STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM
REKENAFDELING

MR 84

The ELAN source text of the MC ALGOL 60 system for the EL × 8.

F.E.J. Kruseman Aretz

and

B.J. Mailloux.

november 1966

The Mathematical Centre at Amsterdam, founded the 11th of February, 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications, and is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.) and the Central Organization for Applied Scientific Research in the Netherlands (T.N.O.), by the Municipality of Amsterdam and by several industries.

Table of contents

Introduction

The ELAN program, presented in this report, constitutes a complete ALGOL 60 system. By complete is meant that this program contains everything that is nescessary for the running of ALGOL programs on an Electrologica X8 installation, without any reference to other (standard) software. It even contains a section that, started after assembly, punches out a system tape in binary code, preceded by a binary loading program in the so-called IP code.

The MC ALGOL 60 system for the EL X8 has been in development for a couple of years, and it had to fulfill four (almost contradictory) requirements:

1) it should be adaptable to the smallest possible X8 configuration (i.e. 16 K memory, I/O by one paper tape reader/punch),
2) it should have high efficiency in extended calculations,
3) it should check against program errors wherever possible,
4) it should have a modular structure, so that it could be easily adapted to future requirements.

The system given here, however, exceeds the first requirement in one respect: it presumes the presence of a line printer with a special character set (the so-called ALGOL set). It is, however, easily adapted to environments without line printer, as will be indicated later on.

The system can be subdivided into three main divisions:

1) the code that must be in memory all the time, called the PICO monitor;
2) the instructions that are used only in the execution phase of an ALGOL program, called the running system;
3) the instructions that are in use only during the compilation phase, called the compiler.

The system can be in a state, in which the "program stack" in the execution phase is allowed to overwrite the compiler. From this, it follows that the position of the compiler in memory immediately follows the part of the memory that is available to the object program and its stack. Some further assumptions about the location of the constituents of the system are:
all instructions must be in directly addressable core;
some parts of the PICO monitor must be below 16 K.

The position of the several parts of the system, as well as the apparatus numbers for the I/O equipment, are specified on page 1 of the system by means of location directions. By changing these appropriately, one can adapt the program to any X8 installation. The apparatus numbers should correspond to:

RE: the paper tape reader used for ALGOL programs and data tapes,
PU: the paper tape punch used for output,
PR: the line printer,
KY: the keyboard of the teleprinter,
TY: the printer section of the teleprinter,
REST: the IP tape reader,
PUST: the paper tape punch of the machine on which the assembly of the system is carried out.

For an installation with 16 K memory the line:

M[27648]: COMP VAR: END OF MEMORY:

should be changed to:

M[11264]: COMP VAR: END OF MEMORY:

The system tape punch program punches, at the end of assembly:
a binary loader;
the contents of M[24] to M[29] (the interrupt instructions);
the contents of M[64] to M[64 + 4x16] (the apparatus registers);
BEGIN OF STACK up to and including END OF STACK;
BEGIN OF PERMANENT up to and including END OF PERMANENT;
BEGIN OF COMPILER up to and including END OF COMPILER;
the punch program itself;
an end marker;
a piece of blank tape.
Next, a second copy of BEGIN OF COMPILER up to and including END OF COMPILER is punched, which tape is to be loaded whenever, during the execution phase of a program, the compiler part has been overwritten.
This punching program makes it desirable that the running system be (almost) contiguous to the PICO monitor.

The ELAN text has been written in a dialect of ELAN. It differs from what the authors know about the official ELAN language in the following respects:

1) as was the official rule at the time that large parts of the system were coded, almost all instructions of the form: " G = :LABEL" have been written as: "F = :LABEL";
2) it uses the function "SUBCD", which for a long while was officially correct ELAN;
3) the set "SUB", "REP", and "REPx", with x = P, Z, or E is extended with "SUBi", "REPj", and "REPjx", with x = P, Z, or E, i = 0 (1) 15, j = 0 (1) 7, such that for i = 0 and j = 7 the old set is obtained.

No use has been made of ELAN facilities like: replica instructions; macros; index expressions attached to identifiers, to be positioned later on; automatic dynamic addressing inside block; renaming of standard identifiers, etc.
All dynamic addresses have been denoted explicitly.

In case of absence of a line printer, the line printer section on pages 33 to 45 may be replaced by the text, given on pages 191 to 194. In that case all information destined for the line printer will be punched. For this purpose, in the assembly parameters on page 1 the line:

M['6530']: SYSTEM:

can be replaced by:

M['4730']: SYSTEM:

On pages 195 to 212 a list of labels is given for all labels of the ELAN text except for the labels local to some short blocks. For each label in this list the page number(s) is given on which it is defined.

The coding of the PICO monitor was done by B.J.Mailloux, assisted by J.A.Th.M. van Berckel, the design of the running system is by J.J.B.M.Nederkoorn and F.E.J.Kruseman Aretz, the standard functions and the operations $\uparrow$ and $\div$ were written and tested mainly by F.J.M.Barning, and the compiler was written by F.E.J.Kruseman Aretz.

The authors of this document are aware of the fact that this report is just a momentary snapshot of a system which is in a state of continuous change and development. It is even their hope and wish that this documentation will contribute to further research and growth.

```
" ASSEMBLY PARAMETERS
M[3]:            RE:     REST:
M[2]:            PU:     PUST:
M[0]:            TY:
M[1]:            KY:
M[8]:            PR:

M[32]:           IBULEN:
M[150]:          PBULEN:
M[140]:          INSTCK:
M[256]:          STK BEG:
M[511]:          STK END:
M['6530']:       SYSTEM:
SYSTEM[64]:      RUN VAR:
RUN VAR[14]:     RUN SYST:
RUN SYST[627]:   LIBRARY:
LIBRARY[742]:    END OF RUN SYST: END OF PERMANENT:
M[27648]:        COMP VAR: END OF MEMORY:
COMP VAR[60]:    BEGIN OF COMPILER: MEMORY END:
BEGIN OF COMPILER[4851]: INSTR LST:
INSTR LST[204]: END OF COMPILER:
M[:END OF COMPILER - :BEGIN OF COMPILER]: LENGTH OF COMPILER:

" INTERFACE VARIABLES AND CONSTANTS
M[512]:          'BEGIN'

FPERMIT:                  'SKIP' 1
PERMIT:                   'SKIP' 1
ERRONEOUS:                'SKIP' 1
DANGEROUS:                'SKIP' 1
RUNNUMBER:                'SKIP' 1
VALUE OF CONSTANT:        'SKIP' 2
LINECOUNTER:              'SKIP' 1
LASTSYMBOL:               'SKIP' 1
LASTIDENTIFIER:           'SKIP' 2
PRESENCE:                 'SKIP' 1
FWANTED:                  'SKIP' 1
WANTED:                   'SKIP' 1
TEXT IN ARRAY:            'SKIP' 1
BEGIN OF STACK:           :STK BEG
BEGIN OF PR AR:           :END OF RUN SYST
START OF TEXT ARRAY:      ((:END OF MEMORY + (2 × :END OF RUN SYST)) / 3)
END OF STACK:             :STK END
PICO:
```

```
                'BEGIN' KAR, SERIAL, DATE, SYM, IBASE2, PBASE2, TBUF,
                        IBUNOE, IBUNOF, IPOS, IBUSY, IHOK, PLENGTH, PBUNOE,
                        PBUNOF, PPOS, PHOK, PSW, ASAFE, SSAFE, FSAFE, SAFEB,
                        BBSAFE, KCASE, INSW, MAYI, XEENW, CASE, CCNT, FLCASE,
                        TEMP, CNT, BEXP, BSAFE, PT, ESIGN, STOCK, HEAD, TAIL,
                        DEXP, MSIGN, MM, HEP, NN, XX, ABS, FIXFLO, IWIGGLE,
                        IWAGGLE, PWIGGLE, PWAGGLE, DYST, D18, D18M1, ILENGTH,
                        KBUF, NTAB, HALF, SGNBIT, SIXMM1, TENTH, SIXMIL,
                        DATMES, SERMES, SMES, SMES1, TRNMES, RUNMES, REMES,
                        TAPMES, AGNMES, XMES, HMES, CRNLMS, CMPMES, PASTE,
                        INTREP, TELXTB, FLEXTB, TENPOW, SCALE, GIANT, NAIGA,
                        PEMPTY, SIGNOFF, RATTLE, THEAD, FIX60, FIXT60, SIGNON,
                        ISKIP, EMPTYI, BRUSH, FLBLNK, INTAB, HOK, START,
                        REEX, ICHK, READER, RESTORE, ISTART, PSTART, PSTART1,
                        PSTART4, PCHER, COM, POK, CTYPE, TYPE, TYPE3, TACT,
                        TWAIT, TOFF, KEYBD, XEENS, XGET, NXTKYBDSL, NKS, DNAH,
                        HANDRD, KYBD, AFXP, NSTART, P, U, V, L, KEYM,
                        READCOMP, NEXTWD, NEXTHP, INOCT, KTAB, CW1, CW2, CW4

                " ASSUMED TO BE DECLARED GLOBALLY ARE: TAR, IAR, PAR, ISS, PSS,
                "       TSS, IBASE1, PBASE1, CBASE, BEGIN OF PERMANENT, DENTST,
                "       ENDRUN, IOINTP, REHEP, PUHEP, STOP, XEEN, HAND, DERRORM,
                "       ERRORM, FINIS, TEST TEXT IN ARRAY, FLEXIS, NXT TAPE SL,
                "       FLEXHP, READ, READ1, DECBIN, FIX, FIX1, FLO, FLO1, GEN,
                "       PUNCH, FIXP, ABSFIXP, FLOP, PO, INITPR1, INITPR2,
                "       INITPR3, PRHEX, NEWPAGE, NLCR, CARRIAGE, PRINT, FIXT,
                "       ABSFIXT, FLOT, PRO, PRNTIS, TAB, SPACE, AFXT6, AFXT

                " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
                "       TY, KY, RE, PU, PICO, IBULEN, PBULEN, BEGIN OF STACK,
                "       FPERMIT, PERMIT, FWANTED, WANTED, ERRONEOUS, DANGEROUS,
                "       PRESCANO, RUNNUMBER, VALUE OF CONSTANT, RUN, INSTCK, RND,
                "       LINECOUNTER, LASTSYMBOL, LAST IDENTIFIER, PRESENCE,
                "       BEGIN OF COMPILER, END OF COMPILER, TEXT IN ARRAY, PR

        " THE FOLLOWING TWO SYMBOLICALLY SHOWN INSTRUCTIONS ARE FILLED IN
        " AFTER ASSEMBLY BY THE ROUTINE BEGINNING AT SYSTAP.
"M[24]:         SUB15(:IOINTP);         "M[28]:         GOTO(:FINIS)
M[64+(4x:TY)]:  TAR:
M[64+(4x:KY)]:  KAR:
M[64+(4x:RE)]:  IAR:
M[64+(4x:PU)]:  PAR:
        " THE FOLLOWING THREE WORDS ARE ALSO FILLED IN AFTER ASSEMBLY.
"TAR[0]:        :TSS[1];        "IAR[0]:        :ISS[1];        "PAR[0]:        :PSS[1]

PICO[0]:
ISS:            'SKIP'1
SERIAL:         'SKIP'2
PSS:            'SKIP'1
DATE:           'SKIP'2
TSS:            'SKIP'1
SYM:            'SKIP'1
IBASE1:         'SKIP' (:IBULEN+1)
IBASE2:         'SKIP' (:IBULEN+1)
PBASE1:         'SKIP' (:PBULEN)
PBASE2:         'SKIP' (:PBULEN+1)
TBUF:           'SKIP'20
```

```
IBUNOE:          'SKIP'1
IBUNOF:          'SKIP'1
IPOS:            'SKIP'1
IBUSY:           'SKIP'1
IHOK:            'SKIP'1
PLENGTH:         'SKIP'1
PBUNOE:          'SKIP'1
PBUNOF:          'SKIP'1
PPOS:            'SKIP'1
PHOK:            'SKIP'1
PSW:             'SKIP'1
ASAFE:           'SKIP'1
SSAFE:           'SKIP'1
FSAFE:           'SKIP'2
SAFEB:           'SKIP'1
BBSAFE:          'SKIP'1
KCASE:           'SKIP'1
INSW:            'SKIP'1
MAYI:            'SKIP'1
XEENW:           'SKIP'1

CBASE:           'SKIP'24
CASE:            'SKIP'1
CCNT:            'SKIP'1
FLCASE:          'SKIP'1
TEMP:CNT:BEXP:   'SKIP'1
BSAFE:PT:ESIGN:  'SKIP'1
STOCK:           'SKIP'1
HEAD:            'SKIP'1
TAIL:            'SKIP'1
DEXP:            'SKIP'1
MSIGN:MM:        'SKIP'1
HEP:NN:          'SKIP'1
XX:              'SKIP'2
ABS:             'SKIP'1
FIXFLO:          'SKIP'1

BEGIN OF PERMANENT:
IWIGGLE:         'SKIP' 1                "PRESENT1
                 :IWAGGLE
                 :IBASE1
IWAGGLE:         'SKIP' 1                "PRESENT2
                 :IWIGGLE
                 :IBASE2
PWIGGLE:         'SKIP' 1                "ROOM1
                 :PWAGGLE
                 :PBASE1
                 :PBASE1[:PBULEN]        "PEND1
PWAGGLE:         'SKIP' 1                "ROOM2
                 :PWIGGLE
                 :PBASE2
                 :PBASE2[:PBULEN]        "PEND2
DYST:            SUBC(D18M1)
```

```
D18:            '001 000 000'
D18M1:          '000 777 777'
ILENGTH:        ('001 000 000'X:IBULEN)
KBUF:           +2                      "CR
                +8                      "NL
                'SKIP'2
NTAB:           +9
                +99
HALF:           '200 000 000'
SGNBIT:         '400 000 000'
SIXMM1:         +6710885
TENTH:          '031 463 146'
                '146 314 632'
SIXMIL:         +6710887
                + 671089
                +  67109

DATMES:         '02 10 44 22 0'         "CR NL BLKLT D
                '30 01 20 77 0'         "A T E END
SERMES:         '02 10 44 24 0'         "CR NL BLKLT S
                '20 12 14 30 0'         "E R I A
                '11 77 00 00 0'         "L END
SMES:           '02 10 40 77 0'         "CR NL BLKDG END
SMES1:          '27 77 00 00 0'         "/ END
TRNMES:         '02 10 44 01 0'         "CR NL BLKLT T
                '12 30 06 24 0'         "R A N S
                '11 30 01 14 0'         "L A T I
                '03 06 77 00 0'         "O N END
RUNMES:         '02 10 44 20 0'         "CR NL BLKLT E
                '27 20 16 34 0'         "X E C U
                '01 14 03 06 0'         "T I O N
                '77 00 00 00 0'         "END
REMES:          '02 10 44 14 0'         "CR NL BLKLT I
                '06 15 34 01 0'         "N P U T
                '04 01 30 15 0'         "SP T A P
                '20 77 00 00 0'         "E END
TAPMES:         '02 10 44 15 0'         "CR NL BLKLT P
                '34 06 16 05 0'         "U N C H
                '04 20 07 15 0'         "SP E M P
                '01 25 77 00 0'         "T Y END
AGNMES:         '02 10 44 30 0'         "CR NL BLKLT A
                '13 30 14 06 0'         "G A I N
                '77 00 00 00 0'         "END
XMES:           '02 10 44 27 0'         "CR NL BLKLT X
                '20 20 06 77 0'         "E E N END
HMES:           '02 10 44 05 0'         "NL CR BLKLT H
                '30 06 22 40 0'         "A N D BLKDG
                '77 00 00 00 0'         "END
CRNLMS:         '02 10 77 00 0'         "CR NL END
CMPMES:         '02 10 44 16 0'         "CR NL BLKLT C
                '03 07 15 14 0'         "O M P I
                '11 20 12 77 0'         "L E R END

PASTE:          ('000 004 000'+:RE)     "TAPE READER
                ('000 004 000'+:PU)     "TAPE PUNCH
                ('000 004 000'+:TY)     "TYPEWRITER
                ('004 004 000'+:KY)     "KEYBOARD
                '777 777 777'           "END MARKER
```

