機械語教育用擬似計算機について

甲南大学　的場裕司

広辞苑によると、「ニュース」とは「珍奇なこと。新しい出来事。また、その報道。」とあった。そうすると、これから筆者が書こうとしている内容は「珍奇なこと」になるのかも知れない。

現在、一般に広く用いられている計算機はフォン・ノイマン型と呼ばれており、それを利用するためにはプログラムが必要となるが、そのプログラムがどのような言語で表現されていても最後の段階では機械語と呼ばれるその計算機独自の命令によって、その計算機の種々のレジスタや指定されたメモリの番地の内容を用いて作業が進められていくようになっている。しかし、一方では、第五世代の計算機(非ノイマン型計算機)ということで、FORTRAN, COBOL, PL／1, PASCAL等の言語による表現もなくして、もっと人間に使いやすい様式、すなわち、プログラムなどをいちいち作らなくても自分のやりたい仕事を計算機でやらせることができるようにしようと研究が進められつつある。(1) ところが、このような時代に、機械語の教育を効率よく行うような方法について述べようとするのであるから、冒頭に述べた「珍奇なこと」と思われても当然かも知れない。

そこで、この「珍奇なこと」、すなわち、計算機の機械語教育を行うための方法であるが、具体的な計算機の機械語そのものを直接利用するのがごく普通の方法として考えられるが、筆者は具体的な計算機の機械語がはたして機械語教育に適しているかどうかについては多少疑問を持っているため、教育に適した機械語をもつような計算機を想定し、その計算機(擬似計算機と呼ぶ)を利用する方法をとることにした。これに関しては意見が種々わかれるものと思われるが、ここでは筆者の独断をお許し願いたい。

ここで、筆者のいう擬似計算機とは、実際に使用する計算機とは異なる命令体系を持った計算機を想定し、その計算機の動きを実際の計算機を用いてシミュレートするためのプログラムのことである。したがって、擬似計算機の機能は実際に使用する計算機によって大きく影響されることになるのは当然である。そこで、いままでに筆者が関係した機械語教育用擬似計算機(以後擬似計算機と呼ぶ)についてふりかえってみることにする。これは筆者の経歴(注)とも関係してくることになるが、まず、擬似計算機第1号は昭和44年頃大阪大学大型計算機センターにあったNEAC2200/500を用いたものであった。当時は計算機というものは誰でも自由に使用できるというものではなかっ

(注)　昭和42年　3月　大阪大学基礎工学部制御工学科助手
　　昭和42年10月　神戸商船大学助教授
　　昭和46年　7月　大阪大学基礎工学部情報工学科助教授
　　昭和55年　4月　甲南大学理学部経営理学科教授

たため、何とかバッチ処理で機械語教育をやりやすくしようと作成した。[2]~[4] つぎに、データ・ステーション（NEAC　N289A－1）を用いてTSS処理が可能となったので、会話形式で利用可能な擬似計算機を作成し、これが第2号となった。しかし、当時は端末から利用可能なプログラムにはきびしい制限があったため擬似計算機の機能は第1号より低いものであった。[5],[6] 以上2つの擬似計算機はその内部表現は2進数表現をとり、プリンタには8進数として印字する方法がとられている。

やがて、ミニコンと呼ばれる計算機があちこちで導入されるようになり、それにともなって大阪大学基礎工学部情報工学科にもDEC社のPDP－11/20が導入された。当時としては最新鋭のミニコンであり、端末装置としてテレタイプライタ（ASR－33）が3台ついていたので、この3台の端末装置から同時に利用可能となるような擬似計算機を作成した。これが第3号である。[7],[8] ここではじめて内部表現が10進数である擬似計算機となった。当時は最新鋭のミニコンであったがメモリ容量は小さく、そのため擬似計算機の機能としてはたいしたことはなかったが、大型計算機センターの使用と異なり、こちらは自由に使用可能であったためそれなりの効果はあった。

PDP－11/20の導入と同時期に同じく情報工学科にFACOM 230－45Sが導入されたが、こちらの方はバッチ処理であったためマークカードを利用するための擬似計算機を作成した。これが第4号である。[9],[10] FACOM 230－45Sは導入当時としては比較的めぐまれたものであったため、擬似計算機としてはかなり高機能のものが実現可能であったが、マークカードの表現からくる制限のために擬似計算機自体の機能はたいしたことはなかった。しかし、マークカードを利用するためにパンチ室を使用しなくても実習用のプログラム作成が可能となったためそれなりに便利に使用することができた。このとき使用したマークカードを図1に示しておく。

