会話型アセンブラの開発

金　野　靖　英*

A New Conversation-type Assembler System

Yasuhide KONNO

要　旨

会話型のアセンブラ・システムを開発したので報告する。このアセンブラは，原始プログラムを格納する領域を持ち，ソースの訂正可能なソース・エディタ機能，アセンブル・アンド・ゴーの機能をそなえた常駐型のシステムである。又アセンブラ言語は既成のシステムに含まれるように作成してある。

Synopsis

This paper describes a new conversation-type assembler. It is a standing type system which has a source pool and a source editor. And its assembler language is included by one of the existing system.

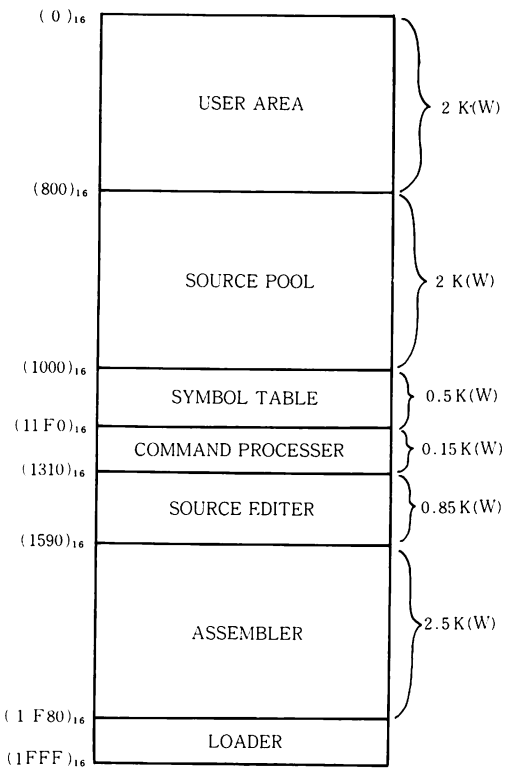
1．ま　え　が　き

コンピュータのハードウェアの，ユーザ側からの理解や，その機能を最大限に引き出して使用する上で，アセンブラ言語でのプログラミング実習は重要な位置をしめている。しかしながらミニ・コンピュータの場合，提供されている既成のシステムでは，その実習に不向きな場合が多い。これはアセンブラがシステム設計に必要であるために，主記憶装置の全領域を使用可能にするよう考慮されているからである。したがって非常駐型のアセンブラ・システムが主であり，原始プログラムや目的プログラムを外部記憶装置又は外部媒体（紙テープの場合が多い）に一旦待避して，それぞれの手順に応じて実行まで行なわれるようになっている。特に入出力媒体として，紙テープを使用するようなミニ・コンピュータ特有のシステムでは，紙テープパンチより実行にいたる手間や修正する手間が無視できず，スムーズなアセンブラ言語の習得に，大きな障害となっている。したがってアセンブラ言語の初期学習時には，使用する主記憶領域は制限されても，会話型で原始プログラムの訂正も可能な常駐型のアセンブラ・システムが望ましいと思われる。このような点から会話型アセンブラ・システムを開発した。使用するアセンブラ言話も既成のシステム（以後M－70アセンブラという）に含まれるようにし，次の学習段階にスムーズに移行できるよう発展性をもたせた。全体的にM－70アセンブラよりも簡略化されている。使用するハードウェアは，MELCOM－70（以後M－70と略す）とテレタイプライタ，高速紙テープ読取機によって構成される簡単なシステムである。M－70の主記憶は8K語（1語は16 bit）である。

2．会話型アセンブラの概要

このアセンブラは，コマンド処理部，ソース・エディタ部，アセンブル部の三つに大きく分けられる。又原始プログラムの格納領域，シンボル表を持っている。主記憶の領域配置は図1に示される。コマンド処理部が，約0.15K語，ソース・エディタ部が約0.85K語，アセンブル部が約2.6K語で処理部全体で約3.5K語である。又原始プログラム格納領域が2K語，シンボル表が約0.5K語であり，システムは6K語である。したがって，ユーザは，0番地から2047番地までの2K語が使用可能である。M－70ハードウェアでは，直接アドレス指定できる番地は0ページ（0～255番地）