INTREP:

```
-'137 400 002'; +'001 247 401'; +'002 241 002'; -'154 400 000';
+'004 243 004'; -'152 400 000'; -'151 400 000'; +'007 262 407';
+'010 261 010'; -'146 400 000'; -'036 400 000'; -'035 400 002';
-'014 400 000'; -'033 400 001'; -'012 400 005'; -'011 400 000';

+'147 275 573'; -'027 400 000'; -'006 400 000'; +'121 241 403';
-'055 400 000'; +'103 255 405'; +'124 262 006'; -'072 400 000';
-'071 400 000'; +'127 276 411'; -'047 400 007'; -'066 400 000';
-'115 400 001'; -'134 400 000'; -'113 400 000'; -'132 400 001';

+'046 250 000'; -'110 400 000'; -'107 400 000'; +'051 234 035';
-'000 000 000'; +'141 435 037'; +'142 435 440'; -'014 200 000';
-'033 200 000'; +'165 437 043'; -'011 200 002'; -'030 200 000';
-'027 200 001'; -'006 200 000'; -'056 200 000'; -'055 200 001';

-'074 200 000'; +'124 445 110'; +'105 433 434'; -'071 200 000';
+'127 434 436'; -'047 200 000'; -'066 200 000'; +'062 436 041';
+'043 436 442'; -'113 200 000'; -'132 200 000'; +'046 474 131';
-'110 200 000'; -'107 200 001'; -'126 200 006'; -'000 000 000';

+'160 246 101'; -'077 400 000'; -'175 200 000'; +'023 430 025';
-'000 000 000'; +'000 031 027'; +'004 431 430'; -'000 000 000';
-'116 400 000'; +'000 033 033'; -'116 200 001'; -'000 000 000';
-'077 200 001'; -'000 000 000'; -'000 000 000'; -'176 200 001';

-'137 200 000'; +'000 027 023'; +'000 027 424'; -'000 000 000';
+'000 030 426'; -'000 000 000'; -'000 000 000'; +'133 232 031';
+'153 232 432'; -'104 400 000'; -'024 200 000'; +'025 475 127';
-'000 000 000'; -'157 600 001'; -'000 000 001'; -'000 000 000';

-'000 000 000'; +'000 022 412'; +'010 423 013'; -'146 200 000';
+'026 424 015'; -'170 200 000'; -'000 000 000'; +'000 025 420';
+'000 026 021'; -'000 000 000'; -'000 000 000'; +'000 076 130';
-'000 000 000'; -'000 000 001'; -'000 000 001'; -'000 000 000';

+'000 074 500'; -'000 000 000'; -'000 000 000'; +'000 023 414';
-'000 000 000'; +'000 024 416'; +'076 025 017'; -'145 600 000';
-'104 200 000'; +'160 426 422'; -'044 200 004'; -'000 000 000';
-'000 000 003'; -'000 000 000'; -'161 400 000'; -'161 200 002';
```

```
TELXTB:     +13                 " 0
            +29                 " 1
            +25                 " 2
            +16                 " 3
            +10                 " 4
            + 1                 " 5
            +21                 " 6
            +28                 " 7
            +12                 " 8
            + 3                 " 9
            +17                 " +
            +24                 " -
            + 7                 " .
            + 5                 " ,
            + 4                 " SPACE

FLEXTB:     + 32                " 0
            +  1                " 1
            +  2                " 2
            + 19                " 3
            +  4                " 4
            + 21                " 5
            + 22                " 6
            +  7                " 7
            +  8                " 8
            + 25                " 9
            +112                " +
            + 64                " -
            +107                " .
            + 59                " ,
            + 16                " SPACE

TENPOW:     +4194 3040          " 10^1 x 2^(26- 4)
            +5242 8800          " 10^2 x 2^(26- 7)
            +6553 6000          " 10^3 x 2^(26-10)
            +4096 0000          " 10^4 x 2^(26-14)
            +5120 0000          " 10^5 x 2^(26-17)
            +6400 0000          " 10^6 x 2^(26-20)
            +4000 0000          " 10^7 x 2^(26-24)
            +5000 0000          " 10^8 x 2^(26-27)
            +6250 0000          " 10^9 x 2^(26-30)
            +3906 2500          " 10^10x 2^(26-34)

SCALE:      + 4
            + 7
            +10
            +14
            +17
            +20
            +24
            +27
            +30
            +34

GIANT:      '377737777'
            '377777777'
```

```
DENTST:     B=BEGIN OF STACK          " INITIALIZE STACK POINTER
            '660 071 046'             " CLEAR INTERRUPT FROM NB
            A=:PASTE[0]               " ADDRESS OF FIRST RESET INST
            G=MA,P                    " RESET INSTR
          Y,SUBC(:BRUSH)              " RESET ONE APPARATUS
          Y,A+1                       " STEP UP FOR NEXT APPAR
          Y,JUMP(-4)                  " RETURN FOR NEXT APPAR
            PHOK=-B                   " INIT PUNCH END OF TAPE
            FPERMIT=B                 " NO OVERWRITING COMPILER
            FWANTED=B                 " COMPILE LINE NUMBERS
            ('760 070 000'+:KY)       " ACCEPT KEYBOARD SYMBOLS
            '760 060 000'             " SET INTERRUPT PERMIT

            F=:TSS[1]
            TAR[0]=G                  " OK INDICATION
            F=0
            TAR[1]=G                  " INTERRUPT COUNTER = 0
            KCASE=-G                  " FIGURE SHIFT
            PWIGGLE=-G
            PWAGGLE=-G                " PUNCH BUFFERS HAVE ROOM
            PSW=-B                    " NO PUNCHING DONE YET
            KAR[0]=-G                 " NO SYMBOL IN KEYBOARD YET
            KAR[1]=B,Z                " CONDITION=NO
            SUBC(:INITPR1)            " INITIALIZE LINE PRINTER
            SUBCD(:CTYPE)             " TYPE "DATE"
              :DATMES
            SUBCD(:DNAH)              " GET DATE FROM KEYBOARD
            DATE=G
            SUBCD(:CTYPE)             " TYPE "SERIAL"
              :SERMES
            SUBCD(:DNAH)              " GET SERIAL NUMBER
            SERIAL=G
            SUBC(:EMPTYI)             " INITIALIZE INPUT

NAIGA:      S=32767
            INSW=S                    " PREPARE KEYBD FOR AUTOSTARTS
            ERRONEOUS=S               " NO MISTAKES DETECTED YET
            DANGEROUS=S               " ALSO, NO DANGEROUS MISTAKES
            MAYI=S                    " XEEN WORD IS UNDEFINED
            STOCK=S                   " INITIALIZE READ
            CCNT=S                    " MORE THAN 150 CHARACTERS ON LINE
            FLCASE=S                  " FLEXHP CASE UNDEFINED
          N,S=FPERMIT
          N,PERMIT=S                  " PROTECT COMPILER

            B=BEGIN OF STACK          " INITIALIZE STACK POINTER
            SUBCD(:CTYPE)             " TYPE NLCR BLKDIG
              :SMES
            G=DATE
            SUBC(:FIXT60)             " TYPE DATE
            SUBCD(:CTYPE)             " TYPE "-"
              :SMES1
            G=SERIAL
          Y,F-1
            SERIAL=G                  " DECREASE SERIAL NUMBER BY ONE
            SUBC(:FIXT60)             " TYPE SERIAL NUMBER
          N,SUBCD(:CTYPE)             " TYPE "TRANSLATION"
              :TRNMES
```

```
PEMPTY:     N,SUBC(:READ COMP)        " ENSURE PRESENCE OF COMPILER
              SUBC(:INITPR2)          " SIGN LINE PRINTER ON
            N,SUBC(:REHEP)            " READ AND DISCARD FIRST HEPTAD
              SUBC(:SIGNON)           " SIGN PUNCH ON
            N,S=FWANTED               " )  INDICATE WHETHER LINE NUMBERS
            N,WANTED=S                " )  ARE TO BE COMPILED OR NOT
              A=1
              SERIAL+A                " INCREMENT SERIAL NUMBER BY ONE
            N,SUBC(:PRESCANO)         " COMPILE THE PROGRAM, IF YOU CAN
              S=ERRONEOUS,Z           " WERE THERE MISTAKES?
            Y,GOTO(:ENDRUN)           " THEN DELETE THE EXECUTION
              SUBCD(:CTYPE)           " TYPE "EXECUTION"
                :RUNMES
              S=500
              RUNNUMBER=S
              F=0
              CASE=G                  " LOWER CASE
              VALUE OF CONSTANT=-F
              SUBC(:INITPR3)          " START NEW PAGE, IF NECESSARY
              GOTO(:RUN)              " START THE OBJECT PROGRAM

"  THE OBJECT PROGRAM RETURNS CONTROL TO ENDRUN AT THE END OF EXECUTION
ENDRUN:       S=-1,P                  " CONDITION = NO FOR NORMAL END
              SUBC(:SIGNOFF)          " SIGN PUNCH OFF
              SUBC(:ISKIP)            " SKIP TO END OF INPUT TAPE
              GOTO(:NAIGA)            " NEXT PROGRAM


SIGNOFF:      S=PSW,P                 " ANY PUNCHING DONE?
            N,GOTOR(MC[-1])           " EXIT SIGNOFF
              S=127
              SUBC(:PUHEP)            " PUNCH ERASE
              A=25
              COUNT=A
BL:           S=0
              SUBC(:PUHEP)            " PUNCH BLANK
              REPP(:BL)               " 25 TIMES
              S=11
              SUBC(:PUHEP)            " PUNCH STOPCODE
              A=150
              COUNT=A
BM:           S=0
              SUBC(:PUHEP)            " PUNCH BLANK
              REPP(:BM)               " 150 TIMES
              S=-32767
              GOTO(:PUHEP)            " ONE MORE BLANK AND FORCE PUNCHING
```

```
THEAD:      +26
           +124
            +59
           +122
           'SKIP'8
            +64
           'SKIP'8
           +124
            +59
            +26

RATTLE:     SUBC(:FLBLNK)          " FILL BUFFER WITH BLANKS
            F=127                  " ERASE
            MA[-1]=G               " AT END OF BUFFER
            S=:THEAD               " )  TRANSPORT
            F=MS                   " )  DATE
            MA[-100]=F             " )  AND
            A+2                    " )  SERIAL
            S+2                    " )  NUMBER
          U,S-:THEAD[24],Z         " )  TO
          N,JUMP(-6)               " )  BUFFER.
            GOTOR(MC[-1])          " EXIT RATTLE

FIX60:      MC=A                   " CONVERSION TABLE
            MC=S                   " DESTINATION
            A=6                    " NN
            S=0                    " MM
            ABS=S                  " UNSIGNED VERSION
            SUBC(:FIX1)            " FIX CONVERSION
            A=:CBASE
            G=MC[-1]
            S=MA                   " CHARACTER FROM FIX
            S+M[B-1]               " ADDRESS CONVERSION TABLE
            S=MS                   " CONVERT TO DESIRED CODE
            MG=S                   " STORE IN DESTINATION
            A+1
            F+1                    " INCREMENT FOR NEXT CHAR
          U,A-:CBASE[8],Z          " ALL DONE?
          N,JUMP(-8)
            B-1
            GOTOR(MC[-1])          " EXIT FIX60

FIXT60:     SUBC(:TWAIT)           " WAIT TIL TYPEWRITER IS FREE
            A=:TELXTB
            S=:TBUF
            SUBC(:FIX60)
            SUBCD(:TYPE)
            ('010 000 000'+:TBUF[-1])
            GOTOR(MC[-1])
```

```
SIGNON:     S=PWIGGLE,Z                 " )  WAIT UNTIL
          Y,S=PWAGGLE,Z                 " )  PUNCHING ACTIVITY
          N,JUMP(-3)                    " )  IS COMPLETED
            S=:PWIGGLE
            PBUNOE=S                    " )  SELECT FIRST
            PBUNOF=S                    " )  PUNCH BUFFER
            PSW=-B                      " NO PUNCHING DONE YET
            G=:PBASE1[:PBULEN[-1]]      " )  FORCE PUNCHING ONLY AFTER
            PPOS=G                      " )  FIRST HEPTAD IS OFFERED
            S=:THEAD[4]
            A=:FLEXTB
            G=DATE
            SUBC(:FIX60)                " CONVERT DATE AND SET IN THEAD
            S=:THEAD[13]
            A=:FLEXTB
            G=SERIAL
            SUBC(:FIX60)                " SAME WITH SERIAL NUMBER
            GOTO(:RATTLE)               " SIGN-ON TO BUFFER AND EXIT SIGNON

ISKIP:      A=IHOK,P                    " ALREADY END OF TAPE?
          Y,JUMP(6)
            A=IBUSY,Z                   " IS READER ACTIVE?
          Y,JUMP(-2)                    " THEN WAIT
            A=IBUNOE
            SUBC(:ISTART)               " START READER
            A=IAR[0],P                  " NOT YET E O T?
          Y,JUMP(-6)
            ('660 070 000'+:RE)         " CANCEL PREVIOUS START
            G=('000 004 000'+:RE)
            SUBCD(:BRUSH)               " RESET READER

EMPTYI:     F=0
            CASE=G                      " LOWER CASE
            A=:IBASE1
            MA[1]=-F                    " )  FILL INPUT
            A+2                         " )  BUFFERS
          U,A-:PBASE1,Z                 " )  WITH
          N,JUMP(-4)                    " )  MINUS ZERO
            S=:IWIGGLE
            IBUNOE=S                    " SELECT FIRST INPUT BUFFER
            IWIGGLE=B
            IWAGGLE=B                   " BOTH BUFFERS ARE EMPTY
            IBUSY=B                     " NO CURRENT INPUT ACTIVITY
            GOTOR(MC[-1])               " EXIT EMPTYI AND ISKIP

BRUSH:      M[216]=G                    " RESET INSTR INTO AR[0]
            '760 070 046'               " START RESET APPARATUS
            '760 075 000'               " READ INTERRUPTS INTO S
            LCS(2),P                    " NO INTER. FROM RESETTER?
          Y,JUMP(-3)                    " WAIT
            '660 071 046'               " REMOVE INTERRUPT
            GOTOR(MC[-1])               " EXIT BRUSH
```

```
FLBLNK:         S=PBUNOE            " BUFFER TO BE FILLED
                BBSAFE=B            " PRESERVE STACK POINTER
                B=MS[2]             " PBASE
                A=B
                A+:PBULEN           " END OF BUFFER
                F=0
                MC[1]=F             " TWO BLANKS
              U, B-A,Z              " ALL DONE?
              N, JUMP(-3)
                B=BBSAFE            " RESTORE STACK POINTER
                GOTOR(MC[-1])       " EXIT FLBLNK

" ALL INPUT-OUTPUT INTERRUPTS CAUSE A TRANSFER TO IOINTP
" THROUGH THE INSTRUCTION IN M[24]
IOINTP:         ASAFE = A
                SSAFE = S
                SAFEB = B
                FSAFE = F
                '660 075 000'       "READ INTERRUPTS INTO A
                NORA
                A = B
                A + :INTAB
                B = :INSTCK         "INTERRUPTION STACK
                GOTO(MA)
INTAB:          :DYST               "CLOCK
                :DENTST             "RESET APPARATUS
                :DYST               "CHARON FAULT HANDLING
                :DYST               "APPARATUS      36
                :DYST               "               35
                :DYST               "               34
                :DYST               "               33
                :DYST               "               32
                :DYST               "                0
                :DYST               "                1
                :DYST               "                2
                :DYST               "                3
                :DYST               "                4
                :DYST               "                5
                :DYST               "                6
                :DYST               "                7
                :DYST               "                8
                :DYST               "                9
                :DYST               "               10
                :DYST               "               11
                :DYST               "               12
                :DYST               "               13
                :DYST               "               14
                :DYST               "               15

INTAB[8+:RE]:   :READER
INTAB[8+:PU]:   :PCHER
INTAB[8+:KY]:   :KEYBD
INTAB[26]:
```

```
HOK:            S = D18
                MA[1] = S               "ACTION COUNTER = 1
                S = MA[-1]              "ADDRESS OF SS[1]
                S'X'D18M1               "CLEAN AWAY NOK SECTION
                G = MS[1]
                MC = G                  "PRESERVE OLD CODEWORD
                F = 0
                MS[1] = G               "CODEWORD
                MA[0] = -G              "INTERRUPT COUNTER=-1
                G = A
                RUA(2)                  "APPARATUS NUMBER + 16
                A + START               "START INSTR IN A
                DO(A)
              U, S = MG[0] , P          "DONE YET?
              N, JUMP(-2)               "THEN WAIT
                A = MC[-1]
                MS[1] = A               "RESTORE OLD CODEWORD
                GOTOR(MC[-1])           "EXIT HOK
START:          ('760 070 000' - 16)

REHEP:          G = IBUNOE
              U, S = MG[0] , Z          "PRESENT?
              N, GOTO(:ICHK)
                A = 1
                PLUSA(IPOS)             "IPOS + 1
                S = MA                  "FETCH CHARACTER
                MA = -A                 "MAKE PLACE EMPTY
                A = MA[1] , P           "MORE CHARACTERS STILL IN BUFFER?
REEX:         Y, A = :MS                "COPY CHARACTER IN A
              Y, GOTOR(MC[-1])          "EXIT REHEP
                A = MG[1]               "NEW BUFFER
                IBUNOE = A              "SELECTION
                A = MA[2]
                IPOS = A                "IPOS = IBASE
                '660 060 000'           "CLEAR INTERRUPT PERMIT
                MG[0] = G               "PRESENT = FALSE
              U, A = IBUSY , Z          "INPUT ACTIVE?
              Y, GOTO(:REEX)
                A = G, P                "OLD IBUNOE. COND = YES
                SUBC(:ISTART)           "START INPUT
                GOTO(:REEX)
ICHK:           S = IBUSY , Z
              N, S = MG[0] , Z          "PRESENT?
              Y, GOTO(:REHEP[1])
                SUBCD(:CTYPE)           "TYPE "INPUT TAPE"
                  :REMES
                IHOK = B                "INDICATE E O T
                A = :IAR[1]
                SUBC(:HOK)
                A = IBUNOE
                S = MA[2]               "IBASE
                S + 2
                IPOS = S                "IPOS = IBASE + 2
                SUBC(:ISTART)
                GOTO(:REHEP)
```

```
READER:          ('660 071 000'+:RE)  "REMOVE INTERRUPT BIT
                 A = IAR[0] , P       "OK?
               Y, JUMP(2)
                 RUA(18) , Z          "NBK?
               Y, SUBC(D18M1)         "IN CASE OF MACHINE FAILURE
                 IHOK = -B            "INDICATE NOT E O T
                 IBUSY = A            "IBUSY = FALSE
                 S = IBUNOF
                 A = 0
                 MS[0] = A            "PRESENT
                 A = MS[1]            "NEW IBUNOF
                 S = MA[0] , Z        "PRESENT?
               N, SUBC(:ISTART)       "START INPUT
RESTORE:         A = ASAFE
                 S = SSAFE
                 B = SAFEB
                 F = FSAFE
                 GOTOR(M[23])         "EXIT INTERRUPT PROGRAMS

ISTART:        U, A = IAR[0] , P      "OK?
               N, GOTOR(MC[-1])       "EXIT ISTART
                 IBUNOF = A
                 A = MA[2]            "IBASE
                 MA[2] = -A           "CLEAR PLACE IN BUFFER
                 A '+' ILENGTH
                 ISS[2] = A           "CODEWORD
                 A = D18
                 IAR[2] = A           "ACTION COUNTER = 1
                 A = 0
                 IBUSY = A            "IBUSY = TRUE
                 IAR[1] = B           "INTERRUPT COUNTER = 0
                 ('760 070 000'+:RE)  "START TAPE READER
                 GOTOR(MC[-1])

PUHEP:           G = PBUNOF
               U, S = MG[0] , Z       "ROOM?
               N, GOTO(:PUHEP[1])     "WAIT UNTIL THERE IS ROOM
                 A = 1
                 PLUSA(PPOS)          "PPOS + 1
                 MA = S , E           "STORE HEPTAD. WAS IT NEG?
               N, A - MG[3], Z        "PEND. IS BUFFER NOW FULL?
               N, GOTOR(MC[-1])       "EXIT PUHEP
                 PSW = G              "NOTE OCCURENCE OF PUNCHING ACTIVITY
                 A = PPOS
                 A - MG[2]            "PBASE
                 PLENGTH = A
                 A = MG[1]
                 PBUNOF = A           "SWITCH OVER
                 S = MA[2]            "PBASE
                 PPOS = S             "INITIALIZE PPOS
                 '660 060 000'        "CLEAR INTERRUPT PERMIT
                 MG[0] = G            "ROOM = FALSE
               U, A = MA[0], Z        "ROOM IN OTHER BUFFER?
               Y, GOTO(:PSTART)       "START PUNCH
                 GOTOR(MC[-1])        "EXIT PUHEP
```

```
PSTART:          S = PLENGTH
PSTART1:         PBUNOE = G
                 RCS(9)                  "SHIFT PLENGTH TO POSITION
                 S '+' MG[2]             "PBASE. MAKE CODEWORD
                 PSS[2] = S              "STORE CODEWORD
PSTART4:         S = D18
                 PAR[2] = S              "ACTION COUNTER = 1
                 PAR[1] = B              "INTERRUPT COUNTER = 0
                 ('760 070 000'+:PU)     "START PUNCH
                 GOTOR(MC[-1])           "EXIT PSTART

PCHER:           ('660 071 000'+:PU)     "REMOVE INTERRUPT BIT
                 S = PBUNOE
                 F = 0
                 A = PAR[0], P           "OK?
               Y, GOTO(:POK)
                 RUA(18), Z              "NBK?
               Y, SUBC(D18M1)            "MACHINE FAILURE
                 A + 1 , Z               "NOK1?
                 A = :PAR[1]
               Y, SUBC(:HOK)             "GO OVER INTO NOK2
               Y, SUBC(:PSTART4)         "RE-OFFER FOR PUNCHING
               Y, GOTO(:RESTORE)         "END NOK1 HANDLING
                 S = PHOK, P             "SECOND TIME NOK2?
               Y, PHOK = -S              "RESET INDICATOR
               Y, SUBC(:HOK)             "GO OVER INTO OK
               Y, SUBC(:RATTLE)          "FILL BUFFER WITH SIGN-ON
               Y, GOTO(:COM)
                 SUBC(:FIBLNK)           "FILL BUFFER WITH BLANKS
                 S = 127
                 MA[-:PBULEN[-1]] = S "ERASE
                 S = 11
                 MA[-:PBULEN[-26]] = S"STOPCODE
                 PHOK = S                "SET INDICATOR
                 SUBCD(:CTYPE)           "TYPE "PUNCH EMPTY"
                   :TAPMES
    COM:         G = PBUNOE
                 S = :PBULEN[-1]         "LENGTH OF BUFFER - 1
                 SUBC(:PSTART1)
                 GOTO(:RESTORE)
    POK:         MS[0] = - G             "ROOM = TRUE
                 G = MS[1]               "NEW PBUNOE
                 S = MG[0], Z            "ROOM?
               N, SUBC(:PSTART)
                 GOTO(:RESTORE)
```

```
CTYPE:      SUBC(:TWAIT)          "WAIT TIL PREV. LINE IS DONE
            A = 1
            PLUSA(M[B - 1])       "INCREASE LINK BY ONE
            G = MA[-1]            "ADDRESS OF FIRST WORD
            KBUF[3] = B           "SAVE STACK POINTER
            B = :TBUF             "BEGIN ADDRESS OF TYPE BUFFER
            A = MG                "PACKED WORD IN A
            S = 0                 "CLEAR S
            LCSA(6) , Z           "SHIFT 6 BITS TO S. DONE?
          Y, F + 1                "INCREMENT FOR NEXT PACKED WORD
          Y, JUMP(-5)             "GET NEXT PACKED WORD
          U, S - 63, Z            "END MARKER?
          N, M[B] = S             "STORE IN TYPE BUFFER
          N, B + 1                "INCREMENT STORE ADDRESS
          N, JUMP(-8)             "RETURN FOR NEXT CHARACTER
            A = -:TBUF
            A + B                 "NO. OF CHARACTERS IN A
            LUA(18)               "SHIFT TO POSITION
            A '+' :TBUF[-1]       "MAKE CODEWORD
            B = KBUF[3]           "RESTORE STACK POINTER
            GOTO(:TYPE3)          "TYPE CONTENTS OF TBUF

TYPE:       SUBC(:TWAIT)          "WAIT TIL PREV. LINE IS DONE
            A = 1
            PLUSA(M[B - 1])       "INCREASE LINK BY ONE
            A = MA[-1]            "CODEWORD
TYPE3:      TSS[2] = A            "STORE IN START LINK
            A - D18M1, P          "INCREMENT / DECREMENT. MORE?
          N, GOTO(:TACT)
            S = MA                "LOOK IN BUFFER
          U, S - 31, P            "IS IT CASE DEFINITION?
          Y, KBUF[2] = S          "STORE IN KEYBOARD BUFFER
            JUMP(-6)              "REPEAT

TACT:       TAR[1] = -B           "INTERRUPT COUNTER = -1
            S = D18
            TAR[2] = S            "ACTION COUNTER = 1
            ('760 070 000'+:TY)   "START TYPEWRITER
            GOTOR(MC[-1])         "EXIT CTYPE, TYPE, TACT

TWAIT:      S = TAR[1], P         "TYPING COMPLETED?
          Y, GOTO(:TOFF)
            S = KAR[0]
            S - 27, Z             "FIGURE SHIFT?
          N, S -  3, P            "LETTER SHIFT
          Y, KCASE = S            "STORE IN KEYBOARD CASE
            ('760 070 000'+:KY)   "ACCEPT SYMBOLS FROM KEYBD
            GOTO(:TWAIT)
TOFF:       A = TAR[0], P         "OK?
          Y, GOTOR(MC[-1])        "EXIT TWAIT
            SUBC(D18M1)           "MACHINE FAILURE
```

```
KEYBD:      S = KAR[0], P          "IS THERE ALREADY A KEY PRESSED?
          N, JUMP(-2)              "WAIT
            KAR[0] = -S
            SYM = S                "STORE KEYBOARD SYMBOL
            KBUF[3] = S            "STORE FOR PRINTING
            ('660 071 000'+:KY)    "CLEAR INTERRUPT BIT
            KAR[1] = B             "INTERRUPT COUNTER = 0
            ('760 070 000'+:KY)    "ACCEPT SYMBOLS FROM KEYBD
            A = S
            A - 27, Z              "FIGURE SHIFT?
          N, A -  3, P             "LETTER SHIFT?
          N, JUMP(4)
            KCASE = A              "STORE IN KEYBOARD CASE
            A = INSW, Z            "IS KEYBD CALLED AS SUBROUTINE?
          N, GOTO(:RESTORE)
            GOTO(:KEYBD)           "WAIT FOR NOT SHIFT
            SUBC(:TWAIT)           "WAIT TIL TYPING IS DONE

            S = KCASE
            LUS(2)
            S + 33                 "RED COLOUR
          U, S - KBUF[2], Z        "COLOUR AND SHIFT OK?
          N, KBUF[2] = S           "CHANGE
          N, A = CW2               "TWO SYMBOLS TO BE PRINTED
          Y, A = CW1               "ONLY ONE SYMBOL
          U, A = INSW, Z           "IS KEYBD CALLED AS SUBROUTINE?
          N, A = CW4               "FOUR SYMBOLS TO BE PRINTED
            TSS[2] = A             "CODEWORD
            SUBC(:TACT)            "PRINT THE SYMBOL
            A = KCASE
            LUA(5)
            A + SYM                "SYMBOL + SHIFT
            A + :KTAB
            A = MA                 "TRANSLATION AND SWITCH ADDRESS
          Y, GOTOR(MC[-1])         "EXIT IF CALLED AS SUBROUTINE
            GOTO(A)                "SWITCH

AFXP:       S = 0                  "MM = 0
            ABS = S                "UNSIGNED VERSION
            SUBC(:FIX1)            "FIXED-POINT CONVERSION
            GOTO(:PO)              "PUNCH IT OUT
```

```
XEENS:          MAYI=A                  " MAKE XEEN WORD UNAVAILABLE
                GOTO(:RESTORE)

STOP:           MAYI=B                  " MAKE XEEN WORD UNAVAILABLE
                GOTOR(MC[-1])

XEEN:           SUBC(:RND)              " ROUND PARAMETER
                MC=G                    " STACK IT
                '660 060 000'           " CLEAR INTERRUPT PERMIT
                A=MAYI,Z                " IS XEEN WORD AVAILABLE?
XGET:         Y,A=XEENW                 " THEN TAKE IT IN A
              Y,A'X'MC[-1]              " COLLATE WITH PARAMETER
                G=A                     " RESULT IN F
              Y,GOTOR(MC[-1])           " AND EXIT XEEN
                SUBC(:CTYPE)            " TYPE "XEEN"
                  :XMES
                A = 0
                INSW = A                " PUT KEYBOARD IN INPUT MODE
                MAYI = A
                SUBC(:INOCT)            " COLLECT OCTAL WORD
                XEENW=A                 " STORE COMPLETED RESULT
                INSW=B,P                " RESET KEYBD FOR AUTOSTARTS. COND=YES
                GOTO(:XGET)             " FETCH THE RESULT AND EXIT

NXT KYBD SL:    SUBC(:KEYBD)            " GET A KEYBOARD SYMBOL
                RUA(18)                 " CLEAN AWAY THE ADDRESS PORTION
                S=A                     " COPY IN A
                GOTOR(MC[-1])           " EXIT NXT KYBD SL

NKS:            SUBC(:NXT KYBD SL)
              U,S-255,Z                 " SHARP SIGN?
              N,GOTOR(MC[-1])           " EXIT NKS
                B-3                     " REMOVE THREE LINKS FROM STACK
                GOTO(:HANDRD)           " BEGIN AGAIN TO READ A NUMBER

HAND:           SUBC(:RND)              " ROUND PARAMETER
                MC=G                    " SAVE IT IN STACK
                '660 060 000'           " CLEAR INTERRUPT PERMIT
                SUBC(:CTYPE)            " TYPE "HAND"
                  :HMES
                G=MC[-1]                " TAKE PARAMETER FROM STACK
                SUBC(:FIXT60)           " TYPE IT OUT
DNAH:           S=STOCK
                MC=S                    " PRESERVE STOCK FROM READ
                S=0
                INSW=S                  " PREPARE KEYBOARD FOR INPUT
HANDRD:         S=32767
                STOCK=S                 " INITIALIZE STOCK FOR READ
                S=KYBD                  " TELL READ TO GET SYMBOLS FROM NKS
                SUBC(:READ1)            " INPUT A NUMBER
                INSW=B                  " RESET KEYBOARD FOR AUTOSTARTS
                S=MC[-1]
                STOCK=S                 " RESTORE OLD STOCK TO READ
                GOTOR(MC[-1])           " EXIT HAND
KYBD:           GOTO(:NKS)
```

```
FINIS:          '760 060 000'          "SET INTERRUPT PERMIT
                B = BEGIN OF STACK     "SET STACK POINTER
                A = 999
                SUBC(:ERRORM)
                GOTO(:ENDRUN)

NSTART:         '760 060 000'          "SET INTERRUPT PERMIT
                A = RUNNUMBER
                A - 500, Z             "RUN TIME ?
              N, GOTO(:RESTORE)        "ELSE IGNORE NEW START
                B = BEGIN OF STACK,P   "CONDITION = YES
                GOTO(:ENDRUN[1])       "CLOSE OUT AND RESTART

P:              FPERMIT = A            "PROTECT COMP AGAINST OVERWRITING
                PERMIT = A
                GOTO(:RESTORE)

U:              A = 0                  "ALLOW OVERWRITING
                GOTO(:P)

V:              A = 0                  "ALLOW OVERWRITING TEMPORARILY
                GOTO(:P[1])

L:              FWANTED = A            "COMPILE LINE NUMBERS
                GOTO(:RESTORE)

KEYM:           FWANTED = -A           "NO LINE NUMBERS
                GOTO(:RESTORE)

FLEXIS:         MC = S                 "SAVE S-REGISTER
                A 'X' 127              "REMOVE EXTRANEOUS BITS
                A + :INTREP            "CONVERSION TABLE
                S = MA                 "CONVERSION WORD
                A = 124                "UPPER CASE
                LCS(10), P             "LOWER CASE NOT REQUIRED?
              N, A = 122               "LOWER CASE
                RCS(1), E              "CASE INDIFFERENT?
                S 'X' 127              "ISOLATE CHARACTER
              Y, JUMP(6)               "PUNCH CHAR WITHOUT CASE
              U, A - FLCASE, Z         "REQUIRED CASE SAME AS CURRENT?
              Y, JUMP(4)               "PUNCH CHAR WITHOUT CASE
                MC = S                 "SAVE CHARACTER
                S = A
                SUBC(:FLEXHP)          "PUNCH PRECEDING CASE
                S = MC[-1]             "RESTORE CHARACTER
                SUBC(:FLEXHP)          "PUNCH CHARACTER
                S = MC[-1]             "RESTORE S
                GOTOR(MC[-1])          "EXIT FLEXIS
```

```
READCOMP:          S = PRESENCE, P        "COMPILER PRESENT?
                 Y, GOTOR(MC[-1])         "EXIT READCOMP
                    SUBCD(:CTYPE)         "TYPE "COMPILER"
                      :CMPMES
                    SUBC(:REHEP)          "READ HEPTAD
                    S + 0, Z              "BLANK?
                 Y, JUMP(-3)              "SKIP TO END OF BLANK
                    F = :BEGIN OF COMPILER
NEXTWD:             S = 32
                    MG = S
NEXTHP:             MC = G                "PRESERVE STORAGE ADDRESS
                    SUBC(:REHEP)          "READ HEPTAD
                    G = MC[-1]            "RESTORE STORAGE ADDRESS
                    S = MG, P             "MORE HEPTADS IN THIS WORD?
                    RCA(7)
                    LCSA(7)               "PROCESS NEW HEPTAD
                    MG = S                "STORE (PARTIAL) RESULT
                 Y, GOTO(:NEXTHP)         "GET NEXT HEPTAD
                    S = MG                "SET LAST PARITY INDICATOR
                    CLP                   "COPY PARITY BIT INTO CONDITION
                 Y, A '+' 1
                    A '×' 1, Z            "PARITY MISTAKE?
                 Y, JUMP(5)
                    F + 1                 "INCREMENT STORE ADDRESS
                    G - :END OF COMPILER, P  "ALL DONE?
                    G + :END OF COMPILER     "RESTORE STORE ADDRESS
                 N, GOTO(:NEXTWD)         "GET NEXT WORD
                    PRESENCE = B          "COMPILER NOW PRESENT
                    SUBC(:ISKIP)          "SKIP TO END OF TAPE
                    GOTO(:READCOMP)

TEST TEXT IN ARRAY:S = -DANGEROUS, P      "ANY DANGEROUS ERRORS?
                 Y, A = 490
                 Y, SUBC(:ERRORM)         "REPORT THIS FACT
                 Y, GOTO(:ENDRUN)         "KICK PROGRAM OFF MACHINE
                    S = TEXT IN ARRAY, Z  "IS PROG TEXT ALL IN STORAGE?
                 Y, GOTOR(MC[-1])         "EXIT TEST TEXT IN ARRAY
                    SUBCD(:CTYPE)         "TYPE "AGAIN"
                      :AGNMES
                    GOTO(:ISKIP)          "SKIP TO END OF TAPE

INOCT:              A = 0
                    MC = A                "INITIALIZE OCTAL WORD TO ZERO
                    SUBC(:NXT KYBD SL)    "GET SYMBOL FROM KEYBOARD
                    A = MC[-1]            "GET RESULT SO FAR
                 U, S - 255, Z            "SHARP SIGN?
                 Y, GOTO(:INOCT)          "START ALL OVER
                 U, S - 7, P              "NOT AN OCTAL DIGIT?
                 N, LCA(3)                "MAKE ROOM FOR NEW DIGIT
                 N, A '+' S               "PUT IT IN
                 Y, S - 254, Z            "IRON CROSS?
                 N, GOTO(:INOCT[1])       "GET NEXT SYMBOL
                    GOTOR(MC[-1])         "EXIT INOCT
```

```
KTAB:   ('105 000 000' + :RESTORE)   "ASTERISK = ↑
        ('005 000 000' + :RESTORE)   "5
        ('167 000 000' + :RESTORE)   "CR
        ('011 000 000' + :RESTORE)   "9
        ('135 000 000' + :RESTORE)   "SPACE
        ('131 000 000' + :RESTORE)   "⌧
        ('127 000 000' + :RESTORE)   ",
        ('130 000 000' + :RESTORE)   ".
        ('167 000 000' + :RESTORE)   "NL
        ('175 000 000' + :RESTORE)   ")
        ('004 000 000' + :RESTORE)   "4
        ('145 000 000' + :RESTORE)   "]
        ('010 000 000' + :RESTORE)   "8
        ('000 000 000' + :RESTORE)   "0
        ('174 000 000' + :RESTORE)   ":
        ('106 000 000' + :RESTORE)   "=
        ('003 000 000' + :RESTORE)   "3
        ('100 000 000' + :RESTORE)   "+
        ('376 000 000' + :RESTORE)   "IRON CROSS
        ('102 000 000' + :RESTORE)   "X
        ('170 000 000' + :RESTORE)   "!
        ('006 000 000' + :RESTORE)   "6
        ('144 000 000' + :RESTORE)   "[
        ('103 000 000' + :RESTORE)   "/
        ('101 000 000' + :RESTORE)   "_
        ('002 000 000' + :RESTORE)   "2
        ('133 000 000' + :RESTORE)   ";
CW1:    ('001 000 000' + :KBUF[2])
        ('007 000 000' + :RESTORE)   "7
        ('001 000 000' + :RESTORE)   "1
        ('142 000 000' + :RESTORE)   "(
CW2:    ('002 000 000' + :KBUF[1])
```

```
                ('377 000 000' + :RESTORE)   "SHARP SIGN
                ('035 000 000' + :RESTORE)   "T
                ('167 000 000' + :RESTORE)   "CR
                ('030 000 000' + :RESTORE)   "0
                ('173 000 000' + :RESTORE)   "SPACE
                ('021 000 000' + :RESTORE)   "H
                ('027 000 000' + :NSTART )   "N
                ('026 000 000' + :KEYM   )   "M
                ('167 000 000' + :RESTORE)   "NL
                ('025 000 000' + :L      )   "L
                ('033 000 000' + :RESTORE)   "R
                ('020 000 000' + :RESTORE)   "G
                ('022 000 000' + :RESTORE)   "I
                ('031 000 000' + :P      )   "P
                ('014 000 000' + :RESTORE)   "C
                ('037 000 000' + :V      )   "V
                ('016 000 000' + :RESTORE)   "E
                ('043 000 000' + :RESTORE)   "Z
                ('015 000 000' + :RESTORE)   "D
                ('013 000 000' + :RESTORE)   "B
                ('034 000 000' + :RESTORE)   "S
                ('042 000 000' + :RESTORE)   "Y
                ('017 000 000' + :FINIS  )   "F
                ('041 000 000' + :XEENS  )   "X
                ('012 000 000' + :RESTORE)   "A
                ('040 000 000' + :RESTORE)   "W
                ('023 000 000' + :RESTORE)   "J
CW4:            ('004 000 000' + :KBUF[-1])
                ('036 000 000' + :U      )   "U
                ('032 000 000' + :RESTORE)   "Q
                ('024 000 000' + :RESTORE)   "K
```

```
FLEXHP:     'BEGIN' COHPTB, NLCR, TAB, LC, UC
            " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
            "       HEP, PUHEP, INTREP, CCNT, FLCASE
            HEP=S                       " SAVE THE HEPTAD
            SUBC(:PUHEP)                " BUFFER IT OUT TO THE PUNCH
            S=HEP                       " RETRIEVE IT
          U,S'X'-127,Z                  " NOT MORE THAN SEVEN BITS?
          N,S=127                       " ELSE NO FLEXOWRITER SYMBOL
            S+:INTREP                   " BASE ADDRESS CONVERSION TABLE
            S=MS,P                      " ELEMENT FROM TABLE. NO SPECIAL HANDLING?
          Y,A=1                         " IN NORMAL CASE,
          Y,CCNT+A                      " CCNT := CCNT + 1
          Y,GOTOR(MC[-1])               " AND EXIT FLEXHP
            LUS(10)                     " SHIFT OUT PUTEXT BITS
            RUS(10)                     " SHIFT BACK
            S+:COHPTB                   " BASE ADDRESS JUMP TABLE
            GOTO(S)                     " JUMP TO HANDLE SPECIAL CASE
            GOTO(:NLCR)
            GOTO(:TAB)
            GOTOR(MC[-1])               " UNDERLINE OR VERTICAL BAR, EXIT
            GOTO(:LC)
            GOTO(:UC)
            GOTOR(MC[-1])               " ERASE, BLANK, STOPCODE, BACKSPACE
            GOTOR(MC[-1])               " NON-FLEXOWRITER CHARACTER
COHPTB:     GOTOR(MC[-1])               " EVEN CHARACTER
NLCR:       S=0
            CCNT=S                      " SET CCNT BACK TO ZERO
            GOTOR(MC[-1])               " AND EXIT FLEXHP
TAB:        S=CCNT
            S+9
            S'X'-7                      " MULTIPLE OF EIGHT
            GOTO(:NLCR[1])
LC:         S=122                       " LOWER CASE
            FLCASE=S                    " ASSIGN NEW CASE TO FLCASE
            GOTOR(MC[-1])               " AND EXIT FLEXHP
UC:         S=124                       " UPPER CASE
            GOTO(:LC[1])
            'END' FLEXHP
```

```
" READ READS THE NEXT NUMBER FROM THE TAPE READER IN MC FLEXOWRITER
" CODE, CONVERTS IT TO A NORMALIZED FLOATING-POINT NUMBER, AND
" DELIVERS IT IN THE F-REGISTER.

READ:          'BEGIN'          LOOP, WORK, D0, D1, P0, P1,
                                ASEMBL, ZERO, SIEVE, SEPEX, SEP, CLEAN,
                                XPONT, NXTSYM, TAPE, ROUND

               " ASSUMED TO BE DECLARED GLOBALLY ARE: READ1, DECBIN
               " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
               "       CNT, PT, HEAD, TAIL, MSIGN, STOCK, SIXMIL, DEXP,
               "       BSAFE, BEXP, TENPOW, SCALE, GIANT, ESIGN,
               "       NXTTAPESL, VALUE OF CONSTANT, SIXMM1
               " STOCK IS SET EXTERNALLY TO 32767 BEFORE FIRST CALL OF READ.
       " A CALL OF SUBC(:DECBIN)
       " MAY BE USED TO CONVERT A DOUBLE-LENGTH POSITIVE INTEGER IN
       " F, ALONG WITH A DECIMAL EXPONENT IN S, INTO
       " A NORMALIZED FLOATING-POINT NUMBER IN F.

               S=TAPE
READ1:         NXTSYM=S        " INITIALIZE FOR TAPE READER
               F=0
               CNT=F           " I.E. CNT:=PT:=0
               HEAD=F          " I.E. HEAD:=TAIL:=0
               DEXP=F          " I.E. DEXP:=MSIGN:=0
               S=STOCK         " +, -, OR SEPARATOR FROM LAST CALL
               JUMP(2)         " GO ANALYZE IT
               SUBC(:SIEVE)    " GET NEXT SYMBOL
             N,GOTO(:LOOP)     " HANDLE FIRST MANTISSA DIGIT
             U,S-32767,Z       " IS NON-DIGIT A SEPARATOR?
             N,MSIGN=S         " STORE SIGN OF MANTISSA
               JUMP(-5)        " AND GET NEXT SYMBOL

LOOP:          S=HEAD
               S-SIXMM1,P      " CAN MULT BY 10 CAUSE OVERFLOW?
             N,S=TAIL          " )
             N,MULAS(10)       " )  [HEAD, TAIL] :=
             N,TAIL=S          " )  [HEAD, TAIL] × 10
             N,S=HEAD          " )  + THE NEW DIGIT
             N,MULAS(10)       " )
             N,HEAD=S          " )
               S=-PT
             Y,S+1
               CNT+S           " COUNT DIGITS
               SUBC(:SIEVE)    " GET NEXT SYMBOL
             N,GOTO(:LOOP)     " IF DIGIT, PROCESS IT
WORK:          A=ESIGN
               A-65,Z          " NEGATIVE EXPONENT SIGN
             Y,A=-DEXP         " THEN TAKE - EXPONENT
             N,A=DEXP          " OTHERWISE POSITIVE
               A+CNT           " CORRECT EXPONENT WITH CNT
               DEXP=A          " STORE DECIMAL EXPONENT
               STOCK=S         " STORE THE FINAL SEPARATOR FOR NEXT CALL
               JUMP(3)
```

```
DECBIN:     HEAD=F
            DEXP=S
            MSIGN=B          " I.E. MSIGN ≠ 65
            BSAFE=B          " PRESERVE THE STACK POINTER
            A=-HEAD
            S=-TAIL          " -MANTISSA IN AS
            NORAS,Z          " NORMALIZE MANTISSA. IS IT ZERO?
          Y,F=0
          Y,GOTO(:ZERO)
            BEXP=-B          " INITIALIZE BINARY EXPONENT
            B=10             " INITIALIZE DECIMAL-BINARY CONVERSION
            HEAD=-A
          U,A=DEXP,P         " IS DEXP POS?
          Y,GOTO(:P1)        " MULTIPLICATION CYCLE
            GOTO(:D1)        " DIVISION CYCLE

D0:       U,A+TENPOW[B-1],P  " CAN WE DIVIDE WITHOUT OVERFLOW?
          N,RUAS(1)          " IF NOT, SHIFT RIGHT ONE
            DIVAS(TENPOW[B-1])  " DIVIDE BY POWER OF 10
            HEAD=-S          " STORE MORE SIGNIFICANT HALF OF QUOTIENT
            DIVAS(TENPOW[B-1])  " DIVIDE FOR LESS SIGNIFICANT HALF
            A=-SCALE[B-1]    " CORRESPONDING BINARY SCALING FACTOR
          N,A+1              " CORRECT FOR PRESHIFT
            BEXP+A           " BRING BEXP UP TO DATE
            A=-HEAD
D1:       U,B+DEXP,P         " IS DEXP > 10(USUALLY)?
          Y,B=-DEXP,Z        " HAS DECIMAL EXPT BEEN REDUCED TO 0 YET?
          N,DEXP+B           " BRING DEXP UP TO DATE
          N,GOTO(:D0)        " DIVIDE
            GOTO(:ASEMBL)    " BINARIZATION IS NOW COMPLETE

P0:         MULS(TENPOW[B-1])   " MULT TAIL BY POWER OF 10
            S=-HEAD
            MULAS(TENPOW[B-1])   " MULT HEAD BY POWER OF 10
            LCAS(1),P        " LEFT ONE. WAS IT ALREADY NORMALIZED?
          Y,RCAS(1)          " THEN SHIFT IT BACK
            HEAD=-A
            A=SCALE[B-1]     " BINARY SCALING FACTOR
          N,A-1              " CORRECT FOR NORMALIZING SHIFT
            BEXP+A           " BRING BEXP UP TO DATE
P1:       U,B-DEXP,P         " DEXP < 10(USUALLY)?
          Y,B=DEXP,Z         " THEN USE DEXP. IS IT ZERO?
          N,DEXP-B           " BRING DEXP UP TO DATE
          N,GOTO(:P0)        " MULTIPLY
```

```
ASEMBL:     A=HEAD
            S=-S            " PLUS MANTISSA IN AS
            B=BEXP
            B+12            " CORRECT BEXP FOR SHIFT
          U,B+2048,P        " IS EXPONENT UNDERFLOW IMPOSSIBLE?
          Y,GOTO(:ROUND)
            B+2087,P        " APPARENT UNDERFLOW CORRECTABLE BY SHIFT?
          N,B=40            " IF NOT, PREPARE TO SHIFT 40 PLACES
          Y,B-39            " MINUS REQUIRED NUMBER OF SHIFTS
          Y,B=-B
          U,B-31,P          " )
          Y,B-31            " )
          Y,RUAS(31)        " ) SHIFT RIGHT
            RUAS(B)         " )
            B=-2047         " UNDERFLOW EXPONENT
ROUND:      S+2048,P        " ROUND MANTISSA TAIL. NO CARRY?
          N,S=0             " IN CASE OF CARRY, TAIL := 0
          N,A+1,P           " CARRY OVER TO HEAD. NO OVERFLOW?
          N,RCA(1)          " REPAIR OVERFLOW IN HEAD
          N,B+1             " AND CORRECT BINARY EXPONENT
          U,B-2047,P        " EXPONENT OVERFLOW?
          Y,F=GIANT
          Y,GOTO(:ZERO)     " THEN PROVIDE LARGEST NUMBER
            RUAS(12)        " SHIFT MANTISSA TO POSITION
            HEAD=A
            A=B,Z           " IS EXPONENT ZERO?
          Y,A=0             " MAKE IT +0 THEN
            A'X'4095        " REMOVE COPIES OF SIGN BIT
            RCA(12)         " SHIFT EXPONENT INTO PLACE
            A'+'HEAD        " COLLATE EXPONENT WITH HEAD OF MANTISSA
            F=A             " PUT THE RESULT IN F
ZERO:       B=MSIGN         " SIGN OF THE MANTISSA
            B-65,Z          " IS IT MINUS?
          Y,F=-F            " THEN COMPLEMENT F
            F+0             " NORMALIZE F
            B=BSAFE         " RESTORE THE STACK POINTER
            VALUE OF CONSTANT=F  " STORE RESULT FOR ERROR MESS.
            GOTOR(MC[-1])   " EXIT READ

SIEVE:      SUBC(:NXTSYM)   " GET A SYMBOL
            A=S             " MAKE A COPY IN A
          U,S-9,P           " IS SYMBOL A NON-DIGIT?
          N,GOTO(MC[-1])    " RETURN WITH DIGIT, CONDITION = NO
          U,S-65,Z          " IS IT MINUS SIGN?
          N,A-64,Z          " OR PLUS SIGN?
          Y,GOTO(MC[-1])    " RETURN WITH SIGN, CONDITION = YES
          U,S-88,Z          " IS IT DECIMAL POINT?
          Y,A=1
          Y,PT=A            " NOTE OCCURENCE OF POINT
          Y,GOTO(:SIEVE)    " AND GET NEXT SYMBOL
          U,S-89,Z          " IS IT ₁₀?
          Y,GOTO(:XPONT)    " GO TO PROCESS THE EXPONENT
          U,S-123,Z         " IS IT A SPACE?
          N,GOTO(:SEP)      " THEN IT CAN ONLY BE A SEPARATOR
            SUBC(:NXTSYM)   " GET SYMBOL WHICH FOLLOWS THE SPACE
          U,S-123,Z         " IS IT ALSO A SPACE?
          N,GOTO(:SIEVE[1]) " THEN GO FIND OUT WHAT IT IS
```

```
SEPEX:      S=32767,P       " SEPARATOR INDICATION IN S, CONDITION = YES
            GOTO(MC[-1])    " RETURN WITH SEPARATOR
SEP:      U,S-120,Z         " IS THE SEPARATOR AN APOSTROPHE?
          N,GOTO(:SEPEX)    " IF NOT, AN ORDINARY SEPARATOR
CLEAN:      SUBC(:NXTSYM)   " NEXT SYMBOL
            S-120,Z         " IS IT APOSTROPHE?
          N,GOTO(:CLEAN)    " IF NOT, FLUSH IT OUT
            GOTO(:SEPEX)    " END OF COMMENT BETWEEN APOSTROPHES

XPONT:      B-1             " REMOVE THE UNUSED LINK
            S=CNT,Z
          Y,A=1,E           " NO DIGITS IN THE MANTISSA?
          Y,TAIL=A          " THEN THE MANTISSA IS 1
            SUBC(:SIEVE)    " NEXT SYMBOL
          N,JUMP(3)         " HANDLE FIRST DIGIT OF EXPONENT
          U,S-32767,Z       " IS THE NON-DIGIT A SEPARATOR?
          N,ESIGN=S         " THEN STORE SIGN OF EXPONENT
            JUMP(-5)        " AND GET NEXT SYMBOL
            DEXP=S          " STORE EXPONENT
            SUBC(:SIEVE)    " GET NEXT SYMBOL
          Y,GOTO(:WORK)     " NON-DIGIT, SO END OF EXPONENT
            S=DEXP
            MULAS(10)       " PROCESS THE NEW DIGIT
          U,S-999,P         " IS THE EXPONENT HUGE?
          Y,S=1000          " THEN 1000 IS BIG ENOUGH
            JUMP(-8)        " REPEAT

NXTSYM:     'SKIP'1
TAPE:       GOTO(:NXTTAPESL)
            'END' READ
```

```
" OUTPUT CONVERSION ROUTINES

            'BEGIN' PARAM, STORE, SPILL, FLO2, TEN, LAST, CONV,
                    DIV, MUL, MULT, AH, RU, GPONE, DIGITS,
                    SHRTCT, RET, NDBTP, STSP, STZ, IZERO,
                    IZERO3, FILL, SMA

            " ASSUMED TO BE DECLARED GLOBALLY ARE: FIX, FIX1,
            "       FLO, FLO1, GEN, PUNCH, FIXP, ABSFIXP, FLOP, PO
            " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
            "       XX, RND, TEMP, ABS, NN, MM, DEXP, SIXMIL,
            "       FIXFLO, BSAFE, BEXP, CBASE, HEAD, TAIL, TENPOW,
            "       SCALE, HALF, SGNBIT, TENTH, NTAB, CCNT,
            "       FLEXHP, FLEXTB

PARAM:      XX=F                    " PRESERVE F
            F=MC[-2]                " TWO LINKS FROM THE STACK
            TEMP=F                  " SAVE THEM
            F=MC[-2]                " SECOND PARAMETER
            SUBC(:RND)              " ROUND IT
            S=G                     " MM IN S
            F=MC[-2]                " FIRST PARAMETER
            SUBC(:RND)              " ROUND IT
            A=G                     " NN IN A
            F=TEMP                  " TWO LINKS
            MC=F                    " RETURN THEM TO THE STACK
            F=XX                    " THIRD PARAMETER, XX, IN F
            GOTOR(LINK)             " EXIT PARAM

STORE:      MG=S                    " STORE CHARACTER
            F+1                     " INCREMENT STORE LOCATION
            GOTOR(MC[-1])           " EXIT STORE

            B-1                     " REMOVE UNUSED LINK FROM STACK
SPILL:      A=13                    " NN := 13
            S=3                     " MM := 3
            ABS=A                   " SIGNED VERSION
            GOTO(:FLO2)             " PERFORM FLOATING CONVERSION

FIX:        SUB(:PARAM)             " GET PARAMETERS IN A, S, AND F
FIX1:       XX=F                    " STORE THE NUMBER TO BE CONVERTED
            NN=A                    " STORE NUMBER OF DIGITS BEFORE POINT
            MM=S                    " STORE NUMBER OF DIGITS AFTER POINT
            A+1,P                   " NN ≥ 0?
          Y,S+1,P                   " MM ≥ 0?
          Y,A+S
          Y,A-2,P                   " MM+NN ≠ 0?
          N,GOTO(:SPILL)
            A-21,P                  " MM+NN > 21?
          Y,GOTO(:SPILL)
            S=1                     " FIXFLO INDICATOR
            SUBC(:CONV)             " CONVERT THE NUMBER
            GOTOR(MC[-1])           " EXIT FIX
```

```
FLO:        SUB(:PARAM)          " GET PARAMETERS IN A, S, AND F
FLO1:       XX=F                 " STORE THE NUMBER TO BE CONVERTED
FLO2:       COUNT=S              " STORE NUMBER OF EXPONENT DIGITS
              S-1,Z              " WAS IT ONE?
            N,S-1,Z              " OR TWO?
            N,S-1,Z              " OR THREE?
            N,GOTO(:SPILL)       " IF NONE OF THESE, THEN ERROR
              MM=A               " STORE NUMBER OF FRACTION DIGITS
            U,A-13,P             " MM > 13?
              A-0,E              " OR MM < 1 ?
            Y,GOTO(:SPILL)       " THEN ERROR
              NN=S               " NUMBER OF DIGITS BEFORE POINT := 0
              SUBC(:CONV)        " CONVERT FRACTION PART OF NUMBER
              S=13               "
              MG[-1]=S           " STORE ␣ OVER SPACE
              A=DEXP,P           " IS DECIMAL EXPONENT POSITIVE?
            Y,S=10               " +
            N,S=11               " -
            N,A=-A,P             " ABS(DEXP). CONDITION = YES
              SUBC(:STORE)       " STORE SIGN OF EXPONENT
              S=A
              A=:SIXMIL[-1]
              A+COUNT
              MULS(MA)           " MULTIPLY DEXP BY .1, .01, OR .001

TEN:          TENAS              " MULT BY TEN
              DEXP=S             " STORE FRACTION
              S=A                " EXPONENT DIGIT TO S
              REPZ(:LAST)        " COUNT DOWN
            Y,S=S,Z              " LEADING ZERO?
            Y,S=14               " THEN REPLACE IT BY A SPACE
              SUBC(:STORE)       " STORE SPACE OR DIGIT
              S=DEXP             " RESTORE FRACTION TO S
              GOTO(:TEN)         " NEXT DIGIT
LAST:         SUBC(:STORE)       " STORE LAST DIGIT
              S=14               " SPACE
              SUBC(:STORE)       " STORE FINAL SPACE
              GOTOR(MC[-1])      " EXIT FLO

CONV:         FIXFLO=S           " STORE FIX/FLO INDICATOR
              BSAFE=B            " PRESERVE STACK POINTER
              F=XX,P
            N,F=-F               " ABSOLUTE VALUE OF NUMBER IN F
              A=-F
              S=-A
              RUS(15)            " BINARY EXPONENT IN S
              BEXP=S
              A'X'32767
              A'+'-32767
              S=-G               " MINUS MANTISSA IN AS
              F=:CBASE[1]        " INITIALIZE STORE
              NORAS              " NORMALIZE MANTISSA
              B-52
              BEXP-B             " BIN EXP WHEN A,S SEEN AS FRACTION
              HEAD=A
              A=-0
              DEXP=A             " INITIALIZE DECIMAL EXPONENT
              B=10
```

```
DIV:        BEXP+A,P          " UPDATE BEXP. STILL MORE DIVIDING TO DO?
          N,A=BEXP
          N,GOTO(:MULT)       " ENTER MULTIPLICATION LOOP
            DEXP-B            " UPDATE DEXP
            A=HEAD
          U,A+TENPOW[B-1],P   " CAN WE DIVIDE WITHOUT OVERFLOW?
          N,RUAS(1)           " PRESHIFT FOR DIVISION
            DIVAS(TENPOW[B-1]) " DIVIDE BY POWER OF TEN
            HEAD=S            " STORE MORE SIGNIFICANT HALF OF RESULT
            DIVA(TENPOW[B-1]) " DIVIDE FOR LESS SIGNIFICANT HALF
            A=-SCALE[B-1]     " CORRESPONDING BINARY SCALING FACTOR
          N,A+1               " CORRECT FOR PRESHIFT
            GOTO(:DIV)        " REPEAT

MUL:        A+SCALE[B-1]      " UPDATE BEXP WITH BINARY SCALING FACTOR
            BEXP=A
            DEXP+B            " UPDATE DEXP
            MULS(TENPOW[B-1]) " MULT BY POWER OF TEN
            S=HEAD
            MULAS(TENPOW[B-1]) " COMPLETE DOUBLE PRECISION MULT
            LCAS(1),P         " NORMALIZE. WAS IT ALREADY NORMALIZED?
          Y,RCAS(1)           " THEN RESTORE IT AS IT WAS
            HEAD=A
            A=BEXP            " BINARY EXPONENT
          N,A-1               " CORRECT FOR NORMALIZING SHIFT

MULT:     U,A+SCALE[B-1],P    " DONE WITH MULTIPLYING?
          N,GOTO(:MUL)
            B-1,P
          Y,GOTO(:MULT)
            B=-A              " REMAINS OF BINARY EXPONENT
          U,B-3,Z             " IS IT POSSIBLE TO MULT BY 10/16?
          N,GOTO(:AH)
            TAIL=S
            MULS(TENPOW)      " MULT BY TEN
            S=HEAD
            MULAS(TENPOW)     " FINISH DOUBLE PRECISION MULT
            LCAS(1),E         " MULT BY TWO. WAS THAT TOO MUCH?
          Y,B=1
          Y,DEXP+B            " UPDATE DEXP
          N,S=TAIL
AH:       N,A=HEAD
          N,RUAS(B)           " NORMALIZE DECIMAL MANTISSA
            HEAD=-A           " STORE POSITIVE
            TAIL=-S           " DECIMAL MANTISSA
            S=-0
            S-DEXP            " ZERO IS NOW -0
            DEXP=S            " STORE DEXP WITH CORRECT SIGN
            B=FIXFLO,Z        " FLO?
          Y,S=-0
            S+MM,P            " DEXP+MM
            M[0]=S            " STORE IN TWO
            M[1]=S            " COUNTER LOCATIONS
            A=HALF            " AS := .5
            S=0               " FOR ROUNDING
          N,JUMP(6)           " AVOID DIVISIONS
```

```
RU:             RUAS(4)                 " )
                DIVAS(TENPOW)           " )
                TEMP=S                  " )  .5 × 10 ↑ (-DEXP - MM)
                DIVA(TENPOW)            " )
                A=TEMP                  " )
                REPOP(:RU)              " )
                S+TAIL,P                " ROUND THE TAIL. NO CARRY?
              N,S'+'SGNBIT              " REMOVE THE OVERFLOWN BIT
              N,A+1                     " AND CARRY IT OVER TO HEAD
                A+HEAD,P                " ROUND HEAD. NO OVERFLOW?
              N,A=1
              N,DEXP+A                  " CORRECT DECIMAL EXPONENT FOR OVERFLOW
              N,M[1]+B                  " CORRECT COUNTER IF FIX
              N,A=TENTH                 " MANTISSA IS
              N,S=TENTH[1]              " .1 IN CASE OF OVERFLOW
                HEAD=A                  " STORE ROUNDED
                TAIL=S                  " MANTISSA

                S=DEXP
                B=BSAFE                 " RESTORE STACK POINTER
              U,S=FIXFLO,Z              " FLO?
              N,JUMP(7)
                A=COUNT                 " NUMBER OF EXPONENT DIGITS
                A+:NTAB[-1]
              U,S-MA,P                  " EXPONENT OVERFLOW?
              N,S=-S
              N,S-MA,P                  " OR EXPONENT UNDERFLOW?
              Y,GOTO(:SPILL[-1])        " HANDLE EXPONENT SPILL
                GOTO(:SHRTCT)           " SHORTCUT FOR FLO
                S-NN,P                  " DEXP-NN>0?
              Y,GOTO(:SPILL[-1])        " OVERFLOW IN FIX
                A=DEXP,P
              N,GOTO(:NDBTP)            " NO DIGITS BEFORE THE POINT
                M[0]=-S,Z               " NUMBER OF LEADING SPACES TO COUNTER
                M[1]=A                  " NUMBER OF DIGITS BEFORE THE POINT
              Y,JUMP(3)                 " IF NO SPACES AT ALL

                S=14,P                  " SPACE. CONDITION = YES
GPONE:          SUBC(:STORE)            " STORE SPACE
                REPOP(:GPONE)
DIGITS:         S=TAIL                  " )
                TENAS                   " )
                TAIL=S                  " )  CALCULATE
                S=HEAD                  " )  AND
                MULAS(10)               " )  STORE
                HEAD=S                  " )  DIGITS
                MG=A                    " )
                F+1                     " )
                REP1P(:DIGITS)          " )
              N,GOTO(:RET)              " AFTER DIGITS AFTER THE POINT
SHRTCT:         A=MM,Z
              Y,GOTO(:RET)              " DONE IF NO DIGITS AFTER THE POINT
                M[1]=A                  " NUMBER OF DIGITS AFTER POINT TO COUNTER
                S=12
                SUBC(:STORE)            " STORE DECIMAL POINT
                GOTO(:DIGITS)           " FOR DIGITS AFTER THE POINT
```

```
RET:        S=14                    " SPACE
            SUBC(:STORE)            " FILL IN TRAILING SPACE
            CBASE=S                 " FILL IN LEADING SPACE
          U,S=ABS,Z                 " ABS VERSION?
          Y,GOTOR(MC[-1])           " EXIT CONV
            A=:CBASE                " INITIALIZE SEARCH FOR FIRST NON-SPACE
          U,S-MA[1],Z               " IS CHARACTER A SPACE?
          Y,A+1                     " THEN INCREMENT SEARCH POSITION
          Y,JUMP(-3)                " AND REPEAT
            S=XX[1],P               " IS THE NUMBER POSITIVE?
          Y,S=10                    " +
          N,S=11                    " -
            MA=S                    " STORE SIGN OVER LAST LEADING SPACE
            GOTOR(MC[-1])           " EXIT CONV

NDBTP:      M[3]=-A                 " NUMBER OF ZEROES AFTER THE POINT
            A=MM,Z                  " NUMBER OF PLACES AFTER POINT. NONE?
          Y,GOTO(:IZERO)            " PRODUCE AN INTEGER ZERO
            S=NN,Z                  " NUMBER OF SPACES BEFORE THE POINT
          Y,JUMP(5)                 " IN CASE NO SPACES
            M[2]=S                  " STORE NUMBER OF SPACES IN COUNTER
            S=14                    " SPACE
STSP:       SUBC(:STORE)            " STORE SPACE
            REP2P(:STSP)
          Y,GOTO(:IZERO3)           " WHEN BEING USED FOR INTEGER ZERO
            A=M[1],P                " SIGNIFICANT DIGITS AFTER POINT?
          N,S=0
          N,HEAD=S                  " CREATE AN
          N,TAIL=S                  " ARTIFICIAL ZERO
          N,GOTO(:SHRTCT)           " AND CONVERT IT
            S=DEXP,Z
          Y,GOTO(:SHRTCT)

            S=12                    " DECIMAL POINT
STZ:        SUBC(:STORE)            " STORE DECIMAL POINT OR DIGIT ZERO
            S=0                     " DIGIT ZERO
            REP3E(:STZ)
            GOTO(:DIGITS)
IZERO:      S=NN
            S-1,P                   " NN-1>0?
          Y,GOTO(:STSP[-2])         " PRODUCE NN-1 SPACES
IZERO3:     S=0
            SUBC(:STORE)            " STORE SINGLE DIGIT ZERO
            GOTO(:RET)
```

```
GEN:        A=F,P
           N,A=-F          " ABSOLUTE VALUE OF HEAD OF F IN A
            A'X'-32767,Z   " IS NUMBER INTEGER?
            A=13           " NN := 13
           Y,S=0           " MM := 0  FOR FIX
           N,S=3           " MM := 3  FOR FLO
            ABS=A          " SIGNED VERSION
           N,SUBC(:FLO1)   " PERFORM FLOATING CONVERSION
           N,GOTOR(MC[-1])
            SUBC(:FIX1)    " PERFORM INTEGER CONVERSION
            S=6            " SIX SPACES EXTRA AFTER INTEGER
            M[0]=S         " COUNTER LOCATION
            S=14           " SPACE
FILL:       SUBC(:STORE)   " STORE SPACE
            REPOP(:FILL)
            GOTOR(MC[-1])  " EXIT GEN

PUNCH:      SUBC(:GEN)     " PERFORM GENERAL CONVERSION
            GOTO(:PO)      " GO AND GET IT PUNCHED OUT

FIXP:       S=1            " FOR SIGNED VERSION
            ABS=S          " STORE SIGN INDICATOR
            SUBC(:FIX)     " PERFORM FIXED-POINT CONVERSION
            GOTO(:PO)      " GO AND GET IT PUNCHED OUT

ABSFIXP:    S=0            " SIGNLESS VERSION
            GOTO(:FIXP[1])

FLOP:       ABS=B          " SIGNED VERSION
            SUBC(:FLO)     " PERFORM FLOATING-POINT CONVERSION

PO:         F-:CBASE       " NUMBER OF CHARACTERS IN CBASE
            COUNT=G        " STORE IN COUNTER LOCATION
            G+CCNT         " ADD POSITION ON FLEXOWRITER LINE
            F-150,P        " BEYOND 150 CHARACTERS ON THE LINE?
           Y,S=26          " THEN A CARRIAGE RETURN FIRST
           Y,SUBC(:FLEXHP) " OUTPUT THE NLCR
            S=122          " LOWER CASE
           U,S-FLCASE,Z    " WAS LAST PUNCHED CASE LOWER CASE?
           N,SUBC(:FLEXHP) " IF NOT, OUTPUT A LOWER CASE
            A=:CBASE       " INITIALIZE LOOP
SMA:        S=MA           " CHARACTER IN SPEC. INT. REPRES.
            S+:FLEXTB      " BASE ADDRESS CONVERSION TABLE
            S=MS           " CHARACTER IN FLEXOWRITER CODE
            MC=A           " SAVE A
            SUBC(:FLEXHP)  " OUTPUT CHARACTER
            A=MC[-1]       " RESTORE A
            A+1            " INCREMENT TO NEXT CHARACTER
            REPP(:SMA)     " COUNT AND RETURN
            GOTOR(MC[-1])  " EXIT PUNCH, FIXP, ABSFIXP, FLOP

            'END'          " OUTPUT CONVERSION ROUTINES
```

```
" LINE PRINTER SECTION

                    'BEGIN'      HERE, PRAR, PRBUNOE, PRBUNOF, PRBUNOF2,
                         PRBUNOF3, PRPOS, HEX, HEX2, PRPARF, PRBUSY, VERTCON,
                         PARMES, TORNMS, YOKMES, PRNTMS, CONTAB, PRNTAB,
                         HEADING, PRBUF1, PRBUF2, PRBUF3, PRBUF4, PRBUF5,
                         PRBUF6, PRBUF7, PRBUF8, PRBUF9, PRBUF10, SHFTAB,
                         MSKTAB, ST8B, CAPRET, PREXIT, SECOND, GOTBUF, SMS,
                         PRNTER, TESTRN, RECFFER, PROK, PRSTART, AMS,  TAB3

                  " ASSUMED TO BE DECLARED GLOBALLY ARE: LINENUMBER, INITPR1,
                  "      INITPR2, INITPR3, PRHEX, NEWPAGE, NLCR, CARRIAGE,
                  "      PRINT, FIXT, ABSFIXT, FLOT, PRO, PRNTIS, DERRORM,
                  "      ERRORM, NXTTAPESL, TAB, SPACE, AFXT6, AFXT
                  " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE: INTAB, PR,
                  "      DATE, SERIAL, BSAFE, RND, INSTCK, D18M1, CTYPE, HOK,
                  "      RESTORE, D18, GEN, FIX, FLO, CBASE, DANGEROUS,
                  "      RUNNUMBER, ERRONEOUS, SMES, FIXT60, WANTED,
                  "      LINECOUNTER, LASTSYMBOL, VALUEOFCONSTANT, ENDRUN,
                  "      LASTIDENTIFIER, REHEP, INTREP, CASE, ABS, FIX1, PO

HERE:
INTAB[8+:PR]:       :PRNTER
M[64+(4X:PR)]:      PRAR:
HERE[0]:

PRBUNOE:            'SKIP' 1
PRBUNOF:            'SKIP' 1
PRBUNOF2:           'SKIP' 1
PRBUNOF3:           'SKIP' 1
PRPOS:              'SKIP' 1
HEX:                'SKIP' 1
HEX2:               'SKIP' 1
PRPARF:             'SKIP' 1
PRBUSY:             'SKIP' 1
VERTCON:            'SKIP' 1
LINENUMBER:         'SKIP' 1

PARMES:             '02 10 44 15 0'      " CR NL BLKLT P
                    '12 14 06 01 0'      " R I N T
                    '20 12 04 15 0'      " E R SP P
                    '30 12 14 01 0'      " A R I T
                    '25 77 00 00 0'      " Y END
TORNMS:             '02 10 44 15 0'      " CR NL BLKLT P
                    '30 15 20 12 0'      " A P E R
                    '04 23 12 20 0'      " SP B R E
                    '30 36 77 00 0'      " A K END
YOKMES:             '02 10 44 25 0'      " CR NL BLKLT Y
                    '03 36 20 04 0'      " O K E SP
                    '03 15 20 06 0'      " O P E N
                    '77 00 00 00 0'      " END
PRNTMS:             '02 10 44 15 0'      " CR NL BLKLT P
                    '12 14 06 01 0'      " R I N T
                    '20 12 04 20 0'      " E R SP E
                    '07 15 01 25 0'      " M P T Y
                    '77 00 00 00 0'      " END
```

```
CONTAB:     +16;    +17;    +18;    +19;
            +20;    +21;    +22;    +23;
            +24;    +25;    +33;    +34;
            +35;    +36;    +37;    +38;

            +39;    +40;    +41;    +42;
            +43;    +44;    +45;    +46;
            +47;    +48;    +49;    +50;
            +51;    +52;    +53;    +54;

            +55;    +56;    +57;    +58;
              -0;   -33;    -34;    -35;
            -36;    -37;    -38;    -39;
            -40;    -41;    -42;    -43;

            -44;    -45;    -46;    -47;
            -48;    -49;    -50;    -51;
            -52;    -53;    -54;    -55;
            -56;    -57;    -58;      -0;

            +11;    +13;    +10;    +15;
              -0;     -0;   +29;      -0;
            +59;      -0;   +60;      -0;
            +63;      -0;     -0;   +61;

            +62;      -0;     -0;     -0;
              -0;     -0;     -0;   +12;
            +14;      +6;     +3;     +4;
              -0;     +0;   +30;      +1;

             +2;      -0;     +8;     +9;
            +26;    +27;      -0;     -0;
              -0;     -0;     -0;     -0;
              -0;     -0;     -0;     -0;

              -0;     -0;     -0;     -0;
              -0;     -0;   -65;     -66;
            +31;      +7;     +5;     +0;
              +3;     +9;   -32;     -28;

PRNTAB:     +16             " 0
            +17             " 1
            +18             " 2
            +19             " 3
            +20             " 4
            +21             " 5
            +22             " 6
            +23             " 7
            +24             " 8
            +25             " 9
            +11             " +
            +13             " -
            +14             " .
              +6            " ⏨
              -1            " SPACE
```

```
HEADING:        'SKIP' 1
                '400 000 000'
                'SKIP' 4

                'SKIP' 1
PRBUF1:         :PRBUF2
                ('045 000 000'+:PRBUF1[2])
                'SKIP' 38

                'SKIP' 1
PRBUF2:         :PRBUF3
                ('045 000 000'+:PRBUF2[2])
                'SKIP' 38

                'SKIP' 1
PRBUF3:         :PRBUF4
                ('045 000 000'+:PRBUF3[2])
                'SKIP' 38

                'SKIP' 1
PRBUF4:         :PRBUF5
                ('045 000 000'+:PRBUF4[2])
                'SKIP' 38

                'SKIP' 1
PRBUF5:         :PRBUF6
                ('045 000 000'+:PRBUF5[2])
                'SKIP' 38

                'SKIP' 1
PRBUF6:         :PRBUF7
                ('045 000 000'+:PRBUF6[2])
                'SKIP' 38

                'SKIP' 1
PRBUF7:         :PRBUF8
                ('045 000 000'+:PRBUF7[2])
                'SKIP' 38

                'SKIP' 1
PRBUF8:         :PRBUF9
                ('045 000 000'+:PRBUF8[2])
                'SKIP' 38

                'SKIP' 1
PRBUF9:         :PRBUF10
                ('045 000 000'+:PRBUF9[2])
                'SKIP' 38

                'SKIP' 1
PRBUF10:        :PRBUF1
                ('045 000 000'+:PRBUF10[2])
                'SKIP' 38
```

```
SHFTAB:         +21
                +14
                 +7
                 +0
MSKTAB:         '770 000 000'
                '003 740 000'
                '000 017 600'
                '000 000 077'

INITPR1:        PRPARF = B              " NO PARITY ERRORS YET
                PRBUSY = -B             " PRINTER NOT ACTIVE
                PRBUNOF2 = -B           " SECOND BUFFER NOT SELECTED
                PRBUNOF3 = -B           " THIRD BUFFER NOT SELECTED
                A = :PRBUF1
                PRBUNOF = A             " SELECT FIRST BUFFER
                S = 19
                COUNT = S
ST8B:           MA[2] = F               " STORE 8 BLANKS IN BUFFER
                A + 2
                REPP(:ST8B)
                S = MA[-38]             " NEXT BUFFER
                S 'X' D18M1             " REMOVE POSSIBLE MARKER BIT
                MA[-38] = S             " PUT IT BACK
                A = S
              U, A - :PRBUF1, Z         " ALL BUFFERS DONE?
              N, GOTO(:ST8B[-2])
                VERTCON = G
                G = ('000 004 000' + :PR)
                SUBCD(:BRUSH)           " RESET PRINTER
                GOTOR(MC[-1])           " EXIT INITPR1

INITPR2:        S = PRBUNOF
                MC = S                  " SAVE PRBUNOF IN STACK
                S = :HEADING[-2]
                PRBUNOF = S             " PREPARE TO FILL HEADING
                S = 3
                PRPOS = S
                F = 0
                HEADING = G
                HEADING[2] = F          " )  FILL HEADING
                HEADING[4] = F          " )  WITH BLANKS
                G = DATE
                SUBC(:AFXT6)            " CONVERT DATE INTO HEADING
                S = 13                  " MINUS
                SUBC(:PRHEX)            " TO HEADING
                G = SERIAL
                SUBC(:AFXT6)            " CONVERT SERIAL NO.
                HEADING = B
                S = MC[-1]
                PRBUNOF = S             " RESTORE PRBUNOF
                GOTO(:NEWPAGE)          " START NEW PAGE AND EXIT INITPR2

INITPR3:        S = 1
              U, S - LINENUMBER, Z      " AT TOP OF NEW PAGE
              Y, S - PRPOS, Z           " ^ BEGINNING OF NEW LINE?
              Y, GOTOR(MC[-1])          " EXIT INITPR3
                GOTO(:NEWPAGE)          " START NEW PAGE AND EXIT INITPR3
```

```
PRHEX:      HEX = S              " SAVE CHARACTER
            A = PRPOS            " POSITION ON LINE
          U, A - 144, P          " BEYOND THE END?
          Y, SUBC(:NLCR)         " START A NEW LINE
            BSAFE = B            " SAVE STACK POINTER
CAPRET:     G = PRBUNOF          " SELECT BUFFER
            S = MG[2], Z         " ROOM IN BUFFER?
          N, JUMP(-2)            " WAIT UNTIL THERE IS
            RUAS(2)              " WORD NUMBER IN A
            LCS(3)
            B = S                " CHARACTER NO. IN B
            S = HEX, P           " ORDINARY CHARACTER?
          N, GOTO(:SECOND)       " HANDLE CAPITALS, UNDERLINE, STROKE
            B = SHFTAB[B]        " NO. OF SHIFT POSITIONS
            LCS(B)               " SHIFT HEX TO POSITION
            A + :MG[3]           " POSITION IN BUFFER
            MA + S               " ADD TO BUFFER WORD
            A = 1
            PRPOS + A            " INCREMENT POSITION ON LINE
PREXIT:     B = BSAFE            " RESTORE STACK POINTER
            GOTOR(MC[-1])        " EXIT PRHEX

SECOND:   U, S + 33, P           " STROKE OR UNDERLINE?
          N, HEX = -S            " STORE CAPITAL LETTER
          N, S = -61             " CAPITAL INDICATOR
            HEX2 = -S            " STORE STROKE, UNDER, OR CAP
            G = PRBUNOF2, P      " IS THERE A SECOND BUFFER YET?
          Y, GOTO(:GOTBUF)
            G = PRBUNOF          " RESELECT PRIMARY BUFFER
            G = MG[0]            " SELECT SECONDARY BUFFER
          U, S = MG[2], Z        " ROOM?
          N, JUMP(-2)            " WAIT FOR ROOM
            PRBUNOF2 = G         " NOTE SECONDARY BUFFER

GOTBUF:     A + :MG[3]           " WORD IN SECOND BUFFER
            S = MSKTAB[B]        " MASK FROM TABLE
            S 'X' MA, Z          " IS POSITION EMPTY?
          Y, JUMP(12)            " THEN SECONDARY BUFFER
            A - :MG[3]           " RESTORE WORD NUMBER IN A
            G = PRBUNOF3, P      " IS THERE A TERTIARY BUFFER YET?
          Y, JUMP(5)
            G = PRBUNOF2         " RESELECT SECONDARY BUFFER
            G = MG[0]            " SELECT TERTIARY BUFFER
          U, S = MG[2], Z        " ROOM?
          N, JUMP(-2)            " WAIT FOR ROOM
            PRBUNOF3 = G         " NOTE TERTIARY BUFFER
            A + :MG[3]           " WORD IN TERTIARY BUFFER
            S = -MSKTAB[B]       " MASK FROM TABLE
            S 'X' MA             " MAKE PLACE EMPTY
            MA = S               " PUT IN BUFFER
            S = HEX2             " CHARACTER
          U, S - 61, Z           " CAPITAL INDICATOR?
            B = SHFTAB[B]        " NO. OF SHIFT POSITIONS
            LCS(B)               " SHIFT HEX2 TO POSITION
            MA + S               " INSERT INTO APPROPRIATE BUFFER
          N, GOTO(:PREXIT)       " DONE WITH STROKE OR UNDERLINE
            A = PRPOS            " POSITION ON LINE
            GOTO(:CAPRET)        " DO LETTER PART OF CAPITAL
```

```
NEWPAGE:        F = -1
                JUMP(1)                 " DO CARRIAGE(-1)
NLCR:           F = 1                   " DO CARRIAGE(+1)
CARRIAGE:       SUBC(:RND)              " ROUND PARAMETER
                S = PRBUNOF             " PRIMARY BUFFER
              U, S = MS[2], Z           " ROOM?
              N, JUMP(-2)               " WAIT FOR ROOM
                A = VERTCON             " VERTICAL CONTROL INDICATION
                LCA(21)                 " SHIFT TO POSITION
                MS[3] + A               " PUT IT INTO BUFFER
                A = PRBUNOF2, P         " IS THERE A SECOND BUFFER?
              Y, MA[2] = B              " ROOM = FALSE
              Y, S = A                  " NOTE HIGHEST BUFFER IN S
              Y, PRBUNOF2 = -A          " CANCEL RESERVATION OF SECONDARY
                A = PRBUNOF3, P         " TERTIARY BUFFER?
              Y, MA[2] = B              " ROOM = FALSE
              Y, S = A                  " NOTE HIGHEST BUFFER IN S
              Y, PRBUNOF3 = -A          " CANCEL RESERVATION

                A = G, Z                " RETRIEVE PARAMETER. ZERO?
              Y, A = 0                  " THEN +0
              U, A - 31, P              " > 31
              U, A + 2,  E              " V < -1?
              Y, A = 1                  " HANDLE OUT OF RANGE
              U, A + 1, Z               " NEW PAGE?
              Y, A = 60
                VERTCON = A             " VERTICAL CONTROL INDICATION
                PLUSA(LINENUMBER)       " INCREMENT LINENUMBER
                A - 60, P               " > 59?
              N, GOTO(:SMS)             " STAY ON SAME PAGE
                A = 2
                VERTCON = A             " DOUBLE SPACE AFTER HEADING
                A = 1
                LINENUMBER = A          " LINENUMBER = 1 AFTER HEADING
                A = D18
                MS + A                  " MARK LAST BUFFER ON PAGE

                S = MS[0]               " SELECT NEXT BUFFER
              U, S = MS[2], Z           " ROOM?
              N, JUMP(-2)               " WAIT FOR ROOM
                F = HEADING             " ) INSERT
                MS[2] = F               " ) HEADING
                F = HEADING[2]          " ) INTO
                MS[4] = F               " ) FOLLOWING
                F = HEADING[4]          " ) BUFFER
                MS[6] = F               " )
SMS:            S = MS[0]               " NEXT BUFFER
                A = PRBUNOF             " OLD PRIMARY BUFFER
                PRBUNOF = S             " NOTE NEW PRIMARY BUFFER
                '660 060 000'           " CLEAR INTERRUPT PERMIT
                MA[2] = B               " ROOM = FALSE, OLD PRIMARY
                S = PRBUSY, P           " PRINTER ACTIVE?
              N, SUBC(:PRSTART)         " START PRINTER
                A = 1
                PRPOS = A               " START AT BEGINNING OF LINE
                GOTOR(MC[-1])           " EXIT NEWPAGE, NLCR, CARRIAGE
```

```
PRNTER:          ('660 071 000' + :PR)" CLEAR PRINTER INTERRUPT
                 PRBUSY = -B          " PRINTER NOT BUSY
                 S = PRBUNOE          " SELECT BUFFER
                 F = 0
                 A = PRAR[0], P       " OK?
               Y, GOTO(:PROK)
                 RUA(18), Z           " NBK?
               Y, SUBC(D18M1)         " MACHINE FAILURE
                 S = MS[-1]           " END WORD
                 RCS(2), P            " ¬ PARITY ERROR?
               Y, GOTO(:TESTRN)
                 A = PRPARF, P        " FIRST TIME?
               N, SUBC(:D18M1)        " GIVE UP
                 PRPARF = -B          " RESET INDICATOR
                 SUBC(:CTYPE)
                    :PARMES           " TYPE "PRINTER PARITY"
                 GOTO(:REOFFER)       " TRY ONCE AGAIN

TESTRN:          RCS(1), P            " ¬ PAPER TORN?
               N, SUBC(:CTYPE)
                    :TORNMS           " TYPE "PAPER BREAK"
               N, SUBC(D18M1)         " GIVE UP
                 SUBC(:CTYPE)
                    :YOKMES           " TYPE "YOKE OPEN"
                 A = :PRAR[1]
                 SUBC(:HOK)           " RESTORE PRINTER OK
REOFFER:         S = PRBUNOE          " RESELECT BUFFER
                 A = MS[3]            " FIRST BUFFER WORD
                 A '×' -MSKTAB[0]
                 MS[3] = A            " CARRIAGE CONTROL = 0
                 A = S                " SELECTED BUFFER
                 SUBC(:PRSTART)       " START PRINTER
                 GOTO(:RESTORE)       " EXIT PRNTER

PROK:            PRPARF = B           " SET INDICATOR
                 A = :MS[40]          " END OF BUFFER
                 B = :MS[2]           " START OF BUFFER
                 MC = F               " 8 BLANKS
               U, B - :MA, Z          " BUFFER FULL?
               N, JUMP(-3)            " 8 MORE
                 B = :INSTCK          " FILL STACK POINTER
                 A = MS[0]            " NEW PRBUNOE
               U, A - D18, P          " END OF PAGE?
               Y, A '×' D18M1
               Y, MS[0] = A           " REMOVE END OF PAGE MARKER
                 MC = A               " SAVE NEW PRBUNOE
               N, JUMP(6)
                 A = MS[-1], Z        " ¬ PAPER LOW?
               Y, JUMP(4)
                 A = :PRAR[1]
                 SUBC(:HOK)           " RESET PRINTER
                 SUBC(:CTYPE)
                    :PRNTMS           " TYPE "PRINTER EMPTY"
                 A = MC[-1]           " PRBUNOE
                 S = MA[2], Z         " ROOM?
               N, SUBC(:PRSTART)      " START PRINTER
                 GOTO(:RESTORE)       " EXIT PRNTER
```

```
PRSTART:        PRBUNOE = A
                PRAR[0] = A
                PRBUSY = B              " PRINTER IS BUSY
                PRAR[1] = B             " INTERRUPT COUNTER = 0
                S = D18
                PRAR[2] = S             " ACTION COUNTER = 1
                ('760 070 000' + :PR)"  START CHARON
                GOTOR(MC[-1])           " EXIT PRSTART

PRINT:          SUBC(:GEN)              " PERFORM GENERAL CONVERSION
                GOTO(:PRO)              " PRINT OUT

FIXT:           S = 1                   " SIGNED VERSION
                ABS = S                 " SIGN INDICATOR
                SUBC(:FIX)              " FIXED-POINT CONVERSION
                GOTO(:PRO)              " PRINT OUT

ABSFIXT:        S = 0                   " UNSIGNED VERSION
                GOTO(:FIXT[1])

FLOT:           ABS = B                 " SIGNED VERSION
                SUBC(:FLO)              " FLOATING-POINT CONVERSION

PRO:            F - :CBASE              " NUMBER OF CHARACTERS
                COUNT = G
                G + PRPOS               " POSITION ON LINE
                F - 145, P              " BEYOND THE END?
              Y, SUBC(:NLCR)            " BEGIN ON NEW LINE
                A = :CBASE
AMS:            S = MA                  " CHARACTER
                S + :PRNTAB             " CONVERSION TABLE
                S = MS, P               " CONVERT CODE. ¬ SPACE?
              N, PRPOS - S              " PRPOS = PRPOS + 1
              Y, MC = A                 " SAVE A
              Y, SUBC(:PRHEX)           " PRINT CHARACTER
              Y, A = MC[-1]             " RESTORE A
                A + 1                   " INCREMENT FOR NEXT CHARACTER
                REPP(:AMS)
                GOTOR(MC[-1])           " EXIT PRINT, FIXT, ABSFIXT, FLOT

PRNTIS:         MC = S                  " PRESERVE S-REGISTER
                A '×' 127               " REMOVE EXTRANEOUS BITS
                A + :CONTAB             " CONVERSION TABLE
                S = MA                  " CONVERT TO PRINTER CODE
              U, S + 65, P              " ¬ TAB OR NLCR?
              Y, SUBC(:PRHEX)           " PRINT
              Y, S = MC[-1]             " RESTORE S
              Y, GOTOR(MC[-1])          " EXIT PRNTIS
                S + 65, Z               " TAB?
              Y, SUBC(:TAB)             " DO TAB
              N, SUBC(:NLCR)            " DO NLCR
                S = MC[-1]              " RESTORE S
                GOTOR(MC[-1])           " EXIT PRNTIS
```

```
        'BEGIN'     FVOC, ELOOPO, ELOOP1

      " ASSUMED TO BE DECLARED GLOBALLY ARE: DERRORM, ERRORM
      " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE: DANGEROUS
      "       RUNNUMBER, ERRONEOUS, PRPOS, CARRIAGE, CTYPE, SMES,
      "       FIXT60, PRHEX, AFXT, WANTED, LINE COUNTER, AFXT6,
      "       LAST SYMBOL, CONTAB, SPACE, PRNTIS, VALUE OF CONSTANT,
      "       PRINT, ENDRUN, LAST IDENTIFIER

DERRORM:    DANGEROUS = -B          " NOTE DANGEROUS ERROR
ERRORM:   U, RUNNUMBER - A, P       " IGNORE ERROR?
          Y, GOTOR(MC[-1])          " EXIT DERRORM AND ERRORM
             MC = S                 " SAVE S-REGISTER
             S = 0
             ERRONEOUS = S          " NOTE ERROR
             S = PRPOS              " POSITION ON LINE
             MC = S                 " PRESERVE IT IN STACK
             MC = A                 " PRESERVE ERROR NUMBER
             S - 1, Z               " AT BEGINNING OF LINE?
          Y, F = 1                  "  ) SINGLE OR
          N, F = 2                  "  ) DOUBLE
             SUBC(:CARRIAGE)        "  ) SPACE VERTICALLY
             SUBCD(:CTYPE)
                :SMES               " TYPE CR NL BLKDIG
             G = M[B - 1]           " ERROR NUMBER
             SUBC(:FIXT60)          " TYPE IT

             S = 37
             SUBC(:PRHEX)           " "e"
             S = 50
             SUBC(:PRHEX)           " "r"
             A = 3
             G = MC[-1]             " ERROR NUMBER
             SUBC(:AFXT)            " PRINT IT
             A = RUNNUMBER
             A - 500, Z             " EXECUTION
          Y, A = -WANTED, P         " ∧ ¬ WANTED?
             G = LINECOUNTER        " OBJECT PROG LINE NUMBER
          N, SUBC(:AFXT6)           " PRINT IT

             A = RUNNUMBER
             A - 500, Z             " EXECUTION?
          Y, GOTO(:FVOC)            " AVOID LAST SYMBOL
             A = LAST SYMBOL
          U, A 'x' -127, Z          " < 128 ?
          N, JUMP(5)                " ELSE PRINT INT REP
          U, A - 93, P              "  ) ≠ 94,
          U, A - 96, E              "  ) 95, 96?
          Y, A + :CONTAB            " CONVERSION TABLE
          Y, A = MA, P              "  ) NOT COMPOUND?
          Y, A - 0, P               "  ) ∧ NOT SPACE?
             G = LAST SYMBOL
          N, SUBC(:AFXT6)           " PRINT INTERNAL REPRESENTATION
          N, GOTO(:FVOC)            " DONE
```

```
                F = 6
                SUBC(:SPACE)          " 6 SPACES
                A = LAST SYMBOL
                SUBC(:PRNTIS)         " PRINT THE SYMBOL
                F = 1, Z              " CONDITION = NO
                SUBC(:SPACE)          " 1 SPACE
FVOC:           F = VALUE OF CONSTANT
                SUBC(:PRINT)          " PRINT IT
              Y, GOTO(:ENDRUN)        " TERMINATE EXECUTION

                S = 2                 " 2 WORDS OF IDENTIFIER
                M[6] = S              " COUNTER LOCATION
                S = LAST IDENTIFIER   " FIRST WORD OF IDENTIFIER
ELOOPO:         A = 4                 " 4 CHARACTERS PER WORD
                COUNT = A             " COUNTER LOCATION
                LCS(2)                " DISCARD UNUSED BITS
ELOOP1:         LUAS(6)               " SHIFT TO POSITION
                A 'X' 63, Z           " CLEAN CHARACTER. DONE?
              N, A = :MA[-1]          " CORRECT CODE
              N, SUBC(:PRNTIS)        " PRINT CHARACTER
                REPP(:ELOOP1)         " NEXT CHARACTER, IF ANY
                S = LAST IDENTIFIER[1], Z  " SECOND WORD. EMPTY?
              N, REP6P(:ELOOPO)       " PRINT SECOND WORD
                F = 2
                SUBC(:CARRIAGE)       " DOUBLE SPACE VERTICALLY
                S = MC[-1]            " OLD POSITION ON LINE
                PRPOS = S             " RESTORE IT
                S = MC[-1]            " RESTORE S-REGISTER
                GOTOR(MC[-1])         " EXIT DERRORM AND ERRORM

                'END'                 " ERROR MESSAGE
```

```
NXTTAPESL:      'BEGIN'     DECIDE, EXIT, SPECIAL, PARITY, RCLN,
                            S119, BAT, UL, LC, UC, NONFLX

                " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
                "           REHEP, INTREP, CASE, PRNTIS, ERRORM,
                "           RUNNUMBER, NLCR, LINECOUNTER, AFXT6,
                "           TAB

                SUBC(:REHEP)          " GET HEPTAD FROM BUFFER
                S '×' 127             " REMOVE EXTRANEOUS BITS
                S + :INTREP           " BASE ADDRESS CONVERSION TABLE
                S = MS, P             " TABLE ENTRY IN S
              N, GOTO(:SPECIAL)       " IF NEGATIVE, SPECIAL HANDLING
DECIDE:       U, S = CASE, Z          " LOWER CASE?
              N, RUS(8)               " SHIFT TO UPPER CASE BITS
                S '×' 255             " MASK OUT EXTRANEOUS BITS
EXIT:           A = RUNNUMBER
                A - 100, Z            " PRESCANO?
              N, GOTOR(MC[-1])        " EXIT NXTAPESL
                A = S
                GOTO(:PRNTIS)         " PRINT CHARACTER AND EXIT NXTTAPESL
SPECIAL:        LUS(10)               " CLEAN AWAY FLEXIS BITS
                RUS(10)               " SHIFT BACK
                S + :PARITY           " BASE ADDRESS OF JUMP TABLE
                GOTO(S)               " SWITCH TO HANDLE SPECIAL CASES
                GOTO(:RCLN)           " -7
                GOTO(:BAT)            " -6
                GOTO(:UL)             " -5
                GOTO(:LC)             " -4
                GOTO(:UC)             " -3
                GOTO(:NXTTAPESL)      " -2 SKIP ERASE, BLANK, STOP, BACKSP
                GOTO(:NONFLX)         " -1

PARITY:         A = 102               " -0  ERROR NUMBER 102
                S = RUNNUMBER
                S - 500, Z            " EXECUTION?
              Y, A + 413              " THEN INCREMENT ERROR NOS. BY 413
                SUBC(:ERRORM)         " REPORT THE ERROR
                GOTO(:NXTTAPESL)      " AND TRY THE FOLLOWING HEPTAD
RCLN:           A = RUNNUMBER
                A - 100, Z            " PRESCANO?
              N, GOTO(:S119)
                S = COUNT
                MC = S                " SAVE COUNT
                SUBC(:NLCR)           " NEW LINE ON LINE PRINTER
                G = LINECOUNTER
                F + 1
                A = 4
                SUBC(:AFXT)           " PRINT NEW LINE NUMBER IN 4 DIG.
                SUBC(:TAB)            " TAB THE LINE PRINTER
                S = MC[-1]
                COUNT = S             " RESTORE COUNT
S119:           S = 119               " CARRIAGE RETURN
                GOTOR(MC[-1])         " EXIT NXTTAPESL
```

```
BAT:        S = 118             " TAB
            GOTO(:EXIT)
UL:         S = 32638           " UNDERLINE OR VERTICAL BAR
            GOTO(:DECIDE)       " DEPENDING ON CURRENT CASE
LC:         S = 0               " LOWER CASE
UC:         CASE = S            " STORE CASE
            GOTO(:NXTTAPESL)    " AND GET NEXT HEPTAD
NONFLX:     A = 103             " ERROR NUMBER 103
            GOTO(:PARITY[1])
            'END' NXTTAPESL

TAB:        A = PRPOS           " POSITION ON LINE
            A 'X' -7
            A + 9               " NEXT TAB POSITION
TAB3:       PRPOS = A
            A - 145, P          " LINE OVERFLOW?
          N, GOTOR(MC[-1])      " EXIT TAB AND SPACE
            A + 1
            MC = A
            SUBC(:NLCR)         " START NEW LINE
            A = MC[-1]
            GOTO(:TAB3)

SPACE:      SUBC(:RND)          " ROUND PARAMETER
            A = G, P            " POSITIVE?
          N, A = 0              " NEGATIVE MEANS 0
            A + PRPOS
            GOTO(:TAB3)

AFXT6:      A = 6               " NN = 6
AFXT:       S = 0               " MM = 0
            ABS = S             " UNSIGNED VERSION
            SUBC(:FIX1)         " FIXED-POINT CONVERSION
            GOTO(:PRO)          " PRINT IT OUT

            'END'               " LINE PRINTER SECTION

            'END'               " PICO, ETC.
```

```
'BEGIN'          bcheck, begin of program, dp0, instr cntr, begin of text array,
                 text array pointer, end of text array, nlp, BEG CAT, END CAT,
                 TEST IC, TEST POINTERS, CONSIDER


SYSTEM[0]:

bcheck:                  'SKIP' 1
begin of program:        'SKIP' 1
dp0:                     'SKIP' 1
instr cntr:              'SKIP' 1
begin of text array:     'SKIP' 1
text array pointer:      'SKIP' 1
end of text array:       'SKIP' 1
nlp:                     'SKIP' 1


TEST IC:
TEST POINTERS:     'BEGIN'  LOOP, STACK TEST, NON SHIFT, ERM

                      G = run number
                      F - 300, Z              " run number = 300 ?
                   Y, A = text array pointer
                   N, A = begin of text array
                      M[6] = - A              " - lower bound
                      S = - end of text array
                      M[6] - S                " + upper bound
                      A - instr cntr
                      S + nlp
                   U, A -  20, P              " enough space for instructions ?
                   N, JUMP (1)
                   U, S -  20, P              " enough space for identifiers ?
                   Y, GOTO (:STACK TEST)
                      A +   S
                   U, A -  60, P              " does shift make any sense ?
                   N, GOTO (:NON SHIFT)
                      RUA (1)
                      S -   A                 " amount of shift
                      M[B] = S, P
                      A = end of text array
                      end of text array + S
                      text array pointer + S
                      begin of text array + S
                   N, A - M[6]
LOOP:                 F = :MA
                      S =  MG
                      G + M[B]
                      MG =  S
                   Y, A -   1
                   N, A +   1
                      REP6E (:LOOP)
STACK TEST:           S = end of stack
                      S - :MC[20], P
                   Y, S = last symbol
                   Y, GOTOR (MC[-1])
                      A = 491
                      GOTO (:ERM)
NON SHIFT:            F + 200, Z              " run number = 100 ?
                   Y, S = text in array, Z    " ∧ text in text array ?
```

```
                Y, text in array = B    " then text in text array = false
                Y, S = begin of text array
                Y, end of text array = S
                Y, GOTO (:TEST POINTERS)
                   A = 492
ERM:               SUBC (:ERRORM)
                   GOTO (:END RUN)      " 46 instructions

                   'END' TEST POINTERS


CONSIDER:          S = permit, Z        " can the compiler be overwritten ?
                Y, permit = B           " then it can't be done a second time
                Y, S = :length of compiler
                Y, bcheck + S
                U, B - bcheck, P        " stack growth still dangerous ?
                N, presence = - B       " else note the absence of the compiler
                N, GOTO (LINK[2])       " and continue execution
                   B = begin of stack
                   A = 609              " end of execution
                   SUBC (:ERRORM)       " 10 instructions
```

```
" running system

RUN VAR[0]:

'BEGIN'         stock, stock1, stock2, stock3, begin objp, end objp,
                loose string, ref, ref1, srt, nbr, ws, ws1, action

stock:                  'SKIP' 1
stock1:                 'SKIP' 1
stock2:                 'SKIP' 1
stock3:                 'SKIP' 1
begin objp:             'SKIP' 1
end objp:               'SKIP' 1
loose string:           'SKIP' 1
ref:                    'SKIP' 1
ref1:                   'SKIP' 1
srt:                    'SKIP' 1
nbr:                    'SKIP' 1
ws:                     'SKIP' 1
ws1:                    'SKIP' 1
action:                 'SKIP' 1                " 14 variables
```

```
RUN SYST[0]:

'BEGIN'         SCHOLTEN, HALF, ERRORTABLE, ENTRIS, ENTRIS4, EXITIS, ENTRPB,
                ENTRB, DPTR, EXITP, CEN, CLPN, CRV, CRV3, CIV, CIV5, CBV, CLV,
                IAD, BAD, RAD, JOINT AD, RAD35, LAD, IND, INDB, INDU, TFSU,
                TSL, TFSL, TFSL1, TASR, TASI, TASB, TASB2, TASST, TASU, STSR,
                STSR4, STSI, SSTSI, STSB, STSB3, STFSU, FAD, STASR, STASI,
                STASB, STASST, STASU, TRSCV, TISCV, TSCVU, FADCV, JUA, REJST,
                TIAV, TAV, TAV1, IDI, TTP, TCST, STST, STSST, STSST4, CSTV,
                STAD, ORAD, OIAD, OBAD, OSTAD, EXITPC, FREE CHAIN, DECREASE,
                DECREASE3, PUNLCR, PUSPACE, RUNOUT, REHEPG, PUHEPG, ENTIER,
                SQRT, LN, EXP, EXP1, COS, SIN, ARCTAN


SCHOLTEN:       + 12288
                +     0
HALF:           - 65535
                +     1             " 4 locations


ERRORTABLE:     'BEGIN'  END OF TABLE

                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)

                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)

                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)
                GOTO (:END OF TABLE)

END OF TABLE:   A = M[B-1]            " link
                A = MA[-1]            " call
                A 'X' 32767
                A - :ERRORTABLE       " isolate error number
                A + 500
                GOTO (:ERRORM)        " 21 instructions

                'END' ERRORTABLE


ENTRIS:         A =   D
                MC =  A               " save dp
                S = MS[1]
                D =   S               " new value of dp = APIC[1]
ENTRIS4:     U, B - bcheck, P         " dangerous growth of stack ?
             N, GOTO (LINK[2])
                GOTO (:CONSIDER)      " 7 instructions
```

```
EXITIS:         S = MC[-1]
                D =  S              " reset dp
                GOTO (MC[-1])       " 3 instructions

ENTRPB:         D =  A              " new value of dp
ENTRB:          MA = B              " wp
                MG = A              " pp
              U, B - :MA[255], P    " still more pp's ?
              N, GOTO (:ENTRIS4)    " else test growth of stack
                F + 1
                A + 512
                GOTO (:ENTRB[1])    " 8 instructions

DPTR:           'BEGIN' DOUBLE, TEST, END

                G =  D
                MC = F              " dp[call] in linkdata
              Y, GOTO (:END)        " no transport in case length = 0
                stock = A
                GOTO (:TEST)
DOUBLE:         S + 2
                F = MS
                MC = F              " double transport
TEST:           A + 2, P
              N, GOTO (:DOUBLE)
                A - 2, Z
              N, A = MS[2]
              N, MC = A             " single transport
                A = stock
END:            A + :MC[-2]         " construct pp in A
                GOTO (LINK)         " 16 instructions

                'END' DPTR

EXITP:          B = :MD             " pp
                S = M[B+1]
                D =  S              " reset dp
                GOTO (MC[-1])       " 4 instructions

CEN:            'BEGIN' FORMAL, MASK0, MASK15

                S =  1
                PLUSS (MA[-1])      " increase link by 1
                S = MS[-1]          " take APD
                F = :MS             " isolate address
                S 'X' - 32767       " isolate code
                S + :MD             " code + (cleared) dp
                MC[1] = S           " together make APIC[1]
                RUS (15)
              U, S 'X' 8, Z         " static address ?
              Y, JUMP (2)
                G + MASK15
                DO (G)              " reduce dynamic address to static one
                RUS (5)             " isolate ordinal number
              U, S 'X' 48, Z        " actual parameter simple ?
```

```
                N, S = - 1              " else select SUBC (:M[0])
                U, S -  15, Z           " passed on formal ?
                Y, GOTO (:FORMAL)
                   S + :MASK0
                   G +  MS              " select function part of APIC[0]
                   MC[-1] = G           " APIC[0]
                   GOTO (LINK)
FORMAL:            S = MG[1]            " APIC[1]
                   G = MG               " and APIC[0]
                   M[B] = S             " copied
                   MC[-1] = G
                   GOTO (LINK)
                   SUBC (:M[0])         " complicated actual parameters
MASK0:             F = M[0]             " real variable
                   G = M[0]             " integer variable
                   S = M[0], P          " Boolean variable
                   A = M[0]             " string variable
                   F = M[0]             " real constant
                   G = M[0]             " integer constant
                   S =   0, Z           " logical value
                   F =   0              " small integer constant
                   A = M[0]             " real array identifier
                   A = M[0]             " integer array identifier
                   A = M[0]             " Boolean array identifier
                   A = M[0]             " string array identifier
                   A = :M[0]            " switch identifier
                   F = - 0              " negative small integer constant
                   A = :M[0]            " label
MASK15:            F = :M0[-256]        " 43 instructions

                   'END' CEN

CLPN:              'BEGIN' LIST, STAI, STABO, STAST, MASK

                   SUB (:CEN)
                   S = M[B-1]           " APIC[1]
                   RUS (20)             " isolate ordinal number
                U, S '×' 12, Z          " actual parameter assignable ?
                N, SUBC (:ERRORTABLE[0])
                U, S -   3, P           " actual parameter complicated ?
                   G =   S
                   S = M[B-2]           " APIC[0]
                Y, S +   2              " if so, increase by 2
                Y, F -  12
                N, S '×' 32767          " if not, isolate address and
                N, S + MASK             " add A = :M[0] as function part
                   MC =  S              " APIC[2]
                   F +   0, Z           " ordinal number = 0 ? real variable ?
                Y, S '+' LIST           " if so, use M[0] = F as function part
                N, F + :LIST
                N, S =  MG              " if not, select instruction from list
                   MC =  S              " APIC[3]
                   GOTO (LINK[1])
LIST:              F ×   0              " (F × 0) '+' (A = :x) -> (x = F)
                   SUBC (:STAI)
                   SUBC (:STABO)
                   SUB2 (:STAST)
                   SUB3 (:STASR)
```

```
                SUB3 (:STASI)
                SUB3 (:STASB)
                SUB3 (:STASST)
STAI:       U, S =  F, Z        " already integer ?
            N, SUBC (:RND)
               S = MS[-1]       " take address from APIC[2]
               MS = G
               GOTO (MC[-1])
STABO:         G =  T           " sign bit of T contains condition
               S = MS[-1]       " take address from APIC[2]
               MS = G
               GOTO (MC[-1])
STAST:         S = MS[-1]       " take address from APIC[2]
               F = :MS          " copy in G
               GOTO (:STST)
LIST[20]:      SUB3 (:STASU)
MASK:          A = :M[0]        " 41 instructions

               'END' CLPN


CRV:           MC = A           " save pp
               SUB (:CEN)
               DOS (M[B-2])     " execute APIC[0]
CRV3:          A = MC[-3]       " reset pp in A
               S = MC[-3]       " take link in S
               M[B-2] = F       " overwrite APIC by real/label value
               GOTO (:MS)       " 7 instructions


CIV:           MC = A           " save pp
               SUB (:CEN)
               DOS (M[B-2])     " execute APIC[0]
               S =  F, Z        " already integer ?
            N, SUBC (:RND)
CIV5:          B -  1
               A = MC[-2]       " reset pp in A
               S = MC[-2]       " take link in S
               M[B-1] = G       " overwrite APIC[0] by int/Bo value
               GOTO (:MS)       " 10 instructions


CBV:           MC = A           " save pp
               SUB (:CEN)
               DOS (M[B-2])     " execute APIC[0]
               G =  T           " sign bit of T contains condition
               GOTO (:CIV5)     " 5 instructions


CLV:           MC = A           " save pp
               SUB (:CEN)
               DOS (M[B-2])     " execute APIC[0]
               F =  MA          " take label value
               GOTO (:CRV3)     " 5 instructions


IAD:           ref = A          " save address of array key
               A =  5           " 4 × delta[0] + 1
               GOTO (:JOINT AD) " 3 instructions
```

```
BAD:            ref = A             " save address of array key
                A =  6              " 4 × delta[0] + 2
                GOTO (:JOINT AD)    " 3 instructions

RAD:            'BEGIN' LOOP, LOOP1, LOOP2

                ref = A             " save address of array key
                A =  8              " 4 × delta[0] + 0
JOINT AD:       srt = A
                nbr = G             " number of arrays
                COUNT = S           " dimension
                LUS (2)
                B - :MS             " B - 4 × dimension
                RUA (2)             " isolate delta[0]
                S = :MA
                A -  2, Z
              N, A =  0             " form - 0 or + 0 for delta[0] = 2 or 1
                F = - M[B]          " save l[0]
                MC = A
                A = :MC[-1], Z      " condition NO
LOOP:         Y, F = - MA           " l[i]
              U, S =  F, Z          " already integer ?
              N, SUBC (:RND)
                F +  1
                MC = - G            " l[i] - 1 constructed, avoiding - 0
                F = MA[2]           " u[i]
              U, S =  F, Z          " already integer ?
              N, SUBC (:RND)
              U, S =  G, Z
              Y, F =  0             " avoid - 0
                MC = G              " u[i] ready
                G - M[B-2], P       " u[i] - l[i] + 1 > 0 ?
              N, SUBC (:ERRORTABLE[1])
                G ×  S              " × delta[i]
                MC = G              " delta[i+1] = (u[i]-l[i]+1) × delta[i]
                S =  G
                A +  4
                REPP (:LOOP)
                A - :MC[-1]         " reconstruct dimension
                MC = A              " ar[-2] = dimension
                A = srt
RAD35:          MC = A              " ar[-1] = array type
LOOP1:        U, A -  6, Z          " Boolean array ?
                RUA (2)             " isolate delta[0]
                F = :MC[1]          " construct address of first element
              Y, F × 27
                G +  S              " + delta[n]
                G -  A              " - delta[0] -> address of last element
                A = ref
                MA = B              " array key = address of ar[0]
                MC = G              " ar[0] = address of last element
              N, JUMP (3)           " calculate address of first free word
                A =  0
                DIVAS (27), Z       " for Boolean arrays: 27 elements/word
              N, S + 1
                B + S               " address of first free word ready
              U, B - bcheck, P      " dangerous growth of stack ?
              Y, SUB2 (:CONSIDER)
```

```
                  A =   1
                  nbr - A, P          " (still) more arrays ?
               N, GOTO (LINK[3])
                  PLUSA (ref)
                  S = MA[-1]          " address of previous ar[0]
                  A = MS[-2]          " dimension
                  COUNT = A
                  S - :MA[3]
                  S - :MA
                  S - :MA             " address of previous ar[-3xdimension-3]
LOOP2:            F =  MS
                  MC =  F
                  A = MS[2]
                  MC =  A
                  S +   3
                  REPE (:LOOP2)
                  G = MS[-3]          " previous ar[-3] = delta[n]
                  S =   G
                  GOTO (:LOOP1)       " 71 instructions

                  'END' RAD


LAD:              'BEGIN'  AGAIN

                  LUS (18)            " display level x d18
                  S + :MD             " (cleared) dp added
                  G = LINK
AGAIN:            A =  MG, P          " take address of label. Any more ?
                  MC[1] = S
               Y, MC[-1] = A
                  F +   1
               Y, GOTO (:AGAIN)
                  MC[-1] = - A
                  GOTO (:MG)          " 10 instructions

                  'END' LAD


IND:              'BEGIN'  LOOP

                  S = :MA             " address ar[0]
               U, S =   F, Z          " (last) index already integer ?
               N, SUBC (:RND)
                  A =   G
               U, A - MS[-5], P
                  A - MS[-4], E       " outside [lowerbound : upperbound] ?
               Y, SUBC (:ERRORTABLE[2])
                  G = MS[-6], P       " take delta[n-1]
               N, LUA (1)             " if needed, take 2 into account
                  F +   0, Z          " n = 1 ?
               Y, A +  MS             " + ar[0], i.e. + address of last elmnt
               Y, GOTO (LINK)
                  G x   A             " (index[n-1] - u[n-1]) x delta[n-1]
                  G +  MS             " + ar[0], i.e. + address of last elmnt
LOOP:             ws =  G
                  S -   3             " i = i - 1
                  F = MC[-2]          " index[i]
               U, S =   F, Z          " already integer ?
```

```
                N, SUBC (:RND)
                   A =   G
                U, A - MS[-5], P
                   A - MS[-4], E        " outside [lowerbound : upperbound] ?
                Y, SUBC (:ERRORTABLE[3])
                   G = MS[-6], P        " take delta[i]
                N, LUA (1)              " if needed, take 2 into account
                   F +   0, Z           " i = 0 ?
                Y, A +  ws
                Y, GOTO (LINK)
                   G ×   A              " (index[i] - u[i]) × delta[i]
                   G +  ws
                   GOTO (:LOOP)         " 31 instructions

                   'END' IND


INDB:              SUB (:IND)
                   S =   A
                   A =   0
                   DIVAS (27)           " for Boolean arrays: 27 elements/word
                   F = :MC              " save B into G
                   B = :MA              " remainder from division into B
                   A = :MS              " address of word containing required
                   S =   1              "                              bit into A
                   LCS (B)              " shift bit in S to the right position
                   B = :MG              " restore B
                   F =   2              " for the sake of INDU
                   GOTOR (LINK[1])      " 12 instructions


INDU:              'BEGIN' SWITCH

                U, A - endobjp, P
                U, A - beginobjp, E     " A outside program range ?
                N, GOTO (:SWITCH)
                   S = MA[-1]           " ar[-1] contains the array type
                   S '×' 3              " isolate the type
                U, S -   2, Z           " Boolean array ?
                Y, GOTO (:INDB)
                   ws1 = S              " save the array type
                   SUB (:IND)
                   G = ws1              " deliver array type in F
                   GOTO (LINK[1])
SWITCH:            stock3 = S           " save the address of APIC[0] for TFSL
                   S = LINK[2]          " save link of TFSU because of danger of
                   MC =  S              "                              recursion
                   S = LINK[1]          " save link of INDU because of danger of
                   MC =  S              "                              recursion
                   GOTO (:TFSL1)        " 17 instructions

                   'END' INDU
```

```
TFSU:           'BEGIN'  R, I, BO, ST, L

                SUB1 (:INDU)
                F +  :R
                DO (MG)              " execute selected instruction
                GOTO (LINK[2])
R:              F =  MA              " TSR
I:              G =  MA              " TSI
BO:             S 'X' MA, Z          " TSB
ST:             A =  MA              " TSST
L:              GOTO (MC[-1])        " 9 instructions

                'END' TFSU


TSL:               S =   F, Z        " index already integer ?
                N, SUBC (:RND)
                   S =   G
                U, S -   MA, P       " index > n ?
                N, S -   0, E        " or < 1 ?
                Y, SUBC (:ERRORTABLE[4])
                   A +   G
                   DO (MA)           " execute selected SWORD
                   GOTO (MC[-1])     " 9 instructions


TFSL:           stock3 = S           " save the address of APIC[0]
TFSL1:          S =   D
                MC =  S              " save dp
                S = stock3
                S = MS[1]
                D =   S              " new value of dp = APIC[1]
                SUBC (:TSL)
                F =   4              " for the sake of TFSU
                GOTO (:EXITIS)       " 9 instructions


TASR:           SUB (:IND)
                F = MC[-2]
                D =   G              " restore dp saved by ENTRIS
                F =   MA             " TSR
                GOTO (MC[-1])        " 5 instructions


TASI:           SUB (:IND)
                F = MC[-2]
                D =   G              " restore dp saved by ENTRIS
                G =   MA             " TSI
                GOTO (MC[-1])        " 5 instructions


TASB:           SUB1 (:INDB)
                S 'X' MA, Z          " TSB
TASB2:          F = MC[-2]
                D =   G              " restore dp saved by ENTRIS
                GOTO (MC[-1])        " 5 instructions
```

```
TASST:          SUB (:IND)
                A = MA                  " TSST
                GOTO (:TASB2)           " 3 instructions

TASU:           S = stock3              " stock3 got its value by the macro SAS
                SUB2 (:TFSU)
                S = MC[-1]
                D =  S                  " restore dp saved by ENTRIS
                B -  1                  " remove extra link
                GOTO (MC[-1])           " 6 instructions

STSR:           stock = F
                A = MC[-1]              " address of ar[0]
                F = MC[-2]              " index[n-1]
                SUB (:IND)
STSR4:          F = stock
                MA =  F                 " STR
                GOTO (LINK[2])          " 7 instructions

STSI:           S =  F, Z               " already integer ?
              Y, GOTO (:SSTSI)
                F + SCHOLTEN
                F - SCHOLTEN
                S =  F, Z               " has the rounding been successful ?
              N, SUBC (:ERRORTABLE[5])
SSTSI:          stock1 = G
                A = MC[-1]              " address of ar[0]
                F = MC[-2]              " index[n-1]
                SUB (:IND)
                G = stock1
                MA =  G                 " SSTI
                GOTO (LINK[2])          " 13 instructions

STSB:           A = MC[-1]              " address of ar[0]
                F = MC[-2]              " index[n-1]
                SUB1 (:INDB)            " INDB does not disturb the condition
STSB3:          stock3 = S
                S = - S
                S '×' MA                " clear old value
              N, S '+' stock3           " substitute new value
                MA =  S
                GOTO (LINK[2])          " 9 instructions

STFSU:          'BEGIN'  R, I, BO, ST, L, INTEGER

                stock = F
                stock2 = A
                A = MC[-1]              " address of ar[0]
                F = MC[-2]              " index[n-1]
                SUB1 (:INDU)
                JUMP (G)                " switch over the array type
R:              GOTO (:STSR4)
I:              GOTO (:INTEGER)
BO:             GOTO (:STSB3)
```

```
ST:             GOTO (:STSST4)
L:              SUBC (:ERRORTABLE[6])
INTEGER:        F = stock
                S =   F, Z          " already integer ?
              N, SUBC (:RND)
                MA =  G             " SSTI
                GOTO (LINK[2])      " 16 instructions

                'END' STFSU


FAD:            MC =  F             " STACK
                MC =  A             " STAA
                A = MS[1]
                D =   A             " restore dp saved by ENTRIS
                GOTO (MS[-1])       " return over link of APIC[2] instructn
                                    " 5 instructions


STASR:          SUB2 (:STSR)
                B -   3             " remove 2 old links and a dp
                GOTO (LINK[3])      " 3 instructions


STASI:          SUB2 (:STSI)
                B -   3             " remove 2 old links and a dp
                GOTO (LINK[3])      " 3 instructions


STASB:          SUB2 (:STSB)
                B -   3             " remove 2 old links and a dp
                GOTO (LINK[3])      " 3 instructions


STASST:         SUB2 (:STSST)
                B -   3             " remove 2 old links and a dp
                GOTO (LINK[3])      " 3 instructions


STASU:          SUB2 (:STFSU)
                B -   3             " remove 2 old links and a dp
                GOTO (LINK[3])      " 3 instructions


TRSCV:          SUB (:IND)
                F =  MA             " TSR
                B -   1             " remove extra link
                GOTO (MC[-1])       " 4 instructions


TISCV:          SUB (:IND)
                G =  MA             " TSI
                B -   1             " remove extra link
                GOTO (MC[-1])       " 4 instructions
```

```
TSCVU:          'BEGIN'  R, I, BO, ST, L

                SUB1 (:INDU)
                F +  :R
                DO (MG)              " execute selected instruction
                B -   1              " remove extra link
                GOTO (MC[-1])
R:              F =  MA              " TSR
I:              G =  MA              " TSI
BO:             SUBC (:ERRORTABLE[7])
ST:             SUBC (:ERRORTABLE[8])
L:              SUBC (:ERRORTABLE[9]) " 10 instructions

                'END' TSCVU


FADCV:          MC =  F              " STACK
                MC =  A              " STAA
                GOTO (MS[-1])        " 3 instructions


RND:            F + SCHOLTEN
                F - SCHOLTEN
              U, S =   F, Z          " has the rounding been successful ?
              Y, GOTO (MC[-1])
                SUBC (:ERRORTABLE[10]) " 5 instructions


JUA:            S = MA[1]
                D =   S              " new value of dp
                RUS (18)             " isolate display level
                S +   D
                S =  MS              " take pp
                B =  MS              " B = wp
                A =  MA
                MC =  A              " create pseudo link
                GOTO (:DECREASE)     " 9 instructions


REJST:          S = loose string, P  " does A contain a string reference ?
              N, GOTO (MC[-1])
                loose string = - S
                S =  MA              " take porch[0]
                GOTO (:DECREASE3)    " 5 instructions


TIAV:           F = - 0
                GOTO (:TAV1)         " 2 instructions


TAV:            'BEGIN'  LOOP, LOOP1

                F =   1
TAV1:           ref = A              " save the address of array key
                DO (MA)              " execute APIC[0]
                ref1 = A             " save address of ar[0] of actual array
                S = MA[-1]           " ar[-1] contains the array type
                S '×' 3              " isolate the type
                S -   G, Z           " special activity required ?
```

```
                  action = S
                  G = MA[-2]
                  COUNT = G          " dimension
                  F × 3
                  A - :MG            " construct address of ar[- 3 × dimension]
                  S = MA[-3]         " form - 0 or + 0 for delta[0] = 2 or 1
              Y, MC = - S
              N, MC = S              " delta[0] adapted
LOOP:             F = MA[-2]
                  MC = F             " u[i] and l[i] - 1 copied
                  S = MA
              N, JUMP (3)
                  G = action, P      " to be halved ?
              Y, RUS (1)
              N, LUS (1), P          " else to be doubled (and reset condition)
                  MC = S             " delta[i] adapted
                  A + 3
                  REPP (:LOOP)
                  G = MA[-2]
                  MC = G             " dimension copied
                  A = MA[-1]
                  A '×' 15
              Y, A '+' 13            " array type adapted
                  nbr = - A          " just for 1 array
                  SUB3 (:RAD35)      " reserve space in the stack
                  A = ref1
                  S = MA[-1]         " ar[-1] of actual array
                  S '×' 15           " contains the array type
              U, S - 6, Z            " Boolean array ?
                  RUS (2)            " isolate delta[0]
                  S + MA             " construct address of elmnt after the last
                  S - MA[-3]         " construct address of first element
              Y, A = 0
              Y, DIVAS (27)
                  A = B              " save B in A
                  B = ref
                  B = M[B]           " array key contains address ar[0]
                  B + 1              " construct address of first element
LOOP1:            G = MS             " take element of actual array
              U, S = action, Z       " special activity required ?
              N, JUMP (6)            " else just copy
              U, S = - 0, E          " to be doubled ?
              Y, MC = F              " then store into 2 words
              Y, JUMP (4)
                  F = MS             " halve: take 2 words
                  SUBC (:RND)        " round to integer
                  S + 1
                  MC = G             " store into 1 word
                  S + 1
              U, B - A, Z            " ready ?
              N, GOTO (:LOOP1)
                  MD = B             " new value of wp
                  GOTO (LINK[4])     " 60 instructions

                  'END' TAV
```

```
IDI:            F +   0
                MC =  F, P
              U, S = - F, E          " divisor not integer ?
              N, F =   0
              N, F + MC[-5], P
              U, S = - F, E          " ∨ dividend not integer ?
              Y, SUBC (:ERRORTABLE[11])
                 F /  MC, P
              N, F = - F             " take absolute value of quotient
                 SUBC (:ENTIER)
              N, F = - F             " entier with correct sign
                 GOTOR (MC[+1])      " 12 instructions

TTP:             'BEGIN' ODD TEST, LN MUL EXP, END, MUL, CYCLE, END1

                 F +   0, Z          " exponent = 0 ?
              Y, F =   1             " then result = 1
              Y, GOTO (:END)
                 MC =  F
                 F = M[B-5]          " take base
                 F +   0, Z          " = 0 ?
              Y, S = M[B-1], P       " and exponent > 0 ?
              Y, F =   0             " then result = 0
              Y, GOTO (:END1)
                 S = - M[B-1], P
              Y, S = - S
                 A = - M[B-2], E     " exponent integer ?
              N, GOTO (:LN MUL EXP)
              U, S +  32, P          " abs (tail of exponent) ≤ 31 ?
              Y, A +   0, Z          " ∧ head of exponent = 0 ?
              Y, GOTO (:MUL)
              U, S = - G, P          " base negative ?
              Y, F = - F             " then invert sign
ODD TEST:     Y, RUA (15), Z         " exponent small enough to be odd ?
              Y, RCS (1), P          " if so, exponent odd indeed ?
LN MUL EXP:      SUBC (:LN)          " ln (base)
                 F × MC[-2]          " × exponent
                 SUBC (:EXP1)        " and exponential of that
              Y, F = - F             " and inversed if necessary
END:             B -   2
                 GOTOR (MC[1])
MUL:             F =   1, P
                 M[B-2] = F          " start cycle with 1 and condition YES
CYCLE:           F = M[B-5]          " base ↑ (2 ↑ i)
              N, F ×   F             " becomes (except for the first time)
              N, M[B-5] = F          " base ↑ (2 ↑ (i+1))
              U, S 'X' 1, Z          " this power of base of interest ?
              Y, F × M[B-2]
              Y, M[B-2] = F          " then incorporate it in the result
                 RUS (1), Z          " ready ?
              N, GOTO (:CYCLE)
                 A +   0, P          " was exponent originally negative ?
              Y, F =   1
              Y, F / M[B-2]          " then invert the result
END1:            B -   4
                 GOTOR (MC[1])       " 41 instructions

                 'END' TTP
```

```
RUN:            B = instr cntr
                A = :MEMORY END
                A - M[B-1]
                A -  20
                bcheck = A
                A = dp0
                D =   A              " dp
                end objp = B
                loose string = - B   " loose string = false
                A = :MC              " TBL (0)
                F = :MD[0]
                B +   1              " ENTRB (1)
                SUB2 (:ENTRB)
                A = begin of program
                begin objp = A
                GOTO (A)             " 16 instructions

TCST:           A = 600
                SUBC (:ERRORM)

STST:           A = 601
                SUBC (:ERRORM)

STSST: STSST4:  A = 602
                SUBC (:ERRORM)

CSTV:           A = 603
                SUBC (:ERRORM)

STAD:           A = 604
                SUBC (:ERRORM)

ORAD: OIAD:
OBAD: OSTAD:    A = 605
                SUBC (:ERRORM)

EXITPC:         A = 606
                SUBC (:ERRORM)

FREE CHAIN:     A = 607
                SUBC (:ERRORM)

DECREASE:       GOTO (MC[-1])

DECREASE3:      A = 608
                SUBC (:ERRORM)       " 19 instructions
```

```
" instruct list

INSTR LST[0]:     MC =  F              " DIV, STACK
                  F = MC[-4]
                  F /  MC
                  F = - F              " SUB, NEG
                  F + MC[-2]           " ADD
                  F × MC[-2]           " MUL
                  SUBC (:IDI)          " IDI
                  SUBC (:TTP)          " TTP
                  F - MC[-2], Z        " EQU, UQU, TEST1
                  S = - T, P           " NON

                  F - MC[-2], P        " LES
               Y, F -   0, P
                  F - MC[-2], P        " MST, MOR
               N, F =   F, Z
                  S = - T, P
                  F - MC[-2], Z        " LST
               N, S = - 1, E
                  S =  T               " STAB
                  MC = S
                  S =  T, P            " QVL

                  S = MC[-1], E
               Y, B -  1               " IMP
               N, S = - MC[-1], P
               Y, B -  1               " OR
               N, S = MC[-1], P
               N, B -  1               " AND
               Y, S = MC[-1], P
                  MC = A               " STAA
                  SUB (:IND)           " TSR
                  F =  MA

                  SUB (:IND)           " TSI
                  G =  MA
                  SUB1 (:INDB)         " TSB
                  S 'X' MA, Z
                  SUB (:IND)           " TSST
                  A =  MA
                  SUB2 (:TFSU)         " TFSU
                  SUBC (:TSL)          " TSL
                  SUBC (:TFSL)         " TFSL
                  SUBC (:TCST)         " TCST

                  SUB2 (:STSR)         " STSR
                  SUB2 (:STSI)         " STSI
                  SUB2 (:SSTSI)        " SSTSI
                  SUB2 (:STSB)         " STSB
                  SUB2 (:STSST)        " STSST
                  SUB2 (:STFSU)        " STFSU
                  SUB2 (:ENTRIS)       " ENTRIS
                  S = MS[1]            " TFD
                  stock3 = S           " SAS
                  S -  2               " DECS
```

```
INSTR LST[50]:    GOTO (:FAD)          " FAD
                  GOTO (:TASR)         " TASR
                  GOTO (:TASI)         " TASI
                  GOTO (:TASB)         " TASB
                  GOTO (:TASST)        " TASST
                  GOTO (:TASU)         " TASU
                  GOTO (:EXITIS)       " EXITIS
                  GOTO (:FADCV)        " FADCV
                  GOTO (:TRSCV)        " TRSCV
                  GOTO (:TISCV)        " TISCV

                  GOTO (:TSCVU)        " TSCVU
                  GOTOR (MC[-1])       " EXIT
               N, F =   F, E           " TEST2
               N, F =   F, Z
                  SUBC (:CRV)          " CRV
                  SUBC (:CIV)          " CIV
                  SUBC (:CBV)          " CBV
                  SUB1 (:CSTV)         " CSTV
                  SUBC (:CLV)          " CLV
                  SUB (:CEN)           " CEN

                  SUB1 (:CLPN)         " CLPN
                  SUB4 (:TAV)          " TAV
                  SUB4 (:TIAV)         " TIAV
                  SUB3 (:RAD)          " RAD
                  SUB3 (:IAD)          " IAD
                  SUB3 (:BAD)          " BAD
                  SUB3 (:STAD)         " STAD
                  SUB3 (:ORAD)         " ORAD
                  SUB3 (:OIAD)         " OIAD
                  SUB3 (:OBAD)         " OBAD

                  SUB3 (:OSTAD)        " OSTAD
                  loose string = B     " LOS
                  GOTO (:EXITP)        " EXITP
                  GOTO (:EXITPC)       " EXITPC
                  SUBC (:REJST)        " REJST
                  GOTO (:JUA)          " JUA
                  F =   F, P           " ABS
               N, F = - F
                  F =   F, Z           " SIGN
               N, F =   1, E

               N, F = - 1
                  SUBC (:ENTIER)       " ENTIER
                  SUBC (:SQRT)         " SQRT
                  SUBC (:EXP)          " EXP
                  SUBC (:LN)           " LN
                  GOTO (:END RUN)      " END
                  F = M[0]             " TRV, TRC
                  G = M[0]             " TIV, TIC
                  F =   0              " TSIC, TNA, STST
                  SUB2 (:STST)
```

