Cmd Zgrass Check Test!

STR EF EVAL ES EA CPL accurate more or less 1/17

```
 .INSERT A:S.ASM
@.REMARK /
@                       *
@                     * * *
@                       ***
@                     *******
@                         *
@                        *
@                       *
@                      *
@                     *
@                     *******
@
@      "WHEN YOU CARE ENOUGH TO PROGRAM
@                 THE VERY BEST"
@
@              ZGRASS V3.00000000
@BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
@                   (C) 1978
@
@/
 .INSERT A:ZRAM.ASM
 .INSERT A:MAC.ASM
@
 .LINK
 .IDENT  STR
 .PREL
 ;+
 ;  INTERNALS
 ;-
 .INTERN ALSTR              ;ALLOCATE STRING
 .INTERN ALSTS              ;ALLOCATE KNOWN SIZE
 .INTERN BALLY              ;PARSES BALANCED PARENS
 .INTERN CAT                ;CONCATENATION
 .INTERN ENDSTR             ;FOR ZLOAD ONLY
 .INTERN GETSTR             ;PARSE STRING LITERAL
 .INTERN INCUSE             ;BUMP USE COUNT
 .INTERN IND                ;INDIRECTION
 .INTERN SDEL               ;DELETE STRING
 .INTERN SFREE              ;FREE STRING
 .INTERN STRCMP             ;ASCIZ STRING COMPARE
 .INTERN SCMP               ;STRING LOGICAL COMPARE
 .INTERN TSTUSE             ;TEST USE COUNT
 ;+
 ;  EXTERNALS
 ;-
 .EXTERN ALLOC              ;ALLOCATE CHUNK OF MEMORY
 .EXTERN ALLOCD             ;FREE FRAGMENT
 .EXTERN CLEARIT            ;CLEAR BYTES
 .EXTERN CNTCK              ;CHECK FOR CONTROL C
 .EXTERN ERRPGM             ;ERROR TRAP
 .EXTERN EVLCPL             ;EVALUATE COMPILED CODE
 .EXTERN FREE               ;FREE MEMORY BLOCK
 .EXTERN GETOPND            ;FETCH OPERAND
 .EXTERN GETVAR             ;PARSE VARIABLE NAME
```

```
                        .EXTERN INPTTY          ;READ FROM TTY
                        .EXTERN OUTCH           ;OUTPUT A CHAR
                        .EXTERN POPOPND         ;POP EVAL OPERAND
                        .EXTERN PUTOPND         ;PUT EVAL OPERAND
                        .EXTERN TEMPCHN         ;CHAIN INTO TEMP LIST
                        .EXTERN TEMPUNC         ;CHAIN OUT OF TEMP LIST
                        ;++++
                        ;
                        ; GETSTR
                        ; PARSES AND BUILDS STRING LITERALS
                        ; STRINGS MAY BE ENCLOSED IN SINGLE OR DOUBLE
                        ; QUOTES (WHICH MAY NOT BE NESTED) OR ANGLE
                        ; OR CURLY BRACKETS WHICH MAY BE NESTED AS LONG
                        ; AS THEY ARE PAIRED
                        ;
                        ; NEEDS:
                        ;   HL -> OPEN DELIMITER
                        ;
                        ; RETURNS:
                        ;   SUCCEED:
                        ;       HL -> AFTER STRING LITERAL (CLOSE DELIM)
                        ;       DE -> TOP OF STRING HEADER WITH LITERAL
                        ;       A = TYPE STRING ($STRADR)
                        ;       CARRY CLEAR
                        ;   FAIL:
                        ;       CARRY SET
                        ;       FAILS IF 1ST CHAR AFTER ASSIGNMENT OPERATOR
                        ;         IS NOT A VALID OPEN DELIMITER
                        ;
                        ; ERRORS:
                        ;   ER.COR - NOT ENOUGH MEMORY FOR STRING
                        ;
                        ;----
0001'                   GETSTR:
0001'  FDE5                     PUSH    Y               ;SAVE REGS
0003'  C5                       PUSH    B               ;SAVE LENGTH
0004'  E5                       PUSH    H               ;SAVE LINE PTR
                        ;+
                        ; HERE WE SEARCH FOR THE OPEN DELIMITER OF THE
                        ; STRING IN THE TABLE (QTAB). IF NOT FOUND, WE FAIL
                        ;-
0005'  7E                       MOV     A,M             ;A=OPEN DELIM
0006'  21 0097'                 LXI     H,QTAB          ;HL->DELIM TABLE
0009'  01 0004                  LXI     B,QSIZ          ;BC=SIZE OF TABLE
000C'  EDB1                     CCIR                    ;SEARCH FOR DELIM
000E'  2806                     JRZ     ..S1            ;NOT THERE?
0010'  37                       STC                     ;FAIL
0011'  E1               ..EX:   POP     H               ;RESTORE REGS
0012'  C1                       POP     B
0013'  FDE1                     POP     Y
0015'  C9                       RET
                        ;+
                        ; BC = OFFSET (0-4) OF DELIMITER WHICH MATCHED
                        ; IT IS CONVERTED INTO A WORD OFFSET
                        ; (0-8) AND USED AS INDEX INTO TABLE OF CORRES-
```

```
                        ; PONDING SUBROUTINES (QSUB) WHICH HANDLE THE
                        ; PARTICULAR DELIMITER
                        ;-
0016'                   ..S1:   MVD     H,B             ;HL=OFFSET[
0016'   60              +       MOV     H,B
0017'   69              +       MOV     H+1,B           +1]
0018'   29                      DAD     H               ;WORD OFFSET
0019'   01 009C'                LXI     B,QSUB          ;BC->SUB TABLE
001C'   09                      DAD     B               ;HL->SUBROUTINE
001D'   4E                      MOV     C,M             ;BC->WHERE TO BRANCH
001E'   23                      INX     H               ;IN THE SUBTAB
001F'   46                      MOV     B,M
                                MVD     Y,B             ;IY->SUBROUTINE[
0020'   C5              +.IFIDN [Y] [Y], [      PUSH    B
0021'   FDE1          +       POP     Y
                        +]
                        ;+
                        ; IY -> SUBROUTINE TO GO TO
                        ; SP -> PTR TO OPEN DELIM
                        ; MEMORY IS ALLOCATED FOR THE STRING AND THE
                        ; POINTER TO THE TOP OF THE BLOCK IS SAVED
                        ; CALL THE PARSING ROUTINE FOR THE TYPE
                        ; OF DELIMITER WE HAVE. THE ENDING NULL IS
                        ; SAVED AND THE UNUSED FRAGMENT RETURNED
                        ;-
0023'   CD 0132'                CALL    ALSTR           ;HL->MEMORY BLOCK
0026'   E3                      XTHL                    ;SAVE PTR,HL->DLIM
0027'   23                      INX     H               ;PT AFTER DELIM
                                JSR     Y               ;PARSE STRING[
0028'   FDE5          +.IFIDN [Y                  ] [Y], [        PUSH    Y
002A'   FD21 0032'      +       LXI     Y,..0001
002E'   FDE3            +       XTIY
0030'   FDE9            +       PCIY
                        +]
0032'   23                      INX     H               ;HL->AFTER CLOSE
0033'   E3                      XTHL                    ;HL->BLOCK
0034'   CD 0039'                CALL    ENDSTR          ;RETURN FRAGMENT
0037'   18D8                    JMPR    ..EX            ;AND EXIT
                        ;++++
                        ;
                        ; ENDSTR
                        ; CLEANS UP AFTER A NEWLY ALLOCATED STRING
                        ; THIS INCLUDES RETURNING THE UNUSED FRAGMENT AND
                        ; SAVING THE PROPER LENGTH OF THE STRING
                        ; IF THE STRING IS NULL, IT IS DELETED
                        ;
                        ; NEEDS:
                        ;   HL -> TOP OF STRING HEADER
                        ;   DE -> AFTER LAST BYTE USED IN THE STRING
                        ;   BC = # OF BYTES LEFT IN THE BLOCK
                        ;
                        ; RETURNS:
                        ;   DE -> TOP OF STRING CREATED
                        ;       0 IF NULL STRING
                        ;   A = TYPE STRING ($STRADR)
```

```
                        ;
                        ; CALLS:
                        ;   ALLOCD - RETURNS UNUSED FRAGMENT
                        ;   FREE - FREE IF NULL
                        ;
                        ;----
0039'                   ENDSTR:
0039'  E5                       PUSH    H               ;SAVE PTR
003A'  D5                       PUSH    D
003B'  11 000A                  LXI     D,$SASCII       ;DE=OFFSET OF DATA
003E'  19                       DAD     D               ;IF NULL, HL=DE
003F'  D1                       POP     D               ;DE->1ST DATA BYTE
                                CLC                     ;CLEAR FOR DSBC[
0040'  B7               +       ORA     A]
0041'  ED52                     DSBC    D               ;HL=HL-DE
0043'  E1                       POP     H               ;HL->TOP OF STRING
0044'  280B                     JRZ     ..NULL          ;IS NULL?