* 助教授　電気工学科



図－1　領域配置

であることと，0ページには，スタックやオートインデキシング等の可能な特殊な番地や割込みの応答番地があるため，システムを8K語の後ろから取り，0番地から2K語までが，ユーザ使用可能領域である。

3．コマンド処理

アセンブラで使用可能なコマンドは8種でシステムの入力要求指示語“>”の次に有効である。

(1) NEW コマンド

NEW ← (←はリターンの意味)

登録中のプログラムを消去し，新たなプログラム登録のための初期化を行なう。“ⓐ”をテレタイプより出力し，プログラム登録受付中であることを知らせる。1行受付けるごとに“ⓐ”を出力する。文として：ENDの入力により，“>”が出力されコマンド受付状態となる。

(2) DEL コマンド

DEL　n←

登録されたプログラムの文を消去する。nは，その文のロケーション番地をあらわし，10進数，16進数（数字の前にスラッシュ記号“/”をつけて表示）どちらでもよい。又数字の後に“+”があると，その番地の前の文，“−”があると，次の文を指定できる。これは，ロケーション番地のないアセンブラ擬似命令やコメントの消去に用いる。

(3) INS コマンド

INS　n 〈文〉←

登録されたプログラムに文を追加する。nは，挿入する文のロケーション番地をあらわし，10進，16進のどちらでもよい。数字の後に“+”がある場合は，指定された番地の次に文を挿入し，“−”なら前に挿入する。ない場合は，指定のロケーション番地の文に換えて挿入する。

(4) LST コマンド

LST ←

登録されたプログラムのリストを，テレタイプより出力する。

(5) ASS コマンド

ASS ←

登録されたプログラムのアセンブルを行ない，実行領域への目的プログラムの作成と，エラーの出力を含めたアセンブルリストの出力を行なう。

(6) RUN コマンド

RUN　n←

nで指定された番地より，目的プログラムを実行する。RUNによる実行の終了は

①HALT命令を検出した時

②プログラムの最後まで実行した時

③エラーを検出した時

④テレタイプのアテンション・キーが打鍵された時

である。

(7) TR コマンド

TR ←

新たなプログラムを登録するための初期化を行ない，紙テープ読取機より，ソース・テープを読み込み，原始プログラム格納領域にいれる。

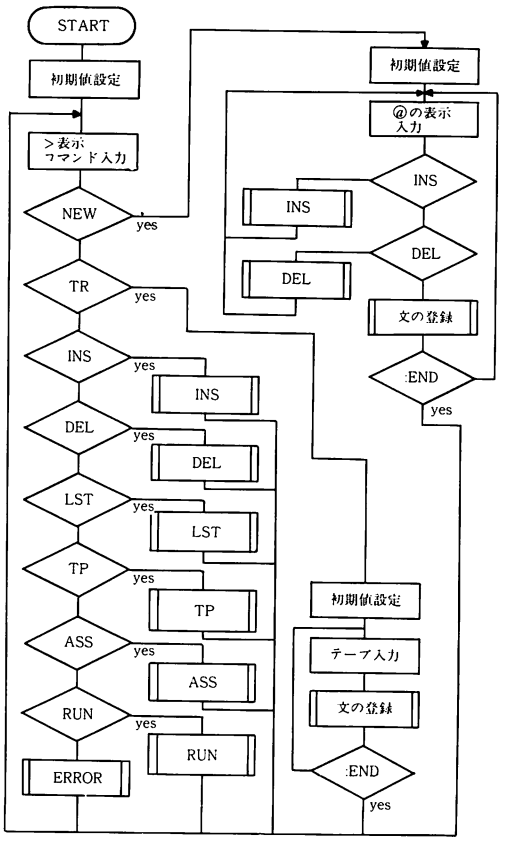
(8) TP コマンド

TP ←

登録されたプログラムを紙テープに穿孔して出

力する。

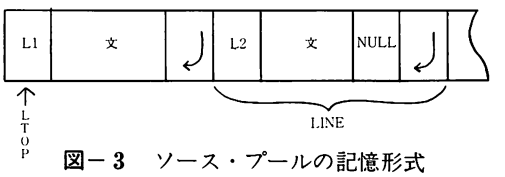
テレタイプからの入力モードは“>”と出力されている場合と，NEW コマンドによって“ⓐ”が出力されている場合の2通りである。“ⓐ”の場合には，文と INS, DEL の両コマンドのみが受付けられ，登録中のプログラムの文の削除，追加が可能である。コマンド処理部での処理のフローチャートは，図2に示される。