やがて昭和50年代に入り半導体技術の急速な発達によってマイクロコンピュータ（マイコン）が

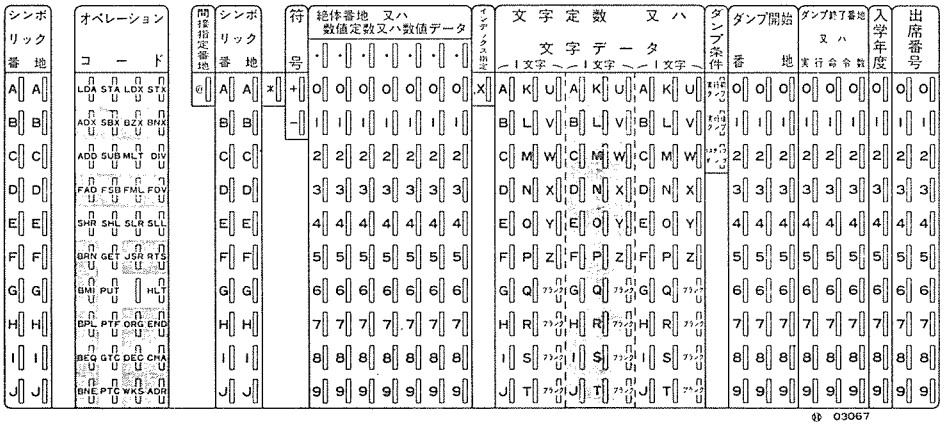


図1

広く利用される時代となり、機械語教育も含めてプログラミング教育というものに対する考え方に大きな変化があらわれてきた。初期のマイコン用プログラムには16進数表現の機械語が用いられ、それが簡単なモニター・プログラムによって処理されるものであったが、半導体技術の発達のみならず周辺機器の著しい発達によって、マイコンもオペレーティング・システム（ＯＳ）の管理下で動作するものとなり、これにともなって、原始プログラムの処理（アセンブルやコンパイル作業）からはじまってそのプログラムが実際に実行されるまでの過程が利用者からかくされてしまうこと

```
  ARCHIMEDES MACRO ASSEMBLER V_02 L_01  DATE 82/09/09 TIME 13:30 PAGE 001

SECTION 001
  LOC  MACHINE  LINE     SOURCE
                   1         ORG   100
                   2         ENTRY X
                   3         EXTRN YEN1
                   4         EXTRN YEN5
  100  02600111    5         LDSP  ASTC
  101  00100109    6   STRT: GET   X
  102  02100108    7         LDX   C5
  103  04900000    8         JSR   YEN5
  104  02100107    9         LDX   C1
  105  04900000   10         JSR   YEN1
  106  03000101   11         BRN   STRT
  107  00000001   12   C1  : DEC   1
  108  00000005   13   C5  : DEC   5
  109  00000000   14   X   : WRKS  1
  110  00000000   15   STCK: WRKS  1
  111  00000110   16   ASTC: ADR   STCK
                   17         END

SYMBOL TABLE  (CROSS)
  X    109 G     YEN1 *** F     YEN5 *** E
  STRT 101       C1   107       C5   108
  STCK 110       ASTC 111

  ERROR  MESSAGE
  NO.  LINE   MESSAGE


       0  WARNINGS
       0  ERRORS


  ARCHIMEDES MACRO ASSEMBLER V_02 L_01  DATE 82/09/09 TIME 13:30 PAGE 001

SECTION 002
  LOC  MACHINE  LINE     SOURCE
                   1         ORG   200
                   2         ENTRY YEN1
                   3         ENTRY YEN5
                   4         EXTRN X
  200  00500000    5   YEN5: LDA   X
  201  02000211    6   YEN1: STX   N
  202  01400211    7         DIV   N
  203  00800212    8         STA   ANS
  204  00200212    9         PUT   ANS
  205  01300211   10         MLT   N
  206  01200000   11         SUB   X
  207  01300210   12         MLT   MONE
  208  00800000   13         STA   X
  209  05000000   14         RTS
  210  10000001   15   MONE: DEC   -1
  211  00000000   16   N   : WRKS  1
  212  00000000   17   ANS : WRKS  1
                   18         END

SYMBOL TABLE  (CROSS)
  YEN1 201 G     YEN5 200 G     X    *** E
  MONE 210       N    211       ANS  212

  ERROR  MESSAGE
  NO.  LINE   MESSAGE


       0  WARNINGS
       0  ERRORS

      *** ASSEMBLE  NORMAL END ***
```

セクション1

セクション2

図2

になった。一般の利用者にとってはこのほうが望ましいかも知れないが、情報工学科の学生などにとってはこのかくされた過程が具体的にどのようなものであるかについてその基本的な事柄を知る必要があるのではないかという考えによって、FACOM 230－45Sを利用して擬似計算機の第5号を作成した。[11] この擬似計算機では、前述の原始プログラム（ここではアセンブリ言語によるもの）から目的モジュール、ロード・モジュールへと変化して行く過程を具体的に目に見えるようにし、また、プログラムをモジュール化した場合の外部記号の具体的な処理なども見えるように考慮されている。

以下では、この擬似計算機について具体的に例題を示すことにより、その機能の一部分をみていくことにする。

そこで、具体例としてあまり大きくない金額をデータとして読み、それが5円および1円硬貨のどのような組合せで表現されるかを計算するプログラムについてみていくことにする。ただし、このプログラムは擬似計算機の特徴を示すために、わざわざ外部記号というものを用いて2つの原始プログラムにより構成してある。具体的な原始プログラムは図2の点線で囲んだ部分である。セクション1ではデータ（金額）を記号Xで示される番地（109番地）に読み込み、5円および1円硬貨の枚数を計算するサブルーチン（セクション2のプログラム）を利用するようになっている。セクション2はセクション1から利用されるサブルーチンでその入口は2ケ所（記号YEN1 および