" instruct list continued

```
INSTR LST[100]:    F = - M[0]                " TNRV, TNRC
                   G = - M[0]                " TNIV, TNIC
                   F = - 0                   " TNSIC
                   F + M[0]                  " ADDRV, ADDRC
                   G + M[0]                  " ADDIV, ADDIC
                   F +   0                   " ADDSIC
                   F - M[0]                  " SUBRV, SUBRC
                   G - M[0]                  " SUBIV, SUBIC
                   F -   0                   " SUBSIC
                   F × M[0]                  " MULRV, MULRC

                   G × M[0]                  " MULIV, MULIC
                   G ×   0                   " MULSIC
                   F / M[0]                  " DIVRV, DIVRC
                   G / M[0]                  " DIVIV, DIVIC
                   F /   0                   " DIVSIC
                   F - M[0], Z               " EQURV, EQURC, UQURV, UQURC
                   S = - T, P
                   G - M[0], Z               " EQUIV, EQUIC, UQUIV, UQUIC
                   S = - T, P
                   F -   0, Z                " EQUSIC, UQUSIC

                   S = - T, P
                   F - M[0], P               " LESRV, LESRC, LSTRV, LSTRC
                N, F =   F, Z
                   S = - T, P
                   G - M[0], P               " LESIV, LESIC, LSTIV, LSTIC
                N, F =   F, Z
                   S = - T, P
                   F -   0, P                " LESSIC, LSTSIC, MORSIC
                N, F =   F, Z
                   S = - T, P

                   F - M[0], Z               " MSTRV, MSTRC
                N, S = - 1, E
                   G - M[0], Z               " MSTIV, MSTIC
                N, S = - 1, E
                   F -   0, Z                " MSTSIC
                N, S = - 1, E
                   F - M[0], P               " MORRV, MORRC
                Y, F -   0, P
                   G - M[0], P               " MORIV, MORIC
                Y, F -   0, P

                   S = M[0], P               " TBV
                   S =   0, Z                " TBC
                   A = M[0]                  " TSTV, TAK
                   A =   0                   " TLV, TSWE, TAA, JU1
                   GOTO (:MA[1])
                   M[0] = F                  " STR
                   S =   F, Z                " STI
                N, SUBC (:RND)
                   M[0] = G                  " SSTI
                   S =   T                   " STB
```