図－2　コマンド処理部フローチャート

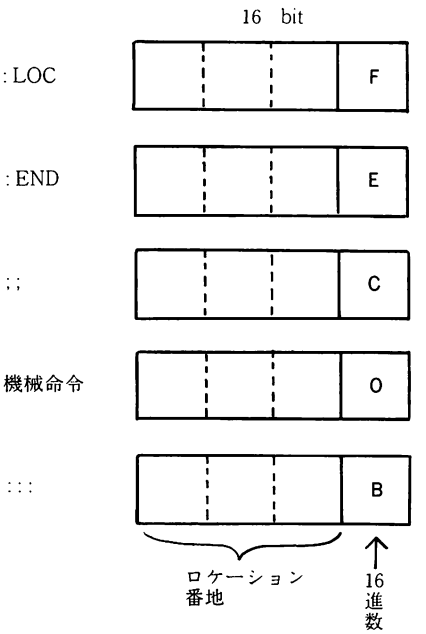
4．ソース・エディタ

原始プログラムを格納するために，ソースプールと名付ける2K語の領域がとられているが，このソース・プールの管理をするのがソース・エディタである。NEW, DEL, INS, LST, TP, TR の6種のコマンドは，その指示によって，ソース・エディタで処理される。ソース・プールはプール・フラグによって，格納される番地の位置が示され，2K語を越えると，注意のメッセージを出力する。メッセージを無視すれば，ユーザ領域はせばめられるが，連続して使用可能である。NEW, TR の両コマンドにより，プール・フラグはクリアされる。ソース・プールでの原始プログラムの記憶形式は，最初の1語（LTOP という）に格納される文のロケーション番地が格納され，次に文が1語に2文字ずつ格納され，最後にリターン・コードが格納される。これが1行である。行が奇数文字で構成される場合，行を語の最後で終らせるため，リターン・コードの前に NULL コードを挿入する。図3に記憶形式を示す。



図－3　ソース・プールの記憶形式

アセンブラ擬似命令やコメントはロケーション番地がないので，LTOP の右4bit に，その種類を示す数字をいれる。図4にその詳細を示す。削除，挿入にあたっては，このLTOP を基準として行ない，削除，挿入後に各行の LTOP を調整しなおす。ソース・プールは上位番地（システム側）より格納され，下位番地にむかっている。ソース・プールの内容を出力するのが，LST, TP の両コマンドであるが，TP により出力される紙テープは，TR