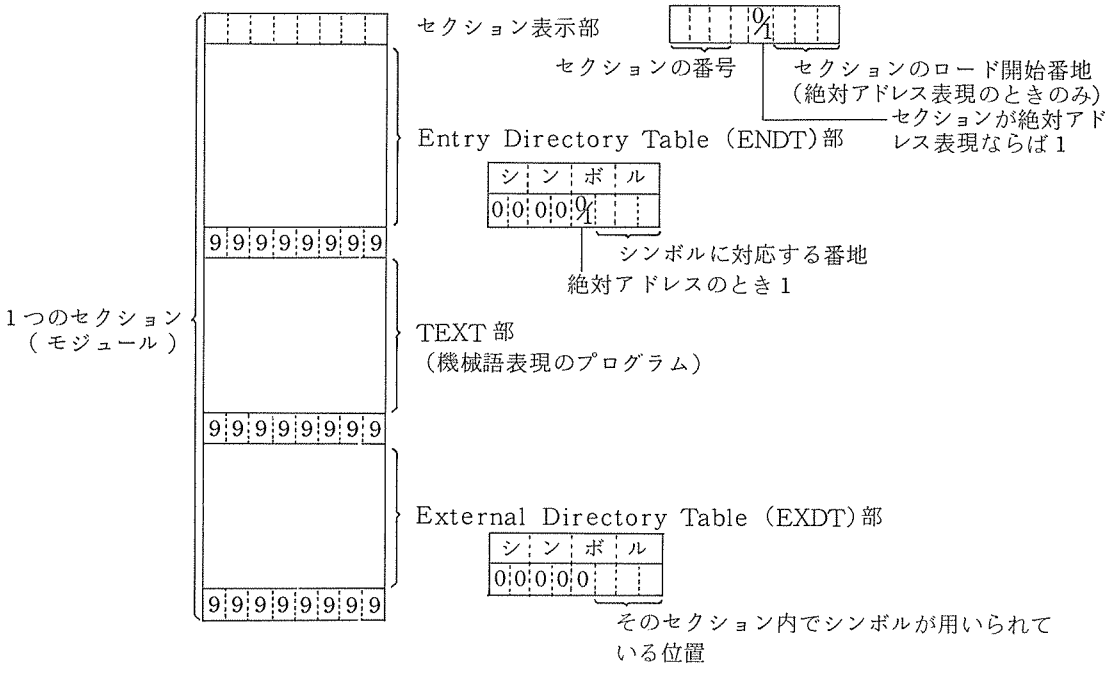


図3　目的モジュールの形式

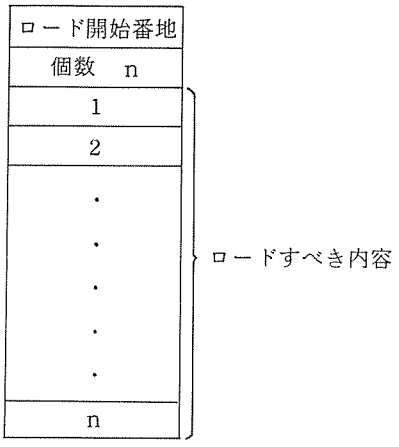


図4　ロード・モジュールの形式

YEN5）ある。ここで、外部記号とはセクション1においては記号YEN1およびYEN5であり、セクション2においては記号Xのことである。すなわち、そのセクション内において定義されていない記号のことである。

さて、このように2つの独立したプログラムとして表現されたものが1つに結合されて主記憶装置内の所定の番地にセットされてはじめて希望する仕事を計算機にやらせることが可能となるのだが、その過程がどのようなものか、そのためにどのような作業が必要になるのか等を示すために、一般に目的モジュールとかロード・モジュールとか呼ばれているものを印字表現することがこの擬似計算機を利用することにより可能となっている。そのためには、この目的モジュールとかロード・モジュールとかがどのような形式になっているかを決めておかなければならないので、ここでは、それぞれ図3および図4のように決めておいた。これによって、図2に示した2つの原始プログラムをアセンブルし、その結果得られた目的モジュールやロード・モジュール等が図5に示されている。そこで、以下において図5についてすこし説明をしておく。

図5(a)に示されている目的モジュールでは、SEQ．0および22に示した部分（□の部分）をみてみると、ここにはセクション番号、絶対番地表現であることおよびそのセクションの最初の番地が示されていることがわかる（図3参照）。また、SEQ．23，24によって、記号YEN1が絶対番地の201番地に、SEQ．25，26によって、記号YEN2が200番地に対応することが示されており、SEQ．42，43によって、このセクションにとっては外部記号であるXが最初の命令で用いられていること、同じくSEQ．44，45とSEQ．46と47によっても外部記号Xがこのセクションの7番目および9番目の命令で用いられていることがわかるようになっている。ただし、擬似計算機による文字の内部表現については付録3を参照のこと。

