"
      CMP/I/A1#3; "compare with A[k]"
      J/SORT2;    "case A[j]>A[k]"
      J/SORT2;    "case A[j]=A[k]"
      L/3;        "case A[j]<A[k]"
      T/2;
      J/SORT2;
A1: ; "address of the bottom of A"
W: ; "work area"
A: ; "beginning of data area"
N: ; "size of data"
```

図 3.16 (a) アセンブラ語 (NASS)

```
      SUBROUTINE SORT(A,N)
      DIMENSION A(100)
      N1=N-1
      DO 10 I=1,N1
      J=I
      K1=I+1
       DO 20 K=K1,N
       IF(A(J)-A(K))21,20,20
   21  J=K
   20  CONTINUE
      W=A(I)
      A(I)=A(J)
      A(J)=W
   10 CONTINUE
      RETURN
      END
```

図 3.16 (b) FORTRAN

```
proc SORT;
def SORT(array A(array A), N(size))
    var i,j,k,W {auxiliary variables};
    begin i←0;
          while i<N do
                begin i←i+1;
                      j←i; k←i;
                      while k<N do
                            begin k←k+1;
                                  if A[j]<A[k] do j←k
                            end;
                      {A[j] > or = A[k] for k=i,...,N}
                      W←A[i]; A[i]←A[j]; A[j]←W
                end
    end {procedure SORT};
```

図 3.16 (c) PLAN

```
1000 REM---SUBROUTINE SORT
1010 FOR I=1 TO N-1
1020  LET J=I
1030  FOR K=I+1 TO N
1040   IF A(J)>=A(K) THEN 1060
1050   LET J=K
1060  NEXT K
1070  LET W=A(I)
1080  LET A(I)=A(J)
1090  LET A(J)=W
1100 NEXT I
1110 RETURN
```

図 3.16 (d) BASIC

3.5 ユーティリティプログラム

3.5.1 イニシャルプログラムローダ (IPL)[15,16]

イニシャルプログラムローダ (IPL) は，主記憶装置に他の記憶装置（媒体）にあるプログラムを格納する基本的なローダである．通常は，機械語で数ステップのプログラムであり，コンソールパネルのレジスタから，記憶装置に格納して実行する．数ステップのプログラムで，一般的なローダの役割を果たすことは不可能であり，IPL によって，本格的なローダを主記憶装置に格納するシステムが多い．したがって IPL は，できるだけ簡単で，短いものが便利である．IPL が読み込むプログラムの形式が少々複雑になっても，IPL のステップ数の短いものが使いやすい．IPL のプログラムは，ハードウェアの仕様に強く依存するので，一般的に述べることは難しいが，HITAC-10[15,16]，MACC 7/S[15] の IPL は，6～7 ステップである．これらによって，20～30 ステップ程度の 2 次プログラムローダを読み込み，本格的な，システムプログラムローダが格納されることになる．

機械によっては，ハードウェアとして，適当なローダが備わっていて，IPL の役割をこのローダにまかせられるものもあるが，そうでない場合には，機械の特徴を生かして，できるだけステップ数の少ない IPL を作るのが望ましい．

3.5.2 エディタ[17~19]

プログラムやデータなどのテキストを，編集したり，訂正したりするためのユーティリティプログラムとして，エディタは欠かせないものである．特に，紙テープを基本的な外部記憶の媒体として用いるシステムでは，エディタの有無で，処理効率が格段に異なる．

時分割システム（TSS）で，通常，タイプライタ端末で行う操作が，ミニコンの場合にも適用できる．したがって，従来開発されていた TSS 用のエディタ[17]が参考にされることが多い．

ミニコンで使用されているエディタで，比較的小型のものとして，CTSS 方式の文脈エディタ (context editor)[18]がある．文脈エディタのほかに，特に，カードを基本とするバッチシステムでよく用いられる，行番号エディタも考えられるが，ミニコンで対話的にテキスト編集を行うには，この方式は不適切であろう．上にあげた文脈エディタは，テキストの，行を示す指針（ポインタ）が用意されていて，テキストのファイルに対して，この指針によって位置を指定する．したがって，編集の指令としては，指針を操作するものと，指針で指されている行に対してテキストを操作するものがある．また，文字列が一致する行に，指針を移動する指令もある．行内の文字列に対しては，文字の列を置き換える操作が可能なように拡張されたエディタも使われている．この形式のエディタのプログラムは，約 250～500 ステップであり，主記憶装置の小さいミニコンでも充分使用できる．

テキストファイルを，文字を単位として操作する方式のエディタ[19]も使われる．この方

式は，タイプライタ端末の場合のほかに，文字ディスプレイ装置による編集に便利なものである．

ディスクやドラムなどの補助記憶装置が備わっているシステムでは，これらをファイルとして使用することができる．この場合にも，エディタが便利である．