" instruct list continued

```
INSTR LST[150]:    M[0] = S
                   DOS (M[0])          " DOS
                   DOS (M[2])          " DOS2
                   DOS (M[3])          " DOS3
                   GOTO (:M[0])        " JU
                   GOTO (M[0])         " IJU
                   GOTO (M[1])         " IJU1
                N, GOTO (:M[0])        " COJU
                Y, GOTO (:M[0])        " YCOJU
                   SUBC (:M[0])        " SUBJ

                   SUBC (M[0])         " ISUBJ
                   B -  0              " DECB
                   DO (M[0])           " DO
                   A = :MC             " TBL
                   F = :MD[0]
                   B +  0              " INCRB, ENTRB
                   SUB2 (:ENTRB)
                   S =  D              " DPTR
                   '0327 77776'
                   SUB (:DPTR)

                   F = :MA[0]          " TDL
                   B +  0              " ENTRPB
                   SUB2 (:ENTRPB)
                   A + M[0]            " NIL
                   A - M[0]            " LAST
                   S =  0              " TDA, LAD
                   SUB (:LAD)
                   MO[0] = B           " SWP
                   B = MD[0]           " EXITB, EXITC
                   SUBC (:FREE CHAIN)

                   S = :MC[0]          " EXITSV
                   GOTO (MS)
                   SUBC (:SIN)         " SIN
                   SUBC (:COS)         " COS
                   SUBC (:ARCTAN)      " ARCTAN
                   SUBC (:READ)        " READ
                   SUBC (:FIXP)        " FIXP
                   SUBC (:ABSFIXP)     " ABSFIXP
                   SUBC (:FLOP)        " FLOP
                   SUBC (:PUNCH)       " PUNCH

                   SUBC (:PUNLCR)      " PUNLCR
                   SUBC (:PUSPACE)     " PUSPACE
                   SUBC (:RUNOUT)      " RUNOUT
                   SUBC (:REHEPG)      " REHEP
                   SUBC (:PUHEPG)      " PUHEP
                   SUBC (:HAND)        " HAND
                   SUBC (:XEEN)        " XEEN
                   SUBC (:STOP)        " STOP
                   S = line counter    " SLNC
                   M[2] = S
```

```
INSTR LST[200]:     S = M[2]              " RLNC
                    line counter = S
                    S =   0               " LNC
                    line counter = S
```

```
'BEGIN'          FIXT L, ABSFIXT L, FLOT L, PRINT L, NLCR L, TAB L,
                 SPACE L, PRINTTEXT L, CARRIAGE L, NEW PAGE L,
                 LINE NUMBER L, RESYM L, PUSYM L, PRSYM L, PUTEXT L,
                 ABS L, SIGN L, SQRT L, SIN L, COS L, ARCTAN L, LN L,
                 EXP L, ENTIER L, READ L, FIXP L, ABSFIXP L, FLOP L,
                 PUNCH L, PUNLCR L, PUSPACE L, RUNOUT L, REHEP L, PUHEP L,
                 HAND L, XEEN L, STOP L, TNRP, TRP, ABS P, SIGN P, SQRT P,
                 SIN P, COS P, ARCTAN P, LN P, EXP P, ENTIER P, FIXP P,
                 ABSFIXP P, FLOP P, PUNCH P, PUSPACE P, PUHEP P, HAND P,
                 XEEN P, PUTEXT, FIXT P, ABSFIXT P, FLOT P, PRINT P,
                 SPACE P, PRINTTEXT, CARRIAGE P, LINE NUMBER P, RESYM,
                 PUSYM, PRSYM, POL IN FF, POL IN F, ENTIER1

LIBRARY[0]:

FIXT L:          :FIXT P
                 :FIXT P
ABSFIXT L:       :ABSFIXT P
                 :ABSFIXT P
FLOT L:          :FLOT P
                 :FLOT P
PRINT L:         :PRINT P
                 :PRINT P
NLCR L:          :NLCR
                 :NLCR
TAB L:           :TAB
                 :TAB
SPACE L:         :SPACE P
                 :SPACE P
PRINTTEXT L:     :PRINTTEXT
                 :PRINTTEXT
CARRIAGE L:      :CARRIAGE P
                 :CARRIAGE P
NEW PAGE L:      :NEW PAGE
                 :NEW PAGE
LINE NUMBER L:   :LINE NUMBER P
                 :LINE NUMBER P
RESYM L:         :RESYM
                 :RESYM
PUSYM L:         :PUSYM
                 :PUSYM
PRSYM L:         :PRSYM
                 :PRSYM
PUTEXT L:        :PUTEXT
ABS L:           :PUTEXT
SIGN L:          :ABS P
SQRT L:          :SIGN P
SIN L:           :SQRT P
COS L:           :SIN P
ARCTAN L:        :COS P
LN L:            :ARCTAN P
EXP L:           :LN P
ENTIER L:        :EXP P
READ L:          :ENTIER P
FIXP L:          :READ
ABSFIXP L:       :FIXP P
FLOP L:          :ABSFIXP P
```

```
PUNCH L:        :FLOP P
PUNLCR L:       :PUNCH P
PUSPACE L:      :PUNLCR
RUNOUT L:       :PUSPACE P
REHEP L:        :RUNOUT
PUHEP L:        :REHEP G
HAND L:         :PUHEP P
XEEN L:         :HAND P
STOP L:         :XEEN P
                :STOP                   " 51 words

END CAT:        'SKIP' 1

                +               0
                +               0
                + 33 554 432                    " 64 x d19
                -               1
                +               0
                + 34 078 720                    " 65 x d19
                -               1
                +               0
                + 34 078 720                    " 65 x d19
                -               1
                ('222 000 000' + 4)             " 010 010 010
                (12 582 912 + :FIXT L)          " 24 x d19
                - '0 53 56 75 71'               " FIXT

                +               0
                +               0
                + 33 554 432                    " 64 x d19
                -               1
                +               0
                + 34 078 720                    " 65 x d19
                -               1
                +               0
                + 34 078 720                    " 65 x d19
                -               1
                ('222 000 000' + 4)
                (12 582 912 + :ABSFIXT L)       " 24 x d19
                - '0 00 56 75 71'
                - '0 46 47 70 53'               " ABSFIXT

                +               0
                +               0
                + 33 554 432                    " 64 x d19
                -               1
                +               0
                + 34 078 720                    " 65 x d19
                -               1
                +               0
                + 34 078 720                    " 65 x d19
                -               1
                ('222 000 000' + 4)
                (12 582 912 + :FLOT L)          " 24 x d19
                - '0 53 61 64 71'               " FLOT
```

```
+               0
+               0
+ 33 554 432                    " 64 x d19
-               1
('222 000 000' + 2)
(12 582 912 + :PRINT L)         " 24 x d19
- '0 00 00 00 71'
- '0 65 67 56 63'               " PRINT

+               0
+               0
+ 33 554 432                    " 64 x d19
-               1
('222 000 000' + 2)
(12 582 912 + :PRINT L)         " 24 x d19
- '0 00 00 00 36'
- '0 32 34 23 30'               " print

+               0
('222 000 000' + 1)
(12 582 912 + :NLCR L)          " 24 x d19
- '0 63 61 50 67'               " NLCR

+               0
('222 000 000' + 1)
(12 582 912 + :TAB L)           " 24 x d19
- '0 00 71 46 47'               " TAB

+               0
+               0
+ 34 078 720                    " 65 x d19
-               1
('222 000 000' + 2)
(12 582 912 + :SPACE L)         " 24 x d19
- '0 00 00 00 52'
- '0 70 65 46 50'               " SPACE

+               0
+               0
+ 51 904 512                    " 99 x d19
-               1
('222 000 000' + 2)
(12 582 912 + :PRINTTEXT L)     " 24 x d19
- '0 00 00 00 71'
- '0 71 71 52 75'
- '0 65 67 56 63'               " PRINTTEXT

+               0
+               0
+ 34 078 720                    " 65 x d19
-               1
('222 000 000' + 2)
(12 582 912 + :CARRIAGE L)      " 24 x d19
- '0 56 46 54 52'
- '0 50 46 67 67'               " CARRIAGE
```

```
+               0
('222 000 000' + 1)
(12 582 912 + :NEW PAGE L)    " 24 x d19
- '0 00 46 54 52'
- '0 63 52 74 65'             " NEW PAGE

+               0
('232 000 000' + 1)           " 010 011 010
(8 912 896 + :LINE NUMBER L)  " 17 x d19
- '0 00 00 52 67'
- '0 63 72 62 47'
- '0 61 56 63 52'             " LINE NUMBER

+               0
('232 000 000' + 1)
(8 912 896 + :RESYM L)        " 17 x d19
- '0 00 00 00 62'
- '0 67 52 70 76'             " RESYM

+               0
+               0
+ 34 078 720                  " 65 x d19
-               1
('222 000 000' + 2)
(12 582 912 + :PUSYM L)       " 24 x d19
- '0 00 00 00 62'
- '0 65 72 70 76'             " PUSYM

+               0
+               0
+ 34 078 720                  " 65 x d19
-               1
('222 000 000' + 2)
(12 582 912 + :PRSYM L)       " 24 x d19
- '0 00 00 00 62'
- '0 65 67 70 76'             " PRSYM

+              76
('272 000 000' + 2)           " 010 111 010
(8 388 608 + :ABS L)          " 16 x d19
- '0 00 13 14 35'             " abs

+              77
('272 000 000' + 2)
(8 912 896 + :SIGN L)         " 17 x d19
- '0 35 23 21 30'             " sign

+              79
('272 000 000' + 2)
(8 388 608 + :SQRT L)         " 16 x d19
- '0 35 33 34 36'             " sqrt

+             129
('272 000 000' + 2)
(8 388 608 + :SIN L)          " 16 x d19
- '0 00 35 23 30'             " sin
```

```
+           130
('272 000 000' + 2)
(8 388 608 + :COS L)          " 16 x d19
- '0 00 15 31 35'             " cos

+           131
('272 000 000' + 2)
(8 388 608 + :ARCTAN L)       " 16 x d19
- '0 00 00 13 30'
- '0 13 34 15 36'             " arctan

+           81
('272 000 000' + 2)
(8 388 608 + :LN L)           " 16 x d19
- '0 00 00 26 30'             " ln

+           80
('272 000 000' + 2)
(8 388 608 + :EXP L)          " 16 x d19
- '0 00 17 42 32'             " exp

+           78
('272 000 000' + 2)
(8 912 896 + :ENTIER L)       " 17 x d19
- '0 00 00 17 34'
- '0 17 30 36 23'             " entier

+           132
('272 000 000' + 1)
(8 388 608 + :READ L)         " 16 x d19
- '0 34 17 13 16'             " read

+           132
('272 000 000' + 1)
(8 388 608 + :READ L)         " 16 x d19
- '0 67 52 46 51'             " READ

+           133
('262 000 000' + 4)           " 010 110 010
(12 582 912 + :FIXP L)        " 24 x d19
- '0 53 56 75 65'             " FIXP

+           134
('262 000 000' + 4)
(12 582 912 + :ABSFIXP L)     " 24 x d19
- '0 00 56 75 65'
- '0 46 47 70 53'             " ABSFIXP

+           135
('262 000 000' + 4)
(12 582 912 + :FLOP L)        " 24 x d19
- '0 53 61 64 65'             " FLOP

+           136
('262 000 000' + 2)
(12 582 912 + :PUNCH L)       " 24 x d19
- '0 00 00 00 55'
- '0 65 72 63 50'             " PUNCH
```

```
                +           137
                ('262 000 000' + 1)
                (12 582 912 + :PUNLCR L)      " 24 x d19
                - '0 00 00 50 67'
                - '0 65 72 63 61'             " PUNLCR

                +           138
                ('262 000 000' + 2)
                (12 582 912 + :PUSPACE L)     " 24 x d19
                - '0 00 46 50 52'
                - '0 65 72 70 65'             " PUSPACE

                +           139
                ('262 000 000' + 1)
                (12 582 912 + :RUNOUT L)      " 24 x d19
                - '0 00 00 72 71'
                - '0 67 72 63 64'             " RUNOUT

                +           140
                ('272 000 000' + 1)
                (8 912 896 + :REHEP L)        " 17 x d19
                - '0 00 00 00 65'
                - '0 67 52 55 52'             " REHEP

                +           141
                ('262 000 000' + 2)
                (12 582 912 + :PUHEP L)       " 24 x d19
                - '0 00 00 00 65'
                - '0 65 72 55 52'             " PUHEP

                +           142
                ('272 000 000' + 2)
                (8 388 608 + :HAND L)         " 16 x d19
                - '0 55 46 63 51'             " HAND

                +           143
                ('272 000 000' + 2)
                (8 912 896 + :XEEN L)         " 17 x d19
                - '0 75 52 52 63'             " XEEN

                +           144
                ('262 000 000' + 1)
                (12 582 912 + :STOP L)        " 24 x d19
                - '0 70 71 64 65'             " STOP

                +             0
                +             0
                + 51 904 512                  " 99 x d19
                -             1
                ('222 000 000' + 2)           " 010 010 010
                (12 582 912 + :PUTEXT L)      " 24 x d19
                - '0 00 00 75 71'
BEG CAT:        - '0 65 72 71 52'             " PUTEXT
```

```
TNRP:       A = MC[-1]          " link TNRP
            S = MC[-1]          " link library routine
            MC = F              " last parameter
            MC = S              " link library routine
            MC = A              " link TNRP
TRP:        A = :MC[-1]
            SUBC (:CRV)
            B - 2
            GOTOR (MC[-1])      " 9 instructions

ABS P:      SUBC (:TRP)
            F = F, P
          N, F = - F
            GOTOR (MC[-1])      " 4 instructions

SIGN P:     SUBC (:TRP)
            F = F, Z
          N, F = 1, E
          N, F = - 1
            GOTOR (MC[-1])      " 5 instructions

SQRT P:     SUBC (:TRP)
            GOTO (:SQRT)        " 2 instructions

SIN P:      SUBC (:TRP)
            GOTO (:SIN)         " 2 instructions

COS P:      SUBC (:TRP)
            GOTO (:COS)         " 2 instructions

ARCTAN P:   SUBC (:TRP)
            GOTO (:ARCTAN)      " 2 instructions

LN P:       SUBC (:TRP)
            GOTO (:LN)          " 2 instructions

EXP P:      SUBC (:TRP)
            GOTO (:EXP)         " 2 instructions

ENTIER P:   SUBC (:TRP)
            GOTO (:ENTIER)      " 2 instructions

FIXP P:     SUBC (:TRP)
            SUBC (:TNRP)
            SUBC (:TNRP)
            GOTO (:FIXP)        " 4 instructions
```

```
ABSFIXP P:      SUBC (:TRP)
                SUBC (:TNRP)
                SUBC (:TNRP)
                GOTO (:ABSFIXP)      " 4 instructions

FLOP P:         SUBC (:TRP)
                SUBC (:TNRP)
                SUBC (:TNRP)
                GOTO (:FLOP)         " 4 instructions

PUNCH P:        SUBC (:TRP)
                GOTO (:PUNCH)        " 2 instructions

PUNLCR:         S = 26
                GOTO (:FLEXHP)       " 2 instructions

PUSPACE P:      SUBC (:TRP)
PUSPACE:        SUBC (:RND)          " round to integer
                COUNT = G, P         " > 0 ?
PUSPACE[2]:     S = 16
              Y, SUBC (:FLEXHP)
              Y, REPP (:PUSPACE[2])
                GOTOR (MC[-1])       " 7 instructions

RUNOUT:         F = 80
                COUNT = G
RUNOUT[2]:      S = 0
                SUBC (:FLEXHP)
                REPP (:RUNOUT[2])
                GOTOR (MC[-1])       " 6 instructions

REHEP G:        SUBC (:REHEP)
                G = S
                GOTOR (MC[-1])       " 3 instructions

PUHEP P:        SUBC (:TRP)
PUHEP G:        SUBC (:RND)          " round to integer
                S = G, Z
              Y, S = 0               " avoid - 0
                GOTO (:FLEXHP)       " 5 instructions

HAND P:         SUBC (:TRP)
                GOTO (:HAND)         " 2 instructions

XEEN P:         SUBC (:TRP)
                GOTO (:XEEN)         " 2 instructions
```

```
PUTEXT:          'BEGIN'  NEW WORD, LOOP, CALL OF TCST

                 S = :FLEXIS
                 MC = S                  " output code = flexo
                 A = :MC[-1]
                 SUB (:CEN)
                 S = MC[-1]              " APIC[1]
                 RUS (20)                " isolate copy of APD code
                 S - 23, Z               " unassignable string expression ?
                 S = MC[-1]              " APIC[0]
                 S 'X' 32767             " isolate begin address of isr
                 A = MS[1]               " take second instruction of isr
               Y, A - CALL OF TCST, Z    " is it a call of TCST ?
               N, A = 611
               N, SUBC (:ERRORM)         " else the execution is ended
NEW WORD:        S + 1
                 MC = S                  " save address of word in isr
                 A = 3
                 COUNT = A               " 3 symbols per word
                 S = MS[2]               " word
                 LCS (2)                 " shift out irrelevant bits
LOOP:            LUAS (8)                " shift next symbol into A
                 A 'X' 255
               U, A - 255, Z             " end marker ?
               N, SUBC (M[B-2])          " do the conversion and put out
               N, REPP (:LOOP)
                 S = MC[-1]              " address of word in isr
               N, GOTO (:NEW WORD)       " handle next string word
                 B - 1
                 GOTOR (MC[-1])
CALL OF TCST:    SUBC (:TCST)            " 29 instructions

                 'END' PUTEXT


FIXT P:          SUBC (:TRP)
                 SUBC (:TNRP)
                 SUBC (:TNRP)
                 GOTO (:FIXT)            " 4 instructions


ABSFIXT P:       SUBC (:TRP)
                 SUBC (:TNRP)
                 SUBC (:TNRP)
                 GOTO (:ABSFIXT)         " 4 instructions


FLOT P:          SUBC (:TRP)
                 SUBC (:TNRP)
                 SUBC (:TNRP)
                 GOTO (:FLOT)            " 4 instructions


PRINT P:         SUBC (:TRP)
                 GOTO (:PRINT)           " 2 instructions


SPACE P:         SUBC (:TRP)
                 GOTO (:SPACE)           " 2 instructions
```

```
PRINTTEXT:          S = :PRNTIS
                    GOTO (:PUTEXT[1])     " 2 instructions

CARRIAGE P:         SUBC (:TRP)
                    GOTO (:CARRIAGE)      " 2 instructions

LINE NUMBER P:      G = line number
                    GOTOR (MC[-1])        " 2 instructions

RESYM:              SUBC (:NXT TAPE SL)
                  U, S - 123, Z           " space ?
                  Y, S =  93
                  U, S - 124, Z           " or colon ?
                  Y, S =  90
                  U, S - 125, Z           " or close ?
                  Y, S =  99
                     G =  S
                     GOTOR (MC[-1])       " 9 instructions

PUSYM:              SUBC (:TRP)
                    SUBC (:RND)           " round to integer
                    A =  G, Z
                  Y, A =  0               " avoid - 0
                    GOTO (:FLEXIS)        " 5 instructions

PRSYM:              SUBC (:TRP)
                    SUBC (:RND)           " round to integer
                    A =  G, Z
                  Y, A =  0               " avoid - 0
                    GOTO (:PRNTIS)        " 5 instructions

POL IN FF:          'BEGIN' START, CYCLE

                    F X  F
POL IN F:           M[B] = F
                  U, GOTO (:START)        " clear OF
START:              F =  MA               " take last coefficient
CYCLE:              F X M[B]              " X argument
                    F + MA[-2]            " + coefficient
                  U, GOTOR (MC[-1])       " if OF then exit
                    A -  2
                    GOTO (:CYCLE)

                    'END' POL IN FF       " 9 instructions

ENTIER:             F +  0, P
ENTIER1:          U, S =  F, E            " argument already integer ?
                  Y, GOTOR (MC[-1])
                     F - HALF, E          " +0 < F < .5 ?
                  Y, JUMP (4)             " then entier = + 0
                  U, S =  F, E            " F - .5 integer ?
                  Y, GOTOR (MC[-1])
```

```
                F + SCHOLTEN
                F - SCHOLTEN, Z
              Y, F =   0
                GOTOR (MC[-1])      " 11 instructions

SQRT:           'BEGIN'  D16, C0

                F -   0, P          " argument ≤ 0 ?
              N, F =   0            " then result = + 0
              N, GOTOR (MC[-1])
                MC =  F             " preserve x
                A =   F
                RUA (15)            " isolate
                MC =  A             " and preserve binary exponent
                A =   F
                S =   G
                F = :MC             " preserve stack pointer
                A 'X' 32767         " isolate mantissa in AS
                NORAS
                B = - :MC
                B -  26
                S =   B             " binary exponent + 2 in S
                B = :MG             " restore stack pointer
                S + MC[-1], P       " form complete binary exponent + 2
              Y, S +   1
              U, S 'X' 1, Z         " was it odd ?
                S 'X' - 1
                MC =  S
              Y, RUA (1)
                MC =  A
                RUA (3)             " estimate sqrt (a):
                MC =  A
                RUA (2)
                A +  C0
                M[B-1] + A          " x0 = .3657 + (5/8) x a
                A = M[B-2]
                RUA (4)
                DIVA (M[B-1])       " first Newton step:
                M[B-1] + S          " x1 = (a/x0 + x0) / 2
                A = MC[-2]
                RUA (2)
                DIVA (MC)           " second Newton step:
                S + M[B+1]          " x2 = (a/x1 + x1) / 2
                A = MC[-1]          " take binary exponent
                LUAS (14), P        " transform 2 x x2 into floating point
                F = MC[-2]          " restore x
                F /   A
                A - D16, E
              N, A - 32767          " AS = x2/2
                F +   A
                GOTOR (MC[-1])
D16:            '000 200 000'
C0:             + 61 35102          " .09142

                'END'  SQRT         " 46 locations
```

```
LN:             'BEGIN' CYCLE, LIST, LN X, LN Y, C1, C3, C5,
                        LN2, ONE, GIANT, BIN EXP 40

                F - 0, P            " argument > 0 ?
              N, F = - GIANT
              N, GOTOR (MC[-1])     " else deliver - infinity
                A = F
                RUA (15)            " isolate
                MC = A              " and preserve binary exponent
                A = F
                A '×' 32767         " replace binary exponent
                A + BIN EXP 40      " by 40
                S = G
                F = A
                F + 0               " standarize
                A = F               " and isolate anew
                RUA (15)            " binary exponent
                M[B-1] + A          " add it to former binary exponent
                A = F               " replace
                A '×' 32767         " binary exponent by 0
                S = G
                F = A               " and consider 40 bits of F as fraction
                LUAS (12)
                S = 0               " analyse first 27 bits of fraction
                M[B] = A
CYCLE:          RUA (3)             " in order to
                PLUSA (M[B]), P
              Y, S - 1              " bring fraction
              Y, GOTO (:CYCLE)
                A = :LIST           " in range sqrt (8/9) < f < sqrt (9/8)
                A - S
                G × MA              " multiply fraction by appropriate power of 9/8
                MC = F              " and preserve resulting fraction
                F + ONE
                MC = F              " compute
                F = ONE             " f1 =
                F - MC[-4]          " (1 - f) /
                F / MC              " (1 + f)
                MC = F              " preserve f1
                F × F
                MC = F
                F × C5              " compute
                F + C3              " polynomial
                F × MC[-2]          " in
                F + C1              " f1 square
                F × MC[-2]          " result × f1
                MC = F
                G = S, Z
              N, F × LN Y           " appropriate multiple of ln (8/9)
                F + MC[-2]          " added
                F + LN X            " and (1/2) × ln (9/8) added
                MC = - F
                G = MC[-3], Z       " take binary exponent
              N, F × LN2            " × ln (2)
                F + MC[-1]
                GOTOR (MC[-1])
LIST:           + 32768             "          2↑15
                + 36864             " (9/8)  × 2↑15
                + 41472             " (9/8)↑2 × 2↑15
```

```
                +     46656          " (9/8)↑3 × 2↑15
                +     52488          " (9/8)↑4 × 2↑15
                +     59049          " (9/8)↑5 × 2↑15
LN X:           -  14 59121
                +  38 95639          " ln (9/8) / 2 = +.588915178282₁₀-1
LN Y:           +  14 26353
                -  38 95640          " ln (8/9) = -.117783035656
C1:             -  12 69759
                +         6          " +.200000000002₁₀+1
C3:             -  13 32565
                + 445 32834          " +.666666478939
C5:             -  13 63134
                + 266 81432          " +.400433275889
LN2:            -  13 32131
                + 351 25202          " ln (2) = +.693147180560
ONE:            +   5 06966
                + 659 90648          " sqrt (8/9) × 2↑55 = +.339682755868₁₀+17
GIANT:          '3777 37777'
                '3777 77777'         " +.177664619751₁₀+629
BIN EXP 40:     +  13 10720          " 40 × d15

                'END'  LN            " 76 locations


EXP:            'BEGIN'  ENTIRE, C0, C1, C2, C3, C4, C5, C6, C7, LOG E,
                         OMEGA

                F +   0
EXP1:           M[B] = F, P
              N, F = - F             " abs (x)
                F - 1447, P          " > 1447 ?
              N, F = M[B]
              Y, F = 1447            " then replace
              Y, S = - M[B+1], P     " x by 1447
              Y, F = - 1447          " with correct sign
                F × LOG E            " × log (e) -> computation of 2↑x
                MC =  F, P
                SUBC (:ENTIER1)
                A =   G              " integer part of x
                F - MC[-2], Z        " fractional part of x = zero ?
              Y, F =    1
              Y, GOTO (:ENTIRE)
                MC =  A
                F = - F              " compute log (fractional part)
                F - HALF, P
                F - HALF
              N, F × HALF            " bring argument in range -1/2 < x < 0
                A = :C7
                SUBC (:POL IN F)     " and compute polynomial
              N, F ×   F             " square, if argument was halved
                A = MC[-1]           " integer part
                A +   1              " corrected for subtraction of 2 × HALF
ENTIRE:       U, A + 2048, P
              U, A - 2047, E         " outside range -2047, +2047 ?
              N, JUMP (5)
                A =   A, P           " > 2047 ?
              N, F / OMEGA
              N, GOTOR (MC[-1])
                F × OMEGA
```

```
                A - 2047
                LUA (15)
                A 'X' - 32767
                S =   1
                F X   A
                GOTOR (MC[-1])
C0:             -  13 27104
                -        0          " +.999999999999
C1:             -  13 32131
                +  351 25162        " +.693147180524
C2:             -  13 93280
                +  324 98472        " +.240226505508
C3:             -  14 60009
                +   42 24865        " +.555040853706₁₀-1
C4:             -  15 30010
                +   98 20595        " +.961794504001₁₀-2
C5:             -  16 27221
                +  234 73550        " +.133256313600₁₀-2
C6:             -  17 26494
                +  299 64123        " +.152132608000₁₀-3
C7:             -  17 35842
                +  348 22787        " +.128376319999₁₀-4
LOG E:          -  12 98901
                +  374 31644        " log (e) = +.144269504089₁₀+1
OMEGA:          +  670 76096
                +        1          " 2 ↑ 2047 = +.161585030357₁₀+617

                'END' EXP           " 58 locations

COS:            'BEGIN'  COSCOEF, SINCOEF, TWO OVER PI

                A =   1             " cos (x) = sin (x + pi/2)
                JUMP (1)
SIN:            A =   0
                F X TWO OVER PI     " replace x by x X 2/pi
                MC =  F, P
              U, S =  F, E          " x integer ?
              N, F + SCHOLTEN       " else round x to integer:
              N, F - SCHOLTEN, P    " n = x
                A +   G, E          " in case of COS: n = n + 1
              N, A 'X' 3, P         " take care of overflow in A
                RCA (1), E          " n even ?
              Y, S = :SINCOEF       " then use polynomial for sin
              N, S = :COSCOEF       " else use polynomial for cos, with
                F - MC[-2]          " y = x - n as argument
              Y, MC = - F           " polynomial for sin starts with y term
                F X   F
                M[B] = F            " evaluate
                F X MS[10]
                F + MS[8]
                F X M[B]            " polynomial
                F + MS[6]
                F X M[B]
                F + MS[4]           " of degree 5
                F X M[B]
                F + MS[2]
                F X M[B]            " in y square
                F + MS
```

```
                Y, F × MC[-2]          " multiply by y in the sin case
                   LCA (1)
                Y, A + 1
                   A 'X' 2, Z          " no extra minus sign ?
                N, F = - F             " else reverse sign
                   GOTOR (MC[-1])
COSCOEF:           +        0
                   +        1          " c0  = 1
                   + 12 42657
                   - 415 22043         " c2  = -.12337 00550 12 ₁₀+1
                   -  13 08641
                   +  40 67027         " c4  = +.25366 95072 72 ₁₀+0
                   +  14 96389
                   - 312 98891         " c6  = -.20863 46891 37 ₁₀-1
                   -  16 01776
                   + 173 93881         " c8  = +.91916 54179 15 ₁₀-3
                   +  12 12415
                   -  17 08115         " c10 = -.24856 34468 03 ₁₀-4
SINCOEF:           -  12 97852
                   + 646 60001         " c1  = +.15707 96326 79 ₁₀+1
                   +  13 32904
                   - 319 28607         " c3  = -.64596 40974 93 ₁₀+0
                   -  14 31346
                   + 316 68041         " c5  = +.79692 62611 83 ₁₀-1
                   +  14 08717
                   - 552 53273         " c7  = -.46817 52592 89 ₁₀-2
                   -  16 98552
                   +  75 84785         " c9  = +.16042 92697 34 ₁₀-3
                   +  11 79647
                   -   1 22197         " c11 = -.35564 00770 32 ₁₀-5
TWO OVER PI:       -  13 33057
                   + 253 90670         " 2/pi= +.63661 97723 67 ₁₀+0

                   'END' COS           " 59 locations

ARCTAN:            'BEGIN'  TOGETHER, C1, C2, C3, C4, C5, C6, TG15, TG30,
                            PI OVER 6

                   F +   0
                   MC =  G, P          " preserve original sign of x
                N, F = - F             " and replace x by absx = abs (x)
                   MC =  F             " preserve absx
                   S =   3             " and start analysis:
                   F -   1, P          " absx > 1 ?
                Y, F =   1
                Y, F / MC[-2]
                Y, MC =  F             " then replace absx by 1/absx
                N, F = M[B-2]          " else restore absx in F
                N, S -   2             " and change indication appropriately
                   F - TG15, E         " abs (x) > 1  =  absx > tg (pi/12) ?
                Y, S -   1             " then change indication appropriately
                U, A =   1, E          " absx > tg (pi/12) ?
                   F = M[B-2]
                N, GOTO (:TOGETHER)
                   F × TG30            " then
                   F +   1             " replace
                   MC =  F             " absx
                   F = M[B-4]          " by
```

```
                    F - TG30            " (absx - tg (pi/6)) /
                    F / MC[-2]          " (1 + absx × tg (pi/6))
                    M[B-2] = F
TOGETHER:           MC =  S             " preserve indication
                    A = :C6
                    SUBC (:POL IN FF)   " compute polynomial in F square
                    F × M[B+1]          " × absx square
                    F +   1
                    F × MC[-3]          " × absx
                    S =MC[1]            " indication
                U,  S -   1, P          " > 1 ?
                Y,  MC = - F
                N,  MC =  F
                    G =   S, Z          " indication = 0 ?
                N,  F × PI OVER 6
                    F + MC[-2]
                    S = MC[-1], P       " original sign of x > 0 ?
                N,  F = - F
                    GOTOR (MC[-1])
C1:                 +  13 65333
                    + 223 69812         " -.33333333246
C2:                 -  13 69702
                    + 402 22387         " +.19999980477
C3:                 +  13 99661
                    - 119 33742         " -.142855496622
C4:                 -  14 27237
                    + 571 72235         " +.111044707738
C5:                 +  14 03157
                    - 595 65289         " -.895216002193_B-1
C6:                 -  14 58249
                    + 433 90644         " +.622201788749_B-1
TG15:               -  13 08524
                    +  26 69885         " tg (pi/12) = +.267949192430
TG30:               -  13 05990
                    + 438 49281         " tg (pi/6) = 1/sqrt(3) = +.577350269190
PI OVER 6:          -  13 34909
                    + 431 06668         " pi/6 = +.523598775599

                    'END' ARCTAN        " 57 locations


'END'

'END'

'END'
```

```
COMP VAR[0]:

'BEGIN'         stock, stock1, quote counter, shift, letter last symbol,
                digit last symbol, own type, type, character, val char,
                arr dec ls, tp dec ls, arr dec mcr, real number, small,
                last nlp, word count, in formal list, in ar dec, int labels,
                int lab, old bcp, dsp lvl, global count, arr pointer,
                dimension, subcount, for cnt, nxt bcp, state, stack0, stack1,
                mcr, par, st cnt, max dpth, max di, max dl, max prl, ret md,
                ret lvl, Ecnt, ifstat frb, cntld var, L0, L1, L2, L3, L4, L5,
                cmpltd, cmpl st, sw idf, nbr of sw e, sw lst, in sw dec,
                adrs of cst

stock:                  'SKIP' 1
stock1:                 'SKIP' 1
quote counter:          'SKIP' 1
shift:                  'SKIP' 1
letter last symbol:     'SKIP' 1
digit last symbol:      'SKIP' 1
own type:               'SKIP' 1
type:                   'SKIP' 1
character:              'SKIP' 1
val char:               'SKIP' 1
arr dec ls:             'SKIP' 1
tp dec ls:              'SKIP' 1
arr dec mcr:            'SKIP' 1
real number:            'SKIP' 1
small:                  'SKIP' 1
last nlp:               'SKIP' 1
word count:             'SKIP' 1
in formal list:         'SKIP' 1
in ar dec:              'SKIP' 1
int labels:             'SKIP' 1
int lab:                'SKIP' 1
old bcp:                'SKIP' 1
dsp lvl:                'SKIP' 1
global count:           'SKIP' 1
arr pointer:            'SKIP' 1
dimension:              'SKIP' 1
subcount:               'SKIP' 1
for cnt:                'SKIP' 1
nxt bcp:                'SKIP' 1
state:                  'SKIP' 1
stack0:                 'SKIP' 1
stack1:                 'SKIP' 1
mcr:                    'SKIP' 1
par:                    'SKIP' 1
st cnt:                 'SKIP' 1
max dpth:               'SKIP' 1
max di:                 'SKIP' 1
max dl:                 'SKIP' 1
max prl:                'SKIP' 1
ret md:                 'SKIP' 1
ret lvl:                'SKIP' 1
Ecnt:                   'SKIP' 1
ifstat frb:             'SKIP' 1
cntld var:              'SKIP' 1
L0:                     'SKIP' 1
```

```
" compiler variables continued

L1:                     'SKIP' 1
L2:                     'SKIP' 1
L3:                     'SKIP' 1
L4:                     'SKIP' 1
L5:                     'SKIP' 1
cmpltd:                 'SKIP' 1
cmpl st:                'SKIP' 1
sw idf:                 'SKIP' 1
nbr of sw e:            'SKIP' 1
sw lst:                 'SKIP' 1
in sw dec:              'SKIP' 1
adrs of cst:            'SKIP' 1
lnc:                    'SKIP' 1
last lnc:               'SKIP' 1
code body:              'SKIP' 1            " 60 variables
```

```
BEGIN OF COMPILER[0]:

'BEGIN'         d18, d19, d21, d22, d22 plus 1, d24, d25, end of list,
                func ltr, func dit, C var, mask, word del, begin of nli,
                correction, begin of catalogue, end of catalogue

d18:                    '001 000 000'
d19:                    '002 000 000'
d21:                    '010 000 000'
d22:                    '020 000 000'
d22 plus 1:             '020 000 001'
d24:                    '100 000 000'
d25:                    '200 000 000'

end of list:            + 64            " function part of A - M[0]
func ltr:               '100 000 000'   " transforms F = :STAT into F = :DYN
func dit:               '040 000 000'   " transforms A = :STAT into A = :DYN
C var:                  '006 000 000'   " transforms A =  STAT into A =  DYN

mask:                   (20 x '004 000 000')    " re
                        (21 x '004 000 000')    " in
                        (22 x '004 000 000')    " bo
                        (23 x '004 000 000')    " st
                        (20 x '004 000 000')    " ar
                        (31 x '004 000 000')    " nondes
                        (28 x '004 000 000')    " des
                        (31 x '004 000 000')    " un
                        (31 x '004 000 000')    " arbo
                        (29 x '004 000 000')    " intlab

word del:               +296926                 " if
                        +477407                 " then
                        +232160                 " else
                        +182120                 " begin
                        +232425                 " end
                        +265297                 " goto
                        +248914                 " for
                        +462574                 " step = string
                        +494548                 " until
                        +526549                 " while
                        +216150                 " do
                        +199777                 " comment
                        +397418                 " own
                        +444267                 " real
                        +297964                 " integer
                        +625773                 " Boolean
                        +183405                 " boolean
                        +167407                 " array
                        +413168                 " procedure
                        +462961                 " switch
                        +345458                 " label
                        +509299                 " value
                        +478708                 " true
                        +247157                 " false

begin of nli:           (:END OF MEMORY - 2)
correction:             (begin of nli - 8191)
begin of catalogue:     :BEG CAT
end of catalogue:       :END CAT                " 49 items
```

```
'BEGIN'         NXT SBL, NXT BSC SBL, IN SBL, UND SBL, OUT SBL, AR OP LS,
                INVERT, REL OP LS, BOOL OP LS, DECL LS, SPEC LS, OP LS,
                UNS INT, MULTIPLY, UNS NBR, RD IDF, RD IDF1, NXT PNR, LOOK UP,
                IN NM LST, NXT IDF, SKP IDF, SKP TP DEC, SKP VA LI, SKP SP LI,
                DISP LVL, TP OF DSP, LOC SPC, PRC LVL, USE CST, STATUS,
                IN CODE, ENTR BLK, EXIT BLK, LOC LAB, NONFRM LAB, CORSP BCP,
                SKIP STRING, SKIP RST OF STAT, INIT, PRESCAN1, TRANSL


NXT SBL:            'BEGIN' SKIP0, SKIP1, SKIP2, ELSE0, ELSE1, END

                    S = stock1, P           " symbol in stock ?
                Y,  stock1 = - S
                N,  SUBC (:NXT BSC SBL)
                U,  S - 97, Z               " next basic symbol = comment ?
                N,  GOTO (:ELSE0)
                    A = last symbol
                U,  A - 91, Z               " last symbol = semicolon ?
                N,  A - 104, Z              " V last symbol = begin ?
                N,  GOTO (:ELSE0)
SKIP0:              SUBC (:NXT BSC SBL)
                U,  S - 91, Z               " next basic symbol = semicolon ?
                N,  GOTO (:SKIP0)
                    GOTO (:NXT SBL)
ELSE0:              A = last symbol
                    A - 105, Z              " last symbol = end ?
                N,  GOTO (:ELSE1)
SKIP1:              A =  S                  " last basic symbol
                U,  A - 105, Z              " = end ?
                N,  A - 91, Z               " = semicolon ?
                N,  A -  5, Z               " = else ?
                Y,  GOTO (:END)             " then accept
                    SUBC (:NXT BSC SBL)
                    GOTO (:SKIP1)
ELSE1:          U,  S - 125, Z              " last basic symbol = close ?
                N,  GOTO (:END)
                    SUBC (:NXT BSC SBL)
                U,  S -  9, P               " next basic symbol
                U,  S - 63, E               " not a letter ?
                Y,  stock1 = S
                Y,  S = 99                  " internal representation of close
                Y,  GOTO (:END)
SKIP2:              SUBC (:NXT BSC SBL)
                U,  S -  9, P               " next basic symbol
                U,  S - 63, E               " not a letter ?
                N,  GOTO (:SKIP2)           " else skip
                U,  S - 90, Z               " last basic symbol = colon ?
                Y,  SUBC (:NXT BSC SBL)
                    stock1 = S
                N,  A = 100
                N,  SUBC (:ERRORM)
                U,  S - 98, Z               " last basic symbol = open ?
                Y,  stock1 = - S
                N,  A = 101
                N,  SUBC (:ERRORM)
                    S = 87                  " internal representation of comma
END:                last symbol = S
                    A = S
                U,  A - 87, P               " (last symbol ≠ period
```

```
                U, A - 89, E            " ∧ last symbol ≠ ten)
                Y, A -  9, P            " implies last symbol > 9 ?
                N, S =  0
                   digit last symbol = S
                Y, S 'X' - 63, Z        " ¬ digit last symbol ∧ last symbol
                N, S =  1               "                          < 64 ?
                   letter last symbol = S
                   S = last symbol
                   A = run number
                   SUBC (:OUT SBL)
                   GOTO (:TEST POINTERS)" 59 instructions

                   'END' NXT SBL


NXT BSC SBL:       A = run number
                   SUBC (:IN SBL)
                U, S - 119, Z           " symbol = new line ?
                N, GOTOR (MC[-1])
                   A =  1
                   line counter + A
                   A = quote counter, Z
                N, GOTOR (MC[-1])
                   A = run number
                   SUBC (:OUT SBL)
                   GOTO (:NXT BSC SBL)  " 11 instructions


IN SBL:            'BEGIN' ELSE0, ELSE1, ELSE2, ELSE3, ELSE4, ELSE5, ELSE6,
                           NEXT0, NEXT1, NEXT2, NEXT3

                U, A - 200, Z
                N, A - 300, Z           " (source = 200 ∨ source = 300)
                Y, A = text in array, Z " ∧ text in text array ?
                N, GOTO (:ELSE1)
                   A = text array pointer
                   S =  MA              " text[text array pointer]
                   A = shift
                U, A - 256, P           " shift > 256 ?
                N, GOTO (:ELSE0)
                   RUS (16)
                   A =  1
                   text array pointer + A
                   shift = A
                   GOTOR (MC[-1])
ELSE0:          U, A -  1, Z            " shift = 1 ?
                N, RUS (8)
                   S 'X' 255
                   LUA (8)              " shift:= 256 × shift
                   shift = A
                   GOTOR (MC[-1])
ELSE1:             S = stock, P         " symbol in stock ?
                Y, stock = - S
                N, SUBC (:NXT TAPE SL)
                U, S - 101, P           " next tape symbol > bus ?
                N, GOTOR (MC[-1])
                U, S - 123, Z
                Y, S =  93              " internal representation of space
                   A = quote counter, Z
```

```
                Y, GOTO (:ELSE3)
                U, S - 127, Z          " last tape symbol = bar ?
                N, GOTO (:ELSE2)
NEXT0:             SUBC (:NXT TAPE SL)
                U, S - 127, Z          " next tape symbol = bar ?
                Y, GOTO (:NEXT0)
                   stock = S
                   A = quote counter
                U, S -  72, Z          " les after bar ?
                Y, A +   2             " then quote counter = quote counter + 1
                N, S -  74, Z          " mor after bar ?
                   S = 127             " internal representation of bar
                N, JUMP (4)
                   A -   1, P          " new value of quote counter > 0 ?
                Y, quote counter = A
                N, S = 103             " internal representation of unquote
                N, stock = - S
ELSE2:          U, S - 124, Z          " last tape symbol = colon ?
                Y, S =  90             " internal representation of colon
                U, S - 125, Z          " last tape symbol = close ?
                Y, S =  99             " internal representation of close
                   GOTOR (MC[-1])
ELSE3:          U, S - 118, P          " last tape symbol ≥ new line ?
                N, GOTO (:ELSE1)       " else skip
                U, S - 127, Z          " last tape symbol = bar ?
                N, GOTO (:ELSE4)
NEXT1:             SUBC (:NXT TAPE SL)
                U, S - 127, Z          " next tape symbol = bar ?
                Y, GOTO (:NEXT1)
                U, S -  80, Z          " and after bar ?
                Y, S = - 2
                N, S -  70, Z          " equ after bar ?
                Y, S -  31
                N, S -   2, Z          " les after bar ?
                Y, S -   1
                N, S -   2, Z          " mor after bar ?
                Y, S + 103
                N, S = 160             " else inadmissable
                   GOTOR (MC[-1])
ELSE4:          U, S - 126, Z          " last tape symbol = underlining ?
                N, GOTO (:ELSE6)
                   SUBC (:UND SBL)
                U, S -  63, P          " after underlining no letter or digit ?
                N, GOTO (:ELSE5)
                U, S -  70, Z          " equ after underlining ?
                Y, S =   4
                N, S -  72, Z          " les after underlining ?
                Y, S -   2
                N, S -   2, Z          " mor after underlining ?
                Y, S -   3
                N, S -   2, Z          " non after underlining ?
                Y, S + 10
                N, S -  48, Z          " colon after underlining ?
                Y, S +  68
                N, S = 161             " else inadmissable
                   GOTOR (MC[-1])
ELSE5:             LUS (7)
                   stock = S
                   SUBC (:NXT TAPE SL)
```

```
                U, S - 126, Z          " next tape symbol = underlining ?
                N, stock = S
                N, GOTO (:ELSE5[-2])   " else inadmissable
                   SUBC (:UND SBL)
                   S + stock
                   LUS (7)
                   A = 23
                   COUNT = A
                   F = :word del
NEXT2:             A = MG              " cycle: look up word delimiter
                   A - S
                U, A 'X' - 127, Z      " found ?
                N, F + 1
                N, REPE (:NEXT2)
                Y, S = A
                N, S = 162             " else inadmissable combination
                   stock = S
NEXT3:             SUBC (:NXT TAPE SL) " cycle: skip rest of word delimiter
                U, S - 126, Z          " next tape symbol = underlining ?
                Y, SUBC (:UND SBL)
                Y, GOTO (:NEXT3)
                   A = stock
                   stock = S
                   S = A
                   GOTOR (MC[-1])
ELSE6:          U, S - 124, Z          " last tape symbol = colon ?
                N, GOTOR (MC[-1])
                   SUBC (:NXT TAPE SL)
                U, S - 70, Z           " equ after colon ?
                Y, S = 92              " internal representation of colonequal
                N, stock = S
                N, S = 90              " internal representation of colon
                   GOTOR (MC[-1])      " 120 instructions

                   'END' IN SBL


UND SBL:           SUBC (:NXT TAPE SL)
                U, S - 126, Z          " next tape symbol = underlining ?
                N, GOTOR (MC[-1])
                   GOTO (:UND SBL)     " 4 instructions
```

```
OUT SBL:        A - 100, Z          " destination = 100 ?
              Y, A = text in array, Z " ^ text in text array ?
              N, GOTOR (MC[-1])
                 A = shift
              U, A - 256, Z         " previous shift = 256 ?
              Y, LUS (8)
              U, A - 256, P         " previous shift > 256 ?
              N, LCSA (8)
              Y, A =  1
                 shift = A          " update shift
                 A = text array pointer
              N, MA[-1] + S         " text[text array pointer] =
              N, GOTOR (MC[-1])     " text[text array pointer]+shift*source
                 A +  1             " text array pointer =
                 text array pointer = A "          text array pointer + 1
                 end of text array = A
                 MA[-1] = S         " text[text array pointer] = source
                 GOTOR (MC[-1])     " 18 instructions


AR OP LS:       S = last symbol
              U, S -  63, P
              U, S -  69, E
INVERT:       U, S = - T, P
                 GOTO (MC[-1])      " 5 instructions


REL OP LS:      S = last symbol
              U, S -  75, P
                 GOTO (:AR OP LS[2]) " 3 instructions


BOOL OP LS:     S = last symbol
              U, S -  76, P
              U, S -  80, E
                 GOTO (:INVERT)     " 4 instructions


DECL LS:        S = last symbol
                 S - 106, Z         " last symbol = own ?
                 own type = S
              Y, SUBC (:NXT SBL)
              Y, S - 106, Z
              U, S -  4, E          " last symbol no <type> ?
              Y, A = 1000
              N, A = :MS[-1]
                 type = A
              N, SUBC (:NXT SBL)
              N, JUMP (6)
                 S -  5, Z          " last symbol = array ?
              Y, type = S
                 A = own type, Z
              Y, A = 104
              Y, SUBC (:ERRORM)
                 S = last symbol
                 S - 111, Z         " last symbol = array ?
                 arr dec ls = S
              N, JUMP (6)
                 S =  8             " chara = 8
```

```
   A = own type, Z
N, A = - 4
   A + type
   arr dec mcr = A
   JUMP (9)
U, S -  1, Z          " last symbol = proced ?
N, JUMP (4)
   S = type
U, S -  3, P          " type > 3 ?
Y, S =  10
N, S =  2, P
N, S -  2, Z          " last symbol = switch ?
Y, S +  14
N, S = type
   character = S
U, S -  8, P          " chara > 8 ?
Y, A = own type, Z    " ∧ own type ?
Y, A = 105
Y, SUBC (:ERRORM)
U, S -  14, Z         " last symbol = switch ?
Y, A = - type
Y, A +  4, P          " ∧ type < 4 ?
Y, A = 106
Y, SUBC (:ERRORM)
   S = - character
U, S +  25, P         " chara < 25 ?
N, GOTO (MC[-1])
U, S +  4, P          " chara < 4 ?
Y, A =  0
N, A =  1
   tp dec ls = A
   A = type
Y, JUMP (3)
U, A -  3, P          " type > 3 ?
Y, A = - S            " then chara
N, A -  S             " else type + chara
U, A = own type, Z
Y, A +  32
   LUA (19), P
Y, character = A
N, character = - A, P
   GOTO (MC[-1])      " 63 instructions
```

```
SPEC LS:        S = last symbol
              U, S - 111, Z          " last symbol = array ?
              Y, S = 112             " then type = 5
              U, S - 106, E          " last symbol not <type> or array ?
              Y, A = 1000            " then type = 1000
              N, A = :MS[-107]
                 type = A
              U, A -   3, P          " type > 3 ?
              N, SUBC (:NXT SBL)
              U, S - 114, Z          " last symbol = label ?
              Y, S = - 8
              N, S - 113, Z          " last symbol = switch ?
              Y, S +  14
              N, S = 1000
                 character = S
                 S + type
              U, S - 999, P          " type + chara ≥ 1000 ?
              N, A = 107
              N, SUBC (:ERRORM)
                 S = last symbol
              U, S - 112, Z          " last symbol = proced ?
              N, JUMP (4)
                 S = type
              U, S -   3, P          " type > 3 ?
              Y, S =  16
              N, S =   8, P
              N, S - 111, Z          " last symbol = array ?
              Y, S +   8
              Y, character = S
                 S = - character
              U, S +  25, P          " chara < 25 ?
              Y, SUBC (:NXT SBL)
              Y, S = - character
                 S - type
                 S + 2000, P         " type + chara < 2000 ?
              N, GOTO (MC[-1])
                 S = character
                 A = type
              U, S -   8, P          " chara > 8 ?
              Y, JUMP (5)
              U, S -   6, Z          " chara = 6 ?
              Y, A =   0
              U, A -   5, Z          " type = 5 ?
              Y, A =   8
              N, A +   S
                 A +  64
                 val char = A
                 A = type
              U, A -   5, P          " type > 5 ?
              Y, JUMP (5)
              U, A -   1, P          " type > 1 ?
              N, A =   4
              U, S - 1000, Z         " chara = 1000 ?
              Y, S =   0
                 S +   A
                 S +  96
                 LUS (19)
                 character = S, P
                 GOTO (MC[-1])       " 59 instructions
```

```
OP LS:          SUBC (:AR OP LS)
              N, SUBC (:REL OP LS[1])
              Y, GOTO (MC[-1])
                 GOTO (:BOOL OP LS[1]) " 4 instructions


UNS INT:          F =   0
                  MC =  F              " head = tail = 0
                  SUBC (:MULTIPLY)
              U, S -   9, P            " last symbol > 9 ?
              N, JUMP (-3)
UNS INT[5]:       G = MC[-1]           " tail
              U, S = MC[-1], Z         " head = 0 ?
              N, F = 32767             " else replace tail by 32767
                 value of constant = F
                 F - 32767, P          " > 32767 ?
              Y, small = S             " then small = false
                 GOTOR (MC[-1])        " 12 instructions


MULTIPLY:         'BEGIN' TENTH

                  A = last symbol
              U, A -   9, P            " last symbol > 9 ?
              Y, A = 109
              Y, SUBC (:ERRORM)
              Y, S = last symbol
              Y, JUMP (9)
                 S = M[B-3]
                 S - TENTH, P          " head > 6710885 ?
              N, S = M[B-2]
              N, MULAS (10)            " tail = 10 × tail + last symbol
              N, M[B-2] = S
              N, S = M[B-3]
              N, MULAS (10)            " head = 10 × head + carry from tail
              N, M[B-3] = S
                 SUBC (:NXT SBL)
                 A =   1
                 GOTO (MC[-1])
TENTH:           + 67 10885            " 18 instructions

                 'END' MULTIPLY
```

```
UNS NBR:        F =   0                 " mantissa = 0
                real number = G         " real number = true
                small = G               " small = true, decimal exponent = 0
                S = last symbol
              U, S -  89, Z             " last symbol = ten ?
              Y, F =   1                " then mantissa = 1
                MC =  F                 " head and tail of mantissa
              U, S -   9, P             " last symbol > 9 ?
              Y, JUMP (3)
                SUBC (:MULTIPLY)
              Y, small + A              " count in case of overflow
                JUMP (-5)
                A = digit last symbol, Z " last symbol = period or ten ?
              N, A = M[B-2]
              N, A -   0, P             " ∨ head ≠ 0 ?
              N, real number = B        " else real number = false
              N, GOTO (:UNS INT[5])     " and deliver single length integer
              U, S -  88, Z             " last symbol = period ?
              N, JUMP (5)
                SUBC (:NXT SBL)
                SUBC (:MULTIPLY)        " handle decimal fraction
              N, small - A              " count except when overflow
              U, S -   9, P             " last symbol > 9 ?
              N, JUMP (-4)
              U, S -  89, Z             " last symbol = ten ?
              N, S = small
              N, JUMP (10)
                SUBC (:NXT SBL)
              U, S -  64, Z             " last symbol = plus ?
              N, S -  65, Z             " ∨ last symbol = minus ?
              Y, MC =  S
              Y, SUBC (:NXT SBL)
                SUBC (:UNS INT)
                S = M[B+2]              " tail of decimal exponent
              Y, A = MC[-1], Z          " was ten followed by minus ?
              Y, S = - S
                S + small               " decimal exponent now completed
                A =   S
                RUA (10), Z             " abs (decimal exponent) < 1024 ?
              N, S = 1024, E            " else decimal exponent = 1024 ×
              N, S = - 1024             "       × sign (decimal exponent)
                F = MC[-2]              " head and tail of mantissa
                small = B               " small = false
                GOTO (:DEC BIN)         " convert to binary number
                                        " 44 instructions

RD IDF:         'BEGIN' NEXTO, ELSEO, ELSE1

                S = nlp
                word count = S          " word count = 0
              U, S = letter last symbol, Z
              Y, A =   0
              N, A =   1
                MS = - A                " name list[nlp] = - 1, word = 0
              N, A = 110
              N, SUBC (:ERRORM)
              N, GOTO (:ELSE1)
                COUNT = A               " count = 0
```

```
                  S = last symbol
NEXTO:          U, S 'X' - 63, Z        " last symbol < 64 ?
                N, GOTO (:ELSEO)
                   S = COUNT
                   S +   4, Z           " count = 4 ?
                Y, COUNT = S            " then count = 0
                   A = word count
                Y, A -   1              " and word count = word count + 1
                Y, word count = A
                N, S =  MA
                N, LUS (6)
                   S - last symbol
                   S -   1
                   MA =  S
                   SUBC (:NXT SBL)
                   REP (:NEXTO)
ELSEO:             A = nlp
                   S = - MA
                   last identifier = S  " last identifier = name list[nlp]
                U, A - word count, Z    " word count = 0 ?
                Y, S =   0
                N, S = - MA[-1]         " else last identifier[1] =
                   last identifier[1] = S "     name list[nlp + 1]
ELSE1:             A = word count
                   S = 127
RD IDF1:           LUS (19)
                   MA[-1] = S           " name list[nlp + word count + 1] =
                   A - nlp              "                          127 × d19
                   word count = - A
                   GOTOR (MC[-1])       " 40 instructions

                   'END' RD IDF


NXT PNR:           A = - MS, P          " word = - name list[pointer]
                N, S -   1              " if word ≤ 0 then pointer = pointer + 1
                N, GOTO (:NXT PNR)
                U, A - d25, P           " word > 33554432 ?
                N, GOTOR (MC[-1])
                   S = :MA
                   GOTO (:NXT PNR)      " 7 instructions


LOOK UP:           'BEGIN' NEXTO, NEXT1, NEXT2

                   S = in formal list, Z
                N, S = in ar dec, Z
                   S = :MD[-4]
                Y, S - 1                " block cell pointer + (4 or 5)
NEXTO:             SUBC (:NXT PNR)
                   A = word count
                   COUNT = A
                   G = last nlp
NEXT1:             A = -MS, P           " name list[pointer + count] =
                Y, A + MG, Z            " name list[last nlp + count] ?
                N, GOTO (:NEXT2)
                   S -   1              " then count = count + 1
                   F -   1
                   REPE (:NEXT1)
```

```
                  A = MS, P            " found ?
               Y, GOTOR (MC[-1])
                  S - 1
NEXT2:            A = MS, P
               Y, GOTO (:NEXTO)
                  GOTO (:NEXT2[-1])    " 20 instructions

                  'END' LOOK UP


IN NM LST:        S = int labels, Z
               N, GOTO (MC[-1])
                  S = real number, Z
               Y, GOTO (:INVERT)
                  S = value of constant[1]
                  RUS (18)             " head = value of constant : d18
                  S + 4096
                  A = nlp
                  last nlp = A         " last nlp = nlp
                  MA = - S             " name list[nlp] = - 4096 - head
                  A - 1
                  S - 4097
                  LUS (18)             " (head - 1) × d18
                  S - value of constant[1]
                  MA = S
                  S = 6
                  SUBC (:RD IDF1)
                  SUBC (:LOOK UP)
                  int lab = S
                  S - nlp, P           " int lab < nlp ?
                  GOTO (MC[-1])        " 21 instructions
```

```
NXT IDF:          SUBC (:NXT PNR)
                  S -  1
                  A =  MS, P
                Y, GOTOR (MC[-1])
                  GOTO (:NXT IDF[1])   " 5 instructions

SKP IDF:          S = last symbol
                U, S 'X' - 63, Z       " last symbol < 64 ?
                Y, SUBC (:NXT SBL)
                Y, GOTO (:SKP IDF[1])
                  GOTOR (MC[-1])       " 5 instructions

SKP TP DEC:       SUBC (:SKP IDF)
                U, S - 87, Z           " last symbol = comma ?
SKP TP DEC[2]:  Y, SUBC (:NXT SBL)
                Y, GOTO (:SKP TP DEC)
                  GOTOR (MC[-1])       " 5 instructions

SKP VA LI:        S = last symbol
                U, S - 115, Z          " last symbol = value ?
                N, GOTOR (MC[-1])
SKP VA LI[3]:     SUBC (:SKP TP DEC[2])
                U, S - 91, Z           " last symbol = semicolon ?
                Y, GOTO (:NXT SBL)
                  GOTOR (MC[-1])       " 7 instructions

SKP SP LI:        SUBC (:SPEC LS)
                N, GOTOR (MC[-1])
                  SUBC (:SKP VA LI[3])
                  GOTO (:SKP SP LI)    " 4 instructions

DISP LVL:         S = MD[-1]           " name list[block cell pointer + 1]
                  S 'X' 63
                  GOTOR (MC[-1])       " 3 instructions

TP OF DSP:        S = MD[-1]           " name list[block cell pointer + 1]
                  RUS (6)
                  GOTO (:DISP LVL[1])  " 3 instructions

LOC SPC:          S = MD[-1]           " name list[block cell pointer + 1]
                  RUS (13)
                  GOTOR (MC[-1])       " 3 instructions

PRC LVL:          S = MD[-2]           " name list[block cell pointer + 2]
                  GOTO (:DISP LVL[1])  " 2 instructions

USE CST:          S = - MD[-2]         " d6 of name list[block cell pointer+2]
                  S 'X' 64, Z          " = 1 ?
                  GOTO (MC[-1])        " 3 instructions
```

```
STATUS:         S = MD[-2]          " name list[block cell pointer + 2]
                RUS (13)
                S + correction
                GOTOR (MC[-1])      " 4 instructions

IN CODE:        A = - MS[-1]        " d24 of name list[n + 1]
                LCA (2), P          " = 1 ?
                GOTO (MC[-1])       " 3 instructions
```

```
ENTR BLK:       S = nxt bcp
                D =   S
                S =  MS
                S 'x' 8191
                S + correction
                nxt bcp = S
                GOTOR (MC[-1])          " 7 instructions

EXIT BLK:       S =  MD
                RUS (13)
                S + correction
                D =   S
                GOTOR (MC[-1])          " 5 instructions

LOC LAB:        SUBC (:NONFRM LAB)
              N, GOTO (MC[-1])
                SUBC (:CORSP BCP)
                G -   D, Z
                GOTO (MC[-1])           " 5 instructions

NONFRM LAB:     A =  MS
                RUA (19)
                A -   6, Z              " code bits (n) = 6 ?
                GOTO (MC[-1])           " 4 instructions

CORSP BCP:      G =   D                 " p = block cell pointer
              U, S -   G, P             " n < p ?
              N, A = MG[-2]
              N, RUA (13)
              N, A + correction
              N, A -   S, P             " n > name list[p + 2] : 8192 ?
              N, GOTOR (MC[-1])
                A =  MG
                RUA (13), Z
              Y, GOTOR (MC[-1])
                A + correction
                G =   A
                GOTO (:CORSP BCP[1])    " 13 instructions

SKIP STRING:    S =   1
                quote counter = S       " quote counter = 1
                SUBC (:NXT SBL)
              U, S - 103, Z             " next symbol = unquote ?
              N, JUMP (-3)              " else skip
                A =   0
                quote counter = A       " quote counter = 0
                GOTOR (MC[-1])          " 8 instructions
```

```
SKIP RST OF STAT:  MC =  A              " pr
                   S = last symbol      " last symbol
                U, S -  86, Z           " = do ?
                Y, SUBC (:NXT SBL)
                N, S -  81, Z           " = goto ?
                N, S -   1, Z           " = for ?
                N, S -  22, Z           " = begin ?
                Y, SUBC (M[B-1])        " then pr
                   S = last symbol
                U, S - 102, Z           " last symbol = quote ?
                Y, SUBC (:SKIP STRING)
                U, S -  91, Z           " last symbol = semicolon ?
                N, S - 105, Z           " ∨ last symbol = end ?
                N, SUBC (:NXT SBL)
                N, GOTO (:SKIP RST OF STAT[2])
                   B -  1
                   GOTOR (MC[-1])       " 17 instructions

INIT:              run number = S
                   F =  0
                   S =  1
                   stock = - S          " stock = - 1
                   stock1 = - S         " stock1 = - 1
                   last symbol = - S    " last symbol = - 1
                   word count = - S     " word count = - 1
                   shift = S            " shift = 1
                   A = begin of text array
                   text array pointer = A
                   value of constant = - F
                   last identifier = F  " last identifier = empty
                   line counter = G     " line counter = 0
                   quote counter = G    " quote counter = 0
                   for cnt = S          " forcount = 1
                   in formal list = S   " in formal list = false
                   in ar dec = S        " in array declaration = false
                   GOTOR (MC[-1])       " 18 instructions
```

```
'BEGIN'         PROGRAM, BLOCK, CMP TL, DECL LST, STATMNT, BEGIN STAT, LAB DEC,
                ILAB DEC, ST NUM CS, IDF, PRCS IDF


PROGRAM:            'BEGIN' PROGRAM0, PROGRAM1, PROGRAM2, PROGRAM3, PROGRAM4,
                          PROGRAM5

                    S =  6
                    LUS (19)
                    character = S          " character = 6 × d19
PROGRAM0:           S = letter last symbol, Z
                  N, GOTO (:PROGRAM2)
                    SUBC (:RD IDF)
                    A = last symbol
                    A -  90, Z             " last symbol = colon ?
                  Y, SUBC (:PRCS IDF)
                  Y, SUBC (:LAB DEC)
                    A = 111
PROGRAM1:         Y, GOTO (:PROGRAM0)
                    GOTO (:PROGRAM4)
PROGRAM2:           S = digit last symbol, Z
                  N, GOTO (:PROGRAM3)
                    SUBC (:UNS NBR)
                    A = last symbol
                    A -  90, Z             " last symbol = colon ?
                  Y, SUBC (:ILAB DEC)
                    A = 112
                    GOTO (:PROGRAM1)
PROGRAM3:           S = last symbol
                  U, S - 104, Z            " last symbol = begin ?
                    A = 113
PROGRAM4:         N, SUBC (:ERRORM)
PROGRAM5:         N, SUBC (:NXT SBL)
                  U, S - 104, Z            " last symbol = begin ?
                  Y, GOTO (:BEGIN STAT)
                    GOTO (:PROGRAM5)       " 29 instructions

                    'END' PROGRAM


BLOCK:              'BEGIN' BLCK, PROC, SEM TEST, SPEC TEST, TEST, BODY, END,
                          NEXT0, NEXT1, NEXT2, NEXT3, NEXT4, NEXT5, ELSE0

                    S = nlp
                    MS =  B                " name list[nlp] = address of dump
                    MC =  A, Z             " dump0 = proc identifier
                    A =   D
                    MC =  A                " dump1 = block cell pointer
                    F =   0
                    MC =  F                " local for count = max for count = 0
                    MC =  F                " local count = label count = 0
                    MC =  F                " internal block depth =
                    A = M1[8]              "                 string occurrence = 0
                    MC =  A                " dump8 = prc level
                    S -   1
                    D =   S                " block cell pointer = nlp + 1
                    A = old bcp            " name list[old block cell pointer] =
                    S - correction         " name list[old block cell pointer] +
                    MA +  S                "                + block cell pointer
```

```
                  S + correction
                  old bcp = S          " old block cell pointer =
                  A = M1[1]            "                 block cell pointer
                  A - correction
                  LUA (13)
                  MS = A               " name list[block cell pointer] =
                  A = 1                "                 8192 × dump1
                  PLUSA (dsp lvl)      " displ level = displ level + 1
                  MS[-1] = A           " name list[block cell pointer + 1] =
                  MS[-3] = G           "                 displ level
                  S - 5                " name list[block cell pointer + 3] = 0
                  nlp = S              " nlp = nlp + 6
                N, GOTO (:PROC)        " if proc identifier ≠ 0 then goto PROC
BLCK:             A = d25
                  A - 4
                  A + M1[1]
                  MS = - A             " name list[nlp] = - 33554436 - dump1
                  S - 1
                  nlp = S              " nlp = nlp + 1
                  S + d25              " name list[block cell pointer + 4] =
                  MD[-4] = - S         "                 - 33554432 - nlp
                N, GOTOR (MC[-1])      " return to body if called from it
                  SUBC (:DECL LST)
                  SUBC (:CMP TL)
                  GOTO (:END)
PROC:             M1[8] = A            " prc level = displ level
                  MC = G               " formal count = 0
                  S + d25              " name list[block cell pointer + 4] =
                  MD[-4] = - S         "                 - 33554432 - nlp
                  S = last symbol
                U, S - 98, Z           " last symbol = open ?
                N, GOTO (:SEM TEST)
                  S = 127
                  LUS (19)
                  character = S        " character = 127 × d19
NEXT0:            SUBC (:NXT SBL)
                  SUBC (:IDF)
                  S = 1
                  M[B-1] + S           " formal count = formal count + 1
                  MINS (nlp)           " nlp = nlp + 1
                  A = 0
                  MS[1] = A            " name list[nlp - 1] = 0
                  S = last symbol
                U, S - 87, Z           " last symbol = comma ?
                Y, GOTO (:NEXT0)
                U, S - 99, Z           " last symbol = close ?
                Y, SUBC (:NXT SBL)
                N, A = 114
                N, SUBC (:ERRORM)
SEM TEST:       U, S - 91, Z           " last symbol = semicolon ?
                Y, SUBC (:NXT SBL)
                N, A = 115
                N, SUBC (:ERRORM)
                  A = MC[-1]
                  A + d22 plus 1
                  S = M1
                  MS[-1] = A           " name list[proc identifier] =
                  S = last symbol      "       d22 + formal count + 1
                U, S - 115, Z          " last symbol = value ?
```

```
                N, GOTO (:SPEC TEST)
NEXT1:             SUBC (:NXT SBL)
                   SUBC (:IDF)
                U, S - last nlp, P      " n < last nlp ?
                Y, A = 95
                Y, LUA (19)
                Y, MS = A               " name list[n] = 95 × d19
                N, A = 116
                N, SUBC (:ERRORM)
                   S = last nlp
                   nlp = S              " nlp = last nlp
                   S = last symbol
                U, S - 87, Z            " last symbol = comma ?
                Y, GOTO (:NEXT1)
                   A = 117
NEXT2:          U, S - 91, Z            " last symbol = semicolon ?
                Y, SUBC (:NXT SBL)
                N, SUBC (:ERRORM)
SPEC TEST:         SUBC (:SPEC LS)      " specifier last symbol ?
                N, GOTO (:BODY)
NEXT3:             SUBC (:IDF)
                U, S - last nlp, P      " n < last nlp ?
                N, A = 118
                N, SUBC (:ERRORM)
                N, GOTO (:TEST)
                   A = MS
                   RUA (19)
                U, A - 127, Z           " name list[n] = 127 × d19 ?
                Y, A = character
                Y, MS = A               " name list[n] = character
                Y, GOTO (:TEST)
                U, A - 95, Z            " name list[n] = 95 × d19 ?
                N, A = 119
                N, GOTO (:NEXT3[3])
                   A = val char
                U, A - 75, P            " value character > 75 ?
                Y, A = 120
                Y, SUBC (:ERRORM)
                N, LUA (19)
                N, MS = A               " name list[n] = value character × d19
                   A = type
                   A - 3, Z             " type = 3 ?
                Y, A = 64
                Y, M1[7] = A            " then stringoccurrence = 64
TEST:              S = last nlp
                   nlp = S              " nlp = last nlp
                   S = last symbol
                U, S - 87, Z            " last symbol = comma ?
                Y, SUBC (:NXT SBL)
                Y, GOTO (:NEXT3)
                   A = 121
                   GOTO (:NEXT2)
BODY:              S = nlp, Z
                   SUBC (:BLCK)
                   S = last symbol
                U, S - 102, Z           " last symbol = quote ?
                N, GOTO (:ELSE0)
                   A = M1
                   S = d24              " name list[proc identifier + 1] =
```

```
                  MA[-1] + S          " name list[proc identifier + 1] + d24
NEXT4:            SUBC (:NXT SBL)
                  S - 103, Z          " next symbol = unquote ?
               N, GOTO (:NEXT4)
                  JUMP (7)
ELSE0:         U, S - 104, Z          " last symbol = begin ?
               N, SUBC (:STATMNT)
               N, GOTO (:END)
                  SUBC (:NXT SBL)
                  SUBC (:DECL LS)     " declarator last symbol ?
               Y, SUBC (:DECL LST)
                  SUBC (:CMP TL)
                  SUBC (:NXT SBL)
END:              S = nlp
                  S - correction
                  LUS (13)            " 8192 x nlp (as status)
                  S + MC[-1]          " + prc level
                  S + MC[-1]          " + string occurrence
                  MD[-2] = S
                  S = MC[-1]          " internal block depth
                  S +  1
                  LUS (6)
                  MD[-1] + S
                  S = M[B+2], Z       " prc level = 0 ?
                  S = MC[-2]          " local count
               N, B -  1
               Y, S +  MC             " + label count
                  S + MC[-1]          " + max for count
               Y, global count + S
               N, LUS (13)
               N, MD[-1] + S
                  S =  MC, Z          " max for count = 0 ?
               N, M[7] = S
                  S = nlp
               N, A = d19
NEXT5:         N, MS =  A             " name list[nlp] = d19
               N, S -  1              " nlp = nlp + 1
               N, REPP (:NEXT5)
                  A = d25
                  A -  5
                  A +  D              " name list[nlp] =
                  MS = - A            " - 33554437 - block cell pointer
                  S -  1
                  nlp = S             " nlp = nlp + 1
                  S + d25             " name list[block cell pointer - 1] =
                  MD[1] = - S         "                  - 33554432 - nlp
                  S = MC[-1]          " dump1
                  A = MS[-1]
                  dsp lvl = A         " displ level = name list[dump1 + 1]
                  D =  S              " block cell pointer = dump1
                  A = MC[5]           " internal block depth
                  A +  1
                  GOTOR (MC[-1])      " 187 instructions

                  'END' BLOCK
```

```
CMP TL:          SUBC (:STATMNT)
                 S = last symbol
               U, S - 91, Z          " last symbol = semicolon ?
               N, GOTOR (MC[-1])
                 SUBC (:NXT SBL)
                 GOTO (:CMP TL)      " 6 instructions

DECL LST:        'BEGIN' STRINGTEST, END, ELSE0, ELSE1, ELSE2, NEXT1,
                       NEXT2, NEXT3, NEXT4, NEXT5

                 S = tp dec ls, Z    " type declarator last symbol ?
               N, GOTO (:ELSE0)
                 MC = S              " count = 0
NEXT1:           S = 1
                 M[B-1] + S          " count = count + 1
                 SUBC (:IDF)
                 S - last nlp, P     " n < last nlp ?
               Y, A = 122
               Y, SUBC (:DERRORM)
                 S = last symbol
               U, S - 87, Z          " last symbol = comma ?
               Y, SUBC (:NXT SBL)
               Y, GOTO (:NEXT1)
                 S = MC[-1]
                 A = type, Z         " type = 0 ?
               N, A - 3, Z           " V type = 3 ?
               Y, LUS (1)            " then count = 2 X count
                 A = own type, Z
               Y, global count + S
               N, M1[4] + S          " local count = local count + count
STRINGTEST:      A = type
                 A - 3, Z            " type = 3 ?
               Y, A = 64
               Y, M1[7] = A          " then stringoccurrence = 64
                 GOTO (:END)
ELSE0:           S = arr dec ls, Z   " arr declarator last symbol ?
               N, GOTO (:ELSE1)
                 MC = S              " count = 0
                 arr pointer = - S   " array pointer = 0
NEXT2:           S = 1
                 M[B-1] + S          " count = count + 1
                 SUBC (:NXT SBL)
                 SUBC (:IDF)
                 S - last nlp, P     " n < last nlp ?
               Y, A = 123
               Y, SUBC (:DERRORM)
                 S = nlp
                 A = arr pointer
                 MS = A              " name list[nlp] = array pointer
                 arr pointer = S     " array pointer = nlp
                 S - 1
                 nlp = S             " nlp = nlp + 1
                 S = last symbol
               U, S - 87, Z          " last symbol = comma ?
               Y, GOTO (:NEXT2)
                 A = 0
                 dimension = A       " dimension = 0
               U, S - 100, Z         " last symbol = sub ?
```

```
                N, A = 125
                N, SUBC (:ERRORM)
                N, GOTO (:NEXT4)
                   subcount = A          " subcount = 0
NEXT3:             SUBC (:NXT SBL)
                U, S = letter last symbol, Z
                Y, SUBC (:SKP IDF)
                Y, JUMP (3)
                U, S = digit last symbol, Z
                Y, SUBC (:UNS NBR)
                Y, SUBC (:ST NUM CS)
                Y, S = last symbol
                U, S - 102, Z          " last symbol = quote ?
                Y, SUBC (:SKIP STRING)
                   A =   1
                U, S -  90, Z          " last symbol = colon ?
                Y, dimension + A       " then dimension = dimension + 1
                Y, GOTO (:NEXT3)
                U, S - 100, Z          " last symbol = sub ?
                Y, subcount + A        " then subcount = subcount + 1
                Y, GOTO (:NEXT3)
                U, S - 101, Z          " last symbol = bus ?
                N, GOTO (:NEXT3)
                   S = subcount, Z     " subcount = 0 ?
                N, subcount - A        " else subcount = subcount - 1
                N, GOTO (:NEXT3)
                   S = dimension, Z    " dimension = 0 ?
                Y, A = 124
                Y, SUBC (:ERRORM)
                N, dimension + A       " else dimension = dimension + 1
                   SUBC (:NXT SBL)
NEXT4:             G = dimension
                   S = arr pointer
                   A =  MS             " n = name list[array pointer]
                   MS =  G             " name list[array pointer] = dimension
                   S =   A, Z          " array pointer = n
                N, GOTO (:NEXT4[2])
                   S = MC[-1]          " count
                U, S = own type, Z
                Y, F X   3
                Y, F +   3
                Y, MULS (G)
                Y, global count + S
                N, M1[4] + S
                   S = last symbol
                   S -  87, Z          " last symbol = comma ?
                Y, GOTO (:NEXT2[-2])
                   GOTO (:STRINGTEST)
ELSE1:             S = last symbol
                U, S - 113, Z          " last symbol = switch ?
                   SUBC (:NXT SBL)
                   SUBC (:IDF)
                N, GOTO (:ELSE2)
                   S - last nlp, P     " n < last nlp ?
                Y, A = 126