図5(b)においては各セクションにおいて外部記号として用いられている記号に対応する具体的な番地が示されており、(c)においてはそれらの外部記号が具体的にどの番地の命令で使用されているかを示しており、(d)においては各セクションが主記憶装置内のどの部分を占有するかが示されている。つぎに、ロード・モジュールが図5(e)に示されているが（図4参照）、これによって、最初のセクションは100番地から12個分であり、つぎのセクションは200番地から13個分であることがわかる。そして、図5(f)においては、この2つのセクションが主記憶装置上に実際にセットされた状態を示している。ただし、未使用の部分はここでは0で示されている。

```
% LINK LOAD=100,RLIST,MAP,ELIST

   OUTPUT LIST FROM ASSEMBLER  (RLIST)  目的モジュール
     SEQ.        0        1        2        3        4        5        6        7        8        9
       0  00101100 24000000 00000109 99999999 02600111 00100109 02100108 04900000 02100107 04900000
      10  03000101 00000001 00000005 00000000 00000000 00000110 99999999 25051465 00000004 25051461
(a)   20  00000006 99999999 00201200 25051461 00001201 25051465 00001200 99999999 00500000 02000211
      30  01400211 00800212 00200212 01300211 01200000 01300210 00800000 05000000 10000001 00000000
      40  00000000 99999999 24000000 00000001 24000000 00000007 24000000 00000009 99999999

   ENTRY SYMBOL TABLE  (MAP)
         SYMBOL  ADDRESS
(b)       X        109
          YEN1     201
          YEN5     200

   EXTERNAL SYMBOL TABLE  (MAP)
   SECTION NO.    SYMBOL  ADDRESS
             1      YEN5     103
             1      YEN1     105
             2      X        200
(c)          2      X        206
             2      X        208

   SECTION TABLE  (MAP)
   SECTION NO.    START      END
             1      100      111
(d)          2      200      212

   OUTPUT LIST FROM LINKER  (ELIST)  ロード・モジュール
     SEQ.        0        1        2        3        4        5        6        7        8        9
       0  00000100 00000012 02600111 00100109 02100108 04900200 02100107 04900201 03000101 00000001
(e)   10  00000005 00000000 00000000 00000110 00000200 00000013 00500109 02000211 01400211 00800212
      20  00200212 01300211 01200109 01300210 00800109 05000000 10000001 00000000 00000000

   PROGRAM LIST ON MEMORY (ELIST)                          ┌セクション1の部分
   ADDRESS      0        1        2        3        4        5        6        7        8        9
       100  02600111 00100109 02100108 04900200 02100107 04900201 03000101 00000001 00000005 00000000
       110  00000000 00000110 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
       120  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
       130  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
       140  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
(f)    150  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
       160  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
       170  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
       180  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
       190  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
       200  00500109 02000211 01400211 00800212 00200212 01300211 01200109 01300210 00800109 05000000
       210  10000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
                                                 └セクション2の部分
      *** LINK       NORMAL END ***
```

図5

つぎに、図2に示したプログラムを実際に擬似計算機で処理した結果を示すと図6のようになるが、1命令ずつ実行させた場合の種々のレジスタや主記憶装置の内容の変化を示したものが図7である。

これまでは図2に示した原始プログラムを利用して、絶対番地表現のもとで種々の説明を行ってきたが、原始プログラム（アセンブリ言語による表現）の作成に際しては絶対番地を指定せずに、一応0番地を基準とした相対番地表現が用いられることが多いこともあり、以下においては、図2に示したプログラムを絶対番地表現ではなく、相対番地表現で作成した場合について簡単に示しておく。

原始プログラムは図8に示すように、図2におけるＯＲＧ擬似命令（絶対番地表現を示すためのもの）がないだけであるが、機械語になった部分をみてわかるように、一応0番地から対応づけがされているが、実際にプログラムの各命令を主記憶装置内の所定の番地にセットしたときに、その

```
% RUN START=100
% DATA
          *** RUN START <START ADDRESS= 100 , LIMIT TIME=   10(S) > ***
     データ 12円……{ 2 … 5円の枚数
                   2 … 1円  〃
     データ  8円……{ 1
                   3

       ﾃﾞｰﾀ ﾉ ｵﾜﾘ ｦ ｹﾝｼｭﾂ ｼﾀ

        PC  MACINE CODE  ACCUMULATOR  INDEX REGISTER  STACK POINTER      ADDRESS            OP CODE
       101  00100109      00000000      00000001      110 (00000106)     109 (00000000)     GET


          *** RUN NORMAL END < HLT AT PC=  101 , RUN TIME=    1(S)    0(MS) >
% JEND
```