```
                                        説　　明
>Q                                      CLEAR FILE

>B                                      MOVE POINTER TO BOTTOM
THIS IS                                 & ENTER TEXT MODE
AN EXAMPLE                              INSERT LINES IN FILE
OF
TEXT EDITING
                                        RETURN COMMAND MODE
>T                                      DISPLAY TOP OF FILE
THIS IS
>N3                                     MOVE POINTER DOWNWARD
TEXT EDITING
>L2                                     MOVE POINTER UPWARD
AN EXAMPLE
>T
THIS IS
>FOF                                    FIND LINE 'OF...'
OF
>T
THIS IS
>IEXAMPLE ---                           INSERT LINE
>T
EXAMPLE ---
>P5                                     PRINT LINES
EXAMPLE ---
THIS IS
AN EXAMPLE
OF
TEXT EDITING
>
```

図 3.17　文脈エディタ使用例

3.5.3　浮動小数点演算ルーチン[20,21]

ミニコンの標準構成として，浮動小数点演算がハードウェアで備わっているものは少数である．浮動小数点演算がハードウェアで備わっていても，倍精度演算が必要なときには，ソフトウェアで面倒をみなくてはならない．浮動小数点数の表現法は，数を仮数部と指数部とに分けて表現するものである．どちらも，2のべき（2，16 など）を，底として

表すことが多い．16 ビットが 1 語のミニコンでは，1 つの浮動小数点数を表現するのに，2 語（32 ビット）を用いて，そのうちの 24 ビットを仮数部，8 ビットを指数部として用いることが多い．仮数部や指数部の表現の方法は，さまざまであるが，仮数部は 2 の補数表示を用いることが多い．また，仮数部の，標準の小数点の位置についても，浮動小数点数と，固定小数点の整数の変換が簡単になるように工夫された形式も使われる[20]．

浮動小数点演算を処理する際に，被演算数を，特定の場所に格納して行うものや，ソフトウェアでスタックの処理を行って，スタック上の 2 数についての演算を行うものなどがある．浮動小数点演算の四則などが単独で用いられることは少ないので，インタプリタを設計して，通常の機械語命令と同じ形でプログラムが書けるようにしたものもある．FORTRAN などに組み込まれる初等関数の計算ルーチンや，浮動小数点数の入出力ルーチンなども用意されているものが多い．

初等関数 (sin, cos, arctan, exp, log, sqrt) などを多用する場合には，CORDIC の方式も考えられる[21,22]．

［武市　正人］

文　献

3.1 節

1) 一松　信：電子計算機のプログラミング，日本評論社 (1968).
2) D.E. Knuth : The Art of Computer Programming, Vol.1, Addison-Wesley (1968) の第 2 章.
3) 森口繁一：システムあばき事始め——逆アセンブラについて——，ミニコン (bit 増刊)，共立出版 (1971)，pp.82-87.

3.4 節

4) 島内剛一：システムプログラムの実際，サイエンス社 (1972).
5) 安楽省吾：ミニフォートラン，ミニコン (bit 臨時増刊)，共立出版 (1971)，pp.48-61.
6) 森口繁一：教育実習用ミニコンピュータのコンパイラ開発コンクールについて，第 13 回プログラミングシンポジウム報告集，情報処理学会 (1972)，pp.241-266.
7) 松下通信工業：E-FORTRAN 説明書.
8) 島内剛一，筧捷彦，辻尚史：FORTRAN の実際，サイエンス社 (1973).
9) 武市正人：ミニ言語のミニ・コンパイラ，bit Vol. 6, No. 8, 10, 11, 12, 13.
10) 島内剛一，他：NHK コンピューター講座テキスト，日本放送出版協会 (1974).
11) 高沢嘉光：ミニコンによる大きな BASIC，bit Vol 6, No. 3.
12) 小林光夫，小林久江：ミニコン向きの算法言語 SIMPLE，bit Vol. 4, No. 3 (1972).
13) 前野年紀：C. Strachey の GPM について，ミニコンのソフトウェアとネットワーク報告集，情報処理学会，pp.25-32.
14) C.L. Liu : Introduction to Combinatorial Mathematics, Example 3-7 (p.68), Mc Graw-Hill (1968).（邦訳　伊理正夫他訳：組合せ数学入門 I，共立出版）.

3.5 節

15) 岩村聯，辻尚史，島内剛一：ミニ IPL，ミニコン (bit 臨時増刊)，共立出版 (1971)，pp.88-95.
16) 島内剛一：システムプログラムの実際，サイエンス社 (1972).
17) P.A. Crisman ed. : The Compatible Time-Sharing System, A Programmer's Guide 2nd Ed., MIT Press (1965).

18) 和田英一：ミニエディタ，ミニコン（bit 臨時増刊），共立出版（1971），pp. 40-47.
19) 島内剛一，他：NHK コンピューター講座テキスト，日本放送出版協会（1973，1974）.
20) 島内剛一，筧捷彦，辻尚史：FORTRAN の実際，サイエンス社（1973）.
21) J. S. Walther : A unified algorithm for elementary functions, SJCC 1971, pp. 379-385.
22) 一松　信：CORDIC およびその変形にもとづく初等函数計算，京都大学数理解析研究所講究録「計算の手間と能率化」（1974）.