                Y, SUBC (:DERRORM)
                   A =   0
                   S =   1
                   MINS (nlp)          " nlp = nlp + 1
```

```
                  MS[1] = A            " name list[nlp - 1] = 0
NEXT5:            SUBC (:NXT SBL)
               U, S = letter last symbol, Z
               Y, SUBC (:SKP IDF)
               Y, JUMP (3)
               U, S = digit last symbol, Z
               Y, SUBC (:UNS NBR)
               Y, SUBC (:ST NUM CS)
               Y, S = last symbol
               U, S - 102, Z           " last symbol = quote ?
               Y, SUBC (:SKIP STRING)
               U, S -  91, Z           " last symbol = semicolon ?
               N, GOTO (:NEXT5)
                  GOTO (:END)
ELSE2:            S - last nlp, P      " n < last nlp ?
               Y, A = 127
               Y, SUBC (:ERRORM)
                  A = nlp
                  S = type
               U, S -   3, P           " type > 3 ?
               N, LUS (19)
               N, MA[-1] = S           " else name list[nlp + 1] = type × d19
                  S = :MA[-1]
               N, S -   1
                  nlp = S              " nlp = nlp + (1 or 2)
                  A + 1
                  SUBC (:BLOCK)
END:              S = last symbol
               U, S -  91, Z           " last symbol = semicolon ?
               Y, SUBC (:NXT SBL)
               N, A = 128
               N, SUBC (:ERRORM)
                  SUBC (:DECL LS)
               Y, GOTO (:DECL LST)
                  GOTOR (MC[-1])       " 142 instructions

                  'END' DECL LST
```

```
STATMNT:        'BEGIN' END, ELSE0, ELSE1, ELSE2, ELSE3, NEXT1

                A = M1[2]
                MC = A                  " lfc = local for count
                A =  6
                LUA (19)
                character = A           " character = 6 × d19
NEXT1:          S = letter last symbol, Z
              N, GOTO (:ELSE0)
                SUBC (:RD IDF)
                A = last symbol
              U, A - 90, Z              " last symbol = colon ?
              Y, SUBC (:PRCS IDF)
              Y, SUBC (:LAB DEC)
              Y, GOTO (:NEXT1)
ELSE0:          S = digit last symbol, Z
              N, GOTO (:ELSE1)
                SUBC (:UNS NBR)
                A = last symbol
              U, A - 90, Z              " last symbol = colon ?
              Y, SUBC (:ILAB DEC)
              Y, GOTO (:NEXT1)
                SUBC (:ST NUM CS)
ELSE1:          S = last symbol
              U, S - 82, Z              " last symbol = for ?
              N, GOTO (:ELSE2)
                S =  1
                PLUSS (M1[2])           " local for count = local for count + 1
              U, S - M1[3], P           " local for count > max for count ?
              Y, M1[3] = S              " then max for count = local for count
                GOTO (:END)
ELSE2:        U, S - 104, Z             " last symbol = begin ?
              Y, SUBC (:BEGIN STAT)
              Y, GOTO (:END)
ELSE3:        U, S - 102, Z             " last symbol = quote ?
              Y, SUBC (:SKIP STRING)
              U, S - 91, Z              " last symbol = semicolon ?
              N, S - 105, Z             " ∨ last symbol = end ?
              Y, S = MC[-1]
              Y, M1[2] = S              " local for count = lfc
              Y, GOTOR (MC[-1])
END:            SUBC (:NXT SBL)
                GOTO (:NEXT1)           " 41 instructions

                'END' STATMNT


BEGIN STAT:     SUBC (:NXT SBL)
                SUBC (:DECL LS)         " declarator last symbol ?
              N, GOTO (:CMP TL)
                A =  0
                SUBC (:BLOCK)           " n = block (0)
              U, A - M1[6], P           " n > internal block depth ?
              Y, M1[6] = A              " then internal block depth = n
                GOTOR (MC[-1])          " 8 instructions
```

```
LAB DEC:        S - last nlp, P      " n < last nlp ?
              Y, A = 129
              Y, SUBC (:DERRORM)
                 A = M1[5], Z        " label count = 0 ?
                 A +   2
                 M1[5] = A           " label count = label count + 2
                 S = nlp
              Y, A = :MS[1]
              Y, A - correction
              Y, LUA (13)            " name list[block cell pointer + 3] =
              Y, MD[-3] = A          "                  8192 × (nlp - 1)
                 A = d18
                 MS =  A             " name list[nlp] = d18
                 S -   1
                 nlp = S             " nlp = nlp + 1
                 GOTO (:NXT SBL)     " 16 instructions

ILAB DEC:       S = real number, Z
              Y, A = 130
              Y, SUBC (:ERRORM)
              Y, GOTO (:NXT SBL)
                 S =   0
                 int labels = S      " int labels = true
                 SUBC (:IN NM LST)
                 S =   3
                 nlp - S             " nlp = nlp + 3
                 S = int lab
                 GOTO (:LAB DEC)     " 11 instructions

ST NUM CS:      S = small, Z
              N, F = value of constant
              N, S =   2
                 PLUSS (instr cntr)  " instruct counter = instruct counter+2
              N, MS[-2] = F
                 GOTOR (MC[-1])      " 6 instructions

IDF:            SUBC (:RD IDF)
PRCS IDF:       S = nlp
                last nlp = S         " last nlp = nlp
                S - word count
                S -   2
                nlp = S              " nlp = nlp + word count + 2
                A = character
                MS[1] = A            " name list[nlp - 1] = character
                GOTO (:LOOK UP)      " 9 instructions
```

```
PRESCAN 0:      A = start of text array
                begin of text array = A
                S = 100
                SUBC (:INIT)
                dsp lvl = G             " displ level = 0
                A = begin of nli        " name list[begin of name list - 1] =
                MA[1] = B               "                 address of dump0
                MC =  F                 " dump0 = dump1 = 0
                MC =  F                 " local for count = max for count = 0
                MC =  F                 " local count = label count = 0
                MC =  G                 " internal block depth = 0
                MC =  F                 " string occurrence = prc level = 0
                global count = G        " global count = 0
                int labels = S          " int labels = false
                text in array = G       " text in text array = true
                D =   A                 " block cell pointer = nlp
                old bcp = A             " old block cell pointer = nlp
                MA[-1] = F              " name list[nlp] = name list[nlp+1] = 0
                MA[-3] = G              " name list[nlp + 3] = 0
                S = :MA[-6]
                nlp = S                 " nlp = nlp + 6
                S + d25                 " name list[block cell pointer + 4] =
                MA[-4] = - S            "                 - 33554432 - nlp
                shift = A               " make shift > 256
                SUBC (:NXT SBL)
                S = begin of pr ar      " instruct counter =
                instr cntr = S          "                 begin of program array
                SUBC (:PROGRAM)
                S = global count
                S + M1[3]
                S + M1[5]
                S + M1[6]               " internal block depth
                LUS (7)                 " x 128
                S + M1[6]               " internal block depth
                LUS (6)                 " x 64 (therefore x 8192 in total)
                S + 8256                " + 8256
                MD[-1] = S
                S = nlp
                S - correction
                LUS (13)                " name list[block cell pointer + 2] =
                MD[-2] = S              "                 8192 x nlp
                S = M1[3], Z            " max for count
             N, COUNT = S
                S = nlp
             N, A = d19
PRESCAN 0[45]: N, MS =  A               " name list[nlp] = d19
             N, S -  1
             N, REPP (:PRESCAN 0[45])
             N, nlp = S                 " nlp = nlp + max for count
                S + d25                 " name list[block cell pointer + 5] =
                MD[-5] = - S            "                 - 33554432 - nlp
                B -  9
                SUBC (:TEST TEXT IN ARRAY)
                GOTO (:PRESCAN 1)       " 54 instructions

'END'
```

```
'BEGIN'         ARITHEXP, S ARITHEXP, SUBS VAR, SBS LST, BOOLEXP, S BOOLEXP,
                STREXP, S STREXP, DESEXP, S DESEXP, EXP, TYPE EXP, S EXP,
                RST OF EXP, ASS STAT, RH SIDE, INSERT, FCT DES, PAR LST,
                ACT PAR, PRC STAT, STATMNT, GT STAT, CMP TL, IFCLAUSE,
                FOR STAT, SW DECL, SW LIST, ARR DEC, BND PR LST, PRC DEC,
                BLOCK, DECL LST, PROGRAM, LAB STAT, ILAB STAT, LAB DEC,
                ADDR BL IDF, STC ADDR, ADD TYPE, BOOLEAN, STRING, ARMETIC,
                ARBOST, DESINAL, ASS TO, SBS VAR, PROC, FNCTN, LST LTH,
                SW LTH, ADDRSS, CHK DIM, IDF, SCAN CODE, ASK LIBR


ARITHEXP:
S ARITHEXP:         'BEGIN' ELSEO, END

                    SUBC (:AR OP LS)      " arithoperator last symbol ?
                Y,  SUBC (:NXT SBL)
                U,  S - 98, Z             " last symbol = open ?
                N,  GOTO (:ELSEO)
                    SUBC (:NXT SBL)
                    SUBC (:ARITHEXP)
                    S = last symbol
                U,  S - 99, Z             " last symbol = close ?
                Y,  SUBC (:NXT SBL)
                    GOTO (:END)
ELSEO:          U,  S - 94, Z             " last symbol = if ?
                Y,  A = :ARITHEXP
                Y,  GOTO (:IFCLAUSE)
                    A = digit last symbol, Z
                Y,  SUBC (:UNS NBR)
                Y,  GOTO (:END)
                    A = letter last symbol, Z
                Y,  SUBC (:IDF)
                Y,  SUBC (:ARMETIC)
                Y,  SUBC (:SUBS VAR)
                Y,  SUBC (:FCT DES)
END:                SUBC (:AR OP LS)
                Y,  GOTO (:ARITHEXP[1])
                    GOTOR (MC[-1])        " 24 instructions

                    'END' S ARITHEXP


SUBS VAR:           A = last symbol
                    A - 100, Z            " last symbol = sub ?
                N,  GOTOR (MC[-1])
                    MC = S                " n
                    SUBC (:SBS VAR)
                    SUBC (:SBS LST)
                    S = MC[-1]            " n
                    GOTO (:LST LTH)       " 8 instructions


SBS LST:            SUBC (:NXT SBL)
                    SUBC (:ARITHEXP)
                    S = last symbol
                U,  S - 87, Z             " last symbol = comma ?
                Y,  SUBC (:SBS LST)
SBS LST[5]:     Y,  A + 1
                Y,  GOTOR (MC[-1])
```

```
                U, S - 101, Z          " last symbol = bus ?
SBS LST[8]:     Y, SUBC (:NXT SBL)
                   A =  1
                   GOTOR (MC[-1])      " 11 instructions


BOOLEXP:           'BEGIN' ELSE0, ELSE1

                   S = last symbol
                U, S -  94, Z          " last symbol = if ?
                Y, A = :BOOLEXP
                Y, GOTO (:IFCLAUSE)
S BOOLEXP:         S = last symbol
                U, S -  76, Z          " last symbol = non ?
                Y, SUBC (:NXT SBL)
                U, S -  98, Z          " last symbol = open ?
                N, GOTO (:ELSE0)
S BOOLEXP[5]:      SUBC (:NXT SBL)
                   SUBC (:EXP)
                   S = last symbol
                U, S -  99, Z          " last symbol = close ?
                Y, MC =  A             " type
                Y, SUBC (:NXT SBL)
                Y, A = MC[-1]          " type
                   GOTO (:RST OF EXP)
ELSE0:             A = letter last symbol, Z
                N, GOTO (:ELSE1)
                   SUBC (:IDF)
                   SUBC (:SUBS VAR)
                   SUBC (:FCT DES)
                   A = last symbol
                U, A -  63, P          " last symbol not an
                U, A -  75, E          " arithoperator or a relatoperator ?
                N, SUBC (:ARMETIC)
                Y, SUBC (:BOOLEAN)
                   GOTO (:RST OF EXP)
ELSE1:          U, S - 116, Z          " last symbol = true ?
                N, S - 117, Z          " ∨ last symbol = false ?
                Y, SUBC (:NXT SBL)
                Y, GOTO (:RST OF EXP)
                   S +  53, Z          " last symbol = plus ?
                N, S -   1, Z          " ∨ last symbol = minus ?
                N, S = digit last symbol, Z  " ∨ digit last symbol ?
                Y, SUBC (:S ARITHEXP)
                   GOTO (:RST OF EXP)  " 37 instructions

                   'END' BOOLEXP


STREXP:            S = last symbol
                U, S -  94, Z          " last symbol = if ?
                Y, A = :STREXP
                Y, GOTO (:IFCLAUSE)
S STREXP:          S = last symbol
                U, S -  98, Z          " last symbol = open ?
                N, JUMP (6)
                   SUBC (:NXT SBL)
                   SUBC (:STR EXP)
S STREXP[5]:       S = last symbol
```

```
                U, S - 99, Z            " last symbol = close ?
S STREXP[7]:    Y, GOTO (:NXT SBL)
                   GOTOR (MC[-1])
                   A = letter last symbol, Z
                Y, SUBC (:IDF)
                Y, SUBC (:STRING)
                Y, SUBC (:SUBS VAR)
                Y, GOTO (:FCT DES)
                U, S - 102, Z           " last symbol = quote ?
                Y, SUBC (:SKIP STRING)
                   GOTO (:S STREXP[7])  " 21 instructions


DESEXP:            S = last symbol
                U, S - 94, Z            " last symbol = if ?
                Y, A = :DESEXP
                Y, GOTO (:IFCLAUSE)
S DESEXP:          S = last symbol
                U, S - 98, Z            " last symbol = open ?
                Y, SUBC (:NXT SBL)
                Y, SUBC (:DESEXP)
                Y, GOTO (:S STREXP[5])
                   A = letter last symbol, Z
                Y, SUBC (:IDF)
                Y, SUBC (:DESINAL)
                Y, GOTO (:SUBS VAR)
                   A = digit last symbol, Z
                N, GOTOR (MC[-1])
                   SUBC (:UNS NBR)
                   SUBC (:IN NM LST)
                N, GOTOR (MC[-1])
                   S = int lab
                   GOTO (:DESINAL)      " 20 instructions


EXP:               S = last symbol
                U, S - 94, Z            " last symbol = if ?
                N, GOTO (:S EXP)
                   SUBC (:NXT SBL)
                   SUBC (:BOOLEXP)
                   SUBC (:NXT SBL)
                   SUBC (:S EXP)
                   S = last symbol
                U, S - 96, Z            " last symbol = else ?
                N, GOTOR (MC[-1])
                   MC = A               " type
                   SUBC (:NXT SBL)
                   A = MC[-1]           " type
TYPE EXP:          S = - A
                   S 'X' 5, Z           " type = 5 V type = 7 ?
                N, MC = A               " type
                   A + :TYPE EXP[7]
                   SUBC (:MA)
                N, A = MC[-1]           " restore type
                   GOTOR (MC[-1])
TYPE EXP[7]:       GOTO (:ARITHEXP)
                   GOTO (:ARITHEXP)
                   GOTO (:BOOLEXP)
                   GOTO (:STREXP)
```

```
                  GOTO (:ARITHEXP)
                  GOTO (:EXP)
                  GOTO (:DESEXP)
                  GOTO (:EXP)          " 28 instructions

S EXP:            'BEGIN' ELSE0, ELSE1

                  S = last symbol
                U, S - 98, Z           " last symbol = open ?
                Y, GOTO (:S BOOLEXP[5])
                  A = letter last symbol, Z
                N, GOTO (:ELSE0)
                  SUBC (:IDF)
                  SUBC (:SUBS VAR)
                  SUBC (:FCT DES)
                  A = last symbol
                U, A - 63, P           " last symbol not an
                U, A - 75, E           " arithoperator or a relatoperator ?
                N, SUBC (:ARMETIC)
                N, GOTO (:RST OF EXP)
                U, A - 80, E           " and not a booloperator ?
                N, SUBC (:BOOLEAN)
                N, GOTO (:RST OF EXP)
                  SUBC (:NON FRM LAB)
                Y, SUBC (:DESINAL)
                  A = MS               " name list[n]
                  RUA (19)
                  A 'X' 7              " type bits (n)
                  GOTOR (MC[-1])
ELSE0:            A = digit last symbol, Z
                N, GOTO (:ELSE1)
                  SUBC (:UNS NBR)
                  SUBC (:IN NM LST)
                Y, S = int lab
                Y, SUBC (:DESINAL)
                Y, A = 7               " type = un
                N, A = 4               " type = ar
                  GOTO (:RST OF EXP)
ELSE1:            S = last symbol
                U, S - 63, P           " last symbol not
                U, S - 65, E           " a plus or minus sign ?
                N, SUBC (:S ARITHEXP)
                N, GOTO (:ELSE1[-2])
                U, S - 102, Z          " last symbol = quote ?
                Y, SUBC (:S STREXP)
                Y, A = 3               " type = st
                Y, GOTOR (MC[-1])
                U, S - 76, Z           " last symbol = non ?
                N, S - 116, Z          " V last symbol = true ?
                N, S - 1, Z            " V last symbol = false ?
                Y, SUBC (: S BOOLEXP)
                Y, A = 2               " type = bo
                N, A = 7               " type = un
                  GOTO (:RST OF EXP)   " 47 instructions

                  'END' S EXP
```

```
RST OF EXP:       SUBC (:AR OP LS)
                Y, SUBC (:S ARITHEXP[1])
                Y, A = 4                " type = ar
                   SUBC (:REL OP LS)
                Y, SUBC (:NXT SBL)
                Y, SUBC (: S ARITHEXP)
                Y, A = 2                " type = bo
                   SUBC (:BOOL OP LS)
                Y, SUBC (:NXT SBL)
                Y, SUBC (: S BOOLEXP)
                Y, A = 2                " type = bo
                   GOTOR (MC[-1])       " 12 instructions

ASS STAT:         'BEGIN' ELSE0, ELSE1, ENDO, END1

                   SUBC (:SUBS VAR)
                   A = last symbol
                U, A - 92, Z            " last symbol = colonequal ?
                N, GOTOR (MC[-1])
RH SIDE:           MC = S               " n
                   SUBC (:ASS TO)
                   A = MS
                   RUA (19)
                   A 'X' 7
                   MC = A               " type n = type bits (n)
                   SUBC (:NXT SBL)
                   A = letter last symbol, Z
                N, GOTO (:ELSE1)
                   SUBC (:IDF)
                   SUBC (:SUBS VAR)
                   A = last symbol
                U, A - 92, Z            " last symbol = colonequal ?
                N, GOTO (:ELSE0)
                   A = M[B-1]           " type n
                   SUBC (:INSERT)
                   SUBC (:RH SIDE)
                   A = MS
                   RUA (19)
                   A 'X' 7              " type = type bits (m)
                   GOTO (:END1)
ELSE0:             SUBC (:FCT DES)
                   A = last symbol
                U, A - 63, P            " last symbol not an
                U, A - 75, E            " arithoperator or a relatoperator ?
                N, SUBC (:ARMETIC)
                N, GOTO (:ENDO)
                U, A - 80, E            " and not a booloperator ?
                N, SUBC (:BOOLEAN)
                N, GOTO (:ENDO)
                   SUBC (:ARBOST)
                   A = M[B-1]           " type = type n
                U, A - 1, P             " type ≠ re ∧ type ≠ in ?
                N, A = 4                " else type = ar
                   SUBC (:INSERT)
                   A = MS
                   RUA (19)
                   A 'X' 7              " type = type bits (m)
                U, A - 1, P             " type ≠ re ∧ type ≠ in ?
```

```
                N, A =   4              " else type = ar
ENDO:              SUBC (:RST OF EXP)
                   GOTO (:END1)
ELSE1:             A = M[B-1]           " type n
                U, A -   5, Z           " = nondes ?
                   SUBC (:TYPE EXP)
                N, A = M[B-1]           " else leave type n unchanged
                U, A -   1, P           " type ≠ re ∧ type ≠ in ?
                N, A =   4              " else type = ar
END1:              B -   1
                   S = MC[-1]           " n
                   GOTO (:INSERT)       " 55 instructions

                   'END' ASS STAT


INSERT:            GOTO (:ADD TYPE)     " 1 instruction


FCT DES:           A = last symbol
                U, A -  98, Z           " last symbol = open ?
                N, GOTOR (MC[-1])
                   SUBC (:FNCTN)
FCT DES[4]:        MC =  S              " n
                   SUBC (:PAR LST)
                   S = MC[-1]           " n
                   GOTO (:LST LTH)      " 8 instructions


PAR LST:           SUBC (:NXT SBL)
                   SUBC (:ACT PAR)
                   S = last symbol
                U, S -  87, Z           " last symbol = comma ?
                Y, SUBC (:PAR LST)
                Y, GOTO (:SBS LST[5])
                U, S -  99, Z           " last symbol = close ?
                   GOTO (:SBS LST[8])   " 8 instructions


ACT PAR:           GOTO (:EXP)          " 1 instruction


PRC STAT:          SUBC (:PROC)
                   A = last symbol
                U, A -  98, Z           " last symbol = open ?
                Y, GOTO (:FCT DES[4])
                   A =   0
                   GOTO (:LST LTH)      " 6 instructions


STATMNT:           'BEGIN' ELSE0, ELSE1

                   A = letter last symbol, Z
                N, GOTO (:ELSE0)
                   SUBC (:IDF)
                   A = last symbol
                U, A -  90, Z           " last symbol = colon ?
                Y, GOTO (:LAB STAT)
                U, A - 100, Z           " last symbol = sub ?
```

```
                N, A - 92, Z            " V last symbol = colonequal ?
                Y, GOTO (:ASS STAT)
                   GOTO (:PRC STAT)
ELSE0:             A = digit last symbol, Z
                N, GOTO (:ELSE1)
                   SUBC (:UNS NBR)
                   A = last symbol
                U, A - 90, Z            " last symbol = colon ?
                Y, GOTO (:ILAB STAT)
                   GOTOR (MC[-1])
ELSE1:             S = last symbol
                U, S - 81, Z            " last symbol = goto ?
                Y, GOTO (:GT STAT)
                U, S - 94, Z            " last symbol = if ?
                Y, A = :STATMNT
                Y, GOTO (:IFCLAUSE)
                U, S - 82, Z            " last symbol = for ?
                Y, GOTO (:FOR STAT)
                U, S - 104, Z           " last symbol = begin ?
                N, GOTOR (MC[-1])
                   SUBC (:NXT SBL)
                   SUBC (:DECL LS)      " declarator last symbol ?
                Y, SUBC (:BLOCK)
                N, SUBC (:CMP TL)
                   GOTO (:NXT SBL)      " 32 instructions

                   'END' STATMNT


GT STAT:           SUBC (:NXT SBL)
                   A = letter last symbol, Z
                N, GOTO (:DESEXP)
                   SUBC (:IDF)
                   SUBC (:LOC LAB)
                Y, GOTOR (MC[-1])
                   SUBC (:DESINAL)
                   GOTO (:SUBS VAR)     " 8 instructions


CMP TL:            SUBC (:STATMNT)
                   S = last symbol
                U, S - 91, Z            " last symbol = semicolon ?
                Y, SUBC (:NXT SBL)
                Y, GOTO (:CMP TL)
                U, S - 105, Z           " last symbol = end ?
                Y, GOTOR (MC[-1])
                   A = :STATMNT
                   SUBC (:SKIP RST OF STAT)
                   GOTO (:CMP TL[1])    " 10 instructions


IFCLAUSE:          MC = A               " pr
                   SUBC (:NXT SBL)
                   SUBC (:BOOLEXP)
                   S = last symbol
                U, S - 95, Z            " last symbol = then ?
                Y, SUBC (:NXT SBL)
                   SUBC (M[B-1])        " pr
                   S = last symbol
```

```
                U, S - 96, Z            " last symbol = else ?
                Y, SUBC (:NXT SBL)
                Y, GOTO (MC[-1])        " then pr once more
                   B - 1
                   GOTOR (MC[-1])       " 13 instructions

FOR STAT:          'BEGIN' NEXTO, ELSEO

                   SUBC (:NXT SBL)
                   A = letter last symbol, Z
                N, GOTOR (MC[-1])
                   SUBC (:IDF)
                   SUBC (:ARMETIC)
                   SUBC (:SUBS VAR)
                   SUBC (:ASS TO)
                   S = last symbol
                U, S - 92, Z            " last symbol = colonequal ?
NEXTO:          Y, SUBC (:NXT SBL)
                   SUBC (:ARITHEXP)
                   S = last symbol
                U, S - 110, Z           " last symbol = step ?
                N, GOTO (:ELSEO)
                   SUBC (:NXT SBL)
                   SUBC (:ARITHEXP)
                   S = last symbol
                U, S - 84, Z            " last symbol = until ?
                Y, SUBC (:NXT SBL)
                Y, SUBC (:ARITHEXP)
                   JUMP (3)
ELSEO:          U, S - 85, Z            " last symbol = while ?
                Y, SUBC (:NXT SBL)
                Y, SUBC (:BOOLEXP)
                   S = last symbol
                U, S - 87, Z            " last symbol = comma ?
                Y, GOTO (:NEXTO)
                U, S - 86, Z            " last symbol = do ?
                Y, SUBC (:NXT SBL)
                   A = 1
                   forcnt + A           " forcount = forcount + 1
                   SUBC (:STATMNT)
                   A = 1
                   forcnt - A           " forcount = forcount - 1
                   GOTOR (MC[-1])       " 35 instructions

                   'END' FOR STAT

SW DECL:           SUBC (:NXT SBL)
                   A = letter last symbol, Z
                N, GOTOR (MC[-1])
                   SUBC (:IDF)
                   A = last symbol
                U, A - 92, Z            " last symbol = colonequal ?
                N, GOTOR (MC[-1])
                   MC = S               " n
                   SUBC (:SW LIST)
                   S = MC[-1]           " n
                   GOTO (:SW LTH)       " 11 instructions
```

```
SW LIST:        SUBC (:NXT SBL)
                SUBC (:DESEXP)
                S = last symbol
             U, S - 87, Z          " last symbol = comma ?
             Y, SUBC (:SW LIST)
             Y, A + 1
             N, A = 1
                GOTOR (MC[-1])     " 8 instructions

ARR DEC:        'BEGIN' NEXTO, NEXT1, ELSEO

                SUBC (:NXT SBL)
                SUBC (:IDF)
                A = 1
                MC = A             " count = 1
                MC = S             " n
NEXTO:          S = last symbol
             U, S - 87, Z          " last symbol = comma ?
             Y, SUBC (:NXT SBL)
             Y, SUBC (:SKP IDF)
             Y, A = 1
             Y, M[B-2] + A         " count = count + 1
             Y, GOTO (:NEXTO)
                S - 100, Z         " last symbol = sub ?
             Y, in ar dec = S      " in array declaration = true
             Y, SUBC (:BND PR LST)
             Y, in ar dec = A      " in array declaration = false
             N, A = 0
                dimension = A
                S = MC[-1]         " n
                SUBC (:CHK DIM)
                A = own type, Z
                A = MC[-1]         " count
             N, GOTO (:ELSEO)
                COUNT = A
NEXT1:          A = instr cntr
                SUBC (:ADDRSS)
                A = dimension
                A + :MA[6]
                A + dimension      " 3 × dimension + 6
                instr cntr + A     " added to instruct counter
                SUBC (:NXT IDF)
                REPP (:NEXT1)
ELSEO:          S = last symbol
             U, S - 87, Z          " last symbol = comma ?
             Y, GOTO (:ARR DEC)
                GOTOR (MC[-1])     " 36 instructions

                'END' ARR DEC

BND PR LST:     SUBC (:NXT SBL)
                SUBC (:ARITHEXP)
                S = last symbol
             U, S - 90, Z          " last symbol = colon ?
             Y, SUBC (:NXT SBL)
             Y, SUBC (:ARITHEXP)
                S = last symbol
```

```
                U, S - 87, Z            " last symbol = comma ?
                Y, SUBC (:BND PR LST)
                   GOTO (:SBS LST[5])   " 10 instructions

PRC DEC:           'BEGIN' NEXTO, ELSEO, END

                   SUBC (:NXT SBL)
                   SUBC (:IDF)
                   MC = S               " n
                   SUBC (:ENTR BLK)
                   S = last symbol
                U, S - 98, Z            " last symbol = open ?
                N, GOTO (:ELSEO)
NEXTO:             A = 0
                   in formal list = A   " in formal list = true
                   SUBC (:NXT SBL)
                   SUBC (:IDF)
                   A = MS
                   RUA (19)
                   A - 95, Z            " name list[m] = 95 × d19 ?
                Y, A = d24              " 32 × d19
                Y, MS + A               " name list[m] = 127 × d19
                   A = 201
                   in formal list = A   " in formal list = false
                Y, SUBC (:ERRORM)
                   S = last symbol
                U, S - 87, Z            " last symbol = comma ?
                Y, GOTO (:NEXTO)
                U, S - 99, Z            " last symbol = close ?
                Y, SUBC (:NXT SBL)
ELSEO:          U, S - 91, Z            " last symbol = semicolon ?
                Y, SUBC (:NXT SBL)
                   SUBC (:SKP VA LI)
                   SUBC (:SKP SP LI)
                   S = M[B-1]           " n
                   SUBC (:IN CODE)
                Y, B - 1
                Y, GOTO (:SCAN CODE)
                   A = 64
                U, A 'x' MD[-2], Z      " ¬ use of counterstack ?
                Y, S = MS
                Y, RUS (19)
                Y, S - 19, Z            " ∧ name list[n] : d19 = 19 ?
                Y, MD[-2] + A           " then add 64 to
                   S = last symbol      "      name list[block cell pointer+2]
                U, S - 104, Z           " last symbol = begin ?
                N, SUBC (:STATMNT)
                N, GOTO (:END)
                   SUBC (:NXT SBL)
                   SUBC (:DECL LS)      " declarator last symbol ?
                Y, SUBC (:DECL LST)
                   SUBC (:CMP TL)
                   SUBC (:NXT SBL)
END:               S = MC[-1]           " n
                   GOTO (:ADDR BL IDF)  " 49 instructions

                   'END' PRC DEC
```

```
BLOCK:          SUBC (:ENTR BLK)
                SUBC (:DECL LST)
                SUBC (:CMP TL)
                S =  0
                GOTO (:ADDR BL IDF)  " 5 instructions

DECL LST:       'BEGIN' END

                A = tp dec ls, Z     " type declarator last symbol ?
              Y, SUBC (:SKP TP DEC)
              Y, GOTO (:END)
                A = arr dec ls, Z    " arr declarator last symbol ?
              Y, SUBC (:ARR DEC)
              Y, GOTO (:END)
                S = last symbol
              U, S - 113, Z          " last symbol = switch ?
              Y, SUBC (:SW DECL)
              N, SUBC (:PRC DEC)
END:            S = last symbol
              U, S -  91, Z          " last symbol = semicolon ?
              Y, SUBC (:NXT SBL)
                 SUBC (:DECL LS)     " declarator last symbol ?
              Y, GOTO (:DECL LST)
                 GOTOR (MC[-1])      " 16 instructions

                 'END' DECL LST

PROGRAM:         A = letter last symbol, Z
              Y, SUBC (:IDF)
PROGRAM[2]:   Y, A = last symbol
              Y, A -  90, Z          " last symbol = colon ?
              Y, SUBC (:LAB DEC)
              Y, GOTO (:PROGRAM)
                 A = digit last symbol, Z
              Y, SUBC (:UNS NBR)
              Y, SUBC (:IN NM LST)   " in name list ?
              Y, S = int lab
              Y, GOTO (:PROGRAM[2])
                 S = last symbol
              U, S - 104, Z          " last symbol = begin ?
                 SUBC (:NXT SBL)
              N, JUMP (-3)           " skip until begin
                 SUBC (:DECL LS)     " declarator last symbol ?
              Y, GOTO (:BLOCK)
                 GOTO (:CMP TL)      " 18 instructions

LAB STAT:        SUBC (:NON FRM LAB)
              Y, SUBC (:LAB DEC)
                 GOTO (:STATMNT)     " 3 instructions

ILAB STAT:       SUBC (:IN NM LST)
              Y, S = int lab
                 GOTO (:LAB STAT[1]) " 3 instructions
```

```
LAB DEC:        A = MD[-2]
                A 'X' 63, Z          " proc level = 0 ?
                A = forcnt
                LUA (20)             " d20 X forcount
              Y, A + instr cntr      " + (possibly) instruct counter
                MS[-1] + A           " added to name list[n + 1]
              N, GOTO (:NXT SBL)
                SUBC (:DESINAL)
                A = instr cntr
                SUBC (:ADDRSS)
                A = instr cntr
                S =  0
                MA = S               " program array[instruct counter] = 0
                SUBC (:DISP LVL)
                LUS (18)
                S + dp0              " add dp0 + display level X d18 to
                MA[1] = S            " program array[instruct counter + 1]
                A +  2
                instr cntr = A       " instruct counter = instruct counter+2
                GOTO (:NXT SBL)      " 20 instructions

ADDR BL IDF:    'BEGIN' FORMALS, FOR VARS, LOCALS, TEST, ELSEO, END

                MC = S, Z            " n = 0 ?
              Y, S = 8192            " then add 8192 to
              Y, MD[-1] + S          " name list[block cell pointer + 1]
                A = MD[-2]
                A 'X' 63, Z          " proc level = 0 ?
              Y, B -  1
              Y, GOTO (:STC ADDR)
                A = M[B-1], Z        " n = 0 ?
                A = 257
              Y, A + d18             " 256 + 1 + d18 in A
              N, SUBC (:TP OF DSP)   " or top of display +
              N, A + :MS[-1]
                 SUBC (:DISP LVL)
              N, A +  S              " + display level + 256 in A
                 LUS (9)             " 512 X display level in S
                 S +  A
                 MC = S              " counter
              Y, GOTO (:FOR VARS)
                 S = :MD[-5]         " f = block cell pointer + 5
                 GOTO (:TEST)
FORMALS:         A = M[B-1]          " counter
                 SUBC (:ADDRSS)
                 RUA (18)            " code1
              U, A - 191, P          " code ≥ 96 ?
              N, JUMP (3)
              U, A 'X' 1, Z          " code1 = 2 X code ?
              N, A =  4
                 JUMP (3)
              U, A - 128, P          " code ≠
              U, A - 132, E          " 65 and ≠ 66 ?
              N, A =  1
              Y, A =  2
                 M[B-1] + A          " increase counter appropriately
TEST:            SUBC (:NXT IDF)     " f = next identifier (f)
              U, S -  D, P           " f < block cell pointer ?
```

```
                N, GOTO (:FORMALS)
                   A = d18
                   M[B-1] + A          " counter = counter + d18
                   S = M[B-2]          " n
                   A = MS
                   RUA (19)            " code = name list[n] : d19
                U, A - 24, Z
                Y, GOTO (:FOR VARS)
                U, A - 16, Z           " code = 16 ?
                N, A - 19, Z           " ∨ code = 19 ?
                   A = M[B-1]          " counter
                   S - 2
                   SUBC (:ADDRSS)
                Y, A = 2
                N, A = 1
                U, A = wanted, P       " wanted ?
                Y, A = 3
                   M[B-1] + A          " increase counter appropriately
                   LUA (13)            " increase appropriately
                   MD[-1] + A          " name list[block cell pointer + 1]
FOR VARS:          SUBC (:STATUS)      " f = status
                   A = M[B-1]          " counter
                U, A = MS, P           " name list[f] > 0 ?
                Y, MS + A              " address (f, counter)
                Y, A + 1               " counter = counter + 1
                Y, S - 1               " f = f + 1
                Y, GOTO (:FOR VARS[2])
                   M[B-1] = A          " counter
                   S = :MD[-4]         " f = block cell pointer + 4
LOCALS:            SUBC (:NXT IDF)     " f = next identifier (f)
                U, S - D, P            " f < block cell pointer ?
                N, A = MD[-2]
                N, RUA (13)
                N, A + correction
                N, A - S, P            " ∨ f > status ?
                N, A = MS
                N, RUA (19)
                N, A - 63, P           " ∨ name list[f] : d19 > 63 ?
                Y, GOTO (:END)
                   A + 39, P           " code > 24 ?
                N, GOTO (:ELSEO)
                   A - 11, P           " code > 35 ?
                Y, GOTO (:LOCALS)
                   A - 0, Z            " code = 35 ?
                N, A + 3, Z            " ∨ code = 32 ?
                   A = instr cntr
                   SUBC (:ADDRSS)
                Y, A = 2
                N, A = 1
                   instr cntr + A
ELSEO:             A + 11, P           " code > 13 ?
                Y, GOTO (:LOCALS)
                   G = A
                   F + 7, Z            " code = 6 ?
                Y, A = - MS[-1]
                Y, A 'X' d18, Z        " ∧ d18 of name list[f + 1] = 1 ?
                Y, GOTO (:LOCALS)
                   A = M[B-1]          " counter
                   SUBC (:ADDRSS)
```

```
                    A =   G, Z          " code = 6 ?
                N, A +   3, Z           " ∨ code = 3 ?
                N, A +   3, Z           " ∨ code = 0 ?
                Y, A =   2
                N, A =   1
                   M[B-1] + A           " increase counter appropriately
                   GOTO (:LOCALS)
END:               SUBC (:DISP LVL)
                   S +   1
                   LUS (9)              " 512 × (display level + 1)
                   S + d18              " + d18
                   S - MC[-1], P        " > counter ?
                N, A = 202
                N, SUBC (:ERRORM)
                   B -   1
                   GOTO (:EXIT BLK)     " 110 instructions

                   'END' ADDR BL IDF


STC ADDR:          'BEGIN' FOR VARS, LOCALS

                   SUBC (:STATUS)       " f = status
                   A = instr cntr
FOR VARS:       U, A =  MS, P           " name list[f] > 0 ?
                Y, MS +  A              " address (f, instruct counter)
                Y, A +   1              " instruct counter = instruct counter+1
                Y, S -   1              " f = f + 1
                Y, GOTO (:FOR VARS)
                   instr cntr = A
                   S = :MD[-4]          " f = block cell pointer + 4
LOCALS:            SUBC (:NXT IDF)      " f = next identifier (f)
                U, S -   D, P           " f < block cell pointer ?
                N, A = MD[-2]
                N, RUA (13)
                N, A + correction
                N, A -   S, P           " ∨ f > status ?
                Y, GOTO (:EXIT BLK)
                   A =  MS
                   RUA (19)             " code = name list[f] : d19
                U, A -  35, P           " code > 35 ?
                Y, GOTO (:LOCALS)
                U, A -  24, P           " code > 24 ?
                Y, JUMP (4)
                U, A -  13, P           " code > 13 ?
                Y, GOTO (:LOCALS)
                U, A -   6, Z           " code = 6 ?
                Y, GOTO (:LOCALS)
                   A 'X' 31, Z          " code = 0 (mod 32) ?
                N, A -   3, Z           " ∨ code = 3 (mod 32) ?
                   A = instr cntr
                   SUBC (:ADDRSS)
                Y, A =   2
                N, A =   1
                   instr cntr + A       " increase instruct counter appropriately
                   GOTO (:LOCALS)       " 34 instructions

                   'END' STC ADDR
```

```
ADD TYPE:          'BEGIN' ELSE0, END0, END1

                   M[B] = A              " t
                   A = MS
                   RUA (19)              " code = name list[n] : d19
                U, A - 95, P             " code > 95 ?
                N, GOTOR (MC[-1])
                U, A - 127, Z            " code = 127 ?
                Y, A = 31                " then (code - new code) = 31
                Y, JUMP (6)
                U, A - 120, Z            " code = 120 ?
                N, GOTO (:ELSE0)
                   A = M[B]              " t
                   A - 5, P              " t > 5 ?
                Y, GOTOR (MC[-1])        " then no gain of information
                   A = 8, P              " else (code - new code) = 8 (and YES)
                   A - M[B]              " - t
                   GOTO (:END1)
ELSE0:             A 'X' 7               " type = code - code : 8 x 8
                U, A - 7, Z              " type = 7 ?
                Y, GOTO (:END0)
                U, A - 5, Z              " type = 5 ?
                Y, GOTO (:END0)
                U, A - 4, Z              " type = 4 ?
                Y, A = 2
                Y, A - M[B], P           " ∧ t < 2 ?
                Y, A = 4
END0:           Y, A - M[B], P           " type > t ?
END1:           Y, LUA (19)
                Y, MS - A
                   GOTOR (MC[-1])        " 29 instructions

                   'END' ADD TYPE


BOOLEAN:           A = 2                 " bo
                   GOTO (:ADD TYPE)      " 2 instructions


STRING:            A = 3                 " st
                   GOTO (:ADD TYPE)      " 2 instructions


ARMETIC:           A = 4                 " ar
                   GOTO (:ADD TYPE)      " 2 instructions


ARBOST:            A = 5                 " nondes
                   GOTO (:ADD TYPE)      " 2 instructions


DESINAL:           SUBC (:NONFRM LAB)    " nonformal label ?
                N, A = 6                 " des
                N, GOTO (:ADD TYPE)
                   A = MS[-1]
                U, A 'X' d18, Z          " d18 of name list[n + 1] = 0 ?
                Y, GOTOR (MC[-1])        " then label known to be complicated
                   A 'X' - d18
                   MS[-1] = A            " d18 of name list[n + 1] = 0
```

```
                  SUBC (:CORSP BCP)   " p = corresponding block cell pointer
                  A = MG[-2]
                  A 'X' 63, Z         " corresponding proc level = 0 ?
               N, A =  1
               N, MG[-3] + A          " else add 1 to name list[p + 3]
               N, A = 16384
               N, MG[-1] + A          " and add 2 to corresponding local space
                  GOTOR (MC[-1])      " 16 instructions

ASS TO:           A =  MS
                  RUA (19)            " code = name list[n] : d19
               U, A - 95, P           " code > 95 ?
               N, GOTOR (MC[-1])
               U, A - 127, Z          " code = 127 ?
               Y, A = 101             " then code = 101
               U, A - 101, P          " code > 101 ?
               Y, GOTO (:ARBOST)
                  LUA (19)            " code x d19
                  A + d18             " + d18
                  MS =  A             " as new value of name list[n]
                  GOTOR (MC[-1])      " 12 instructions

SBS VAR:          A =  MS
                  RUA (19)            " code = name list[n] : d19
               U, A - 95, P           " code > 95 ?
               N, GOTOR (MC[-1])
               U, A - 127, Z          " code = 127 ?
               Y, A = - 16            " then (new code - code) = - 16
               Y, JUMP (2)
               U, A 'X' 24, Z         " code < 104 ?
               Y, A =  8              " then (new code - code) = 8
SBS VAR[9]:    Y, LUA (19)
               Y, MS + A
                  GOTOR (MC[-1])      " 12 instructions

PROC:             A =  MS
                  RUA (19)            " code = name list[n] : d19
               U, A - 95, P           " code > 95 ?
               N, GOTOR (MC[-1])
               U, A - 127, Z          " code = 127 ?
               Y, A = - 7             " then (new code - code) = - 7
               Y, GOTO (:SBS VAR[9])
               U, A - 101, P          " code > 101 ?
               Y, GOTOR (MC[-1])
                  A =  16, P
                  GOTO (:SBS VAR[9])  " 11 instructions

FNCTN:            SUBC (:ARBOST)
                  GOTO (:PROC)        " 2 instructions

LST LTH:          M[B] = A            " dimension
                  A =  MS
                  RUA (19)            " code = name list[n] : d19
               U, A - 95, P           " code > 95 ?
```

```
                N, GOTOR (MC[-1])
                   A = MS[-1]
                U, A = :MA, Z          " dimension not yet known ?
                Y, A + M[B]
                Y, A +   1
                Y, MS[-1] = A          " then add dimension + 1
                   GOTOR (MC[-1])      " 11 instructions

SW LTH:            A +   1
                   MS[-1] + A
                   GOTOR (MC[-1])      " 3 instructions

ADDRSS:            M[B] = A            " m
                   A =  MS             " name list[n]
                   RUA (18)
                   LUA (18)
                   A + M[B]
                   MS =  A
                   GOTOR (MC[-1])      " 7 instructions

CHK DIM:           A +   1
                   A - MS[-1], Z       " name list[n + 1] = dimension + 1 ?
                Y, GOTOR (MC[-1])
                   MS[-1] + A
                   A = 203
                   GOTO (:ERRORM)      " 6 instructions

IDF:               S = nlp
                   last nlp = S        " last nlp = nlp
                   SUBC (:RD IDF)
                   SUBC (:LOOK UP)
                U, S - nlp, P          " n < nlp ?
                Y, GOTOR (MC[-1])
                   SUBC (:ASK LIBR)
                U, S - nlp, P          " n < nlp ?
                Y, GOTOR (MC[-1])
                   A = word count
                   A +   3
                   MINA (nlp)          " nlp = nlp + word count + 3
                   F =   0
                   MA[1] = G           " name list[nlp - 1] = 0
                   A = 204
                   GOTO (:ERRORM)      " 16 instructions

SCAN CODE:         SUBC (:EXIT BLK)
                   SUBC (:NXT SBL)
                U, S -  65, Z          " last symbol = minus ?
                Y, SUBC (:NXT SBL)
                   A = letter last symbol, Z
                Y, SUBC (:IDF)
                N, SUBC (:UNS INT)
                   S = last symbol
                U, S -  87, Z          " last symbol = comma ?
                Y, GOTO (:SCAN CODE[1])
```

```
                U, S - 103, Z           " last symbol = unquote ?
                   GOTO (:S STREXP[7])  " 12 instructions


ASK LIBR:          'BEGIN' NEXTO, END

                   MC = S               " n
                   S = nlp
                   S + d25
                   A = end of catalogue
                   MA = - S             " name list[end of catalogue] =
                   S = begin of catalogue       "      - 33554432 - nlp
                   SUBC (:LOOK UP[4])
                U, S - begin of catalogue, P
                U, S - end of catalogue, E      " outside catalogue ?
                Y, GOTO (:END)
                   A = nlp
                   A - word count
                   A -   1              " nlp = nlp + word count + 1
NEXTO:             F = MS[-1]           " name list[nlp] = name list[m]
                   MA[-1] = F           " name list[nlp + 1] = name list[m + 1]
                   G = MS[-2], P        " name list[m + 2] > 0 ?
                   MA[-2] = G           " name list[nlp + 2] = name list[m + 2]
                   A -   3              " nlp = nlp + 3
                N, S -   3              " m = m + 3
                N, GOTO (:NEXTO)
                   nlp = A
END:               S = MC[-1]
                   GOTOR (MC[-1])       " 23 instructions

                   'END' ASK LIBR
```

```
" main program of prescan1

PRESCAN1:       S = 200
                SUBC (:INIT)
                A = nlp
                MA = - S            " namelist[nlp] = -1
                MA[-1] = G          " namelist[nlp + 1] = 0
                A = begin of nli
                D = A               " blockcell pointer = begin of name list
                nxt bcp = A         " next blockcell pointer =
                A = instr cntr      "                  begin of name list
                dp0 = A             " dp0 = instruct counter
                SUBC (:TP OF DSP)
                instr cntr + S      " instruct counter = instruct counter +
                SUBC (:NXT SBL)     "                    top of display
                SUBC (:ENTR BLK)
                SUBC (:PROGRAM)
                SUBC (:STC ADDR)
                SUBC (:TEST TEXT IN ARRAY)
                GOTO (:TRANSL)      " 18 instructions

'END'
```

```
'BEGIN'     ARITHEXP, IFCLAUSE, FUTURE, S ARITHEXP, NXT TERM, TERM, NXT FCTR,
            FACTOR, NXT PRIM, PRIMARY, REQ CLOSE, AR NAME, MCR DOS, MCR RET,
            RETURN, SUBS VAR, ADR DSCR, EVALON, SBS LST, REQ BUS, BOOLEXP,
            S BOOLEXP, NXT IMP, IMPL, NXT BL TRM, BL TRM, NXT BL FC, BL FAC,
            NXT BL SC, BL SEC, BL PRIM, RELATN, RST OF RL, BL PR RST,
            BL NAME, AR BL EXP, S A BL EXP, RST OF AE, RST OF BE, RST OF ABE,
            AR BL RST, STR EXP, S STR EXP, ST NAME, DES EXP, S DES EXP,
            DS NAME, AR DS EXP, N DS EXP, EXP, S EXP, EXP RST, ASN STAT,
            DST TYPE, PREPARE, IN ASN, RE ASN, BO ASN, ST ASN, AR ASN,
            UN ASN, FCT DES, PR STAT, PR CALL, ORD NUM, PAR LIST, ACT PAR,
            START ISR, NUM DSCR, PRCS OP, NON ASBL, LINE, LINE1, STATMNT,
            UNC STAT, STAT, GOTO STAT, CMP TL, IF STAT, FOR STAT, STORE PR,
            STORE MCR, TAKE MCR, FOR LST, SW DEC, ARR DEC, BND PR LST,
            PR DEC, SAVE LNC, BLOCK, DEC LST, LAB LST, PROGRAM, LAB DEC,
            SUBSTT, SUBST2, ORD CNT, MACRO, MACRO2, LOAD, OPTIMIZE, UNLOAD,
            PRODUCE, PRCS STP, PRCS PAR, SAT, OPT OP, VAL LK, OPT NBR,
            INSTR NBR, CLEAR, PAR PART, INSTR PRT, B REACT, CHARCTR, ARMETIC,
            REAL, INTEGER, BOOLEAN, STRING, DESINAL, NO TYPE, ARBOST,
            UNKNOWN, NON AR, NON RE, NON IN, NON BL, NON STR, NON DES,
            SIMPLE, SIMPLE1, SUBSCR, PROC, FUNCTN, NON SMPL, NON SUBS,
            NON PROC, NON FCT, FORMAL, IN VA LI, ASS TO, DYNAMIC, IN LIBR,
            OP LIKE, OUT DEC, ASS TO FD, DECLARED, SUP LOC, LOC POS,
            SET IN DEC, MRK POS, PR ADR, ADDRSS, LST LTH, TEST FC, TEST RET,
            CHCK DIM, CHCK RET, CHCK LL, CHCK TP, NBR LL, NEXT LL, NXT F I,
            INCR STS, IDF, SKP PR LI, TRL CD, UNS NUM, AR CST, CST STR,
            MCR LST, STACK, NEG, EMPTY, TBC, TLV, STST, JU, SUBJ, NIL, LAST,
            TAA, SWP, EXITSV, CODE, TABEL

ARITHEXP:     S = last symbol
           U, S - 94, Z              " last symbol = if ?
           N, GOTO (:S ARITHEXP)
              SUBC (:IFCLAUSE)
              MC = S                 " future1
           Y, SUBC (:NXT SBL)
           N, A = 300
           N, SUBC (:ERRORM)
              SUBC (:S ARITHEXP)
              SUBC (:FUTURE)
           Y, SUBC (:ARITHEXP)
              A = MC[-1]             " future2
           Y, GOTO (:SUBSTT)
              A = 301
              GOTO (:ERRORM)         " 15 instructions

IFCLAUSE:     SUBC (:NXT SBL)
              SUBC (:BOOLEXP)
              A = 106
              S = 0                  " future1 = 0
              SUBC (:MACRO2)         " MACRO2 (COJU, future1)
              A = last symbol
           U, A - 95, Z              " last symbol = then ?
              GOTO (MC[-1])          " 8 instructions

FUTURE:       S = last symbol
              S - 96, Z              " last symbol = else ?
           Y, A = 102                " then future2 = 0
```

```
                Y, SUBC (:MACRO2)       " and MACRO2 (JU, future2)
                Y, A = M[B-2]           " future1
                Y, M[B-2] = S           " future2
                Y, SUBC (:SUBSTT)
                Y, SUBC (:NXT SBL)
                   GOTO (MC[-1])        " 9 instructions

S ARITHEXP:        S = last symbol
                U, S - 64, Z            " last symbol = plus ?
                N, S - 65, Z            " V last symbol = minus ?
                Y, S -  1
                Y, MC = - S             " (- 63 if plus, + 1 if minus)
S ARITHEXP[5]:  Y, SUBC (:NXT SBL)
                   SUBC (:TERM)
                Y, A = MC[-1], P        " MACRO = NEG/ADD/SUB ?
                Y, SUBC (:MACRO)        " 9 instructions

NXT TERM:          S = last symbol
                U, S - 63, P
                U, S - 65, E            " last symbol not a plus or minus sign ?
                Y, GOTOR (MC[-1])       " then expression completed
                   S - 62               " form macro ADD/SUB
                   MC = S
                   A =  0, Z
                   SUBC (:MACRO)        " MACRO (STACK)
                   GOTO (:S ARITHEXP[5]) " 9 instructions

TERM:              SUBC (:FACTOR)       " 1 instruction

NXT FCTR:          S = last symbol
                U, S - 65, P
                U, S - 68, E            " last symbol # mul or div or idi ?
                Y, GOTOR (MC[-1])       " then term completed
                   S - 62               " form macro MUL/DIV/IDI
                   MC = S
                   A =  0
                   SUBC (:MACRO)        " MACRO (STACK)
                   SUBC (:NXT SBL)
                   SUBC (:FACTOR)
                   A = MC[-1]
                   SUBC (:MACRO)        " MACRO (MUL/DIV/IDI)
                   GOTO (:NXT FCTR)     " 13 instructions

FACTOR:            SUBC (:PRIMARY)      " 1 instruction

NXT PRIM:          S = last symbol
                U, S - 69, Z            " last symbol = ttp ?
                N, GOTOR (MC[-1])       " else factor completed
                   A =  0
                   SUBC (:MACRO)        " MACRO (STACK)
                   SUBC (:NXT SBL)
                   SUBC (:PRIMARY)
                   A =  7
```

```
                    SUBC (:MACRO)          " MACRO (TTP)
                    GOTO (:NXT PRIM)       " 10 instructions

PRIMARY:            S = digit last symbol, Z
                  Y, SUBC (:UNS NUM)
                  Y, GOTO (:AR CST)
                    S = letter last symbol, Z
                  Y, SUBC (:IDF)
                  Y, SUBC (:SUBS VAR)
                  Y, SUBC (:FCT DES)
                  Y, GOTO (:AR NAME)
                    S = last symbol
                  U, S - 98, Z           " last symbol = open ?
                  N, JUMP (7)
                    SUBC (:NXT SBL)
                    SUBC (:ARITHEXP)
                    A = 302
REQ CLOSE:          S = last symbol
                  U, S - 99, Z           " last symbol = close ?
                  Y, GOTO (:NXT SBL)
                    GOTO (:ERRORM)
                    A = 303
                    SUBC (:ERRORM)
                  U, S - 94, Z           " last symbol = if ?
                  N, S - 64, Z           " V last symbol = plus ?
                  N, S - 1, Z            " V last symbol = minus ?
                  Y, GOTO (:ARITHEXP)
                    GOTOR (MC[-1])       " 25 instructions

AR NAME:            SUBC (:NON AR)         " nonarithmetic ?
                  Y, A = 304
                  Y, SUBC (:ERRORM)
                    SUBC (:FORMAL)         " formal ?
                  N, SUBC (:FUNCTN)        " V function ?
                  Y, cmpltd = S            " then complicated = true
                    SUBC (:SIMPLE)         " simple ?
                  N, GOTOR (MC[-1])
                    MC = S                 " n
                    SUBC (:FORMAL)         " formal ?
MCR DOS:          Y, A = 99                " MACRO2 (DOS, n)
                  Y, GOTO (:MCR RET)
                    SUBC (:INTEGER)        " integer ?
                  Y, A = 84                " MACRO2 (TIV, n)
                  N, A = 83                " MACRO2 (TRV, n)
MCR RET:            SUBC (:MACRO2)
RETURN:             S = MC[-1]
                    GOTOR (MC[-1])         " 18 instructions

SUBS VAR:           SUBC (:SUBSCR)         " subscripted ?
                  N, GOTOR (MC[-1])
                    SUBC (:ADR DSCR)
                    A = last symbol
                  U, A - 92, Z           " last symbol = colonequal ?
                  N, GOTO (:EVALON)
                    MC = S                 " n
                    A = 0
```

```
                SUBC (:MACRO)          " MACRO (STACK)
                A = 20                 " MACRO (STAA)
                GOTO (:MCR RET)        " 11 instructions

ADR DSCR:       A = last symbol
              U, A - 100, Z            " last symbol = sub ?
              N, A = 305
              N, GOTO (:ERRORM)
                MC = S                 " n
                SUBC (:NXT SBL)
                SUBC (:SBS LST)
                dimension = A
                S = M[B-1]             " n
                SUBC (:CHCK DIM)
                SUBC (:FORMAL)         " formal ?
              Y, GOTO (:MCR DOS)       " MACRO2 (DOS, n)
                SUBC (:DESINAL)        " designational ?
              Y, A = 93                " MACRO2 (TSWE, n)
              N, A = 92                " MACRO2 (TAK, n)
                GOTO (:MCR RET)        " 16 instructions

EVALON:         MC = S                 " n
                SUBC (:DESINAL)        " designational ?
              N, JUMP (4)
                SUBC (:FORMAL)         " formal ?
              Y, A = 27                " MACRO (TFSL)
              N, A = 26                " MACRO (TSL)
                GOTO (:MCR RET)
                SUBC (:BOOLEAN)        " Boolean ?
              Y, A = 23
              Y, GOTO (:MCR RET)       " MACRO (TSB)
                SUBC (:STRING)         " string ?
              Y, A = 24
              Y, GOTO (:MCR RET)       " MACRO (TSST)
                SUBC (:FORMAL)         " formal ?
              Y, A = 25
              Y, GOTO (:MCR RET)       " MACRO (TFSU)
                SUBC (:INTEGER)        " integer ?
              Y, A = 22                " MACRO (TSI)
              N, A = 21                " MACRO (TSR)
                GOTO (:MCR RET)        " 20 instructions

SBS LST:        SUBC (:ARITHEXP)
                S = last symbol
              U, S - 87, Z             " last symbol = comma ?
              Y, A = 0
              Y, SUBC (:MACRO)         " MACRO (STACK)
              Y, SUBC (:NXT SBL)
              Y, SUBC (:SBS LST)
              Y, A + 1
              Y, GOTOR (MC[-1])
                A = 306
REQ BUS:      U, S - 101, Z            " last symbol = bus ?
              Y, SUBC (:NXT SBL)
              N, SUBC (:ERRORM)
                A = 1
```

```
                    GOTOR (MC[-1])          " 15 instructions

BOOLEXP:            S = last symbol
                  U, S - 94, Z              " last symbol = if ?
                  N, GOTO (:S BOOLEXP)
                     SUBC (:IFCLAUSE)
                     MC = S                 " future1
                  Y, SUBC (:NXT SBL)
                  N, A = 307
                  N, SUBC (:ERRORM)
                     SUBC (:S BOOLEXP)
                     SUBC (:FUTURE)
                  Y, SUBC (:BOOLEXP)
                     A = MC[-1]             " future2
                  Y, GOTO (:SUBSTT)
                     A = 308
                     GOTO (:ERRORM)         " 15 instructions

S BOOLEXP:          SUBC (:IMPL)            " 1 instruction

NXT IMP:            S = last symbol
                  U, S - 77, Z              " last symbol = qvl ?
                  N, GOTOR (MC[-1])         " else expression completed
                     A = 14
                     SUBC (:MACRO)          " MACRO (STAB)
                     SUBC (:NXT SBL)
                     SUBC (:IMPL)
                     A = 16
                     SUBC (:MACRO)          " MACRO (QVL)
                     GOTO (:NXT IMP)        " 10 instructions

IMPL:               SUBC (:BL TRM)          " 1 instruction

NXT BL TRM:         S = last symbol
                  U, S - 78, Z              " last symbol = imp ?
                  N, GOTOR (MC[-1])         " else implication completed
                     A = 14
                     SUBC (:MACRO)          " MACRO (STAB)
                     SUBC (:NXT SBL)
                     SUBC (:BL TRM)
                     A = 17
                     SUBC (:MACRO)          " MACRO (IMP)
                     GOTO (:NXT BL TRM)     " 10 instructions

BL TRM:             SUBC (:BL FAC)          " 1 instruction

NXT BL FC:          S = last symbol
                  U, S - 79, Z              " last symbol = or ?
                  N, GOTOR (MC[-1])         " else Boolean term completed
                     A = 14
                     SUBC (:MACRO)          " MACRO (STAB)
                     SUBC (:NXT SBL)
```

```
                    SUBC (:BL FAC)
                    A = 18
                    SUBC (:MACRO)          " MACRO (OR)
                    GOTO (:NXT BL FC)      " 10 instructions

BL FAC:             SUBC (:BL SEC)         " 1 instruction

NXT BL SC:          S = last symbol
                U,  S - 80, Z              " last symbol = and ?
                N,  GOTOR (MC[-1])         " else Boolean factor completed
                    A = 14
                    SUBC (:MACRO)          " MACRO (STAB)
                    SUBC (:NXT SBL)
                    SUBC (:BL SEC)
                    A = 19
                    SUBC (:MACRO)          " MACRO (AND)
                    GOTO (:NXT BL SC)      " 10 instructions

BL SEC:             S = last symbol
                U,  S - 76, Z              " last symbol = non ?
                N,  GOTO (:BL PRIM)
                    SUBC (:NXT SBL)
                    SUBC (:BL PRIM)
                    A = 15                 " MACRO (NON)
                    GOTO (:MACRO)          " 7 instructions

BL PRIM:            S = letter last symbol, Z
                Y,  SUBC (:IDF)
                Y,  SUBC (:SUBS VAR)
                Y,  GOTO (:BL PR RST)
                    S = last symbol
                U,  S - 98, Z              " last symbol = open ?
                N,  JUMP (14)
                    SUBC (:NXT SBL)
                    SUBC (:AR BL EXP)
                    MC = A                 " type
                    A = 309
                    SUBC (:REQ CLOSE)      " require close as last symbol
                    A = MC[-1]
                U,  A - 4, Z               " type = ar ?
                Y,  GOTO (:RST OF RL)
                U,  A - 8, Z               " type = arbo ?
                N,  GOTOR (MC[-1])
                    SUBC (:AR OP LS)       " arithoperator last symbol ?
                Y,  GOTO (:RST OF RL)
                    SUBC (:RELATN)         " relation ?
                    GOTOR (MC[-1])
                U,  S - 64, Z              " last symbol = plus ?
                N,  S - 65, Z              " ∨ last symbol = minus ?
                N,  A = digit last symbol, Z  " ∨ digit last symbol ?
                Y,  SUBC (:S ARITHEXP)
                Y,  GOTO (:RST OF RL)
                    S - 51, Z              " last symbol = true ?
                N,  S - 1, Z               " ∨ last symbol = false ?
                Y,  S = last symbol
```

```
                Y, A = 89
                Y, SUBC (:MACRO2)          " MACRO2 (TBC, last symbol)
                Y, GOTO (:NXT SBL)
                   A = 310
                   GOTO (:ERRORM)          " 34 instructions

RELATN:            SUBC (:REL OP LS)       " relatoperator last symbol ?
                N, GOTO (MC[-1])
                   S - 62
                   MC = S                  " relmacro
                   A = 0
                   SUBC (:MACRO)           " MACRO (STACK)
                   SUBC (:NXT SBL)
                   SUBC (:S ARITHEXP)
                   A = MC[-1]
                   SUBC (:MACRO)           " MACRO (relmacro)
                   GOTO (MC[-1])           " 11 instructions

RST OF RL:         SUBC (:RST OF AE)
                   SUBC (:RELATN)          " relation ?
                Y, GOTOR (MC[-1])
                   A = 311
                   GOTO (:ERRORM)          " 5 instructions

BL PR RST:         MC = S                  " n
                   SUBC (:FCT DES)
                   SUBC (:ARMETIC)         " arithmetic ?
                N, SUBC (:AR OP LS)        " ∨ arithoperator last symbol ?
                N, SUBC (:REL OP LS)       " ∨ relatoperator last symbol ?
                   S = MC[-1]              " n
                N, GOTO (:BL NAME)
                   SUBC (:AR NAME)
                   GOTO (:RST OF RL)       " 9 instructions

BL NAME:           SUBC (:NON BL)          " nonboolean ?
                Y, A = 312
                Y, SUBC (:ERRORM)
                   SUBC (:SIMPLE)          " simple ?
                N, GOTOR (MC[-1])
                   MC = S                  " n
                   SUBC (:FORMAL)          " formal ?
                Y, A = 99                  " MACRO2 (DOS, n)
                N, A = 88                  " MACRO2 (TBV, n)
                   GOTO (:MCR RET)         " 11 instructions

AR BL EXP:         S = last symbol
                U, S - 94, Z               " last symbol = if ?
                N, GOTO (:S A BL EXP)
                   SUBC (:IFCLAUSE)
                   MC = S                  " future1
                Y, SUBC (:NXT SBL)
                N, A = 313
                N, SUBC (:ERRORM)
                   SUBC (:S A BL EXP)
```

```
                    MC = A              " type
                    A = 314
AR BL EXP[11]:      S = last symbol
                    S - 96, Z           " last symbol = else ?
                 N, SUBC (:ERRORM)
                 N, JUMP (15)
                    A = 102             " future2 = 0
                    SUBC (:MACRO2)      " MACRO2 (JU, future2)
                    A = M[B-2]          " future1
                    M[B-2] = S          " future2
                    SUBC (:SUBSTT)
                    SUBC (:NXT SBL)
                    A = M[B-1]
                 U, A - 6, P            " type=un ∨ type=arbo ∨ type=intlab ?
                 Y, JUMP (1)
                 U, A - 5, Z            " ∨ type = nondes ?
                    A + :MT[5]          " base address of list
                    DO (MA)             " execute selected instruction
                 Y, M[B-1] = A          " replace type by new type
                    A = M[B-2]          " future2
                    SUBC (:SUBSTT)
                    A = MC[-1]          " type
                    B - 1
                    GOTOR (MC[-1])
                    SUBC (:BOOLEXP)
                    SUBC (:STR EXP)
                    SUBC (:ARITHEXP)
                    SUBC (:N DS EXP)
                    SUBC (:DES EXP)
                    SUBC (:EXP)
                    SUBC (:AR BL EXP)
                    SUBC (:AR DS EXP)   " 41 instructions

S A BL EXP:         S = letter last symbol, Z
                 Y, SUBC (:IDF)
                 Y, SUBC (:SUBS VAR)
                 Y, GOTO (:AR BL RST)
                    S = last symbol
                 U, S - 98, Z           " last symbol = open ?
                 N, JUMP (16)
                    SUBC (:NXT SBL)
                    SUBC (:AR BL EXP)
                    MC = A              " type
                    A = 315
                    SUBC (:REQ CLOSE)   " require close as last symbol
                    A = MC[-1]
                 U, A - 8, Z            " type = arbo ?
                 Y, SUBC (:BOOL OP LS)  " ∧ booloperator last symbol ?
                 N, A - 2, Z            " ∨ type = bo ?
                 Y, SUBC (:RST OF BE)
                 Y, JUMP (14)           " type = bo
                    A - 2, Z            " type = ar ?
                 N, SUBC (:AR OP LS)    " ∨ arithoperator last symbol ?
                 N, SUBC (:REL OP LS)   " ∨ relatoperator last symbol ?
                 Y, GOTO (:RST OF ABE)
                    JUMP (13)           " type = arbo
                 U, S - 64, Z           " last symbol = plus ?
                 N, S - 65, Z           " ∨ last symbol = minus ?
```

```
                N, A = digit last symbol, Z  " ∨ digit last symbol ?
                Y, SUBC (:S ARITHEXP)
                Y, GOTO (:RST OF ABE)
                U, S - 11, Z                 " last symbol = non ?
                N, S - 51, Z                 " ∨ last symbol = true ?
                N, S -  1, Z                 " ∨ last symbol = false ?
                Y, SUBC (:S BOOLEXP)
                Y, A =  2                    " type = bo
                Y, GOTOR (MC[-1])
                   A = 316
                   SUBC (:ERRORM)
                   A =  8                    " type = arbo
                   GOTOR (MC[-1])            " 38 instructions

RST OF AE:         SUBC (:NXT PRIM)
                   SUBC (:NXT FCTR)
                   GOTO (:NXT TERM)          " 3 instructions

RST OF BE:         SUBC (:NXT BL SC)
                   SUBC (:NXT BL FC)
                   SUBC (:NXT BL TRM)
                   GOTO (:NXT IMP)           " 4 instructions

RST OF ABE:        SUBC (:RST OF AE)
                   SUBC (:RELATN)            " relation ?
                Y, SUBC (:RST OF BE)
                Y, A =  2                    " type = bo
                N, A =  4                    " type = ar
                   GOTOR (MC[-1])            " 6 instructions

AR BL RST:         MC =  S                   " n
                   SUBC (:FCT DES)
                   SUBC (:BOOLEAN)           " Boolean ?
                N, SUBC (:BOOL OP LS)        " ∨ booloperator last symbol ?
                   S = M[B-1]                " n
                Y, SUBC (:BL NAME)
                Y, SUBC (:RST OF BE)
                Y, A =  2                    " type = bo
                Y, GOTO (:RETURN)
                   SUBC (:ARMETIC)           " arithmetic ?
                N, SUBC (:AR OP LS)          " ∨ arithoperator last symbol ?
                N, SUBC (:REL OP LS)         " ∨ relatoperator last symbol ?
                   S = M[B-1]                " n
                Y, SUBC (:AR NAME)
                Y, SUBC (:RST OF ABE)
                Y, GOTO (:RETURN)
                   SUBC (:STRING)            " string ?
                N, SUBC (:DESINAL)           " ∨ designational ?
                Y, A = 317
                Y, SUBC (:ERRORM)
                   A =  99
                   SUBC (:MACRO2)            " MACRO2 (DOS, n)
                   A =  8                    " type = arbo
                   GOTO (:RETURN)            " 24 instructions
```

```
STR EXP:        S = last symbol
              U, S - 94, Z          " last symbol = if ?
              N, GOTO (:S STR EXP)
                 SUBC (:IFCLAUSE)
                 MC = S             " future1
              Y, SUBC (:NXT SBL)
              N, A = 318
              N, SUBC (:ERRORM)
                 SUBC (:S STR EXP)
                 SUBC (:FUTURE)
              Y, SUBC (:STR EXP)
                 A = MC[-1]         " future2
              Y, GOTO (:SUBSTT)
                 A = 319
                 GOTO (:ERRORM)     " 15 instructions

S STR EXP:      S = letter last symbol, Z
              Y, SUBC (:IDF)
              Y, SUBC (:SUBS VAR)
              Y, GOTO (:ST NAME)
                 S = last symbol
              U, S - 98, Z          " last symbol = open ?
              Y, SUBC (:NXT SBL)
              Y, SUBC (:STR EXP)
              Y, A = 320
              Y, GOTO (:REQ CLOSE)  " require close as last symbol
              U, S - 102, Z         " last symbol = quote ?
              N, A = 321
              N, GOTO (:ERRORM)
                 A = 28
                 SUBC (:MACRO)      " MACRO (TCST)
                 S = 0              " future = 0
                 A = 102
                 SUBC (:MACRO2)     " MACRO2 (JU, future)
                 MC = S             " future
                 SUBC (:CST STR)
                 A = MC[-1]         " future
                 GOTO (:SUBSTT)     " 22 instructions

ST NAME:        SUBC (:NON STR)     " nonstring ?
              Y, A = 322
              Y, SUBC (:ERRORM)
                 SUBC (:FCT DES)
                 SUBC (:SIMPLE)     " simple ?
              N, GOTOR (MC[-1])
                 MC = S             " n
                 SUBC (:FORMAL)     " formal ?
              Y, A = 99             " MACRO2 (DOS, n)
              N, A = 90             " MACRO2 (TSTV, n)
                 GOTO (:MCR RET)    " 11 instructions

DES EXP:        S = last symbol
              U, S - 94, Z          " last symbol = if ?
              N, GOTO (:S DES EXP)
                 SUBC (:IFCLAUSE)
                 MC = S             " future1
```

```
                Y, SUBC (:NXT SBL)
                N, A = 323
                N, SUBC (:ERRORM)
                   SUBC (:S DES EXP)
                   SUBC (:FUTURE)
                Y, SUBC (:DES EXP)
                   A = MC[-1]              " future2
                Y, GOTO (:SUBSTT)
                   A = 324
                   GOTO (:ERRORM)          " 15 instructions

S DES EXP:         S = letter last symbol, Z
                Y, SUBC (:IDF)
                Y, SUBC (:SUBS VAR)
                Y, GOTO (:DS NAME)
                   S = last symbol
                U, S - 98, Z               " last symbol = open ?
                Y, SUBC (:NXT SBL)
                Y, SUBC (:DES EXP)
                Y, A = 325
                Y, GOTO (:REQ CLOSE)       " require close as last symbol
                   S = digit last symbol, Z
                N, A = 327
                N, GOTO (:ERRORM)
                   SUBC (:UNS NUM)
                   SUBC (:IN NM LST)       " in name list ?
                Y, S = int lab
                Y, A = 91
                Y, GOTO (:MACRO2)          " MACRO2 (TLV, integer label)
                   A = 326
                   GOTO (:ERRORM)          " 20 instructions

DS NAME:           SUBC (:NON DES)         " nondesignational ?
                Y, A = 328
                Y, SUBC (:ERRORM)
                   SUBC (:SIMPLE)          " simple ?
                N, GOTOR (MC[-1])
                   MC = S                  " n
                   SUBC (:FORMAL)          " formal ?
                Y, A = 99                  " MACRO2 (DOS, n)
                N, A = 91                  " MACRO2 (TLV, n)
                   GOTO (:MCR RET)         " 10 instructions

AR DS EXP:         SUBC (:EXP)
                   MC = A                  " type
                U, A - 2, Z                " type = bo ?
                N, A - 3, Z                " ∨ type = st ?
                Y, A = 329
                Y, SUBC (:ERRORM)
                   A = MC[-1]
                U, A - 7, Z                " type = un ?
                Y, A = 9                   " then new type = intlab
                U, A - 5, Z                " type = nondes ?
                Y, A = 4                   " then new type = ar
                   GOTOR (MC[-1])          " 12 instructions
```

```
N DS EXP:       SUBC (:EXP)
              U, A - 6, Z          " type = des ?
              Y, MC = A
              Y, A = 330
              Y, SUBC (:ERRORM)
              Y, A = MC[-1]
              U, A - 7, Z          " type = un ?
              Y, A = 5             " then new type = nondes
              U, A - 9, Z          " type = intlab ?
              Y, A = 4             " then new type = ar
                 GOTOR (MC[-1])    " 11 instructions

EXP:            S = last symbol
              U, S - 94, Z         " last symbol = if ?
              N, GOTO (:S EXP)
                 SUBC (:IFCLAUSE)
                 MC = S            " future1
              Y, SUBC (:NXT SBL)
              N, A = 331
              N, SUBC (:ERRORM)
                 SUBC (:S EXP)
                 MC = A            " type
                 A = 332
                 GOTO (:AR BL EXP[11]) " 12 instructions

S EXP:          S = letter last symbol, Z
              Y, SUBC (:IDF)
              Y, SUBC (:SUBS VAR)
              Y, GOTO (:EXP RST)
                 S = last symbol
              U, S - 98, Z         " last symbol = open ?
              N, JUMP (20)
                 SUBC (:NXT SBL)
                 SUBC (:EXP)
                 MC = A            " type
                 A = 333
                 SUBC (:REQ CLOSE) " require close as last symbol
                 A = - M[B-1]
              U, A '×' 5, Z        " type = nondes ∨ type = un ?
              Y, SUBC (:BOOL OP LS) " ∧ booloperator last symbol ?
              N, A + 2, Z          " ∨ type = bo ?
              Y, SUBC (:RST OF BE)
              Y, A = 2             " type = bo
              Y, GOTO (:RETURN)
                 SUBC (:OP LS)     " operator last symbol ?
                 A = MC[-1]        " type
              N, GOTOR (MC[-1])
              U, A - 3, Z          " type = st ?
              Y, GOTOR (MC[-1])
              U, A - 6, Z          " type = des ?
              Y, GOTOR (MC[-1])
                 GOTO (:RST OF ABE)
              U, S = digit last symbol, Z
              N, JUMP (11)
                 SUBC (:UNS NUM)
S EXP[30]:       SUBC (:AR CST)
                 SUBC (:IN NM LST) " in name list ?
```

```
          N, GOTO (:RST OF ABE)
             SUBC (:OP LS)          " operator last symbol ?
          Y, GOTO (:RST OF ABE)
             S = int lab
             A = 91
             SUBC (:MACRO2)         " MACRO2 (TLV, integer label)
             A = 9                  " type = intlab
             GOTOR (MC[-1])
          U, S - 64, Z              " last symbol = plus ?
          N, S - 65, Z              " ∨ last symbol = minus ?
          Y, GOTO (:S A BL EXP)
          U, S - 11, Z              " last symbol = non ?
          N, S - 51, Z              " ∨ last symbol = true ?
          N, S - 1, Z               " ∨ last symbol = false ?
          Y, SUBC (:S BOOLEXP)
          Y, A = 2                  " type = bo
          Y, GOTOR (MC[-1])
          U, S + 15, Z              " last symbol = quote ?
          Y, SUBC (:S STR EXP)
          Y, A = 3                  " type = st
          N, A = 334
          N, SUBC (:ERRORM)
          N, A = 7                  " type = un
             GOTOR (MC[-1])         " 56 instructions

EXP RST:     SUBC (:DESINAL)        " designational ?
          Y, SUBC (:DS NAME)
          Y, A = 6                  " type = des
          Y, GOTOR (MC[-1])
             SUBC (:STRING)         " string ?
          Y, SUBC (:ST NAME)
          Y, A = 3                  " type = st
          Y, GOTOR (MC[-1])
             SUBC (:FCT DES)
             MC = S                 " n
             SUBC (:BOOLEAN)        " Boolean ?
          N, SUBC (:BOOL OP LS)     " ∨ booloperator last symbol ?
             S = M[B-1]             " n
          Y, SUBC (:BL NAME)
          Y, SUBC (:RST OF BE)
          Y, A = 2                  " type = bo
          Y, GOTO (:RETURN)
             SUBC (:ARMETIC)        " arithmetic ?
          N, SUBC (:AR OP LS)       " ∨ arithoperator last symbol ?
          N, SUBC (:REL OP LS)      " ∨ relatoperator last symbol ?
             S = M[B-1]             " n
          Y, SUBC (:AR NAME)
          Y, SUBC (:RST OF ABE)
          Y, GOTO (:RETURN)
             SUBC (:SIMPLE)         " simple ?
          Y, A = 99
          Y, SUBC (:MACRO2)         " MACRO2 (DOS, n)
             S = MC[-1]             " n
             SUBC (:UNKNOWN)        " unknown ?
          Y, A = 7                  " type = un
          N, A = 5                  " type = nondes
             GOTOR (MC[-1])         " 32 instructions
```

```
ASN STAT:          SUBC (:SUBS VAR)
                   A = last symbol
                U, A - 92, Z          " last symbol = colonequal ?
                N, A = 335
                N, GOTO (:ERRORM)     " 5 instructions

DST TYPE:          'BEGIN' LIST, AR, UN

                   SUBC (:REAL)       " real ?
                   A 'X' 7
                   MC = A             " type
                Y, SUBC (:RE ASN)
                N, A + :LIST
                N, DO (MA)            " execute selected instruction
                   A = MC[-1]         " type
                   GOTOR (MC[-1])
LIST:              GOTO (:UN)
                   SUBC (:IN ASN)
                   SUBC (:BO ASN)
                   SUBC (:ST ASN)
                   GOTO (:AR)
                   GOTO (:UN)
                   GOTO (:UN)
                   GOTO (:UN)
AR:                SUBC (:AR ASN)
                   JUMP (1)
UN:                SUBC (:UN ASN)
                   B - 1
                   GOTOR (MC[-1])     " 21 instructions

                   'END' DST TYPE

PREPARE:           SUBC (:FUNCTN)     " function ?
                Y, JUMP (5)
                   SUBC (:SIMPLE)     " simple ?
                Y, SUBC (:FORMAL)     " ^ formal ?
                Y, A = 100
                Y, SUBC (:MACRO2)     " MACRO2 (DOS2, n)
                   JUMP (8)
                   SUBC (:FORMAL)     " formal ?
                Y, A = 336
                Y, JUMP (2)
                   SUBC (:OUT DEC)    " outside declaration ?
                Y, A = 337
                Y, SUBC (:ERRORM)
                N, SUBC (:LOC POS)
                N, M[B-2] = S         " n
                   SUBC (:NXT SBL)
                   S = letter last symbol, Z
                Y, SUBC (:IDF)
                Y, SUBC (:SUBS VAR)
                Y, A = last symbol
                   GOTO (MC[-1])      " 21 instructions

IN ASN:            MC = S             " rounded = false
                   MC = S             " n
```

```
                SUBC (:NON IN)          " noninteger ?
              Y, A = 338
              Y, SUBC (:ERRORM)
                SUBC (:PREPARE)         " Prepare, identifier read ?
              N, SUBC (:ARITHEXP)
              N, JUMP (6)
                A - 92, Z               " last symbol = colonequal ?
              N, SUBC (:FCT DES)
              N, SUBC (:AR NAME)