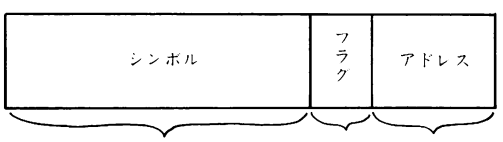


図－4　ＬＴＯＰの詳細

コマンドによりソース・プールに再格納できる。又M−70アセンブラのソース・テープとしても使用できる。

5. アセンブル

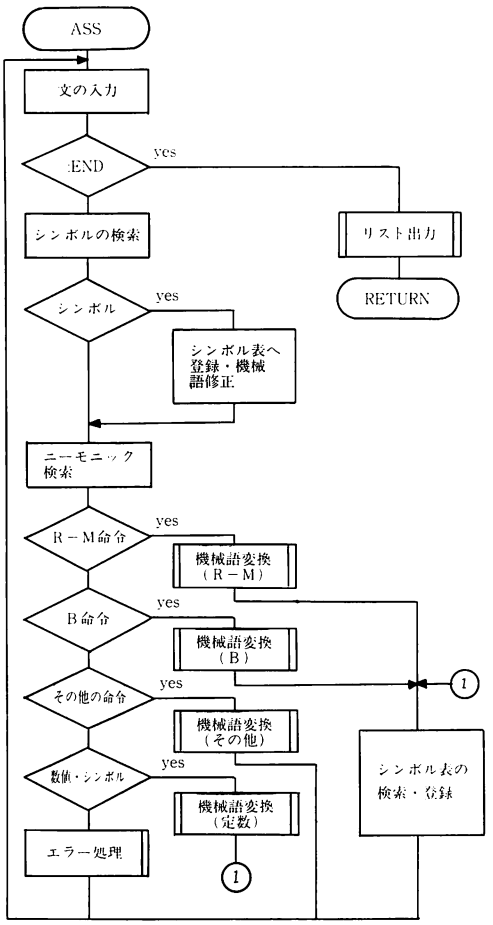
ソース・プールにある原始プログラムを機械語に変換し，実行領域に目的プログラムを作成する処理である。アセンブラ言語はTMA（Tomakomai Technical College M-70 Assembler）と名付け，M−70アセンブラ言語に含まれる。TMAは教育実習用に主眼をおいて作成してある。付録にTMAを示す。処理の方法は，1パスで機械命令なら1命令ずつ機械語に変換して，実行領域に格納する。命令にラベル（シンボル）がついている場合は，シンボル表を調べて未登録なら登録し，既登録未定義なら既定義とし，既登録既定義ならエラーを出力する。TMAのレジスタ・メモリ演算命令，分岐命令，データが，シンボルを引用した場合，シンボル表を調べて既定義であれば，その番地を用いて変換する。未登録，未定義であれば登録する。プログラムによっては複数の未定義シンボルとなる場合もありうる。シンボル表において既登録未定義のシンボルを既定義とする時に，その番地部にある番地の命令のアドレス部を変換する。シンボル表の記憶形式は図5に示す。シンボル表の1記録はシンボル部2語半，未既定義の別半語，登録番地1語の合計4語によって構成される。



フラグ＝$(00)_{16}$　未定義機械命令
＝$(F0)_{16}$　未定義定数
＝$(FF)_{16}$　既定数

図−5　シンボル表の記憶形式

アセンブルにおいては，アセンブルリストを出力する。格納番地，変換された機械語(16進表示)，原始ステートメントが1行として出力される。文法エラーは，その行の先頭に“＊”を出力する。又シンボルの対応がとれない場合は，リストの終りに，その名前と引用された番地をエラーとして出力する。アセンブル部の概要フローチャートを図6に示す。



図−6　アセンブル部概要フローチャート

6. 使　用　例

簡単なプログラムでソース・エディタの使用，アセンブル，実行の例を図7に示す。なおCAL△/30は前もって作成した10進出力ルーチンである。

```
>NEW

@:LOC /200
@A;20
@B;10
@LD R0,A
@AD R0,B
@CAL /30
@HALT
@:END
```

```
>ASS

                :LOC /200
0200   0014   A;20
0201   000A   B;10
0202   01FD   LD R0,A
0203   41FD   AD R0,B
0204   F030   CAL /30
0205   E4FF   HALT
                :END

ERROR   000

>RUN /202

!  30

>DEL /203

>INS/202
  ERROR C.
>INS /202+ SR R0,B
>LST

:LOC /200
A;20
B;10
LD R0,A
SR R0,B
CAL /30
HALT
:END

>
```

図－7　使用例

7. あ と が き

この会話型アセンブラはアセンブラ言語によるプログラムの初期教育用として，作成したものである。システムは処理部3.5 K語，テーブル等2.5 K語の6 K語であり，ユーザは2 K語使用可能である。原始プログラムは約200ステップ，シンボルは100個が使用できる。処理速度はソース・エディタ部において，プログラムが大きい場合，DEL，INSの両コマンドの処理に若干かかるが，他は会話型としてスムーズな取扱いができる。従来，ソース・テープの作成，アセンブラのロードとアセンブル，オブジェクト・テープの作成，オブジェクト・テープのロードと実行というように，3段階になっていたアセンブラの実験実習が，このシステムによって，より使いやすい，習得しやすいものとなった。今後，大いに活躍が期待される。課題としては，INS，DEL，両コマンドの処理速度の改善，アセンブラ擬似命令の追加等が考えられる。最後にシステムのアセンブル部や各種ルーチンの作成に協力いただいた卒研生，戸田勉君（8期），松尾徹君（9期）に深謝いたします。

参 考 文 献

(1) 島内剛一　システムプログラムの実際　サイエンス社（1972）
(2) MELCOM 70　システム説明書（1973）
(3) MELCOM 70　アセンブラ・システム説明書（1973）

（昭和53年11月30日受理）

付　　録

TMA (Tomakomai Technical College M-70 Assembler)