                                CLR     A               ;NULL AT END[
0046'  AF               +.IFN A         -A, [   MVI     A                       ,0][
                        XRA     A]]
0047'  12                       STAX    D               ;OF STRING
0048'  13                       INX     D               ;FOR ALLOCD
0049'  CD 0000:05               CALL    ALLOCD          ;RETURN FRAGMENT
004C'  EB                       XCHG                    ;DE->TOP OF STRING
                        ;+
                        ; BC = NO OF BYTES LEFT IN BLOCK
                        ; DE -> TOP OF STRING
                        ; $SLEN(DE) = ORIGINAL LENGTH OF STRING
                        ; WE MUST COMPUTE THE ACTUAL NUMBER OF CHARACTERS
                        ; USED IN THE STRING: $SLEN(DE) - BC
                        ; AND SAVE THIS IN THE LENGTH FIELD
                        ;-
                        ;       PUSH    D               ;SAVE TOP PTR
                        ;       LXI     H,$SLEN         ;HL=OFFSET OF LENGTH
                        ;       DAD     D               ;HL->LENGTH FIELD
                        ;       MOV     E,M             ;DE=OLD LENGTH
                        ;       INX     H
                        ;       MOV     D,M
                        ;       XCHG                    ;HL=OLD LENGTH
                        ;       DSBC    B               ;HL=ACTUAL LENGTH
                        ;       XCHG                    ;DE=LENGTH
                        ;       MOV     M,D             ;SAVE LENGTH
                        ;       DCX     H
                        ;       MOV     M,E
                        ;       POP     D               ;RESTORE TOP PTR
004D'  3E08             ..EX:   MVI     A,$STRADR       ;RETURN TYPE STRING
                                CLC                     ;SUCCEED[
004F'  B7               +       ORA     A]
0050'  C9                       RET
                        ;+
                        ; COME HERE WHEN STRING IS NULL
                        ; RETURN DE=0
                        ;-
0051'  11 0000          ..NULL: LXI     D,0             ;ZERO RETURN VAL
0054'  CD 0000:12               CALL    TEMPUNC
```

```
0057'   CD 0000:0A             CALL    FREE            ;FREE THE STRING
005A'   18F1                    JMPR    ..EX
                        ;++++
                        ;
                        ; CAT
                        ; CONCATENATES TWO STRING OPERANDS FROM EVALUATOR
                        ; OPERAND STACK, PUTS RESULT ON STACK
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B (STRING)
                        ;        OPERAND A (STRING)
                        ;
                        ; RETURNS:
                        ;   IY->RESULT (OPERAND A & OPERAND B)
                        ;
                        ; CALLS:
                        ;   ALSTR - ALLOCATE STRING
                        ;   ENDSTR - RETURN FRAGMENT & CLEAN UP
                        ;   CAT - INTERNAL CAT ROUTINE
                        ;   POPOPND,GETOPND,PUTOPND - GET & SAVE OPERANDS
                        ;
                        ;----
005C'   08             CAT:    .BYTE   $STRADR
005D'   CD 0000:0F              CALL    POPOPND         ;GET OPERAND B
0060'   D5                      PUSH    D               ;SAVE OPND B
0061'   CD 0132'               CALL    ALSTR           ;ALLOCATE MEMORY
0064'   E5                      PUSH    H               ;SAVE TOP PTR
0065'   D5                      PUSH    D               ;SAVE DATA PTR
0066'   CD 0000:0B              CALL    GETOPND         ;GET OPERAND A
0069'   E1                      POP     H               ;HL->DATA AREA
006A'   EB                      XCHG                    ;HL->FROM,DE->TO
006B'   CD 007A'                CALL    CATS            ;COPY OVER A
006E'   E1                      POP     H               ;HL->TOP OF MEMORY
006F'   E3                      XTHL                    ;HL->OPND B
0070'   CD 007A'                CALL    CATS            ;COPY OVER B
0073'   E1                      POP     H               ;HL->TOP OF MEMORY
0074'   CD 0039'                CALL    ENDSTR          ;CLEAN UP
0077'   C3 0000:10              JMP     PUTOPND         ;SAVE NEW STRING
                        ;++++
                        ;
                        ; CATS
                        ; INTERNAL CAT ROUTINE, COPIES OVER A STRING FROM
                        ; A PREVIOUSLY CREATED STRING TO A NEW BLOCK
                        ; (ATTEMPTS TO FREE OLD STRING)
                        ;
                        ; NEEDS:
                        ;   HL -> TOP OF NEW STRING TO COPY
                        ;   DE -> NEXT FREE DATA BYTE IN NEW BLOCK
                        ;   BC = # OF BYTES LEFT IN BLOCK
                        ;
                        ; RETURNS:
                        ;   DE -> AFTER LAST BYTE COPIED (IN NEW
                        ;        ALLOCATED BLOCK)
                        ;   BC = # OF BYTES LEFT
                        ;   HL DESTROYED
```

```
                        ;
                        ; ERRORS:
                        ;   ER.COR - NOT ENOUGH MEMORY FOR STRING
                        ;
                        ; CALLS:
                        ;   SFREE - FREE COPIED STRING
                        ;
                        ;----
007A'                   CATS:
                                TSTD    H               ;NULL STRING?[
007A'   7C              +       MOV     A,H
007B'   B5              +       ORA     H               +1]
007C'   C8                      RZ                      ;RETURN THEN
007D'   E5                      PUSH    H               ;SAVE STRING PTR
007E'   D5                      PUSH    D               ;SAVE MEM PTR
007F'   11 000A                 LXI     D,$SASCII       ;GET 1ST DATA
0082'   19                      DAD     D               ;ADDR IN HL
0083'   D1                      POP     D               ;DE -> NEW BLOCK
0084'                   ..S1:   TST     M               ;SOURCE BYTE NULL?[
0084'   7E              +.IFN M         -A, [   MOV     A,M             ]
0085'   B7              +       ORA     A]
0086'   2807                    JRZ     ..EX            ;EXIT THEN
0088'   EDA0                    LDI                     ;COPY OVER ONE
008A'   E2 0093'                JPO     CORERR          ;OUT OF MEMORY!
008D'   18F5                    JMPR    ..S1            ;TRY NEXT ONE
008F'   E1              ..EX:   POP     H               ;HL -> TOP
0090'   C3 0191'                JMP     SFREE           ;TRY TO FREE
0093'                   CORERR: ERROR   ER.COR[
0093'   CD 0000:08     +       CALL ERRPGM
0096'   1B              +       .BYTE ER.COR
                        +]
                        ;++++
                        ;
                        ; QTAB - QSUB
                        ; TABLE OF ALLOWABLE OPEN DELIMITERS FOR STRINGS
                        ; AND THE CORRESPONDING SUBROUTINES TO HANDLE THEM
                        ;
                        ;----
0097'   5B7B222700      QTAB:   .ASCIZ  /[{"'/
0004                    QSIZ    = . - QTAB - 1
009C'   00A4'           QSUB:   .WORD   ..SING  ;SINGLE QUOTE
009E'   00B4'                   .WORD   ..DOUB  ;DOUBLE QUOTE
00A0'   00C4'                   .WORD   ..CURB  ;CURLY BRACKET
00A2'   00D9'                   .WORD   ..SQRB  ;SQUARE BRACKET
                        ;++++
                        ;
                        ; ..SING ..DOUB ..SQRB ..CURB
                        ; ROUTINES TO PARSE STRINGS & COPY FOR
                        ; DIFFERENT SETS OF DELIMITERS
                        ;
                        ; NEEDS:
                        ;   HL -> AFTER OPEN DELIMITER IN INPUT STRING
                        ;   DE -> WHERE TO START PUTTING DATA IN BLOCK
                        ;   BC = SIZE OF BLOCK (BYTES)
                        ;
```

```
                        ; RETURNS:
                        ;   HL -> AT ENDING DELIMITER
                        ;   DE -> AFTER LAST CHAR IN STRING
                        ;   BC = NO OF BYTES LEFT
                        ;
                        ; CALLS:
                        ;   GETLIN - GET NEW LINE OF INPUT
                        ;
                        ; ERRORS:
                        ;   ER.COR - OUT OF MEMORY
                        ;
                        ;----
   00A4'                 ..SING: TST     M                       ;END OF LINE?[
LL 00A4'   7E           +.IFN M         -A?, ?[ MOV     A,M                     ]
   00A5'   B7           +               ORA     A]
   00A6'   CC 0107'             CZ      GETLIN                  ;GET NEW LINE
   00A9'   7E                   MOV     A,M                     ;A=CURRENT CHAR
   00AA'   FE27                 CPI     "'"                     ;SINGLE QUOTE?