              N, SUBC (:RST OF AE)
              Y, SUBC (:IN ASN)
              Y, M[B-2] = A             " rounded
                S = M[B-1]              " n
                SUBC (:SUBSCR)          " subscripted ?
              N, JUMP (7)
                SUBC (:FORMAL)          " formal ?
              Y, A = 34                 " MACRO (STFSU)
              Y, JUMP (10)
                A = M[B-2], Z           " rounded ?
              Y, A = 31                 " MACRO (SSTSI)
              N, A = 30                 " MACRO (STSI)
                JUMP (6)
                SUBC (:FORMAL)          " formal ?
              Y, A = 101                " MACRO2 (DOS3, n)
              Y, JUMP (3)
                A = M[B-2], Z           " rounded ?
              Y, A = 96                 " MACRO2 (SSTI, n)
              N, A = 95                 " MACRO2 (STI, n)
                SUBC (:MACRO2)
                S = MC[-1]              " n
                SUBC (:FORMAL)          " formal ?
                A = MC[-1]              " rounded
              N, A = 0
                GOTOR (MC[-1])          " 36 instructions

RE ASN:         MC = S                  " n
                SUBC (:NON RE)          " nonreal ?
              Y, A = 339
              Y, SUBC (:ERRORM)
                SUBC (:PREPARE)         " Prepare, identifier read ?
              N, SUBC (:ARITHEXP)
              N, JUMP (5)
                A - 92, Z               " last symbol = colonequal ?
              Y, SUBC (:RE ASN)
              N, SUBC (:FCT DES)
              N, SUBC (:AR NAME)
              N, SUBC (:RST OF AE)
                S = M[B-1]              " n
                SUBC (:SUBSCR)          " subscripted ?
              N, JUMP (4)
                SUBC (:FORMAL)          " formal ?
              Y, A = 34                 " MACRO (STFSU)
              N, A = 29                 " MACRO (STSR)
                GOTO (:MCR RET)
                SUBC (:FORMAL)          " formal ?
              Y, A = 101                " MACRO2 (DOS3, n)
              N, A = 94                 " MACRO2 (STR, n)
                GOTO (:MCR RET)         " 23 instructions
```

```
BO ASN:          MC = S              " n
                 SUBC (:NON BL)      " nonboolean ?
              Y, A = 340
              Y, SUBC (:ERRORM)
                 SUBC (:PREPARE)     " Prepare, identifier read ?
              N, SUBC (:BOOLEXP)
              N, JUMP (4)
                 A - 92, Z           " last symbol = colonequal ?
              Y, SUBC (:BO ASN)
              N, SUBC (:BL PR RST)
              N, SUBC (:RST OF BE)
                 S = M[B-1]          " n
                 SUBC (:SUBSCR)      " subscripted ?
              Y, A = 32
              Y, GOTO (:MCR RET)     " MACRO (STSB)
                 SUBC (:FORMAL)      " formal ?
              Y, A = 101             " MACRO2 (DOS3, n)
              N, A = 97              " MACRO2 (STB, n)
                 GOTO (:MCR RET)     " 19 instructions

ST ASN:          MC = S              " n
                 SUBC (:NON STR)     " nonstring ?
              Y, A = 341
              Y, SUBC (:ERRORM)
                 SUBC (:PREPARE)     " Prepare, identifier read ?
              N, SUBC (:STREXP)
              N, JUMP (3)
                 A - 92, Z           " last symbol = colonequal ?
              Y, SUBC (:ST ASN)
              N, SUBC (:ST NAME)
                 S = M[B-1]          " n
                 SUBC (:SUBSCR)      " subscripted ?
              Y, A = 33
              Y, GOTO (:MCR RET)     " MACRO (STSST)
                 SUBC (:FORMAL)      " formal ?
              Y, A = 101             " MACRO2 (DOS3, n)
              N, A = 98              " MACRO2 (STST, n)
                 GOTO (:MCR RET)     " 18 instructions

AR ASN:          MC = S              " n
                 SUBC (:NON AR)      " nonarithmetic ?
              Y, A = 342
              Y, SUBC (:ERRORM)
                 SUBC (:PREPARE)     " Prepare, identifier read ?
                 F = 4
                 MC = G              " type = ar
              N, SUBC (:ARITHEXP)
              N, JUMP (10)
                 A - 92, Z           " last symbol = colonequal ?
              N, SUBC (:FCT DES)
              N, SUBC (:AR NAME)
              N, SUBC (:RST OF AE)
              N, JUMP (5)
                 SUBC (:NON AR)      " nonarithmetic ?
              Y, A = 343
              Y, SUBC (:ERRORM)
                 SUBC (:DST TYPE)
```

```
                    M[B-1] = A          " type
                    S = M[B-2]          " n
                    SUBC (:SUBSCR)      " subscripted ?
                 Y, A = 34              " MACRO (STFSU)
AR ASN[22]:      N, A = 101             " MACRO2 (DOS3, n)
AR ASN[23]:        SUBC (:MACRO2)
                    A = MC[-1]          " type
                    GOTO (:RETURN)      " 26 instructions

UN ASN:             MC = S              " n
                    SUBC (:NO TYPE)     " nontype ?
                 Y, A = 344
                 Y, SUBC (:ERRORM)
                    SUBC (:PREPARE)     " Prepare, identifier read ?
                 N, SUBC (:N DS EXP)
                 N, JUMP (7)
                    SUBC (:NO TYPE)     " nontype ?
                 Y, A = 345
                 Y, SUBC (:ERRORM)
                    A = last symbol
                    A - 92, Z           " last symbol = colonequal ?
                 Y, SUBC (:DST TYPE)
                 N, SUBC (:EXP RST)
                    MC = A              " type
                    S = M[B-2]          " n
                    SUBC (:SUBSCR)      " subscripted ?
                 N, GOTO (:AR ASN[22])  " MACRO2 (DOS3, n)
                    A = M[B-1]          " type
                 U, A - 1, P            " type ≠ bo
                 U, A - 3, E            " ∧ type ≠ st ?
                 Y, A = 34              " MACRO (STFSU)
                 N, A + 30              " MACRO (STSB/STSST)
                    GOTO (:AR ASN[23])  " 24 instructions

FCT DES:            SUBC (:PROC)        " proc ?
                 N, GOTOR (MC[-1])
                    SUBC (:NON FCT)     " nonfunction ?
                 Y, A = 346
                 Y, SUBC (:ERRORM)
                    GOTO (:PR CALL)     " 6 instructions

PR STAT:            SUBC (:PROC)        " proc ?
                 N, A = 347
                 N, GOTO (:ERRORM)
                    SUBC (:PR CALL)
                    SUBC (:IN LIBR)     " in library ?
                 N, SUBC (:LINE1)
                    SUBC (:FUNCTN)      " function ?
                 Y, SUBC (:STRING)      " ∧ string ?
                 N, SUBC (:FORMAL)      " ∨ formal ?
                 N, GOTOR (MC[-1])
                    A = 73              " MACRO (REJST)
                    GOTO (:MACRO)       " 12 instructions
```

```
PR CALL:          SUBC (:OP LIKE)      " operatorlike ?
                Y, GOTO (:PRCS OP)
                  SUBC (:LST LTH)
                  A -  0, P            " number of parameters > 0 ?
                Y, GOTO (:PAR LIST)
PR CALL[5]:       MC = S               " n
                  SUBC (:FORMAL)       " formal ?
                Y, A = 99
                Y, GOTO (:MCR RET)     " MACRO2 (DOS, n)
                  SUBC (:IN LIBR)      " in library ?
                Y, A = 109             " MACRO2 (ISUBJ, n)
                N, A = 108             " MACRO2 (SUBJ, n)
                  GOTO (:MCR RET)      " 13 instructions

ORD NUM:          SUBC (:FORMAL)       " formal ?
                Y, A = 15
                Y, GOTOR (MC[-1])
                  A 'X' 31             " character
                U, A - 15, P           " proc ?
                N, JUMP (4)
                U, A 'X' 8, Z          " function ?
                Y, A + 8               " then 24/25/26/27
                N, A = 30              " else 30
                  GOTOR (MC[-1])
                U, A - 14, Z           " switch ?
                Y, A = 12              " then 12
                U, A - 6, Z            " label ?
                Y, A = 14              " then 14
                  GOTOR (MC[-1])       " 15 instructions
```

```
PAR LIST:       'BEGIN' PAR LIST0, PAR LIST1, PAR LIST2, PAR LIST3,
                        PAR LIST4, PAR LIST5, PAR LIST6, PAR LIST7,
                        PAR LIST8, PAR LIST9, PAR LIST10, PAR LIST11,
                        PAR LIST12, PAR LIST13

                  A +   B
                  MA =  B              " address first APD
                  MA[1] = B            " address current APD
                  B = :MA[2]
                  MC =  S              " n
                  MC =  S              " f = n
                  F =   0
                  MC =  F              " type, future = 0
                  S = last symbol
               U, S -   98, Z          " last symbol = open ?
               N, A = 362              " PL15
               N, GOTO (:PAR LIST12)
PAR LIST0:        SUBC (:NXT SBL)
                  SUBC (:ACT PAR)
                  M[B-2] = S           " type
                  S = M[B-5]           " address current APD
               U, S - :MC[-6], Z       " > address last APD ?
               Y, A = 359              " PL12
               Y, GOTO (:PAR LIST10)
                  MS =  A              " APD
                  S +   1
                  M[B-5] = S           " address next APD
                  S = M[B-4]           " n
                  SUBC (:FORMAL)       " formal ?
               Y, GOTO (:PAR LIST11)
                  S = M[B-3]           " f
                  SUBC (:NXT F I)
                  M[B-3] = S           " f = next formal identifier
                  F +   0, Z           " simple identifier ?
               N, GOTO (:PAR LIST4)
                  SUBC (:SUBSCR)       " f subscripted ?
               N, GOTO (:PAR LIST1)
                  S = M[B-2]           " type
                  SUBC (:NON SUBS)     " nonsubscripted ?
               Y, A = 348              " PL1
               Y, SUBC (:ERRORM)
                  A = M[B-3]           " f
                  SUBC (:CHCK TP)
                  A = M[B-3]           " f
                  SUBC (:CHCK LL)
                  GOTO (:PAR LIST11)
PAR LIST1:        SUBC (:PROC)         " f proc ?
               N, GOTO (:PAR LIST3)
                  S = M[B-2]           " type
                  SUBC (:NON PROC)     " nonproc ?
               Y, A = 349              " PL2
               Y, SUBC (:ERRORM)
                  A = M[B-3]           " f
                  SUBC (:CHCK LL)
                  S = M[B-3]           " f
                  SUBC (:FUNCTN)       " function ?
               N, GOTO (:PAR LIST11)
                  S = M[B-2]           " type
                  SUBC (:NON FCT)      " nonfunction ?
```

```
                Y, A = 350              " PL3
PAR LIST2:      Y, SUBC (:ERRORM)
                   A = M[B-3]           " f
                   SUBC (:CHCK TP)
                   GOTO (:PAR LIST11)
PAR LIST3:         SUBC (:SIMPLE1)      " f simple1 ?
                N, GOTO (:PAR LIST11)
                   S = M[B-2]           " type
                   SUBC (:NON SMPL)     " nonsimple ?
                Y, A = 351              " PL4
                   GOTO (:PAR LIST2)
PAR LIST4:         SUBC (:SUBSCR)       " f subscripted ?
                N, SUBC (:PROC)         " ∨ proc ?
                Y, A = 352              " PL5
                Y, SUBC (:ERRORM)
                   SUBC (:ASS TO)       " assigned to f ?
                Y, A = M[B-5]           " address current APD
                Y, A = MA[-1]           " APD
                Y, SUBC (:NON ASBL)     " nonassignable ?
                Y, A = 353              " PL6
                Y, SUBC (:ERRORM)
                   SUBC (:ARMETIC)      " f arithmetic ?
                N, GOTO (:PAR LIST5)
                   A = 354              " PL7
                   S = M[B-2]           " type
                U, S 'X' - 3, Z         " bo ∨ st ?
                N, S -  6, Z            " ∨ des ?
                   GOTO (:PAR LIST10)
PAR LIST5:         SUBC (:BOOLEAN)      " f Boolean ?
                N, GOTO (:PAR LIST7)
                   A = 355              " PL8
                   S = - M[B-2]         " type
                U, S +  2, Z            " bo ?
PAR LIST6:      N, S 'X' 5, Z           " ∨ nondes ∨ un ?
                N, SUBC (:ERRORM)
                   GOTO (:PAR LIST11)
PAR LIST7:         SUBC (:STRING)       " f string ?
                N, GOTO (:PAR LIST8)
                   A = 356              " PL9
                   S = - M[B-2]         " type
                U, S +  3, Z            " st ?
                   GOTO (:PAR LIST6)
PAR LIST8:         SUBC (:DESINAL)      " f designational ?
                N, GOTO (:PAR LIST9)
                   A = 357              " PL10
                   S = - M[B-2]         " type
                U, S +  6, P            " bo ∨ st ∨ ar ∨ nondes ?
                   GOTO (:PAR LIST10)
PAR LIST9:         SUBC (:ARBOST)       " f arbost ?
                Y, S = M[B-2]           " type
                Y, S -  6, Z            " des ?
                Y, A = 358              " PL11
PAR LIST10:     Y, SUBC (:ERRORM)
PAR LIST11:        S = last symbol
                U, S -  87, Z           " last symbol = comma ?
                Y, GOTO (:PAR LIST0)
                U, S -  99, Z           " last symbol = close ?
                N, A = 361              " PL14
                N, GOTO (:PAR LIST12)
```

```
                    SUBC (:NXT SBL)
                    S = M[B-5]              " address current APD =
                U,  S - :MC[-6], Z          " = address last APD + 1 ?
                N,  A = 360                 " PL13
PAR LIST12:     N,  SUBC (:ERRORM)
                    A = M[B-1], Z           " future = 0 ?
                N,  SUBC (:SUBSTT)
                    S = M[B-4]              " n
                    SUBC (:PR CALL[5])      " MACRO2 (DOS/ISUBJ/SUBJ, n)
                    S = M[B-6]              " address first APD
PAR LIST13:         M[B-2] = S              " address next APD
                U,  S - M[B-5], Z           " = address current APD ?
                Y,  S = M[B-4]              " n
                Y,  B = M[B-6]
                Y,  GOTOR (MC[-1])
                    A = 128
                    S =  MS                 " APD
                    SUBC (:MACRO2)          " MACRO2 (CODE, APD)
                    S = M[B-2]              " address next APD
                    S +  1
                    GOTO (:PAR LIST13)      " 134 instructions

                    'END' PAR LIST


ACT PAR:            'BEGIN'  ACT PAR0, ACT PAR1, ACT PAR2, ACT PAR3, ACT PAR4,
                             ACT PAR5, ACT PAR6, ACT PAR7, ACT PAR8, ACT PAR9,
                             ACT PAR10, ACT PAR11, ACT PAR12, ACT PAR13,
                             ACT PAR14, ACT PAR15, ACT PAR16

                    SUBC (:ORD CNT)
                    A = M[B-2], Z           " future = 0 ?
                Y,  S +  1
                    MC =  S                 " begin address
                    S = letter last symbol, Z
                N,  GOTO (:ACT PAR5)
                    SUBC (:IDF)
                    MC =  S                 " n
                    A = last symbol
                U,  A -  87, Z              " last symbol = comma ?
                N,  A -  99, Z              " V last symbol = close ?
                N,  GOTO (:ACT PAR2)
                    SUBC (:PROC)            " proc ?
                N,  GOTO (:ACT PAR0)
                    SUBC (:FORMAL)          " formal ?
                Y,  GOTO (:ACT PAR0)
                    S = M[B-4], Z           " future = 0 ?
                Y,  A = 102
                Y,  SUBC (:MACRO2)          " MACRO2 (JU, future)
                Y,  M[B-4] = S              " future
                    A =  36
                    SUBC (:MACRO)           " MACRO (TFD)
                    S = M[B-1]              " n
                    SUBC (:IN LIBR)         " in library ?
                Y,  A = 105                 " MACRO2 (IJU1, n)
                N,  A = 103                 " MACRO2 (JU1, n)
                    SUBC (:MACRO2)
                    GOTO (:ACT PAR1)
ACT PAR0:           SUBC (:DYNAMIC)         " dynamic ?
```

```
                    SUBC (:ADDRSS)
                Y,  A + d18
                    M[B-2] = A
ACT PAR1:           S = MC[-1]              " n
                    SUBC (:ORD NUM)
                    LUA (20)
                    A + MC[-1]              " APD
                    F =  0                  " simple identifier = true
                    GOTOR (MC[-1])
ACT PAR2:           SUBC (:START ISR)
                    S = M[B-1]              " n
                    SUBC (:SUBSCR)          " subscripted ?
                Y,  SUBC (:ADR DSCR)
                    A = last symbol
                U,  A - 87, Z               " last symbol = comma ?
                N,  A - 99, Z               " ∨ last symbol = close ?
                N,  GOTO (:ACT PAR3)
                    SUBC (:DESINAL)         " designational ?
                Y,  GOTO (:ACT PAR3)
                    SUBC (:UNKNOWN)         " unknown ?
                Y,  A =  37
                Y,  SUBC (:MACRO)           " MACRO (SAS)
                    A = 127
                    S = - dimension
                    LUS (1)
                    SUBC (:MACRO2)          " MACRO2 (EXITSV, - 2 × dimension)
                    SUBC (:ORD CNT)
                    MC =  S
                    S = M[B-2]              " n
                    SUBC (:BOOLEAN)         " Boolean ?
                N,  SUBC (:STRING)          " ∨ string ?
                Y,  JUMP (3)
                    SUBC (:FORMAL)          " formal ?
                Y,  A =  16
                N,  A 'X' 3                 " type
                    A + 16
                    LUA (20)
                    M[B-1] + A              " APD
                    SUBC (:ARMETIC)         " arithmetic ?
                Y,  A =  4                  " ar
                N,  SUBC (:INTEGER)         " isolate type bits
                    M[B-2] = A              " type
                    A = 108
                    S = - M[B-3]            " begin address
                    SUBC (:MACRO2)          " MACRO2 (SUBJ, - begin address)
                    A = M[B-1]              " APD
                    RUA (20)
                U,  A - 32, Z
                Y,  A =  44
                N,  A + 24
                    SUBC (:MACRO)           " MACRO (TASR/TASI/TASB/TASST/TASU)
                    A =  38
                    SUBC (:MACRO)           " MACRO (DECS)
                    A = 108
                    S = - M[B-3]            " begin address
                    SUBC (:MACRO2)          " MACRO2 (SUBJ, - begin address)
                    A =  39
                    SUBC (:MACRO)           " MACRO (FAD)
                    A = MC[-1]              " APD
```

```
                  S = MC[-1]          " type
                  G = MC[-1]          " simple identifier = false
                  GOTOR (MC[-1])
ACT PAR3:         SUBC (:SUBSCR)      " subscripted ?
                Y, SUBC (:EVALON)
                  SUBC (:EXP RST)
ACT PAR4:         M[B-1] = A          " type
                  A = 45
                  SUBC (:MACRO)       " MACRO (EXITIS)
                  S = MC[-1]          " type
                  A = S
                  A + :mask
                  A = MA
                  A + MC[-1]          " begin address
                  F = 1               " simple identifier = false
                  GOTOR (MC[-1])
ACT PAR5:         S = digit last symbol, Z
                N, GOTO (:ACT PAR8)
                  SUBC (:UNS NUM)
                  S = last symbol
                U, S - 87, Z          " last symbol = comma ?
                N, S - 99, Z          " ∨ last symbol = close ?
                N, GOTO (:ACT PAR7)
                  SUBC (:IN NM LST)   " in name list ?
                Y, GOTO (:ACT PAR7)
ACT PAR6:         SUBC (:NUM DSCR)
                  S = 4               " type = ar
                  G = MC[-1]          " simple identifier = false
                  GOTOR (MC[-1])
ACT PAR7:         B + 1               " reserve word for type
                  SUBC (:START ISR)
                  SUBC (:S EXP[30])
                  GOTO (:ACT PAR4)
ACT PAR8:         S = last symbol
                U, S - 64, Z          " last symbol = plus ?
                N, GOTO (:ACT PAR10)
                  SUBC (:NXT SBL)
                  S = digit last symbol, Z
                N, GOTO (:ACT PAR9)
                  SUBC (:UNS NUM)
                  S = last symbol
                U, S - 87, Z          " last symbol = comma ?
                N, S - 99, Z          " ∨ last symbol = close ?
                Y, GOTO (:ACT PAR6)
                  GOTO (:ACT PAR7)
ACT PAR9:         B + 1               " reserve word for type
                  SUBC (:START ISR)
                  SUBC (:AR BL EXP)
                  GOTO (:ACT PAR4)
ACT PAR10:      U, S - 65, Z          " last symbol = minus ?
                N, GOTO (:ACT PAR14)
                  SUBC (:NXT SBL)
                  S = digit last symbol, Z
                N, GOTO (:ACT PAR13)
                  SUBC (:UNS NUM)
                  S = last symbol
                U, S - 87, Z          " last symbol = comma ?
                N, S - 99, Z          " ∨ last symbol = close ?
                Y, S = small, Z       " ∧ small ?
```

```
                N, GOTO (:ACT PAR11)
                   A = 13
                   LUA (20)
                   A + value of constant[1]
                   GOTO (:ACT PAR6[1])
ACT PAR11:         B + 1                  " reserve word for type
                   SUBC (:START ISR)
                   SUBC (:AR CST)
                   SUBC (:NXT PRIM)
                   SUBC (:NXT FCTR)
ACT PAR12:         A = 1
                   SUBC (:MACRO)          " MACRO (NEG)
                   SUBC (:RST OF ABE)
                   GOTO (:ACT PAR4)
ACT PAR13:         B + 1                  " reserve word for type
                   SUBC (:START ISR)
                   SUBC (:TERM)
                   GOTO (:ACT PAR12)
ACT PAR14:      U, S - 115, P             " last symbol ≠ true
                U, S - 117, E             " ∧ last symbol ≠ false ?
                Y, GOTO (:ACT PAR16)
                   MC = S                 " n = last symbol
                   SUBC (:NXT SBL)
                U, S - 87, Z              " last symbol = comma ?
                N, S - 99, Z              " ∨ last symbol = close ?
                N, GOTO (:ACT PAR15)
                   A = 6
                   LUA (20)
                   A + MC[-1]             " n
                   A - 116
                   S = 2                  " type = bo
                   G = MC[-1]             " simple identifier = false
                   GOTOR (MC[-1])
ACT PAR15:         SUBC (:START ISR)
                   A = 89
                   S = M[B-1]
                   SUBC (:MACRO2)         " MACRO2 (TBC, n)
                   SUBC (:RST OF BE)
                   A = 2                  " type = bo
                   GOTO (:ACT PAR4)
ACT PAR16:         B + 1                  " reserve word for type
                   SUBC (:START ISR)
                   SUBC (:EXP)
                   GOTO (:ACT PAR4)       " 191 instructions

                   'END' ACT PAR


START ISR:         S = M[B-5], Z          " future = 0 ?
                Y, A = 102
                Y, SUBC (:MACRO2)         " MACRO2 (JU, future)
                Y, M[B-5] = S             " future
                   A = 35                 " MACRO (ENTRIS)
                   GOTO (:MACRO)          " 6 instructions


NUM DSCR:          S = small, Z           " small ?
                Y, A = 7
                Y, JUMP (3)
```

```
                  S = real number, Z    " real number ?
               N, A =   5
               Y, A =   4, Z            " (condition NO)
                  LUA (20)
               Y, A + value of constant[1]
               N, A + adrs of cst
                  GOTOR (MC[-1])        " 10 instructions

PRCS OP:          MC =  S               " n
                  A =   0
                  MC =  A               " count = 0
                  S = last symbol
               U, S -  98, Z            " last symbol = open ?
               N, JUMP (11)
PRCS OP[6]:       SUBC (:NXT SBL)
                  SUBC (:ARITHEXP)
                  A =   1
                  M[B-1] + A            " count = count + 1
                  S = last symbol
               U, S -  87, Z            " last symbol = comma ?
               Y, A =   0
               Y, SUBC (:MACRO)         " MACRO (STACK)
               Y, GOTO (:PRCS OP[6])
                  A = 361
                  SUBC (:REQ CLOSE[1]) " require close as last symbol
                  S = M[B-2]            " n
                  SUBC (:LST LTH)
                  A - MC[-1], Z         " list length = count ?
               N, A = 363
               N, SUBC (:ERRORM)
                  A = MS[-2]            " MACRO (operator macro (n))
                  GOTO (:MCR RET)       " 24 instructions

NON ASBL:         RUA (20)              " rank
               U, A -  14, P
               U, A -  15, E            " rank ≠ 15 ?
               Y, A '×' 15
               Y, A -   3, P            " ∧ rank - rank ÷ 16 × 16 > 3 ?
                  GOTO (MC[-1])         " 6 instructions

LINE:             A = lnc
                  A - last lnc, Z       " lnc = last lnc ?
LINE1:         N, A = - wanted, P       " ∨ ¬ wanted ?
               Y, GOTOR (MC[-1])
                  MC =  S
                  S = lnc
                  last lnc = S          " last lnc = lnc
                  A = 147               " MACRO2 (LNC, lnc)
                  GOTO (:MCR RET)       " 9 instructions

STATMNT:          S =   1               " ifstatement forbidden = false
                  JUMP (1)              " 2 instructions
```

```
UNC STAT:          S = 0                " ifstatement forbidden = true
                   ifstat frb = S       " 2 instructions

STAT:              S = line counter
                   lnc = S              " lnc = line counter
                   S = letter last symbol, Z
                N, JUMP (10)
                   SUBC (:IDF)
                   SUBC (:DESINAL)      " designational ?
STAT[6]:        Y, SUBC (:LAB DEC)
                Y, GOTO (:STAT)
                   SUBC (:LINE)
                   SUBC (:SUBSCR)       " subscripted ?
                N, A = last symbol
                N, A - 92, Z            " ∨ last symbol = colonequal ?
                Y, GOTO (:ASN STAT)
                   GOTO (:PR STAT)
                   S = digit last symbol, Z
                N, JUMP (6)
                   SUBC (:UNS NUM)
                   SUBC (:IN NM LST)    " in name list ?
                Y, S = int lab
                Y, GOTO (:STAT[6])
                   A = 364
                   GOTO (:ERRORM)
                   SUBC (:LINE)
                   S = last symbol
                U, S - 81, Z            " last symbol = goto ?
                Y, GOTO (:GOTO STAT)
                U, S - 104, Z           " last symbol = begin ?
                N, JUMP (5)
                   SUBC (:NXT SBL)
                   SUBC (:DECL LS)      " declarator last symbol ?
                Y, SUBC (:BLOCK)
                N, SUBC (:CMP TL)
                   GOTO (:NXT SBL)
                U, S - 94, Z            " last symbol = if ?
                N, JUMP (4)
                   S = ifstat frb, Z    " ifstatement forbidden ?
                Y, A = 365
                Y, SUBC (:ERRORM)
                   GOTO (:IF STAT)
                U, S - 82, Z            " last symbol = for ?
                Y, SUBC (:FOR STAT)
                Y, S = last symbol
                Y, S - 96, Z            " last symbol = else ?
                N, GOTOR (MC[-1])
                   A = 366
                   GOTO (:ERRORM)       " 46 instructions

GOTO STAT:         SUBC (:NXT SBL)
                   S = letter last symbol, Z
                N, SUBC (:DES EXP)
                N, JUMP (6)
                   SUBC (:IDF)
                   SUBC (:SUBS VAR)
                   SUBC (:LOC LAB)      " local label ?
```

```
                Y, SUBC (:TEST FC)
                Y, A = 102              " MACRO2 (JU, n)
                N, SUBC (:DS NAME)
                N, A = 74               " MACRO (JUA)
                   GOTO (:MACRO)        " 12 instructions

CMP TL:            SUBC (:STATMNT)
                   S = last symbol
                U, S - 91, Z            " last symbol = semicolon ?
                Y, SUBC (:NXT SBL)
                Y, GOTO (:CMP TL)
                U, S - 105, Z           " last symbol = end ?
                Y, GOTOR (MC[-1])
                   A = 367
                   SUBC (:ERRORM)
                   A = :STATMNT
                   SUBC (:SKIP RST OF STAT)
                   GOTO (:CMP TL[1])    " 12 instructions

IF STAT:           SUBC (:IFCLAUSE)
                   MC = S               " future1
                Y, SUBC (:NXT SBL)
                N, A = 368
                N, SUBC (:ERRORM)
                   S = line counter
                   MC = S               " save line counter
                   SUBC (:UNC STAT)
                   S = MC[-1]
                   last lnc = S         " last lnc = saved line counter
                   SUBC (:FUTURE)
                Y, S = line counter
                Y, MC = S               " save line counter
                Y, SUBC (:STATMNT)
                Y, S = MC[-1]
                Y, last lnc = S         " last lnc = saved line counter
                   A = MC[-1]           " future1 or future2
                   GOTO (:SUBSTT)       " 18 instructions

FOR STAT:          G = line counter
                   MC = F               " save line counter
                   S = 0
                   LO = S               " LO = 0
                   SUBC (:NXT SBL)
                   SUBC (:FOR LST)
                   A = 102
                   S = 0                " future = 0
                   SUBC (:MACRO2)       " MACRO2 (JU, future)
                   M[B-2] = S           " future
                   A = LO, Z
                N, SUBC (:SUBSTT)
                   S = last symbol
                U, S - 86, Z            " last symbol = do ?
                Y, SUBC (:NXT SBL)
                N, A = 369
                N, SUBC (:ERRORM)
                   A = 8192
```

```
                    SUBC (:INCR STS)
                    A =   1
                    for cnt + A          " forcount = forcount + 1
                    SUBC (:STATMNT)
                    A = - 8192
                    SUBC (:INCR STS)
                    A =   1
                    for cnt - A          " forcount = forcount - 1
                    S = MC[-1]
                U,  S - lnc, Z           " lnc = saved line counter ?
                N,  lnc = S              " else lnc = saved line counter
                N,  SUBC (:LINE1)
                    SUBC (:STATUS)
                    A = 104
                    SUBC (:MACRO2)       " MACRO2 (IJU, status)
                    A = MC[-1]           " future
                    GOTO (:SUBSTT)       " 35 instructions

STORE PR:           S = cntld var        " controlled variable
                    SUBC (:SUBSCR)       " subscripted ?
                Y,  A = 108
                Y,  S = - L2             " MACRO2 (SUBJ, - L2)
                Y,  GOTO (:MACRO2)
                    SUBC (:FORMAL)       " formal ?
                N,  GOTOR (MC[-1])
                    A = 100              " MACRO2 (DOS2, controlled variable)
                    GOTO (:MACRO2)       " 9 instructions

STORE MCR:          S = cntld var        " controlled variable
                    SUBC (:SUBSCR)       " subscripted ?
                N,  JUMP (8)
                    SUBC (:FORMAL)       " formal ?
                Y,  A =  34              " MACRO (STFSU)
                N,  A '×' 1
                N,  A +  29              " MACRO (STSR/STSI)
                    SUBC (:MACRO)
                    A = 110
                    S =   2
                    GOTO (:MACRO2)       " MACRO2 (DECB, 2)
                    SUBC (:FORMAL)       " formal ?
                Y,  A = 101              " MACRO2 (DOS3, controlled variable)
                N,  A '×' 1
                N,  A +  94              " MACRO2 (STR/STI, controlled variable)
                    GOTO (:MACRO2)       " 16 instructions

TAKE MCR:           S = cntld var        " controlled variable
                    SUBC (:SUBSCR)       " subscripted ?
                N,  GOTO (:AR NAME)
                    A = 108
                    S = - L1             " MACRO2 (SUBJ, - L1)
                    GOTO (:MACRO2)       " 6 instructions
```

```
FOR LST:        'BEGIN'  FOR LST0, FOR LST1, FOR LST2, FOR LST3, FOR LST4

                S = letter last symbol, Z
             N, A = 375
             N, GOTO (:ERRORM)
                SUBC (:IDF)
                cntld var = S          " controlled variable
                SUBC (:NON AR)         " nonarithmetic ?
             Y, A = 370
             Y, SUBC (:ERRORM)
                SUBC (:SUBSCR)         " controlled variable subscripted ?
             N, GOTO (:FOR LST0)
                A = 102
                S =  0                 " L3 = 0
                SUBC (:MACRO2)         " MACRO2 (JU, L3)
                L3 =  S                " L3
                SUBC (:ORD CNT)
                L4 =  S                " L4 = order counter
                S = cntld var          " controlled variable
                SUBC (:ADR DSCR)
                A = 127
                S = - dimension
                LUS (1)
                S + 1
                SUBC (:MACRO2)         " MACRO2 (EXITSV, 1 - 2 × dimension)
                SUBC (:ORD CNT)
                L1 =  S                " L1 = order counter
                A = 108
                S = - L4
                SUBC (:MACRO2)         " MACRO2 (SUBJ, - L4)
                S = cntld var          " controlled variable
                SUBC (:FORMAL)         " formal ?
             Y, A =  49                " MACRO (TSCVU)
             N, A '×' 1
             N, A + 47                 " MACRO (TRSCV/TISCV)
                SUBC (:MACRO)
                SUBC (:ORD CNT)
                L2 =  S                " L2 = order counter
                A = 108
                S = - L4
                SUBC (:MACRO2)         " MACRO2 (SUBJ, - L4)
                A =  46
                SUBC (:MACRO)          " MACRO (FADCV)
                A =  L3
                SUBC (:SUBSTT)
                JUMP (3)
FOR LST0:       SUBC (:FUNCTN)         " function ?
             Y, A = 371
             Y, SUBC (:ERRORM)
                S = last symbol
             U, S -  92, Z             " last symbol = colonequal ?
             N, A = 372
             N, SUBC (:ERRORM)
FOR LST1:       SUBC (:ORD CNT)
                L3 =  S                " L3 = order counter
                A =  87
                S =  0
                SUBC (:MACRO2)         " MACRO2 (TSIC, 0)
                SUBC (:STATUS)
```

```
                  A = 96
                  SUBC (:MACRO2)          " MACRO2 (SSTI, status)
                  SUBC (:ORD CNT)
                  L4 = S                  " L4 = order counter
                  SUBC (:STORE PR)
                  SUBC (:NXT SBL)
                  SUBC (:ARITHEXP)
                  S = last symbol
               U, S - 87, Z               " last symbol = comma ?
               N, S - 86, Z               " V last symbol = do ?
               N, GOTO (:FOR LST2)
                  SUBC (:STORE MCR)
                  A = 102
                  S = L0
                  SUBC (:MACRO2)          " MACRO2 (JU, L0)
                  L0 = S                  " L0
                  A = L3
                  SUBC (:SUBSTT)
                  GOTO (:FOR LST4)
FOR LST2:         S + 1, Z                " last symbol = while ?
               N, GOTO (:FOR LST3)
                  SUBC (:STORE MCR)
                  SUBC (:NXT SBL)
                  SUBC (:BOOLEXP)
                  A = 107
                  S = L0
                  SUBC (:MACRO2)          " MACRO2 (YCOJU, L0)
                  L0 = S                  " L0
                  A = L3
                  S = L4
                  SUBC (:SUBST2)
                  GOTO (:FOR LST4)
FOR LST3:         S - 25, Z               " last symbol = step ?
               N, A = 374
               N, SUBC (:ERRORM)
               N, GOTO (:FOR LST4)
                  A = 102
                  SUBC (:MACRO2)          " MACRO2 (JU, L5)
                  L5 = S                  " L5
                  SUBC (:ORD CNT)
                  L4 = S                  " L4 = order counter
                  cmpltd = - S            " complicated = false
                  SUBC (:NXT SBL)
                  SUBC (:ARITHEXP)
                  SUBC (:ORD CNT)
                  S - L4
                  S - 1, P                " order counter > L4 + 1 ?
               N, S = cmpltd, P           " V complicated ?
                  cmpl st = S             " complex step element
               Y, A = 50
               Y, SUBC (:MACRO)           " MACRO (EXIT)
                  A = L3
                  SUBC (:SUBSTT)
                  SUBC (:STORE PR)
                  SUBC (:TAKE MCR)
                  A = 0
                  SUBC (:MACRO)           " MACRO (STACK)
                  S = cmpl st, P          " complex step element ?
               Y, A = 108
```

```
                Y, S = - L4             " MACRO2 (SUBJ, - L4)
                N, A = 111
                N, S =  L4              " MACRO2 (DO, L4)
                   SUBC (:MACRO2)
                   A =  2
                   SUBC (:MACRO)        " MACRO (ADD)
                   A =  L5
                   SUBC (:SUBSTT)
                   SUBC (:STORE MCR)
                   S = cntld var        " controlled variable
                   SUBC (:SUBSCR)       " subscripted ?
                N, SUBC (:FORMAL)       " ∨ formal ?
                Y, SUBC (:TAKE MCR)
                   A =  0
                   SUBC (:MACRO)        " MACRO (STACK)
                   S = last symbol
                U, S - 84, Z            " last symbol = until ?
                Y, SUBC (:NXT SBL)
                Y, SUBC (:ARITHEXP)
                N, A = 373
                N, SUBC (:ERRORM)
                   A =  51
                   SUBC (:MACRO)        " MACRO (TEST1)
                   S = cmpl st, P       " complex step element ?
                Y, A = 108
                Y, S = - L4             " MACRO2 (SUBJ, - L4)
                N, A = 111
                N, S =  L4              " MACRO2 (DO, L4)
                   SUBC (:MACRO2)
                   A =  52
                   SUBC (:MACRO)        " MACRO (TEST2)
                   A = 107
                   S =  L0
                   SUBC (:MACRO2)       " MACRO2 (YCOJU, L0)
                   L0 =  S              " L0
FOR LST4:          S = last symbol
                U, S - 87, Z            " last symbol = comma ?
                Y, GOTO (:FOR LST1)
                   GOTOR (MC[-1])       " 155 instructions

                   'END' FOR LST


SW DEC:            'BEGIN'  SW DEC0, SW DEC1, SW DEC2, SW DEC3, SW DEC4,
                            SW DEC5, SW DEC6, SWORD0, SWORD1, SWORD2

                   SUBC (:NXT SBL)
                   S = letter last symbol, Z
                N, A = 381
                N, GOTO (:ERRORM)
                   SUBC (:IDF)
                   sw idf = S           " switch identifier
                   SUBC (:LST LTH)
                   nbr of sw e = A      " number of switch elements
                   S = last symbol
                   S - 92, Z            " last symbol = colonequal ?
                N, A = 380
                N, GOTO (:ERRORM)
                   sw lst = B           " sword list
```

```
                  in sw dec = S          " in switch declaration = true
SW DEC0:          SUBC (:NXT SBL)
                  S = letter last symbol, Z
               N, GOTO (:SW DEC3)
                  SUBC (:IDF)
                  SUBC (:NON DES)        " nondesignational ?
               Y, A = 376
               Y, SUBC (:ERRORM)
                  SUBC (:SUBSCR)         " subscripted ?
               N, GOTO (:SW DEC2)
                  MC = S                 " m
                  SUBC (:ORD CNT)
                  S + SWORD0
                  A = MC[-1]             " m
                  MC = S                 " SWORD
                  S = A
                  SUBC (:SUBS VAR)
SW DEC1:          A = 50
                  SUBC (:MACRO)          " MACRO (EXIT)
                  GOTO (:SW DEC5)
SW DEC2:          SUBC (:FORMAL)         " formal ?
               Y, A = SWORD1
               Y, JUMP (3)
SW DEC2[3]:       SUBC (:DYNAMIC)        " dynamic ?
                  A = SWORD2
               Y, A + func dit
                  MC = A
                  SUBC (:ADDRSS)
                  M[B-1] + A             " SWORD
                  GOTO (:SW DEC5)
SW DEC3:          S = digit last symbol, Z
               N, GOTO (:SW DEC4)
                  SUBC (:UNS NUM)
                  SUBC (:IN NM LST)      " in name list ?
               N, A = 377
               N, SUBC (:ERRORM)
                  S = int lab            " integer label
                  GOTO (:SW DEC2[3])
SW DEC4:          SUBC (:ORD CNT)
                  S + SWORD0
                  MC = S                 " SWORD
                  SUBC (:DES EXP)
                  GOTO (:SW DEC1)
SW DEC5:          S = sw lst
                  S + nbr of sw e
               U, B - S, P               " switch list count >
               Y, A = 378                "       number of switch elements ?
               Y, SUBC (:ERRORM)
                  S = last symbol
               U, S - 87, Z              " last symbol = comma ?
               Y, GOTO (:SW DEC0)
                  S = sw lst
                  in sw dec = S          " in switch declaration = false
                  S + nbr of sw e
               U, S - B, P               " switch list count <
               Y, A = 379                "       number of switch elements ?
               Y, SUBC (:ERRORM)
                  S = sw idf             " switch identifier
                  SUBC (:MRK POS)
```

```
                    A = 128
                    S = nbr of sw e
                    SUBC (:MACRO2)          " MACRO2 (CODE,
SW DEC6:            S =   1                 "          number of switch elements)
                    PLUSS (in sw dec)       " m = m + 1
                U,  S -   B, P              " m > switch list count ?
                Y,  B = sw lst
                Y,  GOTOR (MC[-1])
                    A = 128
                    S = MS[-1]
                    SUBC (:MACRO2)          " MACRO2 (CODE, sword list[m])
                    GOTO (:SW DEC6)
SWORD0:             GOTO (:M[0])
SWORD1:             DOS (MO[-256])
SWORD2:             A = :M[0]               " 87 instructions

                    'END' SW DEC


ARR DEC:            S = line counter
                    lnc = S                 " lnc = line counter
                    SUBC (:LINE)
                    SUBC (:NXT SBL)
                    SUBC (:IDF)
                    MC =  S                 " n
                    S =   1
                    MC =  S                 " count = 1
ARR DEC[8]:         S = last symbol
                U,  S -  87, Z              " last symbol = comma ?
                Y,  SUBC (:NXT SBL)
                Y,  SUBC (:IDF)
                Y,  S =   1
                Y,  M[B-1] + S              " count = count + 1
                Y,  GOTO (:ARR DEC[8])
                    S - 100, Z              " last symbol = sub ?
                Y,  in ar dec = S           " in array declaration = true
                Y,  SUBC (:BND PR LST)
                    A = 382
                Y,  in ar dec = A           " in array declaration = false
                N,  SUBC (:ERRORM)
                    A = 122
                    S = MC[-1]              " count
                    SUBC (:MACRO2)          " MACRO2 (TNA, count)
                    S = M[B-1]              " n
                    SUBC (:LST LTH)
                    S =   A
                    A = 121
                    SUBC (:MACRO2)          " MACRO2 (TDA, dimension)
                    A = 123
                    S = MC[-1]              " n
                    SUBC (:MACRO2)          " MACRO2 (TAA, n)
                    A = arr dec mcr
                    A +  66
                    SUBC (:MACRO)           " MACRO (arr decla macro)
                    S = last symbol
                U,  S -  87, Z              " last symbol = comma ?
                Y,  GOTO (:ARR DEC)
                    GOTOR (MC[-1])          " 39 instructions
```

```
BND PR LST:     SUBC (:NXT SBL)
                SUBC (:ARITHEXP)
                A = 0
                SUBC (:MACRO)          " MACRO (STACK)
                S = last symbol
              U, S - 90, Z             " last symbol = colon ?
              Y, SUBC (:NXT SBL)
              Y, SUBC (:ARITHEXP)
              Y, A = 0
              Y, SUBC (:MACRO)         " MACRO (STACK)
              N, A = 383
              N, SUBC (:ERRORM)
                S = last symbol
              U, S - 87, Z             " last symbol = comma ?
              Y, GOTO (:BND PR LST)
                A = 384                " require bus as last symbol
                GOTO (:REQ BUS)        " 17 instructions

PR DEC:         'BEGIN' PR DEC0, PR DEC1, PR DEC2, PR DEC3,
                        PR DEC4, PR DEC5, PR DEC6

                SUBC (:NXT SBL)
                SUBC (:IDF)
                MC = S                 " n
                SUBC (:SKP PR LI)
                SUBC (:SKP VA LI)
                SUBC (:SKP SP LI)
                S = M[B-1]             " n
                SUBC (:IN LIBR)        " in library ?
              N, SUBC (:MRK POS)
                S = M[B-1]             " n
                SUBC (:IN CODE)        " in code ?
              Y, B - 1
              Y, GOTO (:TRL CD)
                SUBC (:FUNCTN)         " function ?
              Y, SUBC (:SET IN DEC)
                SUBC (:ENTR BLK)
                SUBC (:DISP LVL)
                A = 114
                SUBC (:MACRO2)         " MACRO2 (DPTR, display level)
                SUBC (:TP OF DSP)
                A = 115
                SUBC (:MACRO2)         " MACRO2 (INCRB, top of display)
                S = M[B-1]             " n
                SUBC (:LST LTH)
                COUNT = A
                GOTO (:PR DEC3)
PR DEC0:        SUBC (:NXT F I)
                MC = S                 " f = next formal identifier
                SUBC (:IN VA LI)       " in value list ?
              N, GOTO (:PR DEC1)
                SUBC (:SUBSCR)         " subscripted ?
              Y, A = 58                " MACRO (CEN)
              Y, GOTO (:PR DEC2)
                SUBC (:DESINAL)        " designational ?
              Y, A = 57                " MACRO (CLV)
              N, A + 53                " MACRO (CRV/CIV/CBV/CSTV)
                GOTO (:PR DEC2)
```

```
PR DEC1:        SUBC (:ASS TO)          " assigned to f ?
              Y, A = 59                 " MACRO (CLPN)
              N, A = 58                 " MACRO (CEN)
PR DEC2:        SUBC (:MACRO)
                S = MC[-1]              " f
PR DEC3:        REPE (:PR DEC0)
                SUBC (:DISP LVL)
                A = 116
                SUBC (:MACRO2)          " MACRO2 (TDL, display level)
                SUBC (:LOC SPC)
                A = 117
                SUBC (:MACRO2)          " MACRO2 (ENTRPB, local space)
                SUBC (:LAB LST)
                S = M[B-1]              " n
                SUBC (:LST LTH)
                COUNT = A
                GOTO (:PR DEC5)
PR DEC4:        SUBC (:NXT F I)         " f = next formal identifier
                SUBC (:IN VA LI)        " in value list ?
              Y, SUBC (:SUBSCR)         " ∧ subscripted ?
              N, GOTO (:PR DEC5)
                MC = S                  " f
                SUBC (:INTEGER)         " integer ?
                A = 123
                SUBC (:MACRO2)          " MACRO2 (TAA, f)
              Y, A = 61                 " MACRO (TIAV)
              N, A = 60                 " MACRO (TAV)
                SUBC (:MACRO)
                S = MC[-1]              " f
PR DEC5:        REPE (:PR DEC4)
                last lnc = - S          " make last lnc ≠ line counter
                S = M[B-1]              " n
                F = 145
                SUBC (:SAVE LNC)        " MACRO2 (SLNC, n)
                S = last symbol
              U, S - 104, Z             " last symbol = begin ?
              N, SUBC (:STATMNT)
              N, JUMP (5)
                SUBC (:NXT SBL)
                SUBC (:DECL LS)         " declarator last symbol ?
              Y, SUBC (:DEC LST)
                SUBC (:CMP TL)
                SUBC (:NXT SBL)
                S = M[B-1]              " n
                SUBC (:NON FCT)         " nonfunction ?
              Y, GOTO (:PR DEC6)
                SUBC (:SET IN DEC)
                SUBC (:LOC POS)
                SUBC (:ARMETIC)         " arithmetic ?
              Y, SUBC (:AR NAME)
              Y, GOTO (:PR DEC6)
                SUBC (:BOOLEAN)         " Boolean ?
              Y, SUBC (:BL NAME)
              N, SUBC (:ST NAME)
              N, A = 70
              N, SUBC (:MACRO)          " MACRO (LOS)
PR DEC6:        S = MC[-1]              " n
                F = 146
                SUBC (:SAVE LNC)        " MACRO2 (RLNC, n)
```

```
                    SUBC (:USE CST)      " use of counter stack ?
                Y,  A =  72              " MACRO (EXITPC)
                N,  A =  71              " MACRO (EXITP)
                    SUBC (:MACRO)
                    GOTO (:EXIT BLK)     " 101 instructions

                    'END' PR DEC


SAVE LNC:           A = wanted, P        " wanted ?
                Y,  SUBC (:FUNCTN)       " ∧ function ?
                N,  GOTOR (MC[-1])
                    A =  G
                    S -  2               " local position
                    GOTO (:MACRO2)       " 6 instructions


BLOCK:              SUBC (:ENTR BLK)
                    SUBC (:DISP LVL)
                    A = 112
                    SUBC (:MACRO2)       " MACRO2 (TBL, display level)
                    SUBC (:LOC SPC)
                    A = 113
                    SUBC (:MACRO2)       " MACRO2 (ENTRB, local space)
                    SUBC (:LAB LST)
                    SUBC (:DEC LST)
                    SUBC (:CMP TL)
                    SUBC (:USE CST)      " use of counter stack ?
                    SUBC (:DISP LVL)
                Y,  A = 126              " MACRO2 (EXITC, display level)
                N,  A = 125              " MACRO2 (EXITB, display level)
                    SUBC (:MACRO2)
                    GOTO (:EXIT BLK)     " 16 instructions


DEC LST:            'BEGIN'  DEC LST0, DEC LST1, DEC LST2

                    F =  0
                    MC =  F              " future = arr dec = 0
DEC LST0:           S = tp dec ls, Z     " type declarator last symbol ?
                Y,  SUBC (:SKP TP DEC)
                Y,  GOTO (:DEC LST2)
                    S = arr dec ls, Z    " arr declarator last symbol ?
                N,  GOTO (:DEC LST1)
                    A = M[B-1], Z        " future = 0 ?
                N,  M[B-1] = S           " future = 0
                N,  SUBC (:SUBSTT)
                    A =  1
                    M[B-2] = A           " arr dec = 1
                    SUBC (:ARR DEC)
                    GOTO (:DEC LST2)
DEC LST1:           S = M[B-1], Z        " future = 0 ?
                Y,  A = 102
                Y,  SUBC (:MACRO2)       " MACRO2 (JU, future)
                Y,  M[B-1] = S           " future
                    S = last symbol
                U,  S - 113, Z           " last symbol = switch ?
                Y,  SUBC (:SW DEC)
                N,  SUBC (:PR DEC)
```

```
DEC LST2:          S = last symbol
                U, S - 91, Z          " last symbol = semicolon ?
                Y, SUBC (:NXT SBL)
                N, A = 385
                N, SUBC (:ERRORM)
                   SUBC (:DECL LS)    " declarator last symbol ?
                Y, GOTO (:DEC LST0)
                   A = MC[-1], Z      " future = 0 ?
                N, SUBC (:SUBSTT)
                   A = MC[-1], Z      " arr dec = 0 ?
                Y, GOTOR (MC[-1])
                   SUBC (:DISP LVL)
                   A = 124            " MACRO2 (SWP, display level)
                   GOTO (:MACRO2)     " 36 instructions

                   'END' DEC LST


LAB LST:           SUBC (:NBR LL)
                U, S -  0, Z          " number of local labels = 0 ?
                Y, GOTOR (MC[-1])
                   MC = S             " count
                   LUS (1)
                   A = 110
                   SUBC (:MACRO2)     " MACRO2 (DECB, 2 × count)
                   SUBC (:DISP LVL)
                   A = 120
                   SUBC (:MACRO2)     " MACRO2 (LAD, display level)
                   S =  0             " n = 0
LAB LST[11]:       SUBC (:NEXT LL)    " n = next local label
                   SUBC (:SUP LOC)    " super local ?
                Y, GOTO (:LAB LST[11])
                   A =  1
                   M[B-1] - A, Z      " count = 1 ?
                Y, A = 119            " MACRO2 (LAST, n)
                N, A = 118            " MACRO2 (NIL, n)
                N, MC = S             " n
                   SUBC (:MACRO2)
                   S = MC[-1]         " n
                N, GOTO (:LAB LST[11])
                   GOTOR (MC[-1])     " 23 instructions


PROGRAM:           A = letter last symbol, Z
                Y, SUBC (:IDF)
PROGRAM[2]:     Y, A = last symbol
                Y, A - 90, Z          " last symbol = colon ?
                Y, SUBC (:LAB DEC)
                Y, GOTO (:PROGRAM)
                   A = digit last symbol, Z
                Y, SUBC (:UNS NUM)
                Y, SUBC (:IN NM LST)  " in name list ?
                Y, S = int lab
                Y, GOTO (:PROGRAM[2])
                   S = last symbol
                U, S - 104, Z         " last symbol = begin ?
                   SUBC (:NXT SBL)
                N, JUMP (-3)
                   SUBC (:DECL LS)    " declarator last symbol ?
```

```
                Y, SUBC (:BLOCK)
                N, SUBC (:CMP TL)
                   A = 82                " MACRO (END)
                   GOTO (:MACRO)         " 20 instructions

LAB DEC:           last lnc = - S        " make last lnc ≠ line counter
                   SUBC (:SUBSCR)        " subscripted ?
                Y, A = 388
                Y, SUBC (:ERRORM)
                Y, SUBC (:SUBS VAR)
                N, SUBC (:MRK POS)
                   S = last symbol
                U, S - 90, Z             " last symbol = colon ?
                Y, GOTO (:NXT SBL)
                   A = 389
                   GOTO (:ERRORM)        " 11 instructions
```

```
SUBSTT:          MC = A                  " address
                 SUBC (:ORD CNT)
                 A = MC[-1]              " 3 instructions

SUBST2:          MC = S                  " address1
                 G = A, P
              N, F = - F                 " abs (address2)
                 S = MG                  " instruction
                 A = S
                 S 'X' - 32767           " instruct part
                 A 'X' 32767, Z          " address part
                 S + M[B-1]              " address1
                 MG = S
              N, GOTO (:SUBST2[1])
                 A = S
                 RUA (15)
              U, A - end of list, Z      " instruct part = end of list ?
              Y, MG = - S
                 GOTO (:RETURN)          " 15 instructions

ORD CNT:         A = 75
                 SUBC (:MACRO)           " MACRO (EMPTY)
                 S = instr cntr
                 GOTOR (MC[-1])          " 4 instructions

MACRO:
MACRO2:          'BEGIN' STATE0, STATE1, STATE2, STATE3, END

                 A + :MCR LST
                 A = MA
                 mcr = A
                 par = S
                 S = state
                 JUMP (S)                " switch according to state
                 GOTO (:STATE0)
                 GOTO (:STATE1)
                 GOTO (:STATE2)
STATE3:       U, A - NEG, Z              " macro = NEG ?
              Y, GOTO (:OPTIMIZE)
                 SUBC (:UNLOAD)
                 A = mcr
STATE0:       U, A - STACK, Z            " macro = STACK ?
              Y, A = 1
              Y, state = A               " state = 1
              Y, GOTOR (MC[-1])
                 SUBC (:SAT)             " simple arithmetic take macro ?
              Y, S = 3
              Y, GOTO (:LOAD)
                 S = par
                 SUBC (:PRODUCE)
END:             S = - instr cntr
                 S + 1
                 GOTOR (MC[-1])
STATE1:          S = 2
                 SUBC (:LOAD)
                 SUBC (:SAT)             " simple arithmetic take macro ?
```

```
                Y, GOTOR (MC[-1])
                   A = STACK
                   SUBC (:PRODUCE)
                   SUBC (:UNLOAD)
                   GOTO (:END)
STATE2:            SUBC (:OPT OP)        " optimizable operator ?
                Y, GOTO (:OPTIMIZE)
                   A = STACK
                   SUBC (:PRODUCE)
                   A = mcr
                   GOTO (:STATE3)        " 39 instructions

                   'END' MACRO


LOAD:              state = S             " state = state i
                   stack0 = A            " stack0 = macro
                   S = par
                   stack1 = S            " stack1 = parameter
                   GOTOR (MC[-1])        " 5 instructions


OPTIMIZE:          A = mcr
                   SUBC (:OPT NBR)
                   MULS (5)
                   MC = S
                   A = stack0
                   SUBC (:OPT NBR)
                   S + MC[-1]
                   S + :TABEL
                   A = MS[-5]
                   stack0 = A            " 10 instructions


UNLOAD:            S = 0
                   state = S             " state = 0
                   A = stack0
                   S = stack1            " 4 instructions


PRODUCE:           'BEGIN' CODE0, NEXT

                U, A - EMPTY, Z          " macro = EMPTY ?
                Y, GOTOR (MC[-1])
                U, A - CODE, Z           " macro = CODE ?
                N, JUMP (4)
CODE0:             A = 1
                   PLUSA (instr cntr)
                   MA[-1] = S
                   GOTO (:TEST IC)
                   MC = S                " parameter
                   SUBC (:INSTR NBR)
                   MC = S                " number
                   SUBC (:PAR PART)
                   MC = S, Z             " par number = 0 ?
                   SUBC (:INSTR PRT)
                   S + :INSTR LST
                   MC = S                " entry
                   SUBC (:PRCS STP)
```

```
                N, SUBC (:PRCS PAR)
NEXT:              A =   1
                   M[B-2] - A, Z        " par number = count ?
                   PLUSA (M[B-1])       " entry = entry + 1
                   S = MA[-1]
                Y, S + M[B-4]           " parameter
                   SUBC (:CODEO)
                   A =   1
                   M[B-3] - A, P        " number = number - 1
                Y, GOTO (:NEXT)
                   B -   4
                   GOTOR (MC[-1])       " 29 instructions

                   'END' PRODUCE


PRCS STP:          'BEGIN'  REACT10, REACT11, REACT12,
                            REACT13, REACT14, REACT15

                   S = code body, P
                Y, GOTOR (MC[-1])
                   SUBC (:B REACT)
                U, S -   8, P           " reaction ≥ 9 ?
                Y, JUMP (S)
                   S -   4
                   PLUSS (st cnt)       " B
                U, S - max dpth, P      " B > max depth ?
                Y, max dpth = S
                   GOTOR (MC[-1])
REACT11:           S = dimension
                   LUS (1)
                   S -   2
                   st cnt - S           " B = B - 2 × (dimension - 1)
                   GOTOR (MC[-1])
                   GOTO (:REACT10)
                   GOTO (:REACT11)
                   GOTO (:REACT12)
                   GOTO (:REACT13)
                   GOTO (:REACT14)
REACT15:           S = st cnt           " B
                   st cnt - S
                U, S - max prl, P       " B > max proc level ?
                Y, max prl = S
REACT15[4]:        S = ret md
                   max dpth = S         " max depth = ret max depth
                   GOTOR (MC[-1])
REACT12:           S = Ecnt, Z          " Ecount = 0 ?
                   S +   1
                   Ecnt = S
                N, GOTOR (MC[-1])
                   S = st cnt
                   ret lvl = S          " ret level = B
                   S = max dpth
                   ret md = S           " ret max depth = max depth
                   S = max di
                   max dpth = S         " max depth = max depth isr
REACT10:           S =   0
                   st cnt = S           " B = 0
                   GOTOR (MC[-1])
```

```
REACT13:        U, A - EXITSV, Z        " macro = EXITSV ?
                N, JUMP (4)
                   S = st cnt           " B
                U, S - max di, P        " B > max depth isr ?
                Y, max di = S
                   SUBC (:REACT11)
                   S = Ecnt, Z          " Ecount = 0 ?
                Y, GOTOR (MC[-1])
                   S -   1, Z           " Ecount = 1 ?
                   Ecnt = S
                N, GOTOR (MC[-1])
                   S = max dpth
                U, S - max di, P        " max depth > max depth isr ?
                Y, max di = S
                   S = ret lvl
                   st cnt = S           " B = ret level
                   GOTO (:REACT15[4])
REACT14:           SUBC (:DISP LVL)
                   st cnt = S
                   SUBC (:TP OF DSP)
                   PLUSS (st cnt)       " B = display level + top of display
                U, S - max dl, P        " B > max display length ?
                Y, max dl = S
                   S = max dpth
                   ret md = S           " ret max depth = max depth
                   GOTOR (MC[-1])       " 66 instructions

                   'END' PRCS STP


PRCS PAR:          'BEGIN'  ABS, NON VAL, VARIANT

                   SUBC (:VAL LK)       " value like ?
                   S = M[B-5]           " parameter
                N, GOTO (:NON VAL)
                U, A - TBC, Z           " macro = TBC ?
                Y, S - 116
                U, A - SWP, Z           " macro = SWP ?
                Y, LUS (9)
                U, A - EXITSV, Z        " macro = EXITSV ?
                N, M[B-5] = S, P
ABS:            N, M[B-5] = - S         " parameter
                   GOTOR (MC[-1])
NON VAL:           MC =  A              " macro
                U, A -  JU, Z           " macro = JU ?
                Y, JUMP (6)
                U, A - SUBJ, Z          " macro = SUBJ ?
                Y, JUMP (4)
                U, A - NIL, Z           " macro = NIL ?
                Y, JUMP (2)
                U, A - LAST, Z          " macro = LAST ?
                N, JUMP (5)
                   S -   0, P           " parameter > 0 ?
                Y, SUBC (:PR ADR)
                Y, M[B-6] = A           " parameter
                   A = MC[-1]           " macro
                   GOTO (:ABS)
                   SUBC (:DYNAMIC)      " dynamic ?
                   SUBC (:ADDRSS)
```

```
                    M[B-6] = A          " parameter
                    A = MC[-1]          " macro
                N, GOTOR (MC[-1])
                    S = func dit
                U, A - TLV, Z           " macro = TLV ?
                Y, GOTO (:VARIANT)
                U, A - TAA, Z           " macro = TAA ?
                Y, GOTO (:VARIANT)
                U, A - STST, Z          " macro = STST ?
                Y, S = func ltr
                N, S = C var
VARIANT:        M[B-5] + S              " parameter
                GOTOR (MC[-1])          " 40 instructions

                'END' PRCS PAR

SAT:        U, A 'X' 1, Z
                GOTO (:INVERT)          " 2 instructions

OPT OP:     U, A 'X' 2, Z
                GOTO (:INVERT)          " 2 instructions

VAL LK:     U, A 'X' 8, Z
                GOTO (:INVERT)          " 2 instructions

OPT NBR:        S =  A
                RUS (4)
                S 'X' 15
                GOTOR (MC[-1])          " 4 instructions

INSTR NBR:      S =  A
                RUS (8)
CLEAR:          S 'X' 3
                GOTOR (MC[-1])          " 4 instructions

PAR PART:       S =  A
                RUS (10)
                GOTO (:CLEAR)           " 3 instructions

INSTR PRT:      S =  A
                RUS (12)
                S 'X' 511
                GOTOR (MC[-1])          " 4 instructions

B REACT:        S =  A
                RUS (21)
                GOTOR (MC[-1])          " 3 instructions
```

```
CHARCTR:        A = MS
                RUA (19)
                A 'X' 31            " character
              U, A - 24, Z          " = 24 ?
                GOTO (:INVERT)      " 5 instructions

ARMETIC:        SUBC (:CHARCTR)     " character ≠ 24 ?
              N, GOTO (MC[-1])
              U, A 'X' 6, Z         " type bits = 0, 1 ?
              N, A 'X' 3, Z         " ∨ type bits = 4 ?
                GOTO (MC[-1])       " 5 instructions

REAL:           SUBC (:CHARCTR)     " character ≠ 24 ?
              Y, A 'X' 7, Z         " ∧ type bits = 0 ?
                GOTO (MC[-1])       " 3 instructions

INTEGER:        A = MS
                RUA (19)
                A 'X' 7             " type bits
              U, A - 1, Z           " = 1 ?
                GOTO (MC[-1])       " 5 instructions

BOOLEAN:        SUBC (:INTEGER)
              U, A - 2, Z           " type bits = 2 ?
                GOTO (MC[-1])       " 3 instructions

STRING:         SUBC (:INTEGER)
              U, A - 3, Z           " type bits = 3 ?
                GOTO (MC[-1])       " 3 instructions

DESINAL:
NO TYPE:        SUBC (:INTEGER)
              U, A - 6, Z           " type bits = 6 ?
                GOTO (MC[-1])       " 3 instructions

ARBOST:         SUBC (:CHARCTR)     " character ≠ 24 ?
              N, GOTO (MC[-1])
                GOTO (:NON DES)     " 3 instructions

UNKNOWN:        SUBC (:INTEGER)
              U, A - 7, Z           " type bits = 7 ?
                GOTO (MC[-1])       " 3 instructions

NON AR:         SUBC (:CHARCTR)     " character ≠ 24 ?
              N, GOTO (:INVERT)
                A 'X' 7             " type bits
              U, A - 2, Z           " = 2 ?
              N, A - 3, Z           " ∨ = 3 ?
              N, A - 3, Z           " ∨ = 6 ?
                GOTO (MC[-1])       " 7 instructions
```

```
NON RE:          SUBC (:NON AR)        " nonarithmetic ?
              N, A +  5, Z             " ∨ type bits = 1 ?
                 GOTO (MC[-1])         " 3 instructions

NON IN:          SUBC (:NON AR)        " nonarithmetic ?
              N, A +  6, Z             " ∨ type bits = 0 ?
                 GOTO (MC[-1])         " 3 instructions

NON BL:          SUBC (:BOOLEAN)       " Boolean ?
              N, A -  5, Z             " ∨ type bits = 5 ?
              N, A -  2, Z             " ∨ type bits = 7 ?
                 GOTO (:INVERT)        " 4 instructions

NON STR:         SUBC (:STRING)        " string ?
                 GOTO (:NON BL[1])     " 2 instructions

NON DES:         SUBC (:INTEGER)
              U, A -  5, P             " type bits > 5 ?
                 GOTO (:INVERT)        " 3 instructions

SIMPLE:          A = MS
                 RUA (19)              " code bits
              U, A - 127, Z            " = 127 ?
              N, A 'X' 24, Z           " ∨ character : 8 = 0 ?
                 GOTO (MC[-1])         " 5 instructions

SIMPLE1:         A = MS
                 RUA (22)
                 A 'X' 3, Z            " character : 8 = 0 ?
                 GOTO (MC[-1])         " 4 instructions

SUBSCR:          SUBC (:SIMPLE1)
                 A -  1, Z             " character : 8 = 1 ?
                 GOTO (MC[-1])         " 3 instructions

PROC:            A = MS
                 RUA (19)
              U, A - 127, Z            " code bits = 127 ?
              N, A 'X' 16, Z           " ∨ character : 8 < 2 ?
                 GOTO (:INVERT)        " 5 instructions

FUNCTN:          SUBC (:SIMPLE1)
                 A -  2, Z             " character : 8 = 2 ?
                 GOTO (MC[-1])         " 3 instructions

NON SMPL:        SUBC (:SIMPLE)        " simple ?
              Y, GOTO (:INVERT)
                 A -  16, Z            " function ?
              N, SUBC (:FORMAL)        " ∨ formal ?
```

```
                Y, SUBC (:PROC)          " ∧ proc ?
                Y, A = MS[-1]
                Y, A 'X' 32766, Z        " ∧ list length < 1 ?
                   GOTO (:INVERT)        " 8 instructions

NON SUBS:          SUBC (:SIMPLE1)       " simple1 ?
                Y, GOTO (MC[-1])
                   GOTO (:PROC)          " 3 instructions

NON PROC:          A = MS
                   RUA (18)
                U, A 'X' 32, Z           " character : 8 < 2 ?
                N, GOTO (MC[-1])
                U, A - 191, P            " formal ?
                Y, A 'X' 49, Z           " ∧ simple1 ∧ ¬ assigned to ?
                   GOTO (:INVERT)        " 7 instructions

NON FCT:           SUBC (:FUNCTN)        " function ?
                N, SUBC (:FORMAL)        " ∨ formal ?
                   GOTO (:INVERT)        " 3 instructions

FORMAL:            A = MS
                   RUA (19)
                U, A - 95, P             " code bits > 95 ?
                   GOTO (MC[-1])         " 4 instructions

IN VA LI:          SUBC (:FORMAL)        " formal ?
                U, A - 63, E             " ∨ code bits < 64 ?
                   GOTO (:INVERT)        " 3 instructions

ASS TO:            A = - MS
                   LCA (8), P            " d18 of name list[n] = 1 ?
                   GOTO (MC[-1])         " 3 instructions

DYNAMIC:           SUBC (:ASS TO)        " assigned to ?
                N, A 'X' 64, Z           " ∨ code bits > 63 ?
                   GOTO (MC[-1])         " 3 instructions

IN LIBR:           A = - MS[-1]
                   LCA (1), P            " d25 of name list[n + 1] = 1 ?
                   GOTO (MC[-1])         " 3 instructions

OP LIKE:           A = - MS[-1]
                   LCA (3), P            " d23 of name list[n + 1] = 1 ?
                   GOTO (MC[-1])         " 3 instructions

OUT DEC:           SUBC (:OP LIKE)
                   GOTO (:IN LIBR[1])    " 2 instructions
```

```
ASS TO FD:      A = - MS[-1]
                LCA (5), P          " d21 of name list[n + 1] = 1 ?
                GOTO (MC[-1])       " 3 instructions

DECLARED:       A = - MS[-1]
                LCA (7), P          " d19 of name list[n + 1] = 1 ?
                GOTO (MC[-1])       " 3 instructions

SUP LOC:        A = - MS[-1]
                GOTO (:ASS TO[1])   " 2 instructions

LOC POS:           SUBC (:ASS TO FD)  " ass to function designator ?
                N, A = d21
                N, MS[-1] + A       " change (d21)
                   S -   2
                   GOTOR (MC[-1])   " 5 instructions

SET IN DEC:        A = - MS[-1]
                   A '+' d22
                   MS[-1] = - A     " change (d22)
                N, LCA (5), P       " bool ∨ ass to function designator ?
                Y, GOTOR (MC[-1])
                   A = 390
                   GOTO (:ERRORM)   " 7 instructions

MRK POS:           SUBC (:DECLARED) " declared ?
                Y, A = 391
                Y, GOTO (:ERRORM)
                   MC =  S          " n
                   SUBC (:PR ADR)   " program address
                   A +   0, Z       " = 0 ?
                N, SUBC (:SUBSTT)
                   S = MC[-1]       " n
                   A = d19
                   MS[-1] + A       " change (d19)
                   GOTOR (MC[-1])   " 11 instructions

PR ADR:            SUBC (:NONFRM LAB) " code bits = 6 ?
                   A = MS[-1]
                Y, S -   1          " m
                   LCA (7), P       " ¬ declared ?
                   A =  MS          " name list[m]
                N, JUMP (9)
                   MC =  S          " m
                   MC =  A          " word
                   SUBC (:ORD CNT)
                   A = MC[-1]       " word
                   A 'X' - 32767    " head
                   A +   S
                   S = MC[-1]       " m
                   MS =  A          " name list[m] = head + order counter
                   A = M[B+1]       " word
                   A 'X' 32767
```

```
                    GOTOR (MC[-1])      " 17 instructions

ADDRSS:             A = MS
                    RUA (19)            " code bits
                 U, A - 13, P
                 U, A - 24, E           " ≤ 13 ∨ ≥ 25 ?
                 N, GOTO (:PR ADR)
                    SUBC (:DYNAMIC)     " dynamic ?
                    A = MS
                    A 'X' 32767
                 N, GOTOR (MC[-1])
                 U, A = in sw dec, Z
                 Y, GOTOR (MC[-1])
                    SUBC (:PRC LVL)
                    LUS (9)
                    S '+' A
                    S 'X' 32256, Z      " level = proc level ?
                 Y, A 'X' 511
                 Y, A + 32256
                    GOTOR (MC[-1])      " 18 instructions

LST LTH:            A = MS[-1]
                    A 'X' 32767
                    A - 1
                    GOTOR (MC[-1])      " 4 instructions

TEST FC:            A = MS[-1]
                    RUA (20)
                    A - for cnt, P      " name list[n + 1] : d20 > for count ?
                    A = 392
TEST RET:        N, GOTOR (MC[-1])
                    GOTO (:ERRORM)      " 6 instructions

CHCK DIM:           A = 393
                    G = dimension
CHCK RET:           MC = A              " error number
                    SUBC (:FORMAL)      " code bits
                    A - 14, Z           " non formal switch ?
                 Y, A = 1
                 N, SUBC (:LST LTH)
                 U, A + 1, Z            " - 1 ?
                 N, A - G, Z            " correct ?
                    A = MC[-1]          " error number
                 Y, GOTOR (MC[-1])
                    GOTO (:ERRORM)      " 12 instructions

CHCK LL:            A = MA[-1]
                    A 'X' 32767, Z      " list length (f) = - 1 ?
                 Y, GOTOR (MC[-1])
                    F = :MA[-1]
                    A = 394
                    GOTO (:CHCK RET)    " 6 instructions
```

```
CHCK TP:        A = MA
                RUA (19)
                A 'X' 7                 " type bits (f)
                A + :CHCK TP[8]
                DO (MA)                 " type erroneous ?
              Y, SUBC (:CHARCTR)        " ∧ character ≠ 24 ?
                A = 395
                GOTO (:TEST RET)
CHCK TP[8]:     SUBC (:NON AR)
                SUBC (:NON AR)
                SUBC (:NON BL)
                SUBC (:NON STR)
                SUBC (:NON AR)
                SUBC (:NO TYPE)
                SUBC (:NON DES)
                GOTOR (MC[-1])          " 16 instructions

NBR LL:         S = MD[-3]              " name list[block cell pointer + 3]
                S 'X' 8191
                GOTOR (MC[-1])          " 3 instructions

NEXT LL:        S +   0, Z              " n = 0 ?
              N, GOTO (:NXT IDF)
                S = MD[-3]
                GOTO (:STATUS[1])       " 4 instructions

NXT F I:        SUBC (:FORMAL)          " formal ?
              N, SUBC (:IN VA LI)       " ∨ in value list ?
              N, SUBC (:IN LIBR)        " ∨ in library ?
              Y, JUMP (2)
                 SUBC (:FUNCTN)         " function ?
                 S -  8
              Y, S -  1
                 GOTO (:NXT IDF)        " 8 instructions

INCR STS:       MD[-2] - A
                GOTOR (MC[-1])          " 2 instructions

IDF:            SUBC (:RD IDF)
                GOTO (:LOOK UP)         " 2 instructions
```

```
SKP PR LI:      S = last symbol
              U, S - 98, Z          " last symbol = open ?
              N, JUMP (5)
                 SUBC (:NXT SBL)
                 SUBC (:SKP TP DEC)
                 S = last symbol
              U, S - 99, Z          " last symbol = close ?
              Y, SUBC (:NXT SBL)
              U, S - 91, Z          " last symbol = semicolon ?
              Y, GOTO (:NXT SBL)
                 GOTOR (MC[-1])     " 11 instructions

TRL CD:         'BEGIN' TRL CD0, TRL CD1, TRL CD2, TRL CD3

                 S = last symbol
              U, S - 102, Z         " last symbol = quote ?
                 A = 401
              N, GOTO (:TRL CD3)
                 code body = A      " code body = true
TRL CD0:         SUBC (:NXT SBL)
                 S = digit last symbol, Z
              N, A = 399
              N, SUBC (:ERRORM)
              N, GOTO (:TRL CD2)
                 SUBC (:UNS INT)
                 A = value of constant[1]
              U, A - 511, P         " macro ≥ 512 ?
              N, A + :MCR LST
              N, A = MA
                 mcr = A            " macro
                 SUBC (:PAR PART)
                 S - 0, P           " par part > 0 ?
              N, GOTO (:TRL CD1)
                 S = last symbol
              U, S - 87, Z          " last symbol = comma ?
              Y, SUBC (:NXT SBL)
              N, A = 396
              N, SUBC (:ERRORM)
                 S = letter last symbol, Z
              Y, SUBC (:IDF)
              Y, GOTO (:TRL CD1)
                 S = digit last symbol, Z
              Y, SUBC (:UNS INT)
              Y, S = value of constant[1]
              Y, GOTO (:TRL CD1)
                 S = last symbol
              U, S - 65, Z          " last symbol = minus ?
              N, A = 398
              N, JUMP (3)
                 SUBC (:NXT SBL)
                 S = digit last symbol, Z
              N, A = 397
              N, SUBC (:ERRORM)
              Y, SUBC (:UNS INT)
              Y, S = - value of constant[1]
TRL CD1:         A = mcr
                 SUBC (:MACRO2[3])
TRL CD2:         S = last symbol
```

```
                U, S - 87, Z            " last symbol = comma ?
                Y, GOTO (:TRL CD0)
                U, S - 103, Z           " last symbol = unquote ?
                Y, SUBC (:NXT SBL)
                   A = 400
                   code body = - A      " code body = false
TRL CD3:        N, SUBC (:ERRORM)
                   SUBC (:ENTR BLK)
                   GOTO (:EXIT BLK)     " 53 instructions

                   'END' TRL CD


UNS NUM:           SUBC (:UNS NBR)
                   S = small, Z
                Y, GOTOR (MC[-1])
                   S = begin of pr ar
UNS NUM[4]:     U, S - dp0, Z
                Y, JUMP (6)
                   F = value of constant
                   F - MS, Z
                N, S + 2
                N, GOTO (:UNS NUM[4])
                U, S = real number, Z
                N, S + 1
                   adrs of cst = S
                   GOTOR (MC[-1])       " 14 instructions


AR CST:            S = small, Z
                Y, A = 87
                Y, S = value of constant[1]
                Y, GOTO (:MACRO2)       " MACRO2 (TSIC, value of constant)
                   S = real number, Z
                Y, A = 85               " MACRO2 (TRC, address of constant)
                N, A = 86               " MACRO2 (TIC, address of constant)
                   S = adrs of cst
                   GOTO (:MACRO2)       " 9 instructions


CST STR:           'BEGIN' CST STR0, CST STR1, CST STR2, CST STR3

                   S = 1
                   quote counter = S    " quote counter = 1
CST STR0:          F = 3                " word = 0, count = 3
CST STR1:          MC = F
                   SUBC (:NXT SBL)
                U, S - 103, Z           " last symbol = unquote ?
                   S = MC[-1]           " count
                   A = MC[-1]           " word
CST STR2:          LUA (8)              " × 256
                Y, GOTO (:CST STR3)
                   A + last symbol      " + last symbol
                   S - 1, Z             " count = count - 1
                N, F = A
                N, GOTO (:CST STR1)
                   S = A
                   A = 128
                   SUBC (:MACRO2)       " MACRO2 (CODE, word)
```

```
                GOTO (:CST STR0)
CST STR3:       A + 255
                S -   1, P              " count = count - 1
              Y, GOTO (:CST STR2)
                quote counter = - S     " quote counter = 0
                S =   A
                A = 128
                SUBC (:MACR02)          " MACR02 (CODE, word)
                GOTO (:NXT SBL)         " 26 instructions

                'END' CST STR
```

```
" main program of translator

TRANSL:         S = 300
                SUBC (:INIT)
                state = G               " state = 0
                st cnt = G              " B = 0
                max dpth = G            " max depth = 0
                max di = G              " max depth isr = 0
                max dl = G              " max display level = 0
                max prl = G             " max proc level = 0
                Ecnt = G                " Ecount = 0
                in sw dec = S           " in switch declaration = false
                code body = - S         " code body = false
                last lnc = - S          " make last lnc = line counter
                A = instr cntr
                begin of program = A    " begin of program = instruct counter
                A = begin of nli        " next block cell pointer =
                nxt bcp = A             "                 = begin of name list
                A = nlp
                last nlp = A            " last nlp = nlp
                SUBC (:ENTR BLK)
                SUBC (:NXT SBL)
                SUBC (:PROGRAM)
                S = max dpth
                S + max di
                S + max dl
                S + max prl
                A = 128                 " MACRO2 (CODE, sum of maxima)
                GOTO (:MACRO2)          " 27 instructions
```