ハードウェアは，M－70であり，アキュムレータ4個（R0，R1，R2，R3とあらわす）のマルチ・アキュムレータ方式である。又，プログラム・ステータス・レジスタをもっている。これは，キャリー（CR），オーバーフロー（OF），ネガティブ（NG），ゼロ（ZR）の4個のフラグを持ち，演算結果によって，セット，リセットされる。主記憶は1語16 bitである。

1．1　アセンブラのプログラムは，ソース・ステートメントおよびコメントから構成される。プログラムは，：ENDで終わらねばならない。ステートメントは自由様式である。

(1)ステートメント
(a)擬似命令ステートメント（アセンブラ制御命令）
(b)ラベル・ステートメント（主記憶のある番地に記号をつける）
(c)ストレッジワード・ステートメント（機械命令とデータ）

(2)コメント
行の空白を除くはじめが，；；であれば　注釈行であり，プログラムの実行に関係しない。

1．2　ステートメントに使用できる文字
英文字，数字，特殊記号（空白：；=，"ⓐ$＋－＊/！&（））

1．3　シンボル（Symbol）
英文字，数字，コロン（：）によって構成され，英文字，コロンが先頭である5文字以内の名前である。

1．4　定数（C_o）

(a)　10進整数

(b)　16進整数　例　/2BF（/を16進数の前につける）

1．5　ロケーションカウンタ・レファレンス

アドレスを指定する場合に，その命令が格納されている番地を“*”で示し，それよりの増減を＋，－であらわす。例（*－10, *＋5）

1．6　アドレス（A_d）

アドレスは番地を表すが，シンボル，定数，ロケーションカウンタ・レファレンスで構成される。

2．擬似命令ステートメント

これはアセンブル時に必要な命令で，アセンブラに対する実行制御命令である。

2．1　ロケーションカウンタ設定命令

：LOC　C_o

メモリ領域割当て機能に対する初期値設定命令であり，ロケーションカウンタにC_oをセットする。

2．2　サブルーチン・エントリ命令

：：：

サブルーチンの入口として用いられ，内容がゼロの2語のメモリが確保される。

2．3　アセンブル終了指示命令

：END

アセンブルの終了を指示する命令であり，プログラムの最後に必らずなければならない。

3．ラベル・ステートメント

アセンブラが割りつけるメモリの番地に記号名をつける。Symbolがその時のロケーションカウンタの値に定義される。Symbolの後には必らず“；”をつけなければならない。

Symbol ；

4．ストレッジワード・ステートメント

対応した機械語の作成ができ，オブジェクトに変換されるステートメントで，次の二つに分けられる。

4．1　データステートメント

シンボル，定数をデータとして記したものである。

Symbol or Co

4．2　機械命令ステートメント

命令シンボルと，そのいくつかの命令オペランドから構成される。

OP－symbol Operand

・OP-symbolは機械命令シンボル（ニーモニック・コード）

・Operandは機械命令を完成するのに必要な項

・OP-symbolとOperand間は1文字以上の空白，Operandの各項はコンマで区切る

以下各命令について記述するが，使用される記号について述べる。

(1)　〔　〕でかこまれている項は省略可能である。

(2)　（　）は，その内容を示す。

(3)　OPはニーモニック・コードを示す。

(4)　PCはプログラムカウンタを示す。

(5)　PSはプログラムステータスを示す。

(6)　R_i, R_j, R_xは使用するレジスタを指定し，i,j,xで0～3までのレジスタを示す。

(7)　→は左の演算結果を右の項にいれることを示す。

(8)　OPAeは命令のアドレス部による実行番地のことである。各アドレス指定を次に示す。

R_i, Ad……直接・相対アドレス

R_i, $Ad……間接アドレス

R_i, Ad, R_x……R_xインデックス修飾アドレス

R_i, $Ad, R_x……R_xインデックス修飾間接アドレス

(9)　Skipは，プログラムステータスの内容によって，条件が満足されれば次の命令をスキップすることを示す。Skipのニーモニック・コードと内容は次のようである。