図 6

```
% RUN START=100,STEPDUMP=100-200-1
% DATA
          *** RUN START <START ADDRESS= 100 , LIMIT TIME=   10(S) > ***

     /////     DUMP   STEP BY STEP     /////   < PC   FROM  100  COUNT  200  BY    1 >

        PC  MACINE CODE  ACCUMULATOR  INDEX REGISTER  STACK POINTER      ADDRESS            OP CODE
       100  02600111      00000000      00000000      110 (00000000)     111 (00000110)     LDSP
       101  00100109      00000000      00000000      110 (00000000)     109 (00000012)     GET
       102  02100108      00000000      00000005      110 (00000000)     108 (00000005)     LDX
       103  04900200      00000000      00000005      109 (00000012)     200 (00500109)     JSR
       200  00500109      00000012      00000005      109 (00000012)     109 (00000012)     LDA
       201  02000211      00000012      00000005      109 (00000012)     211 (00000005)     STX
       202  01400211      00000002      00000005      109 (00000012)     211 (00000005)     DIV
       203  00800212      00000002      00000005      109 (00000012)     212 (00000002)     STA
                  2
       204  00200212      00000002      00000005      109 (00000012)     212 (00000002)     PUT
       205  01300211      00000010      00000005      109 (00000012)     211 (00000005)     MLT
       206  01200109      10000002      00000005      109 (00000012)     109 (00000012)     SUB
       207  01300210      00000002      00000005      109 (00000012)     210 (10000001)     MLT
       208  00800109      00000002      00000005      109 (00000002)     109 (00000002)     STA
       209  05000000      00000002      00000005      110 (00000104)       0 (00000000)     RTS
       104  02100107      00000002      00000001      110 (00000104)     107 (00000001)     LDX
       105  04900201      00000002      00000001      109 (00000002)     201 (02000211)     JSR
       201  02000211      00000002      00000001      109 (00000002)     211 (00000001)     STX
       202  01400211      00000002      00000001      109 (00000002)     211 (00000001)     DIV
       203  00800212      00000002      00000001      109 (00000002)     212 (00000002)     STA
                  2
       204  00200212      00000002      00000001      109 (00000002)     212 (00000002)     PUT
       205  01300211      00000002      00000001      109 (00000002)     211 (00000001)     MLT
       206  01200109      00000000      00000001      109 (00000002)     109 (00000002)     SUB
       207  01300210      00000000      00000001      109 (00000002)     210 (10000001)     MLT
       208  00800109      00000000      00000001      109 (00000000)     109 (00000000)     STA
       209  05000000      00000000      00000001      110 (00000106)       0 (00000000)     RTS
       106  03000101      00000000      00000001      110 (00000106)     101 (00100109)     BRN
       101  00100109      00000000      00000001      110 (00000106)     109 (00000008)     GET
       102  02100108      00000000      00000005      110 (00000106)     108 (00000005)     LDX
       103  04900200      00000000      00000005      109 (00000008)     200 (00500109)     JSR
       200  00500109      00000008      00000005      109 (00000008)     109 (00000008)     LDA
       201  02000211      00000008      00000005      109 (00000008)     211 (00000005)     STX
       202  01400211      00000001      00000005      109 (00000008)     211 (00000005)     DIV
       203  00800212      00000001      00000005      109 (00000008)     212 (00000001)     STA
                  1
       204  00200212      00000001      00000005      109 (00000008)     212 (00000001)     PUT
       205  01300211      00000005      00000005      109 (00000008)     211 (00000005)     MLT
       206  01200109      10000003      00000005      109 (00000008)     109 (00000008)     SUB
       207  01300210      00000003      00000005      109 (00000008)     210 (10000001)     MLT
       208  00800109      00000003      00000005      109 (00000003)     109 (00000003)     STA
       209  05000000      00000003      00000005      110 (00000104)       0 (00000000)     RTS
       104  02100107      00000003      00000001      110 (00000104)     107 (00000001)     LDX
       105  04900201      00000003      00000001      109 (00000003)     201 (02000211)     JSR
       201  02000211      00000003      00000001      109 (00000003)     211 (00000001)     STX
       202  01400211      00000003      00000001      109 (00000003)     211 (00000001)     DIV
       203  00800212      00000003      00000001      109 (00000003)     212 (00000003)     STA
                  3
       204  00200212      00000003      00000001      109 (00000003)     212 (00000003)     PUT
       205  01300211      00000003      00000001      109 (00000003)     211 (00000001)     MLT
       206  01200109      00000000      00000001      109 (00000003)     109 (00000003)     SUB
       207  01300210      00000000      00000001      109 (00000003)     210 (10000001)     MLT
       208  00800109      00000000      00000001      109 (00000000)     109 (00000000)     STA
       209  05000000      00000000      00000001      110 (00000106)       0 (00000000)     RTS
       106  03000101      00000000      00000001      110 (00000106)     101 (00100109)     BRN
       ﾃﾞｰﾀ ﾉ ｵﾜﾘ ｦ ｹﾝｼｭﾂ ｼﾀ

        PC  MACINE CODE  ACCUMULATOR  INDEX REGISTER  STACK POINTER      ADDRESS            OP CODE
       101  00100109      00000000      00000001      110 (00000106)     109 (00000000)     GET


          *** RUN NORMAL END < HLT AT PC=  101 , RUN TIME=    1(S)  500(MS) >
% JEND
```