   00AC'   C8                   RZ                              ;YES? RETURN
   00AD'   EDA0                 LDI                             ;XFER A CHAR
   00AF'   E2 0093'             JPO     CORERR                  ;OUT OF CORE?
   00B2'   18F0                 JMPR    ..SING
   00B4'                 ..DOUB: TST     M                       ;END OF LINE?[
   00B4'   7E           +.IFN M         -A, [   MOV     A,M                     ]
   00B5'   B7           +               ORA     A]
   00B6'   CC 0107'             CZ      GETLIN                  ;GET NEW LINE
   00B9'   7E                   MOV     A,M                     ;A=CURRENT CHAR
   00BA'   FE22                 CPI     '"'                     ;DOUBLE QUOTE?
   00BC'   C8                   RZ                              ;YES? RETURN
   00BD'   EDA0                 LDI                             ;XFER A CHAR
   00BF'   E2 0093'             JPO     CORERR                  ;OUT OF CORE?
   00C2'   18F0                 JMPR    ..DOUB
   00C4'                 ..CURB: TST     M                       ;NO MORE?[
   00C4'   7E           +.IFN M         -A, [   MOV     A,M                     ]
   00C5'   B7           +               ORA     A]
   00C6'   CC 0107'             CZ      GETLIN                  ;GET NEW LINE
   00C9'   7E                   MOV     A,M                     ;A=CURRENT CHAR
   00CA'   FE7D                 CPI     '}'                     ;CLOSE BRACKET?
   00CC'   C8                   RZ                              ;YES? RETURN
   00CD'   FE7B                 CPI     '{'                     ;OPEN BRACKET?
   00CF'   CC 00D2'             CZ      ..CURI                  ;DOWN A LEVEL
   00D2'   EDA0          ..CURI: LDI                             ;XFER A CHAR
   00D4'   E2 0093'             JPO     CORERR                  ;OUT OF CORE?
   00D7'   18EB                 JMPR    ..CURB
   00D9'                 ..SQRB: TST     M                       ;NO MORE?[
   00D9'   7E           +.IFN M         -A, [   MOV     A,M                     ]
   00DA'   B7           +               ORA     A]
   00DB'   CC 0107'             CZ      GETLIN                  ;GET NEW LINE
   00DE'   7E                   MOV     A,M                     ;A=CURRENT CHAR
   00DF'   FE5D                 CPI     ']'                     ;CLOSE BRACKET
   00E1'   C8                   RZ                              ;YES? RETURN
   00E2'   FE5B                 CPI     '['                     ;OPEN BRACKET?
   00E4'   CC 00E7'             CZ      ..SQRI                  ;DOWN A LEVEL
   00E7'   EDA0          ..SQRI: LDI                             ;XFER A CHAR
   00E9'   E2 0093'             JPO     CORERR                  ;OUT OF CORE?
```

```
00EC'  18EB                    JMPR    ..SORB
                              ;++++
                              ;
                              ; BALLY
                              ; PARSES OVER BALANCED DELIMITERS
                              ;
                              ; NEEDS:
                              ;   HL -> OPEN DELIM
                              ;   D = OPEN DELIM, E = CLOSE DELIM
                              ;
                              ; RETURNS:
                              ;   HL -> AFTER LAST CLOSE DELIM
                              ;
                              ; ERRORS:
                              ;   ER.PAR - UNBALANCED PARENS
                              ;
                              ;----
00EE'                          BALLY:  TST     M               ;NULL?[
00EE'  7E                     +.IFN M          -A, [  MOV     A,M             ]
00EF'  B7                     +        ORA     A]
                                       ZERROR  ER.PAR[
00F0'  2004                   +        JRNZ ..0002
00F2'  CD 0000:08             +        CALL ERRPGM
00F5'  2A                     +        .BYTE ER.PAR
00F6'                         +..0002:]
00F6'  FE0A                            CPI     NL              ;NEWLINE?
                                       ZERROR  ER.PAR[
00F8'  2004                   +        JRNZ ..0003
00FA'  CD 0000:08             +        CALL ERRPGM
00FD'  2A                     +        .BYTE ER.PAR
00FE'                         +..0003:]
00FE'  23                              INX     H               ;HL->NEXT CHAR
00FF'  BB                              CMP     E               ;CLOSE?
0100'  C8                              RZ                      ;RETURN
0101'  BA                              CMP     D               ;OPEN?
0102'  CC 00EE'                       CZ      BALLY           ;PARSE AGAIN
0105'  18E7                            JMPR    BALLY
                              ;++++
                              ;
                              ; GETLIN
                              ; READ A LINE OF INPUT OFF TTY
                              ; STORES NEWLINE AT END OF OLD LINE
                              ;
                              ; NEEDS:
                              ;   DE -> NEXT AVAILABLE SPOT FOR A CHAR
                              ;   BC = NO OF BYTES LEFT FOR CHARS
                              ;
                              ; RETURNS:
                              ;   HL -> 1ST CHAR OF NEW LINE READ
                              ;
                              ; CALLS:
                              ;   OUTCH - OUTPUT PROMPT
                              ;   INPTTY - INPUT CHAR
                              ;
                              ;----
```

```
0107'   3E0A          GETLIN: MVI     A,NL            ;OUTPUT NEWLINE
0109'   12                    STAX    D               ;SAVE IN BLOCK
010A'   13                    INX     D               ;DE->NEXT SPOT
010B'   0B                    DCX     B               ;BUMP # BYTES
010C'   78                    MOV     A,B             ;IF BC = 0
010D'   B1                    ORA     C               ;NO MORE CORE
010E'   CA 0093'              JZ      CORERR
0111'   3E2B                  MVI     A,'+'           ;PROMPT WITH +
0113'   CD 0000:0E            CALL    OUTCH
0116'   21 660C                LXI     H,KBLOCK+$MIBEND ;HL->BUFFER COUNT
0119'   3A 6260                LDA CSFLAG
011C'   B7                    ORA A                   ;TEST IT
011D'   2803                  JRZ ..OK                ;IS NOT ~S
011F'   21 66A8                LXI H,CSBLOK+$MIBEND    ;PROPER BUFFER
0122'   3600          ..OK:   MVI     M,0             ;CLEAR BUFFER COUNT
0124'   CD 0000:07     ..G1:   CALL    CNTCK           ;CHECK FOR CONTROL C
0127'   CD 0000:0D            CALL    INPTTY          ;READ IN CHARS
012A'   20F8                  JRNZ    ..G1            ;UNTIL NULL
012C'   23                    INX     H               ;HL->1ST CHAR
                              TST     M               ;NO MORE?[
012D'   7E            +.IFN M         -A, [   MOV     A,M             ]
012E'   B7            +       ORA     A]
012F'   C0                    RNZ                     ;RETURN IF SO
0130'   18D5                  JMPR    GETLIN          ;ELSE GET NUTHER LINE
                    ;++++
                    ;
                    ; ALSTR
                    ; ALLOCATES MEMORY FOR A STRING
                    ;
                    ; RETURNS:
                    ;   HL -> TOP OF STRING BLOCK
                    ;   DE -> 1ST AVAILABLE CHAR FOR DATA
                    ;   BC = NO OF DATA BYTES IN BLOCK
                    ;
                    ; HL--->**********
                    ;       * TYPE   * $STRADR
                    ;       **********
                    ;       * SIZE   * SET BY ALLOC
                    ;       **********
                    ;       * FLAGS * CLEARED
                    ;       **********
                    ;       * TEMP   * LINKED INTO
                    ;       * PTR    * TEMP LIST
                    ;       **********