MCR LST:

```
STACK:      + 12583168 ;
            +  4210978 ;
            +  4215106 ;
            +  4219136 ;
            +  4227426 ;

            +  4235906 ;
            +  4244386 ;
            + 10555904 ;
            +  6369792 ;
            +  6386176 ;

            + 10596608 ;
            + 23192064 ;
            + 23208448 ;
            +  8540416 ;
            +  8548608 ;

            + 21139712 ;
            + 21147904 ;
            + 21156096 ;
            +  8581376 ;
            +  8589568 ;

            +  8597760 ;
            +  8605952 ;
            +  8614144 ;
            +  8622336 ;
            +  8630528 ;


            +  8638720 ;
            +  8643072 ;
            + 10752256 ;
            + 12857600 ;
            + 12865792 ;

            +  8679680 ;
            + 21270784 ;
            + 21278976 ;
            + 21287168 ;
            + 21295360 ;

            +  8720640 ;
            +  8728832 ;
            +  8737024 ;
            +  8741376 ;
            +  8761600 ;

            +  8769792 ;
            +  8777984 ;
            +  8787217 ;
            +  8787257 ;
            +  8963328 ;

NEG:        +  8401168 ;
            +  4207154 ;
            +  4195154 ;
            +  4223232 ;
            +  4227698 ;

            +  4244114 ;
            +  4256434 ;
            +  8425728 ;
            +  6377984 ;
            +  6394368 ;

            + 23183872 ;
            + 23200256 ;
            + 23216384 ;
            +  8544512 ;
            + 21135616 ;

            + 21143808 ;
            + 21152000 ;
            + 25354496 ;
            +  8585472 ;
            +  8593664 ;

            +  8601856 ;
            +  8610048 ;
            + 27492608 ;
            +  8626432 ;
            +  8634624 ;


            +  4227328 ;
            + 12845312 ;
            + 10756352 ;
            + 12861696 ;
            + 17064192 ;

            +  8683776 ;
            + 21274880 ;
            + 21283072 ;
            + 21291264 ;
            + 21299456 ;

            +  8724736 ;
            +  8732928 ;
EMPTY:      +  8388608 ;
            +  8749824 ;
            +  8765696 ;

            +  8773888 ;
            +  8783105 ;
            +  8783145 ;
            +  8791369 ;
TBC:        +  8967432 ;
```

```
MCR LST[90]:

                + 8971520 ;         TLV:    + 8975616 ;
                + 8971520 ;                 + 8975616 ;
                + 8983808 ;                 + 8990464 ;
                + 8996096 ;                 + 9001472 ;
STST:           + 8791552 ;                 + 9008384 ;


                + 9012480 ;                 + 9016576 ;
JU:             + 9020676 ;                 + 8975872 ;
                + 9024768 ;                 + 9028864 ;
                + 9032972 ;                 + 9037068 ;
SUBJ:           + 9041152 ;                 + 9045248 ;

                + 9049352 ;                 + 9053448 ;
                + 9058824 ;                 + 9065992 ;
                + 30046984 ;                + 9065736 ;
                + 9086216 ;                 + 32159240 ;
NIL:            + 9098496 ;         LAST:   + 9102592 ;

                + 9106952 ;                 + 9106696 ;
                + 8791304 ;         TAA:    + 8975616 ;
SWP:            + 9114888 ;                 + 9118984 ;
                + 9119240 ;         EXITSV: + 28001800 ;
CODE:           + 9098504 ;                 + 9134336 ;

                + 9138432 ;                 + 9142528 ;
                + 9146624 ;                 +  762112 ;
                +  766208 ;                 +  770304 ;
                + 9163008 ;                 + 9167104 ;
                + 9171200 ;                 + 9175296 ;

                + 9179392 ;                 + 9183488 ;
                + 9187584 ;                 + 9191680 ;
                + 9195776 ;                 + 9202176 ;
                + 9209344 ;                 + 9217544 ;
```

```
" table of optimized macro's

TABEL:

      + 8799488 ;
      + 8799496 ;
      + 8807688 ;
      + 8815872 ;
      + 8815880 ;

      + 8803584 ;
      + 8803592 ;
      + 8811776 ;
      + 8811784 ;
      + 8819976 ;

      + 8824064 ;
      + 8824072 ;
      + 8832264 ;
      + 8840448 ;
      + 8840456 ;

      + 8828160 ;
      + 8828168 ;
      + 8836352 ;
      + 8836360 ;
      + 8844552 ;

      + 8848640 ;
      + 8848648 ;
      + 8856840 ;
      + 8869120 ;
      + 8869128 ;

      + 8852736 ;
      + 8852744 ;
      + 8860928 ;
      + 8860936 ;
      + 8877320 ;

      + 8861184 ;
      + 8861192 ;
      + 8877576 ;
      + 8898304 ;
      + 8898312 ;

      + 8869376 ;
      + 8869384 ;
      + 8886016 ;
      + 8886024 ;
      + 8910600 ;

      + 8922624 ;
      + 8922632 ;
      + 8939016 ;
      + 8955392 ;
      + 8955144 ;

      + 8930816 ;
      + 8930824 ;
      + 8947200 ;
      + 8946952 ;
      + 8910088 ;

      + 8885760 ;
      + 8885768 ;
      + 8910344 ;

      + 8898048 ;
      + 8898056 ;

'END'

'END'

'END'

'END'

'END'
```

```
" THIS SECTION PUNCHES THE SYSTEM AND COMPILER TAPES AT THE END
" OF ASSEMBLY. SINCE THIS PROGRAM IS ASSEMBLED IN THE OUTPUT
" BUFFER AREA, IT IS OVERWRITTEN BY THE FIRST ALGOL PROGRAM.

                'BEGIN' STIAR, STPAR, STISS, SPBUF, SPAR, READBI, SKPBLK,
                        SNXTWD, SCW, IWORD, PWORD, SREHEP, SREWRD, GEOP,
                        SIPINT, INTINS, KEYINS, CWP1, SIPCW1, SIPCW2,
                        SPHBLK, SPBLOK, SPWORD, SPBLNK, SPUHEP, ENDSYS

                " ASSUMED TO BE DECLARED GLOBALLY IS:  SYSTAP
                " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
                "       REST, PUST, ISS, IAR, PSS, PAR, TSS, TAR, PRESENCE,
                "       DENTST, BEGIN OF STACK, END OF STACK,
                "       BEGIN OF PERMANENT, END OF PERMANENT,
                "       BEGIN OF COMPILER, END OF COMPILER,
                "       IOINTP, FINIS

M[64+(4x:REST)]:        STIAR:
M[64+(4x:PUST)]:        STPAR:
M[0]:                   STISS:
PBASE1[1]:
SPBUF:          'SKIP' 1
SPAR:           'SKIP' 1

READBI:         ('660 071 000' + :REST) " CLEAR INTERRUPT FROM IP-READER
                S=:STISS[1]
                STIAR[0]=S              " INITIALIZE INPUT APPARATUS REGISTER
                S=SCW
                STISS[2]=S              " CODE WORD TO READ ONE HEPTAD
SKPBLK:           SUB(:SREHEP)
                Y,JUMP(-2)              " SKIP BLANK TAPE
                  SUB1(:SREWRD)         " READ WORD FROM TAPE IN A
                  B=A,Z                 " BEGIN ADDRESS IN B. ALL DONE?
                Y,PRESENCE=A            " COMPILER IS NOW PRESENT
                Y,A=IWORD
                Y,STIAR[0]=A            " INITIALIZE POINTER TO INPUT SS
                Y,A=PWORD
                Y,STPAR[0]=A
                Y,GOTO(:DENTST)         " START PICO
                  SUB1(:SREWRD)         " WORD FROM TAPE
                  M[6]=A                " NUMBER OF WORDS TO BE READ
SNXTWD:           SUB1(:SREWRD)
                  MC=A                  " STORE WORD
                  REP6P(:SNXTWD)        " GET NEXT WORD
                  GOTO(:SKPBLK)         " NEXT BLOCK OF WORDS
SCW:            ('001 000 000' + :STISS[0])
IWORD:          'SKIP' 1
PWORD:          'SKIP' 1

SREHEP:         G=-0
                STIAR[1]=G              " INTERRUPT COUNTER = -1
                G = SCW
                STIAR[2]=G              " ACTION COUNTER = 1
                ('760 070 000' + :REST) " START TAPE READER
                  G=STIAR[1],P          " IS THE HEPTAD PRESENT YET?
                N,JUMP(-2)              " WAIT FOR IT
                  G=STISS[1],Z          " HEPTAD IN G. IS IT BLANK?
                  GOTO(LINK)            " EXIT SREHEP
```

```
SREWRD:         S=4
                COUNT=S                 " FOUR HEPTADS PER WORD
                SUB(:SREHEP)            " READ HEPTAD IN G
                LUSA(7)                 " MAKE ROOM FOR NEW HEPTAD
                A'+'G                   " ADD IT IN
                REPP(:SREWRD[2])        " GET NEXT HEPTAD
                M[0]=A                  " SET LAST PARITY INDICATOR
                A = M[0]                " BY READING THE WORD FROM MEMORY
                CLP                     " COPY LAST PARITY INTO CONDITION
              Y,S'+'1
                S'X'1,Z                 " PARITY ERROR?
              N,GOTOR(LINK[1])          " EXIT SREWRD
ENDRBI:         S=MS[-2]                " ERROR STOP

" CONTROL IS TRANSFERRED TO SYSTAP AT THE END OF ASSEMBLY
SYSTAP:         S=INTINS
                M[24]=S                 " I/O INTERRUPT ENTRY
                S=KEYINS
                M[28]=S                 " OPERATOR'S INTERRUPT ENTRY
                S=:PSS[1]
                PAR[0]=S                " POINTER TO OUTPUT START CHAIN
                S=:ISS[1]
                IAR[0]=S                " POINTER TO INPUT START CHAIN
                S=STIAR[0]
                IWORD=S                 " SAVE STIAR[0]
                S=STPAR[0]
                PWORD=S                 " SAVE STPIAR[0]
                S=:PSS[1]
                STPAR[0]=S              " POINTER DURING SYSTEM PUNCHING
                S=CWP1                  " CODE WORD TO PUNCH ONE WORD
                PSS[2]=S                " INSERT IN CHAIN

                S=1                     " PARITY BIT FOR IP SECTION
                SPAR=S                  " MUST ALWAYS BE A 1
                SUB1(:SPBLNK)           " 100 BLANKS
                SUB1(:SPBLNK)           " ANOTHER 100
                A=SIPCW1                " ONE WORD INTO LOCATION 24
                SUB1(:SPWORD)           " PUNCH THE CODE WORD
                A=SIPINT                " THE ENTRY FOR M[24]
                SUB1(:SPWORD)           " PUNCH IT
                A=SIPCW2                " CODE WORD TO LOAD BINARY LOADER
                SUB1(:SPWORD)           " PUNCH IT
                B=:READBI               " BEGIN ADDRESS OF BINARY READER
                G=:ENDRBI               " END ADDRESS OF BINARY LOADER
                SUB2(:SPBLOK)           " PUNCH BLOCK BETWEEN TWO ADDRESSES
                SUB1(:SPBLNK)           " 100 BLANKS
                A=0
                SPAR=A                  " PUNCH WITH PARITY FROM NOW ON
                SUB1(:SPWORD)           " PUNCH END MARKER OF IP SECTION
```

```
                B=:M[24]
                G=:M[28]
                SUB2(:SPHBLK)           " PUNCH INTERRUPT ENTRY INSTRUCTIONS
                B=:M[63]
                G=:STIAR[-1]
                SUB2(:SPHBLK)           " PUNCH SOME APPARATUS REGISTERS
                B=:STIAR[4]
                G=:M[64+(16x4)]
                SUB2(:SPHBLK)           " PUNCH REMAINING APPARATUS REGISTERS
                B=:BEGIN OF STACK
                G=:END OF STACK
                SUB2(:SPHBLK)           " PUNCH INTERFACE CONSTANTS
                B=:BEGIN OF PERMANENT
GEOP:           G=:END OF PERMANENT
                SUB2(:SPHBLK)           " PUNCH MONITOR, COMPLEX, AND LIBRARY
                B=:BEGIN OF COMPILER
                G=:END OF COMPILER
                SUB2(:SPHBLK)           " PUNCH COMPILER
                B=:READBI
                G=:ENDSYS
                SUB2(:SPHBLK)           " PUNCH THIS PUNCHER PROGRAM
                SUB1(:SPBLNK)           " 100 BLANKS
                S=127
                SUB(:SPUHEP)            " PUNCH BEGIN MARKER
                A=0
                SUB1(:SPWORD)           " END MARKER OF SYSTEM TAPE
                SUB1(:SPBLNK)           " 100 BLANKS
                SUB1(:SPBLNK)           " ANOTHER 100

                SUB1(:SPBLNK)           " 100 BLANKS FOR BEGIN COMPILER TAPE
                SUB1(:SPBLNK)           " ANOTHER 100
                S=127                   " ERASE
                SUB(:SPUHEP)            " START MARKER FOR COMPILER TAPE
                B=:BEGIN OF COMPILER
                G=:END OF COMPILER
                SUB2(:SPBLOK)           " PUNCH COMPILER
                SUB1(:SPBLNK)           " 100 BLANKS
                SUB1(:SPBLNK)           " ANOTHER 100
                S=PWORD
                STPAR[0]=S              " RESTORE FOR NORMAL SERVICE
                S=1
                PRESENCE=S              " NOTE COMPILER PRESENT
                GOTO(:DENTST)           " START PICO

SIPINT:         GOTO(:READBI)
INTINS:         SUB15(:IOINTP)
KEYINS:         GOTO(:FINIS)
CWP1:           ('001 000 000'+:SPBUF[-1])
SIPCW1:         ('001 000 000'+:M[23])
SIPCW2:         (((:ENDRBI-:READBI[-1])x'001 000 000')+:READBI[-1])
```

```
SPHBLK:     SUB1(:SPBLNK)           " 100 BLANKS
            S=127
            SUB(:SPUHEP)            " PUNCH BEGIN MARKER
            A=B
            SUB1(:SPWORD)           " PUNCH BEGIN ADDRESS
            A=G
            A-B
            A+1
            SUB1(:SPWORD)           " PUNCH NUMBER OF WORDS
SPBLOK:     G-B
            COUNT=G                 " NUMBER OF WORDS -1
            A=M[B]                  " WORD TO BE PUNCHED
            SUB1(:SPWORD)           " PUNCH IT
            B+1                     " INCREMENT
            REPE(:SPBLOK[2])        " GET NEXT WORD
            GOTOR(LINK[2])          " EXIT SPHBLK AND SPBLOK

SPWORD:     M[0]=A                  " SET LAST PARITY INDICATOR
            A=M[0]                  " BY READING THE WORD FROM MEMORY
            CLP                     " COPY LAST PARITY INTO CONDITION
          N,S=1
          Y,S=SPAR                  " PARITY BIT IN S
            LCSA(6)                 " FIRST SEVEN BITS IN S
            SUB(:SPUHEP)            " PUNCH THEM
            LCSA(7)                 " SECOND SEVEN BITS
            SUB(:SPUHEP)
            LCSA(7)                 " THIRD SEVEN BITS
            SUB(:SPUHEP)
            LCSA(7)                 " LAST SEVEN BITS
            SUB(:SPUHEP)
            GOTOR(LINK[1])          " EXIT SPWORD

SPBLNK:     S=100
            COUNT=S                 " 100 TIMES
            S=0                     " BLANK
            SUB(:SPUHEP)            " PUNCH IT
            REPP(:SPBLNK[2])        " RETURN FOR NEXT ONE
            GOTOR(LINK[1])          " EXIT SPBLNK

SPUHEP:     SPBUF=S                 " HEPTAD INTO BUFFER
            S=-0
            STPAR[1]=S              " INTERRUPT COUNTER = -1
            S=CWP1                  " D18 + ...
            STPAR[2]=S              " ACTION COUNTER = 1
            ('760 070 000'+:PUST)   " START PUNCH
            S=STPAR[1],P            " PUNCHING COMPLETED?
          N,JUMP(-2)                " WAIT FOR IT
ENDSYS:     GOTOR(LINK)             " EXIT SPUHEP

            'END'                   " PUNCHING OF SYSTEM TAPES

            'END'                   " ALGOL SYSTEM

            'START' :SYSTAP
```