図 7

```
  ARCHIMEDES MACRO ASSEMBLER V-02 L-01  DATE 82/09/09 TIME 13:33 PAGE 001

SECTION 001
  LOC  MACHINE  LINE     SOURCE
                   1          ENTRY  X
                   2          EXTRN  YEN1
                   3          EXTRN  YEN5
  000  92600011    4          LDSP   ASTC
  001  90100009    5     STRT:GET    X
  002  92100008    6          LDX    C5
  003  04900000    7          JSR    YEN5     セクション1
  004  92100007    8          LDX    C1
  005  04900000    9          JSR    YEN1
  006  93000001   10          BRN    STRT
  007  00000001   11     C1   :DEC    1
  008  00000005   12     C5   :DEC    5
  009  00000000   13     X    :WRKS   1
  010  00000000   14     STCK:WRKS   1
  011  90000010   15     ASTC:ADR    STCK
                  16          END

SYMBOL TABLE  (CROSS)
  X    009RG     YEN1 *** E     YEN5 *** E
  STRT 001R      C1   007R      C5   008R
  STCK 010R      ASTC 011R

  ERROR  MESSAGE
  NO.  LINE   MESSAGE


       0  WARNINGS
       0  ERRORS



  ARCHIMEDES MACRO ASSEMBLER V-02 L-01  DATE 82/09/09 TIME 13:34 PAGE 001

SECTION 002
  LOC  MACHINE  LINE     SOURCE
                   1          ENTRY  YEN1
                   2          ENTRY  YEN5
                   3          EXTRN  X
  000  00500000    4     YEN5:LDA    X
  001  92000011    5     YEN1:STX    N
  002  91400011    6          DIV    N
  003  90800012    7          STA    ANS
  004  90200012    8          PUT    ANS
  005  91300011    9          MLT    N        セクション2
  006  01200000   10          SUB    X
  007  91300010   11          MLT    MONE
  008  00800000   12          STA    X
  009  05000000   13          RTS
  010  10000001   14     MONE:DEC    -1
  011  00000000   15     N    :WRKS   1
  012  00000000   16     ANS  :WRKS   1
                  17          END

SYMBOL TABLE  (CROSS)
  YEN1 001RG     YEN5 000RG     X    *** E
  MONE 010R      N    011R      ANS  012R

  ERROR  MESSAGE
  NO.  LINE   MESSAGE


       0  WARNINGS
       0  ERRORS

      *** ASSEMBLE  NORMAL END ***
```

図8

命令の番地部分が影響を受けることになるものについては最上位桁が9となっている。そして、図8に示された2つのプログラムについて、セクション1を100番地から対応させ、セクション2はセクション1が終ったすぐつぎの番地から対応させることにして、目的モジュールやロード・モジュールおよびその他の対応表を示したものが図9であり、これは図5に対応するものである。また、図8のプログラムについて、セクション1を200番地から対応させることにして上と同じ操作をして得られたものが図10である。したがって、図5、図9、図10を比較してみることによって、絶

```
% LINK LOAD=100,RLIST,MAP,ELIST

   OUTPUT LIST FROM ASSEMBLER  (RLIST)   目的モジュール
     SEQ.        0        1        2        3        4        5        6        7        8        9
       0  00100000 24000000 00000009 99999999 92600011 90100009 92100008 04900000 92100007 04900000
      10  93000001 00000001 00000005 00000000 00000000 90000010 99999999 25051465 00000004 25051461
(a)   20  00000006 99999999 00200300 25051461 00000001 25051465 00000000 99999999 00500000 92000011
      30  91400011 90800012 90200012 91300011 01200000 91300010 00800000 05000000 10000001 00000000
      40  00000000 99999999 24000000 00000001 24000000 00000007 24000000 00000009 99999999

   ENTRY SYMBOL TABLE  (MAP)
         SYMBOL  ADDRESS
          X        109
(b)       YEN1     113
          YEN5     112

   EXTERNAL SYMBOL TABLE  (MAP)
   SECTION NO.   SYMBOL  ADDRESS
             1     YEN5     103
             1     YEN1     105
(c)          2     X        112
             2     X        118
             2     X        120

   SECTION TABLE  (MAP)
   SECTION NO.   START     END
             1     100     111
(d)          2     112     124

   OUTPUT LIST FROM LINKER  (ELIST)   ロード・モジュール
     SEQ.        0        1        2        3        4        5        6        7        8        9
       0  00000100 00000012 02600111 00100109 02100108 04900112 02100107 04900113 03000101 00000001
      10  00000005 00000000 00000000 00000110 00000112 00000013 00500109 02000123 01400123 00800124
(e)   20  00200124 01300123 01200109 01300122 00800109 05000000 10000001 00000000 00000000

   PROGRAM LIST ON MEMORY (ELIST)          セクション1の部分
   ADDRESS      0        1        2        3        4        5        6        7        8        9
       100  02600111 00100109 02100108 04900112 02100107 04900113 03000101 00000001 00000005 00000000
(f)    110  00000000 00000110 00500109 02000123 01400123 00800124 00200124 01300123 01200109 01300122
       120  00800109 05000000 10000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000
                                                          セクション2の部分
        *** LINK       NORMAL END ***
```