                    ;       * USE    * SET TO 0
                    ;       * COUNT * (ONE BYTE ONLY)
                    ;       **********
                    ;       * STRING* SET TO MAX
                    ;       * LENGTH* POSSIBLE
                    ;       **********
                    ;       * NULL   *
                    ;       **********
                    ; DE--->* DATA   *
                    ;       *        *
                    ;       **********
```

```
                        ;
                        ; CALLS:
                        ;   ALLOC - ALLOCATE A BLOCK
                        ;   CLEARIT - CLEAR BYTES
                        ;   TEMPCHN - CHAIN INTO TEMP LIST
                        ;
                        ;----
0132'                   ALSTR:
0132'  3EFF                     MVI     A,-1            ;UNKNOWN SIZE
0134'  CD 0000:04       ALSTS:  CALL    ALLOC           ;ALLOCATION
0137'  E5                       PUSH    H               ;HL->TOP OF BLOCK
0138'  E5                       PUSH    H               ;TWICE
0139'  21 FFF5                  LXI     H,-$SASCII-1    ;HL=-OFFSET OF DATA
013C'  09                       DAD     B               ;HL=MAX # CHARS
013D'  E3                       XTHL                    ;SAVE IT
013E'  3608                     MVI     M,$STRADR       ;SET TYPE
0140'  23                       INX     H
0141'  23                       INX     H               ;SKIP SIZE
0142'  01 0008                  LXI     B,8             ;CLEAR FLAGS-NULL
0145'  CD 0000:06               CALL    CLEARIT         ;(8 BYTES)
0148'  C1                       POP     B               ;GET SIZE BACK
                        ;       LXI     D,$SASCII-2     ;DE=OFFSET OF DATA
                        ;       DAD     D               ;HL->1ST DATA BYTE
                        ;       PUSH    H               ;SAVE IT
                        ;       DCX     H               ;HL->ZERO BYTE
                        ;       DCX     H               ;HL->HI-ORDER LENGTH
                        ;       MOV     M,B             ;STORE LENGTH
                        ;       DCX     H               ;HL->LO-ORDER LENGTH
                        ;       MOV     M,C
                        ;       POP     D               ;DE -> DATA
0149'  11 0008                  LXI     D,$SASCII-2     ;DE=OFFSET OF DATA
014C'  19                       DAD     D               ;HL -> DATA
014D'  EB                       XCHG                    ;DE -> DATA
014E'  E1                       POP     H               ;RESTORE TOP PTR
014F'  C3 0000:11               JMP     TEMPCHN         ;INTO TEMP LIST
                        ;++++
                        ;
                        ; TSTUSE
                        ; TESTS THE USE COUNT (OF A STRING)
                        ;
                        ; NEEDS:
                        ;   HL -> TOP OF BLOCK (HEADER)
                        ;
                        ; RETURNS:
                        ;   REGS UNCHANGED
                        ;   Z BIT SET IN CC'S IF USE COUNT ($USE) IS 0
                        ;   C BIT SET IF STRING IS NULL (HL=0)
                        ;
                        ;----
0152'                   TSTUSE:
                                TSTD    H               ;IF NULL, SET CARRY[
0152'  7C               +       MOV     A,H
0153'  B5               +       ORA     H               +1]
0154'  37                       STC                     ;AND RETURN
0155'  C8                       RZ
```

```
0156'   E5                      PUSH    H
0157'   D5                      PUSH    D
0158'   11 0005                LXI     D,$USE          ;OFFSET OF USE COUNT
015B'   19                      DAD     D               ;HL->USE COUNT
                                TST     M               ;SET CC'S[
015C'   7E              +.IFN M         -A, [   MOV     A,M             ]
015D'   B7              +       ORA     A]
015E'   D1                      POP     D
015F'   E1                      POP     H
0160'   C9                      RET
                        ;++++
                        ;
                        ; INCUSE
                        ; INCREMENTS THE USE COUNT OF AN ALLOCATED BLOCK
                        ;
                        ; NEEDS:
                        ;   DE -> HEADER OF BLOCK
                        ;       $USE FIELD BUMPED
                        ;
                        ; CALLS:
                        ;       TEMPUNC - UNCHAIN FROM TEMP LIST
                        ;
                        ;----
0161'                   INCUSE: TSTD    D               ;IF DE=0, NO CHECK[
0161'   7A              +       MOV     A,D
0162'   B3              +       ORA     D               +1]
0163'   C8                      RZ
0164'   E5                      PUSH    H
0165'   21 0005                LXI     H,$USE          ;DE=OFFSET OF USE COUNT
0168'   19                      DAD     D               ;HL->USE FIELD
                                TST     M               ;CHECK IT[
0169'   7E              +.IFN M         -A, [   MOV     A,M             ]
016A'   B7              +       ORA     A]
016B'   EB                      XCHG                    ;HL->TOP OF BLOCK
016C'   CC 0000:12             CZ      TEMPUNC         ;OUT OF TEMP LIST
016F'   EB                      XCHG
0170'   34                      INR     M               ;BUMP USE COUNT
0171'   F2 0179'                JP      ..OUT           ;>0? OK TO EXIT
0174'   35                      DCR     M               ;RESTORE IT
                                ERROR   ER.USE          ;BAD USE COUNT[
0175'   CD 0000:08      +       CALL ERRPGM
0178'   42              +       .BYTE ER.USE
                        +]
0179'   E1                      ..OUT:  POP     H
017A'   C9                      RET
                        ;++++
                        ;
                        ; SFREE, SDEL
                        ; DELETE A STRING IF ITS USE COUNT IS 0
                        ; SDEL WILL DECREMENT THE USE COUNT BEFORE FREEING
                        ; WORKS FOR NULL STRINGS
                        ;
                        ; NEEDS:
                        ;   HL -> TOP OF STRING
                        ;
```

```
                        ; RETURNS:
                        ;   REGS UNCHANGED
                        ;   STRING GARBAGE COLLECTED IF USE COUNT IS 0
                        ;
                        ; CALLS:
                        ;   TSTUSE - CHECK USE COUNT
                        ;   TEMPUNC - CHAIN OUT OF TEMP LIST
                        ;   FREE - FREE THE STRING
                        ;
                        ;----
017B'                   SDEL:   TSTD    H               ;NULL?[
017B'   7C              +       MOV     A,H
017C'   B5              +       ORA     H               +1]
017D'   C8                      RZ                      ;RETURN THEN
017E'   E5                      PUSH    H
017F'   D5                      PUSH    D
0180'   11 0005                LXI     D,$USE          ;USE COUNT OFFSET
0183'   19                      DAD     D               ;HL->USE COUNT
0184'   D1                      POP     D
0185'   35                      DCR     M               ;DEC IT
0186'   E1                      POP     H
0187'   CA 0000:0A              JZ      FREE            ;FREE IF 0
018A'   F0                      RP                      ;IF >0, RETURN
018B'   CD 0000:12     SFEX:   CALL    TEMPUNC         ;OUT OF TEMPS
018E'   C3 0000:0A             JMP     FREE            ;FREE IF 0
0191'                   SFREE:
0191'   CD 0152'               CALL    TSTUSE          ;TEST USE COUNT
0194'   D8                      RC                      ;DONT FREE IF NULL
0195'   FA 0000:0A              JM      FREE            ;FREE IF NEG
0198'   28F1                    JRZ     SFEX            ;UNCHAIN, FREE
019A'   C9                      RET