SKP……無条件スキップ

SCZ……CR＝0の時スキップ

SOZ……OF＝0の時スキップ

SRN……NG＝1の時スキップ

SRP……NG＝0の時スキップ

SRZ……ZR＝1の時スキップ

SNZ……ZR＝0の時スキップ

(10)　Valueは16進数で0～1FFの数値で左の先頭ビットが上位まで拡張されて演算の対象となる。

(11)　Deviceは入出力機器の機番であり，16進数であらわされる。

TMAの機械命令は，8種に分けられ，31命令である（この場合，テスト命令は1，入出力命令は4として）。次に機械命令とその内容について述べる。

4．2．1　レジスタ・メモリ演算命令

OP　R_i〔$〕Ad〔, R_x〕（x＝2，3）

LD——（OPAe）→（R_i）

ST——（R_i）→（OPAe）

AD——（R_i）＋（OPAe）→（R_i）

SB——（R_i）－（OPAe）→（R_i）

4．3．1　レジスタ・オペランド演算命令

OP　R_i, Value

LDI——Value→（R_i）

ＡＮＩ——（R_i）．ＡＮＤ．Value→（R_i）
ＸＯＩ——（R_i）⊕Value→（R_i）
ＡＤＩ——（R_i）＋Value→（R_i）

4．3．2 レジスタ・レジスタ演算命令

ＯＰ　R_i, R_j〔, Skip〕

ＡＤＲ——（R_i）＋（R_j）→（R_i）
ＳＢＲ——（R_i）－（R_j）→（R_i）
ＭＯＶ——（R_j）→（R_i）
ＣＭＰ——（R_i）－（R_j）の結果をＰＳにセット．
（R_i），（R_j）は変化せず。

4．3．3 レジスタ変更命令

ＯＰ　R_i〔, SKip〕

ＣＰＬ——．ＮＯＴ．（R_i）→（R_i）
ＩＮＣ——（R_i）＋1→（R_i）
ＤＥＣ——（R_i）－1→（R_i）
ＭＰＹ——（R_1）×（R_i）→（R_0）2_{16}＋（R_1）
（i＝2 or 3）
ＤＩＶ——（（R_o）2^{16}＋（R_1））/（R_i）→（R_1）
…商，（R_0）…余り（i＝2 or 3）

4．3．4 分岐命令

ＯＰ　〔$〕Ａｄ〔, R_x〕　（ｘ＝2，3）

ＢＲＮ——ＯＰＡｅ→（ＰＣ）
ＢＳＳ——（ＰＣ）→（ＯＰＡｅ），（ＰＳ）→（ＯＰＡｅ＋1），ＯＰＡｅ＋2→（ＰＣ）
ＢＲＳ——（ＯＰＡｅ）→（ＰＣ）
（ＯＰＡｅ＋1）→（ＰＳ）

4．3．5 テスト命令

ＯＰ

SKip のニーモニック・コードの先頭にＴをつけ，単独の命令とした。
（例）ＴＳＣＺ，ＴＳＯＺ

4．3．6 シフト命令

ＯＰ　R_i, Ｎ〔, Skip〕

ＳＲＡ——R_iの内容を右へＮビット算術シフトする。
ＳＲＬ——R_iの内容を右へＮビット，シフトし，あふれは捨てられ，左からは0が入れられる。
ＳＬＬ——R_iの内容を左へＮビット，シフトし，あふれは捨てられ，右からは0が入れられる。
ＲＬＬ——R_iの内容を左へＮビット循環シフトする。

4．3．7 入出力命令

ＯＰ　R_i, Device

ＲＤＡ——データの入力
ＲＤＢ——データの入力
ＲＤＣ——ステータス入力
ＷＲＡ——データの出力
ＷＲＢ——コントロール情報の出力
ＷＲＣ——コントロール情報の出力

ＯＰ　Device

SNSi —— ｉは0～3の数字である。Device の状態をチェックし，ある条件を満足していれば，次の命令をスキップする。
CNTi —— ｉは0～3の数字である。Device への制御指令を与える。
いずれの場合も使用機器によって制御や条件チェックが異なるので，機器使用説明書を参照しなければならない。

4．3．8 その他の命令

ＣＡＬ　Value

ＣＡＬはニーモニック・コードで，これが実行されると，ＰＣは（108）$_{16}$にセットされる。（108）$_{16}$番地からの命令は Value を参照して処理をする。

ＨＡＬＴ

ＣＰＵの全動作を停止する。