図9

```
% LINK LOAD=200,RLIST,MAP,ELIST

   OUTPUT LIST FROM ASSEMBLER  (RLIST)   目的モジュール
     SEQ.        0        1        2        3        4        5        6        7        8        9
       0  00100000 24000000 00000009 99999999 92600011 90100009 92100008 04900000 92100007 04900000
      10  93000001 00000001 00000005 00000000 00000000 90000010 99999999 25051465 00000004 25051461
(a)   20  00000006 99999999 00200000 25051461 00000001 25051465 00000000 99999999 00500000 92000011
      30  91400011 90800012 90200012 91300011 01200000 91300010 00800000 05000000 10000001 00000000
      40  00000000 99999999 24000000 00000001 24000000 00000007 24000000 00000009 99999999

   ENTRY SYMBOL TABLE  (MAP)
         SYMBOL  ADDRESS
          X        209
(b)       YEN1     213
          YEN5     212

   EXTERNAL SYMBOL TABLE  (MAP)
   SECTION NO.   SYMBOL  ADDRESS
             1     YEN5     203
             1     YEN1     205
             2     X        212
(c)          2     X        218
             2     X        220

   SECTION TABLE  (MAP)
   SECTION NO.   START     END
             1     200     211
(d)          2     212     224

   OUTPUT LIST FROM LINKER  (ELIST)   ロード・モジュール
     SEQ.        0        1        2        3        4        5        6        7        8        9
       0  00000200 00000012 02600211 00100209 02100208 04900212 02100207 04900213 03000201 00000001
      10  00000005 00000000 00000000 00000210 00000212 00000013 00500209 02000223 01400223 00800224
(e)   20  00200224 01300223 01200209 01300222 00800209 05000000 10000001 00000000 00000000

   PROGRAM LIST ON MEMORY (ELIST)          セクション1の部分
   ADDRESS      0        1        2        3        4        5        6        7        8        9
       200  02600211 00100209 02100208 04900212 02100207 04900213 03000201 00000001 00000005 00000000
(f)    210  00000000 00000210 00500209 02000223 01400223 00800224 00200224 01300223 01200209 01300222
       220  00800209 05000000 10000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000
                                                          セクション2の部分
        *** LINK       NORMAL END ***
```

図10

対番地表現によるプログラムと相対番地表現によるプログラムの差やそれぞれについての目的モジールやロード・モジュールの具体的な相異点などについて、これらに関する基本的な概念を理解することが容易になると筆者は考えている。

以上、機械語教育用擬似計算機を使用した場合の具体的な一例について述べることにより、現在どの程度のことが教育可能であるかについて簡単に示したが、ここに示したような教育目標がはたして必要かどうかということになると意見のわかれるのは当然であろうと思われる。冒頭に示したように「珍奇なこと」を述べたにすぎないといえるかも知れないが何かの参考にでもしていただければ筆者の望外のよろこびである。なお、ここで示した擬似計算機を用いて初心者用として書かれた機械語によるプログラミングの入門書[(12)]があるので興味のある方は参照していただきたい。また、本文中では擬似計算機の具体的な機能や命令体系等については示さなかったので、付録としてその概略を示しておくことにする。

最後に、この擬似計算機に関して大変お世話になった大阪大学豊中地区データ・ステーション助手の工藤英男氏に深謝致します。そして、大阪大学基礎工学部情報工学科よりFACOM230-45Sがリプレースのため姿を消すことになったということを記しペンをおくことに致します。