                        ;++++
                        ;
                        ; IND
                        ; INDIRECTION OPERATOR - CONVERTS A STRING TO A NAME
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B (STRING)
                        ;       OPERAND A (0)
                        ;
                        ; RETURNS:
                        ;   IY->RESULT (NAME)
                        ;   MAKES A NAME OUT OF THE STRING OPERAND, ATTEMPTS
                        ;   TO FREE THE STRING (IF $USE IS 0)
                        ;    DESTROYS DE, HL
                        ;
                        ; CALLS:
                        ;   GETOPND, PUTOPND, POPOPND - PUSH, POP OPERANDS
                        ;   GETVAR - PARSE NAME
                        ;   SFREE - FREE STRING
                        ;
                        ; ERRORS:
                        ;   ER.NAM - STRING NOT A LEGAL NAME
                        ;
                        ;----
```

```
019B'  08              IND:    .BYTE   $STRADR
019C'  CD 0000:0F              CALL    POPOPND         ;DE->STRING HDR
019F'  D5                      PUSH    D               ;SAVE STRING
                               MVD     H,D             ;HL->STRING[
01A0'  62              +       MOV     H,D
01A1'  6B              +       MOV     H+1,D           +1]
01A2'  11 000A                 LXI     D,$SASCII       ;DE=OFFSET OF DATA
01A5'  19                      DAD     D               ;HL->1ST CHAR OF STRING
01A6'  FDE5                    PUSH    Y               ;SAVE OUR STACK PTR
01A8'  CD 0000:0C              CALL    GETVAR          ;PARSE THE NAME
                               CERROR  ER.NAM          ;NOT A NAME![
01AB'  3004            +       JRNC    ..0004
01AD'  CD 0000:08      +       CALL    ERRPGM
01B0'  29              +       .BYTE   ER.NAM
01B1'                  +..0004:]
01B1'  280D                    JRZ     ..I1            ;WAS VARIABLE
01B3'  DDE5                    PUSH    X               ;SAVE OUR X
                               MVD     X,Y             ;IX->FUNCTION CODE[
01B5'  FDE5            +.IFIDN [X] [X], [      PUSH    Y
01B7'  DDE1            +       POP     X
                       +]
01B9'  CD 0000:09              CALL    EVLCPL          ;EVALUATE FUNCTION
01BC'  DDE1                    POP     X               ;RESTORE X
01BE'  1803                    JMPR    ..I2            ;SKIP AROUND PUT
01C0'  CD 0000:10      ..I1:   CALL    PUTOPND         ;PUSH THE RESULT
01C3'  FDE1            ..I2:   POP     Y               ;GET IY PTR
01C5'  E1                      POP     H               ;HL->STRING
01C6'  C3 0191'                JMP     SFREE           ;TRY TO FREE
                       ;++++
                       ;
                       ; STRCMP
                       ; COMPARES TWO ASCIZ STRINGS, SETS CCS
                       ;
                       ; NEEDS:
                       ;   DE -> STRING2 (OPERAND B)
                       ;   HL -> STRING1 (OPERAND A)
                       ;
                       ; RETURNS:
                       ;   DESTROYS HL, DE
                       ;       Z BIT SET       STRING1 = STRING2
                       ;       N BIT SET       STRING1 < STRING2
                       ;       ELSE            STRING1 > STRING2
                       ;
                       ;----
01C9'                  STRCMP:
01C9'  1A                      LDAX    D               ;GET STRING1 CHAR
01CA'  96                      SUB     M               ;FIND DIFF
01CB'  C0                      RNZ                     ;NOT EQ? RETURN
01CC'  B6                      ORA     M               ;END OF STRING?
01CD'  C8                      RZ                      ;RETURN
01CE'  23                      INX     H               ;NEXT CHARS
01CF'  13                      INX     D
01D0'  18F7                    JMPR    STRCMP          ;GO AGAIN
                       ;++++
                       ;
```

```
                        ; SCMP
                        ; LOGICAL STRING COMPARE ROUTINE
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND A (STRING)
                        ;        OPERAND B (STRING)
                        ;
                        ; RETURNS:
                        ;   DE = 0
                        ;   BOTH STRING OPERANDS POPPED OFF AND FREED
                        ;   CC'S SET FOR COMPARISON
                        ;        Z BIT   OPERAND A = OPERAND B
                        ;        N BIT   OPERAND A < OPERAND B
                        ;        ELSE    OPERAND A > OPERAND B
                        ;   THE NULL STRING IS CONSIDERED SMALLER THAN ALL
                        ;   OTHER STRINGS
                        ;
                        ; CALLS:
                        ;   STRCMP - COMPARE ASCIZ STRINGS
                        ;   SFREE - GARBAGE COLLECT STRINGS
                        ;   POPOPND - POP OFF STRING
                        ;
                        ;----
01D2'                   SCMP:
                        ;+
                        ; FIRST WE POP OFF THE 2 STRING OPERANDS AND
                        ; CHECK IF THE POINTERS ARE EQUAL. IF BOTH STRINGS
                        ; ARE NULL OR BOTH ARE THE SAME STRING THIS WILL
                        ; OCCUR. IF THERE ARE 2 POINTERS TO THE SAME STRING,
                        ; IT CAN BE ASSUMED THIS STRING HAS A NON-ZERO USE
                        ; COUNT AND NEED NOT BE DELETED BEFORE EXIT. HENCE
                        ; IF THE PTRS ARE EQUAL WE JUST RETURN WITH Z SET
                        ;-
01D2'   CD 0000:0F              CALL    POPOPND         ;GET OPNDB
01D5'   D5                      PUSH    D               ;SAVE PTR TO B
01D6'   EB                      XCHG                    ;HL->OPND B
01D7'   CD 0000:0F              CALL    POPOPND         ;DE->OPND A
                                CLC                     ;FOR DSBC[
01DA'   B7              +       ORA     A]
01DB'   ED52                    DSBC    D               ;HL=DE?
01DD'   E1                      POP     H               ;HL->B,DE->A
01DE'   2823                    JRZ     ..SRET          ;RETURN IF EQ
                        ;+
                        ; HERE WE CHECK IF OPERAND A OR OPERAND B IS NULL
                        ; WE KNOW THAT THEY ARE NOT BOTH NULL OR WE WOULD
                        ; NOT HAVE GOTTEN THIS FAR. CONSEQUENTLY, IF A
                        ; IS NULL WE RETURN WITH N BIT SET (<)
                        ; IF B IS NULL WE RETURN WITH CCS CLEAR (>)
                        ;-
01E0'   E5                      PUSH    H               ;SAVE OPNDB
01E1'   D5                      PUSH    D               ;SAVE OPNDA
01E2'   7C                      MOV     A,H             ;CHECK IF B IS
01E3'   85                      ADD     L               ;NULL STRING
01E4'   3E01                    MVI     A,1             ;A NULL, RETURN N
01E6'   2810                    JRZ     ..OUT           ;BIT SET FOR <
```

```
01E8'  7A                    MOV     A,D             ;CHECK IF A IS
01E9'  83                    ADD     E               ;NULL STRING
01EA'  3EFF                  MVI     A,-1            ;B NULL, RETURN NO
01EC'  280A                  JRZ     ..OUT           ;CC'S SET FOR >
                        ;+
                        ; NOW WE KNOW NEITHER STRING IS NULL AND WE MUST
                        ; COMPARE THE ASCII PARTS UNTIL AN INEQUALITY OR
                        ; ENDING NULL IS FOUND
                        ; HL -> OPERAND B
                        ; DE -> OPERAND A
                        ;-
01EE'  01 000A               LXI     B,$SASCII       ;GET DATA PTR
01F1'  09                    DAD     B               ;HL->OPNDB DATA
01F2'  EB                    XCHG                    ;DE->B DATA,HL->A
01F3'  09                    DAD     B               ;HL->A DATA