APPENDIX

```
" THIS SECTION REPLACES THE LINE-PRINTER SECTION FOR
" MACHINES WHICH DO NOT INCLUDE ANY LINE PRINTERS.

                'BEGIN'      HERE

              " ASSUMED TO BE DECLARED GLOBALLY ARE: PRINT, FIXT,
              "       ABSFIXT, FLOT, PRNTIS, LINENUMBER, INITPR1,
              "       INITPR2, INITPR3, NEWPAGE, NLCR, CARRIAGE,
              "       TAB, SPACE, NXTTAPESL, DERRORM, ERRORM
              " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE: PUNCH,
              "       FIXP, ABSFIXP, FLOP, FLEXIS, FLEXHP, RND,
              "       REHEP, INTREP, CASE, RUNNUMBER, DANGEROUS,
              "       ERRONEOUS, CTYPE, SMES, FIXT60, AFXP, WANTED,
              "       LINE COUNTER, LAST SYMBOL, VALUE OF CONSTANT,
              "       ENDRUN, LAST IDENTIFIER

HERE:
PUNCH[0]:     PRINT:
FIXP[0]:      FIXT:
ABSFIXP[0]:   ABSFIXT:
FLOP[0]:      FLOT:
FLEXIS[0]:    PRNTIS:
HERE[0]:

LINENUMBER:      +1                  " LINENUMBER IS PERMANENTLY 1

NEWPAGE:
NLCR:
CARRIAGE:        S = 26
                 GOTO(:FLEXHP)

TAB:             S = 62
                 GOTO(:FLEXHP)

SPACE:           SUBC(:RND)          " ROUND PARAMETER
                 COUNT = G, P        " >0 ?
                 S = 16              " SPACE
              Y, SUBC(:FLEXHP)       " BUFFER IT OUT
              Y, REPP(:SPACE[2])     " COUNT DOWN
INITPR1:
INITPR2:
INITPR3:         GOTOR(MC[-1])
```

```
          'BEGIN'      ELOOP0, ELOOP1, SMC

        " ASSUMED TO BE DECLARED GLOBALLY ARE: DERRORM, ERRORM
        " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE: DANGEROUS,
        "       ERRONEOUS, CTYPE, SMES, FIXT60, FLEXHP, AFXP, WANTED,
        "       LINE COUNTER, LAST SYMBOL, VALUE OF CONSTANT, PUNCH,
        "       ENDRUN, LAST IDENTIFIER, FLEXIS

DERRORM:    DANGEROUS = -B          "NOTE DANGEROUS ERROR
ERRORM:     MC = S                  "PRESERVE S-REGISTER
            S = 0
            ERRONEOUS = S           "ERRONEOUS = TRUE
          U, RUN NUMBER - A, P      "ERROR NUMBER < RUN NUMBER?
          Y, GOTO(:SMC)             "RESTORE S AND EXIT
            MC = A                  "SAVE ERROR NUMBER
            SUBCD(:CTYPE)           "CR NL BLKDG
              :SMES
            G = M[B - 1]
            SUBC(:FIXT60)           "TYPE ERROR NUMBER
            S = 26
            SUBC(:FLEXHP)           "PUNCH NLCR
            S = 122
            SUBC(:FLEXHP)           "PUNCH l.c.
            S = 117
            SUBC(:FLEXHP)           "PUNCH e
            S = 73
            SUBC(:FLEXHP)           "PUNCH r
            A = 3
            G = MC[-1]
            SUBC(:AFXP)             "PUNCH ERROR NUMBER IN 3 DIGITS
            A = RUN NUMBER
            A - 500, Z              "EXECUTION
          Y, A= -WANTED, P          " ∧ ¬WANTED?
            G = LINE COUNTER
            A = 6
          N, SUBC(:AFXP)            "PUNCH LINE COUNTER IN 6 DIGITS
            A = RUNNUMBER
            A - 500, Z              "EXECUTION?
            G = LAST SYMBOL
            A = 6
          N, SUBC(:AFXP)            "PUNCH LAST SYMBOL IN 6 DIGITS
            F = VALUE OF CONSTANT
            SUBC(:PUNCH)            "PUNCH LAST CONSTANT
          Y, GOTO(:ENDRUN)          "KICK PROGRAM OFF MACHINE
            S = 2
            M[6] = S                "INITIALIZE FOR TWO WORDS IDENT
            S = LASTIDENTIFIER      "FIRST FOUR CHARACTERS
ELOOP0:     A = 4
            COUNT = A               "FOUR CHARACTERS PER WORD
            LCS(2)                  "REMOVE IRRELEVANT BITS
ELOOP1:     LUAS(6)                 "6 BIT CHAR IN A, REMDR IN S
            A 'x' 63, Z             "ISOLATE CHAR. BLANK?
          N, A = :MA[-1]            "CORRECT CODE
          N, SUBC(:FLEXIS)          "PUNCH CHARACTER
            REPP(:ELOOP1)           "OTHER CHARS IN WORD
```

```
                    S = LASTIDENTIFIER[1], Z  " SECOND FOUR CHARACTERS
                  N, REP6P(:ELOOP0)          " RETURN WITH SECOND WORD
SMC:                S = MC[-1]               " RESTORE S-REGISTER
                    GOTOR(MC[-1])            " EXIT ERRORM

                    'END'                    " ERROR MESSAGE

NXTTAPESL:          'BEGIN'  DECIDE, PARITY, NLCR, TAB, UL, LC, UC, NONFLX

                    " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
                    "       REHEP, INTREP, CASE, ERRORM, RUNNUMBER

                    SUBC(:REHEP)             " GET HEPTAD FROM BUFFER
                    S'X'127                  " REMOVE EXTRANEOUS BITS
                    S+:INTREP                " ADD BASE ADDRESS OF CONVERSION TABLE
                    S=MS,P                   " CORRESPONDING TABLE ENTRY IN S
                  N, JUMP(4)                 " IF NEGATIVE, SPECIAL HANDLING
DECIDE:           U, S=CASE,Z                " ARE WE IN LOWER CASE?
                  N, RUS(8)                  " SHIFT FOR UPPER CASE BITS
                    S'X'255                  " MASK OUT EXTRANEOUS BITS
                    GOTOR(MC[-1])            " EXIT NXTTAPESL
                    LUS(10)                  " CLEAN AWAY PUTEXT BITS
                    RUS(10)                  " SHIFT BACK
                    S+:PARITY                " BASE ADDRESS OF JUMP TABLE
                    GOTO(S)                  " SWITCH TO HANDLE SPECIAL CASES
                    GOTO(:NLCR)              " -7
                    GOTO(:TAB)               " -6
                    GOTO(:UL)                " -5
                    GOTO(:LC)                " -4
                    GOTO(:UC)                " -3
                    GOTO(:NXTTAPESL)         " -2 SKIP ERASE BLANK STOPCODE BACKSPACE
                    GOTO(:NONFLX)            " -1
PARITY:             A=102                    " -0  ERROR NUMBER 102
                    S=RUNNUMBER
                    S-500,Z                  " DURING PROGRAM EXECUTION?
                  Y, A+413                   " THEN INCREMENT ERROR NUMBERS BY 413
                    SUBC(:ERRORM)            " REPORT THE ERROR
                    GOTO(:NXTTAPESL)         " AND SEE IF NEXT SYMBOL IS ANY BETTER
NLCR:               S=119                    " CARRIAGE RETURN
                    GOTOR(MC[-1])            " EXIT NXTTAPESL
TAB:                S=118
                    GOTOR(MC[-1])            " EXIT NXTTAPESL
UL:                 S=32638                  " UNDERLINE OR VERTICAL BAR
                    GOTO(:DECIDE)            " DEPENDING ON CURRENT CASE
LC:                 S=0                      " LOWER CASE
UC:                 CASE=S                   " STORE CASE, 0 = LOWER CASE
                    GOTO(:NXTTAPESL)         " GET NEXT SYMBOL
NONFLX:             A=103                    " ERROR  103. NON-FLEXOWRITER SYMBOL
                    GOTO(:PARITY[1])

                    'END' NXTTAPESL

                    'END'                    " PSEUDO LINE-PRINTER SECTION

                    'END'                    " PICO, ETC.
```

LABEL LIST

WS1	47
XEEN	17
XEENL	68
XEENP	74
XEENS	17
XEENW	3
XGET	17
XMES	4
XPONT	26
XX	3
YOKMES	33
ZERO	25