〔参 考 文 献〕

1）元岡　達　：第５世代コンピュータの構想，情報処理，Vol.23，No.5（1982－5）

2）中北，的場　：教育用アセンブラ言語の解釈プログラムについて，昭和44年電気四学会連合大会　3293（1969）

3）的場，中北　：シンボリック言語によるプログラム教育用システム，電子通信学会論文誌，Vol.53－C，No.1（1970－1）

4）的場　進　：大型計算機センターの利用法について，神戸商船大学紀要，第二類第17号（1970－3）

5）的場，家長：会話形式によるアセンブラ言語教育用システム，電子通信学会論文誌，Vol.54－C，No.6（1971－6）

6）下条，的場，家長　：ＴＳＳの端末を利用するシンボリック・プログラム教育用システム，神戸商船大学紀要，第二類第19号　（1972－3）

7）佐藤，吉岡，的場　：多端末から利用可能な教育用仮想計算機システム，昭和49年度電子通信学会全国大会　1671（1974）

8）的場，吉岡，佐藤　：プログラミング教育用擬似計算機システムについて，情報処理，Vol.17，No.2（1976－2）

9）的場　裕司　：マークカードを利用した教育用擬似計算機について，第２回教育工学研究協議会研究発表資料　（1976－10）

10）林，的場，吉岡　：マークカードを用いた教育用擬似計算機システムについて，昭和52年度情報処理学会第18回全国大会　314（1977）

11）的場，工藤，松浦，吉岡　：連結編集過程を考慮したソフトウェア教育用ツール（ＴＳＥ），情報処理学会論文誌，Vol.22，No.5（1981－9）

12）的場　裕司　：わかりやすいプログラミング入門，日刊工業新聞社　（1982－2）

付録１　機械語命令一覧表（ 42種類 ）

命令 コード	命令 記号	機能	命令 コード	命令 記号	機能
00	HLT	ストップ	21	LDX	(m) → XR
01	GET	データをmにセット	22	INCX	(XR) + 1 → XR
02	PUT	(m) を出力	23	ADDX	(XR) + (m) → XR
03	FRUT	(m) を浮動小数点型で出力	24	SUBX	(XR) − (m) → XR
04	CPUT	Accの値をnとすると、mからm+n	26	LDSP	(m) → SP
		/4番地の内容を文字型で出力	27	STSP	(SP) → m
05	LDA	(m) → Acc	28	PUSH	(Acc) → (SP), (SP) − 1 → SP
06	LDLA	(m[L4]) → Acc[L4]	29	POP	(SP) + 1 → SP, (SP) → Acc
07	LDRA	(m[R4]) → Acc[R4]	30	BRN	mへブランチ (m → PC)
08	STA	(Acc) → m	31	BMI	(Acc) ＜ 0ならmへブランチ
09	STLA	(Acc[L4]) → m[L4]	32	BPL	(Acc) ≧ 0ならmへブランチ
10	STRA	(Acc[R4]) → m[R4]	33	BEQ	(Acc) ＝ 0ならmへブランチ
11	ADD	(Acc) + (m) → Acc	34	BNE	(Acc) ≒ 0ならmへブランチ
12	SUB	(Acc) − (m) → Acc	37	BZX	(XR) ＝ 0ならmへブランチ
13	MLT	(Acc) × (m) → Acc	38	BNX	(XR) ≒ 0ならmへブランチ
14	DIV	(Acc) ÷ (m) → Acc	40	SHRL	(Acc) を右シフト (含符号)
15	FADD	(Acc) + (m) → Acc	41	SHLL	(Acc) を左シフト (含符号)
16	FSUB	(Acc) − (m) → Acc	42	SHR	(Acc) を右シフト (除符号)
17	FMLT	(Acc) × (m) → Acc	43	SHL	(Acc) を左シフト (除符号)
18	FDIV	(Acc) ÷ (m) → Acc	49	JSR	(PC) → (SP), (SP) − 1 → SP,
19	CMP	(Acc) ＝ (m) なら0 → Acc			mへブランチ
		(Acc) ≒ (m) なら1 → Acc	50	RTS	(SP) + 1 → SP, (SP) へブランチ
20	STX	(XR) → m			

(X)　：Xの内容を意味する
m　：実際の番地を意味する
Acc　：演算用レジスタ
XR　：インデックス・レジスタ
SP　：スタック・ポインタ
PC　：プログラム・カウンタ
[L4]：左側4けたの部分
[R4]：右側4けたの部分

付録２　擬似命令一覧表（14種類）

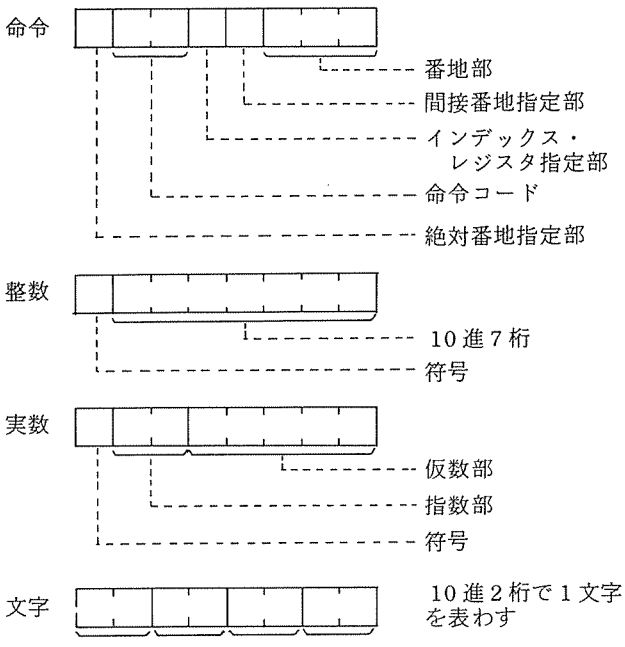
記　号	機　能	記　号	機　能
ORG	以下の命令をセットする番地を指定する	WRKS	作業用番地を指定した個数だけ確保する
LIST	マクロ展開したプログラムのリストを出す	ADR	アドレス定数を定義する
		MACRO	マクロ定義の開始
NLIST	マクロ展開しない	IFT	条件付きマクロ展開（真）
ENTRY	ENTRYシンボルの定義	IFF	条件付きマクロ展開（偽）
EXTRN	EXTERNALシンボルの定義	MEND	マクロ定義の終了
DEC	10進整数をセットする	END	プログラムの終りを示す
CHAR	文字定数をセットする		

付録３　擬似計算機用文字コード一覧表（50種類）

	上位けた 0	1	2	3	6
0		J	T	(	0
1	A	K	U	)	1
2	B	L	V	,	2
下 3	C	M	W	.	3
位 4	D	N	X	▼	4
け 5	E	O	Y	;	5
た 6	F	P	Z	:	6
7	G	Q	+	&	7
8	H	R	−	@	8
9	I	S	*	=	9

付録４　内部割込み一覧表

割込みの種類	番　地
データ・エラー	0
Accのオーバー・フロー	1
Accのアンダー・フロー	2
0による除算	3
不正命令	4
PCのオーバー・フロー	5



付録5　語の構成

```
¥  JOB    使用者名，ACCT=▼会計コード▼
¥  TALE   *
%  JOB    学生番号
%  ASSEMBLER   { [LIST,CROSS]
                 NOLIST,NOCROSS
```

↑
原始プログラム・カード
↓

```
%  LINK    { [LOAD=0,NORLIST,NOMAR,NOELIST]
             LOAD=番地,RLIST,MAP,ELIST
%  RUN     { [START=0]
             START=実行開始番地，
             DUMP=ダンプ開始番地－終了番地，
             STEPDUMP=開始番地－終了番地－間隔
%  DATA
```

↑
データ・カード
↓

```
%  JEND
¥  JEND
```

（注）[　]の中のパラメータは無指定のとき採用される

付録6　カード・デックの構成