01F4'  EB                    XCHG                    ;HL->B,DE->A
01F5'  CD 01C9'              CALL    STRCMP          ;COMPARE THEM
                        ;+
                        ; EXIT POINT - A HAS CCS FOR STRING COMPARE
                        ; HL,DE BOTH POINT TO OPERANDS OR ARE NULL
                        ; WE ATTEMPT TO FREE BOTH OPERANDS AND LOAD
                        ; A 0 INTO DE FOR RETURN. THE CCS ARE SET WITH
                        ; COMPARE INFO
                        ;-
01F8'  47               ..OUT:  MOV     B,A             ;SAVE CC SET
01F9'  E1                       POP     H               ;HL->OPND
01FA'  CD 0191'                 CALL    SFREE           ;KILL IT
01FD'  E1                       POP     H               ;HL->NUTHER OPND
01FE'  CD 0191'                 CALL    SFREE           ;KILL THIS ONE
                                TST     B               ;TEST RETURN CC[
0201'  78               +.IFN B         -A, [   MOV     A,B             ]
0202'  B7               +               ORA     A]
0203'  11 0000          ..SRET: LXI     D,0             ;LOAD RETURN
0206'  C9                       RET
                        .END
```

STR - STRING - STRING ASSIGNMENT MODULE
++++++ SYMBOL TABLE ++++++

AASN	005F		ALLOC	0000:04	X	ALLOCD	0000:05	X	ALSTR	0132'	I
ALSTS	0134'	I	ARGEND	6252		ARGSTK	60CC		BACKGR	65CC	
BALLY	00EE'	I	BLANK	0020		BOTRAM	6000		BOTTOM	64A9	
CAT	005C'	I	CATS	007A'		CHARSL	65DD		CLEARI	0000:06	X
CLEARS	60CA		CNTCK	0000:07	X	CNTRL	000C		CNTRLC	65CD	
CNTRLO	65D9		CNTRLU	0015		CNTRLZ	65E4		CORERR	0093'	
CPLFLG	6540		CPLIN	681C		CPLTMP	672C		CPREF	691C	
CP.ARG	0001		CP.MXL	0080		CP.MXR	0014		CP.NO	0002	
CR	000D		CSBLOK	668D		CSFLAG	6260		CURCX	6813	
CURCY	6815		CURLIN	6934		CURREN	64A5		C.CO	0011	
C.C1	0012		C.CX	000B		C.CY	000D		C.DP	0013	
C.ST	0002		C.X	0003		C.XF	000F		C.XS	0007	
C.Y	0005		C.YF	0010		C.YS	0009		DDTON	65CE	
DEVBL	6589		DEVCL0	6579		DEVCL1	657B		DEVCL2	657D	
DEVCL3	657F		DEVCL4	6581		DEVCL5	6583		DEVCL6	6585	
DEVCL7	6587		DEVFB	65CB		DEVHCB	658F		DEVMO	658B	
DEVNM	65B7		DEVNT	658D		DEVTNA	65BB		DEVTNB	65BF	
DEVTNC	65C3		DEVVA	65BD		DEVVAR	6579		DEVVB	65C1	
DEVVBL	6591		DEVVC	65C5		DEVVD	65C9		DEVVN	65B9	
DEVVS	65C7		DEVXCD	65B3		DEVYCD	65B5		DOLPLH	62E8	
DOLPPT	62EA		DUMBST	6577		EDBCNT	64AD		EDCCNT	64A7	
EDLONG	6811		EDMODE	681B		EDNAME	64A1		EDNCX	680B	

EF -

```
.INSERT A:MAC.ASM
@
```

```
 .INSERT A:S.ASM
@.REMARK /
@                       *
@                     * * *
@                      ***
@                     *******
@                        *
@                       *
@                      *
@                     *
@                    *
@                   *******
@
@      "WHEN YOU CARE ENOUGH TO PROGRAM
@                 THE VERY BEST"
@
@              ZGRASS V3.00000000
@BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
@                    (C) 1978
@
@/
 .INSERT A:NCUEQU.ASM
 .INSERT A:ZRAM.ASM
 .LINK
 .IDENT  EF
 .PREL
 ;+
 ;  INTERNALS
 ;-
 .INTERN ARCCOS             ;ARC COSINE FUNCTION
 .INTERN ARCSIN             ;ARC SINE FUNCTION
 .INTERN ARCTAN             ;ARC TANGENT FUNCTION
 .INTERN CNVFS              ;CONVERT FLOATING TO STRING
 .INTERN COSINE             ;COSINE FUNCTION
 .INTERN EXP                ;EXPONENT FUNCTION
 .INTERN FORMAT             ;FORMAT ASCII OUTPUT
 .INTERN GETNUM             ;PARSE FLOATING NUMBER
 .INTERN GETSKP             ;PARSE SKIP NUM
 .INTERN INT                ;INTEGER CONVERT
 .INTERN LN                 ;LOG NATURAL FUNCTION
 .INTERN LOG                ;LOG BASE 10 FUNCTION
 .INTERN PI                 ;RETURN VALUE OF PI
 .INTERN POWER              ;POWER FUNCTION
 .INTERN SINE               ;SINE FUNCTION
 .INTERN SQRT               ;SQUARE ROOT FUNCTION
 .INTERN TANGENT            ;TANGENT FUNCTION
 ;+
 ;  EXTERNALS
 ;-
 .EXTERN ALSTS              ;ALLOCATE STRING
 .EXTERN ARG                ;PARSE ARGUMENTS
 .EXTERN CARTYP             ;FIND CHARACTER TYPE
 .EXTERN CVDSTR             ;CONVERT DOUBLE TO STRING
```

```
                        .EXTERN CVFSTR          ;CONVERT FRACTION TO STRING
                        .EXTERN DONCU           ;DO NCU OPERATION
                        .EXTERN ENDSTR          ;CLEAN UP STRING
                        .EXTERN ERRPGM          ;ERROR TRAP
                        .EXTERN FPUSH           ;PUSH FLT NUMBER
                        .EXTERN GETMSB          ;GET MOST SIGNIF BYTE
                        .EXTERN GETOPND         ;GET OPERAND
                        .EXTERN INC5IY          ;POINT TO NEXT ARG
                        .EXTERN IPOP            ;POP INT FROM NCU STACK
                        .EXTERN IPSHO           ;PUSH 0 ONTO NCU STACK
                        .EXTERN IPUSH           ;PUSH INT ONTO NCU STACK
                        .EXTERN IYNCU           ;GET FROM IY TO NCU
                        .EXTERN LPAD            ;FOR LOG LINKAGE
                        .EXTERN NCUIY           ;GET FROM NCU TO IY
                        .EXTERN PLAY            ;FOR POWER LINK
                        .EXTERN POPOPND         ;POP OPERAND
                        .EXTERN PSHF10          ;PUSH FLOATING 10
                        .EXTERN PSHOPND         ;PUSH OPERAND
                        .EXTERN PUTOPND         ;PUT OPERAND
                        .EXTERN RETNONE         ;RETURN
                        .EXTERN SUBSTR          ;FOR SQRT LINK
                        ;++++
                        ;
                        ; CNVFS
                        ; CONVERT FLOATING POINT OPERAND TO STRING OPERAND
                        ;
                        ; NEEDS:
                        ;  IY-> FLOATING POINT OPERAND
                        ;
                        ; RETURNS:
                        ;  IY-> STRING OPERAND
                        ;
                        ; CALLS:
                        ;   ALSTS - ALLOCATE STRING
                        ;   ENDSTR - CLEAN UP AFTER STRING
                        ;   GETOPND, PUTOPND - FETCH, PUT OPERAND
                        ;   GETMSB - GET MOST SIGNIFICANT BYTE
                        ;   INTPT - CONVERT INTEGER PART
                        ;   FRACPT - CONVERT FRACTIONAL PART
                        ;   DONCU - DO NCU OPERATION
                        ;
                        ;----
0001'                   CNVFS:
0001'  C5               ;       PUSH    B               ;SAVE BC
0002'  3E01             ;       MVI     A,($ASCII+13)/16
0004'  CD 0000:04       ;       CALL    ALSTS           ;ALLOCATE STRING
0007'  E5               ;       PUSH    H               ;SAVE PTR
0008'  EB               ;       XCHG                    ;HL->WHERE TO PUT DATA
0009'  CD 0000:0E       ;       CALL    GETOPND         ;GET NUM ON NCU
000C'  CD 0042'         ;       CALL    MSGN            ;DO THE SIGN
000F'  CD 0050'         ;       CALL    NORMLIZ         ;NORMALIZE THE NUMBER
                        ;+
                        ; BC = BASE 10 EXPONENT
                        ; IF THE BASE 10 EXPONENT IS BETWEEN -3 AND 7 WE DO NOT
                        ; USE SCIENTIFIC NOTATION
```

```
                        ;-
   0012'   79           ..CA:   MOV     A,C             ;A=BASE 10 EXP
   0013'   FE07                 CPI     6+1             ;EXP<7? NORMAL
   0015'   3810                 JRC     ..C2            ;EXP<7? NORMAL
   0017'   FEFD                 CPI     -3              ;EXP<-3? SCI NO
   0019'   300C                 JRNC    ..C2
   001B'   CD 00FF'     ..C1:   CALL    SCINO           ;SCIENTIFIC NOTATION
                        ;+
                        ; EXIT, CLEAN UP AFTER STRING AND PUSH IT
                        ;-
   001E'                ..EX:   CLR     M               ;TRAILING NULL[
LL 001E'   3600         +.IFN M         -A?, ?[ MVI     M                ,0?][
                        XRA     A]]
   0020'   D1                   POP     D               ;HL->TOP OF STRING
   0021'   C1                   POP     B               ;RESTORE BC
   0022'   3E08                 MVI     A,$STRADR       ;TYPE STRING
   0024'   C3 0000:1A           JMP     PUTOPND         ;SAVE STRING
                        ;+
                        ; C = BASE 10 EXPONENT
                        ;   IF 0, .NNNNNN
                        ;   IF <0, .00NNNNNN
                        ;   ELSE, NN.NNNN
                        ;-
   0027'   CD 002C'     ..C2:   CALL    NUMIT           ;NORMAL NOTATION
   002A'   18F2                 JMPR    ..EX            ;AND EXIT
                        ;++++
                        ;
                        ; NUMIT
                        ; CONVERTS A FLOATING POINT NUMBER TO AN ASCII INTEGER
                        ; AND FRACTINAL PART
                        ;
                        ; NEEDS:
                        ;   BC = BASE 10 EXPONENT
                        ;   HL -> WHERE TO PUT FIRST DIGIT
                        ;   NCU STACK HAS NORMALIZED NUMBER
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER LAST DIGIT
                        ;   BC DESTROYED
                        ;
                        ; CALLS:
                        ;   FLTOUT - GET 6 DIGITS
                        ;   FRAC, FRACPT - FRACTION PART
                        ;   INTPT - INTEGER PART
                        ;
                        ;----
   002C'   E5           NUMIT:  PUSH    H               ;SAVE OUR PTR
   002D'   21 64A3              LXI     H,NUMBUF        ;HL->NUMBER BUFFER
   0030'   E5                   PUSH    H
   0031'   CD 00E8'             CALL    FLTOUT          ;GET 7 DIGITS
   0034'   D1                   POP     D               ;DE->NUMBER BUF
   0035'   E1                   POP     H               ;HL->STRING
                                TST     C               ;CHECK BASE 10 EXP[
LL 0036'   79           +.IFN C         -A?, ?[ MOV     A,C                ]
   0037'   B7           +               ORA     A]
```

```
0038'   CC 00C7'            CZ      FRAC            ;.NNNNN
003B'   FC 00BC'            CM      FRACPT          ;.OONNN
003E'   C4 00B3'            CNZ     INTPT           ;NNN.NNN
0041'   C9                  RET
                    ;++++
                    ;
                    ; MSGN
                    ; TAKES CARE OF THE SIGN OF THE MANTISSA OF A
                    ; FLOATING POINT NUMBER. IF IT IS POSITIVE, NOTHING
                    ; IS DONE. IF IT IS NEGATIVE, A '-' IS PUT IN THE
                    ; STRING AND THE SIGN OF THE FLOATING NUMBER ON
                    ; THE TOP OF THE NCU STACK IS CHANGED
                    ;
                    ; NEEDS:
                    ;   HL -> WHERE TO PUT SIGN IN STRING
                    ;   NCU -> FLOATING POINT NUMBER
                    ;
                    ; RETURNS:
                    ;   NCU -> POSITIVE FLOATING POINT NUMBER
                    ;   HL -> AFTER SIGN (IF ANY)
                    ;
                    ; CALLS:
                    ;   GETMSB - GET EXPONENT
                    ;   DONCU - DO NCU OPERATION
                    ;
                    ;----
0042'   CD 0000:0D  MSGN:   CALL    GETMSB          ;GET EXPONENT
0045'   E680                ANI     N.MSGN          ;CHECK MANTISSA SIGN
0047'   C8                  RZ                      ;POSITIVE? LEAVE IT
0048'   362D                MVI     M,'-'           ;SIGN IN STRING
004A'   23                  INX     H               ;AFTER THE SIGN
004B'   3E15                MVI     A,N.CHSF        ;GET ABSOLUTE VALUE
004D'   C3 0000:09          JMP     DONCU

                    ;++++
                    ;
                    ; NORMLIZ, NORMLZ
                    ; NORMALIZES A FLOATING POINT NUMBER WITH RESPECT TO
                    ; BASE 10
                    ;
                    ; NEEDS:
                    ;   NCU->NUMBER .NNNNNN * 10**M
                    ;   BC = BASE 10 EXPONENT (NORMLZ ONLY)
                    ;
                    ; RETURNS:
                    ;   NCU-> BASE 10 MANTISSA .NNNNNN
                    ;   BC = BASE 10 EXPONENT + M
                    ;
                    ;----
0050'               NORMLIZ:
0050'   01 0000            LXI     B,0             ;BASE 10 EXP = 0
0053'   3E17                MVI     A,N.PTOF        ;IF IS 0,
0055'   CD 0000:09          CALL    DONCU           ;RETURN
                            TST     A[
0058'   B7          +       ORA     A]
```

```
0059'   C8                      RZ
005A'   CD 0000:0D              CALL    GETMSB          ;GET EXPONENT
005D'   FE40                    CPI     N.ESGN          ;EXPONENT NEGATIVE?
005F'   3804                    JRC     ..N1            ;NO? LEAVE IT BE
0061'   FE56                    CPI     #N.MSGN&-42     ;IS TOO SMALL?
0063'   3814                    JRC     NORMLZ          ;DONT ROUND THEN
0065'   C6EB            ..N1:   ADI     -21             ;SHIFT TO THE RIGHT
0067'   E67F                    ANI     #N.MSGN         ;KILL THE SIGN
0069'   57                      MOV     D,A
006A'   1EE0                    MVI     E,0E0H          ;.111 * 2**-21
006C'   D5                      PUSH    D
006D'   CD 0000:11              CALL    IPSHO
0070'   D1                      POP     D
0071'   CD 0000:12              CALL    IPUSH
0074'   3E10                    MVI     A,N.FADD        ;ROUND IT UP
0076'   CD 0000:09              CALL    DONCU
                        ;+
                        ; NORMLZ - ACTUAL NORMALIZATION
                        ; THE BASE 2 EXPONENT IS CHECKED
                        ;   IF IT IS 0 - WE EXIT
                        ;   IF IT IS > 0 - WE DIVIDE BY 10 & CHECK AGAIN
                        ;-
0079'   CD 0000:0D     NORMLZ: CALL    GETMSB          ;GET EXPONENT
                                TST     A               ;IS IT 0?[
007C'   B7              +       ORA     A]
007D'   C8                      RZ                      ;RETURN IF SO
007E'   E640                    ANI     N.ESGN          ;IS IT NEGATIVE?
0080'   2826                    JRZ     ..ND            ;DIVIDE BY 10
                        ;+
                        ; BASE 2 EXPONENT IS < 0. IF THE NUMBER WE HAVE IS
                        ; LARGER THAN OR EQUAL TO .1, WE CAN EXIT. OTHERWISE
                        ; WE MUST MULTIPLY BY 10 AND CHECK AGAIN
                        ;-
0082'   3E17            ..NM:   MVI     A,N.PTOF        ;SAVE NUMBER
0084'   CD 0000:09              CALL    DONCU
0087'   E5                      PUSH    H               ;SUBTRACT .1
0088'   21 0364'                LXI     H,PT1           ;HL->.1
008B'   CD 0000:0C              CALL    FPUSH           ;ON NCU STACK
008E'   E1                      POP     H
008F'   3E11                    MVI     A,N.FSUB        ;SUBTRACT TO
0091'   CD 0000:09              CALL    DONCU           ;COMPARE THEM
                                TST     A               ;SAVE CC'S[
0094'   B7              +       ORA     A]
0095'   F5                      PUSH    PSW
0096'   3E18                    MVI     A,N.POPF        ;KILL RESULT
0098'   CD 0000:09              CALL    DONCU
009B'   F1                      POP     PSW             ;GET CC'S
009C'   F0                      RP                      ;RETURN IF >= .1
009D'   CD 0000:18              CALL    PSHF10          ;ELSE PUSH 10.
00A0'   0B                      DCX     B               ;DEC BASE 10 EXP
00A1'   3E12                    MVI     A,N.FMUL        ;AND MULTIPLY
00A3'   CD 0000:09              CALL    DONCU
00A6'   18DA                    JMPR    ..NM            ;CHECK AGAIN
                        ;+
                        ; BASE 2 EXPONENT > 0, WE DIVIDE BY 10 AND
```

```
                        ; CHECK AGAIN
                        ;-
00A8'  CD 0000:18       ..ND:   CALL    PSHF10          ;PUSH 10.
00AB'  3E13                     MVI     A,N.FDIV        ;AND DIVIDE
00AD'  CD 0000:09               CALL    DONCU
00B0'  03                       INX     B               ;BUMP BASE 10 EXP
00B1'  18C6                     JMPR    NORMLZ
                        ;++++
                        ;
                        ; INTPT
                        ; PUTS THE INTEGER PART OF A FLOATING POINT NUMBER
                        ; INTO THE GIVEN STRING
                        ;
                        ; NEEDS:
                        ;   DE -> FIRST DIGIT OF NUMBER IN BUFFER
                        ;   HL -> NEXT BYTE IN STRING TO STORE INTO
                        ;   A = BASE 10 EXPONENT (IF <=0, RETURN)
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER INTEGER PART (IF ANY)
                        ;   DE -> AFTER LAST INTEGER DIGIT
                        ;   B DESTROYED
                        ;
                        ;----
00B3'  47               INTPT:  MOV     B,A             ;B=BASE 10 EXP
00B4'  1A               ..I1:   LDAX    D               ;A=NEXT DIGIT
00B5'  13                       INX     D
00B6'  77                       MOV     M,A             ;COPY DIGIT
00B7'  23                       INX     H
00B8'  10FA                     DJNZ    ..I1            ;UNTIL ALL DONE
00BA'  180B                     JMPR    FRAC            ;DO FSACTION
                        ;++++
                        ;
                        ; FRACPT
                        ; COPIES THE REST OF THE FRACUIONAL DIGITS FROM
                        ; BUFFER INTO STRING. DECIMAL POINT ADDED, TRAILING
                        ; ZEROS SUPPRESSED
                        ;
                        ; NEEDS:
                        ;   DE -> FIRST FRACTIONAL DIGIT
                        ;   HL -> WHERE TO PUT DIGITS IN STRING
                        ;
                        ; RETURNS:
                        ;   DE -> AFTER LAST DIGIT
                        ;   HL -> NEXT BYTE IN STRING
                        ;
                        ;----
00BC'  362E             FRACPT: MVI     M,'.'           ;PUT IN DEC PT
00BE'  23                       INX     H
00BF'  3630             ..F1:   MVI     M,'0'           ;LEADING ZEROS
00C1'  23                       INX     H
00C2'  3C                       INR     A               ;BUMP EXPONENT
00C3'  20FA                     JRNZ    ..F1            ;UNTIL 0
00C5'  1803                     JMPR    FRACX           ;REST OF DIGITS
                        ;+
```

```
                        ; FRAC - ENTRY POINT WHEN LEADING ZEROS NOT DESIRED
                        ;-
00C7'  362E             FRAC:   MVI     M,'.'           ;DECIMAL POINT
00C9'  23                       INX     H
00CA'  1A               FRACX:  LDAX    D               ;GET A·DIGIT
00CB'  13                       INX     D
00CC'  77                       MOV     M,A             ;COPY IT
00CD'  23                       INX     H
                                TST     A               ;WAS IT NULL?[
00CE'  B7               +       ORA     A]
00CF'  20F9                     JRNZ    FRACX           ;NO? CONTINUE
00D1'  2B                       DCX     H               ;AT NULL
00D2'  3E30                     MVI     A,'0'
00D4'  2B                ..F2:  DCX     H               ;DEC PTR
00D5'  BE                       CMP     M               ;TRAILING 0?
00D6'  28FC                     JRZ     ..F2            ;KILL IT
00D8'  7E                       MOV     A,M             ;A=LAST DIGIT
00D9'  FE2E                     CPI     '.'             ;WAS THE DP?
00DB'  2008                     JRNZ    ..F3            ;NO? EXIT
00DD'  2B                       DCX     H               ;HL->BEFORE DP
                                TST     M               ;IS NULL?[
LL 00DE'  7E            +.IFN M         -A?, ?[ MOV     A,M                 ]
00DF'  B7               +       ORA     A]
00E0'  2003                     JRNZ    ..F3            ;NO? EXIT
00E2'  23                       INX     H               ;POINT AT DP
00E3'  3630                     MVI     M,'0'           ;MAKE IT 0
00E5'  23                ..F3:  INX     H               ;HL->LAST 0 OR NULL
                                CLR     A               ;SET Z BIT[
00E6'  AF               +.IFN A         -A, [   MVI     A                   ,0][
                         XRA    A]]
00E7'  C9                       RET
                        ;++++
                        ;
                        ; FLTOUT
                        ; CONVERT NORMALIZED FLOATING POINT NUMBER TO A
                        ; DIGIT STRING (NO DECIMAL POINT, ASSUMED TO BE
                        ; AT THE FAR LEFT)
                        ;
                        ; NEEDS:
                        ;   HL -> BUFFER TO STORE!AT LEAST 6 DIGITS & NULL
                        ;   NCU-> BASE 10 NORMALIZED FLOATING NUMBER
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER LAST DIGIT
                        ;   DESTROYS B, NCU
                        ;
                        ;----
00E8'  0605             FLTOUT: MVI     B,5             ;FIELD WIDTH
00EA'  E5                       PUSH    H               ;PUSH 10**6
00EB'  21 0368'                 LXI     H,TEN6          ;ON NCU STACK
00EE'  CD 0000:0C               CALL    FPUSH
00F1'  E1                       POP     H
00F2'  3E12                     MVI     A,N.FMUL        ;GET 7 DIGITS
00F4'  CD 0000:09               CALL    DONCU
00F7'  3E1E                     MVI     A,N.FIXD        ;CONVERT TO INT
```

```
00F9'   CD 0000:09              CALL    DONCU
00FC'   C3 0000:08              JMP     CVFSTR          ;PUT IN STRING
                        ;++++
                        ;
                        ; SCINO
                        ; PUTS A NORMALIZED FLOATING NUMBER INTO SCIENTIFIC
                        ; NOUATION IN A STRING .NNNNNNNENNN
                        ;
                        ; NEEDS:
                        ;   HL -> AFTER SIGN IN STRING
                        ;   NCU-> FLOATING MANTISSA OF NUMBER
                        ;   BC = BASE 10 EXPONENT
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER EXPONENT DIGITS
                        ;   BC, DE, NCU DESTROYED
                        ;
                        ;----
00FF'   C5              SCINO:  PUSH    B               ;SAVE BASE 10 EXP
0100'   362E                    MVI     M,'.'           ;DECIMAL POINT
0102'   23                      INX     H
0103'   CD 00E8'               CALL    FLTOUT          ;GET 7 DIGITS
0106'   3645                    MVI     M,'E'           ;EXPONENT
0108'   23                      INX     H
0109'   D1                      POP     D               ;DE = BASE 10 EXP
010A'   CD 0000:12              CALL    IPUSH           ;DOUBLE IT
                                TST     D               ;CHECK HI BYTE[
LL 010D'   7A           +.IFN D         -A?, ?[ MOV     A,D             ]
010E'   B7              +               ORA     A]
010F'   11 0000                 LXI     D,0             ;ASSUME POS
0112'   2803                    JRZ     ..S1            ;OK
0114'   11 FFFF                 LXI     D,-1            ;NEGATIVE
0117'   CD 0000:12      ..S1:   CALL    IPUSH           ;PUSH HI BYTE
011A'   C3 0000:07              JMP     CVDSTR          ;INTO STRING
                        ;++++
                        ;
                        ; GETNUM
                        ; PARSES A FLOATING POINT NUMBER FROM A STRING
                        ;
                        ; NEEDS:
                        ;   HL -> FIRST DIGIT OF STRING
                        ;
                        ; RETURNS:
                        ;   FAIL:
                        ;       C BIT SET
                        ;       FAILS IF NO DIGITS AT ALL
                        ;   SUCCEED:
                        ;       HL -> AFTER LAST DIGIT, EXPONENT, ETC.
                        ;       TOP OF NCU STACK HAS FLOATING NUMBER
                        ;       A = $FVAL (TYPE FLOAT)
                        ;   DESTROYS DE, BC
                        ;
                        ; CALLS:
                        ;   CARTYP - CHECK CHARACTER TYPE
                        ;   DONCU - DO!NCU OPERATION
```

```
                        ;    PSHF10- PUSH FLOATING 10
                        ;    GETINT, GETI - GET INUEGER FROM STRING
                        ;    GETEXP - PARSE EXPONEOT
                        ;
                        ;-----
   011D'   7E           GETNUM: MOV     A,M             ;CHECK FOR DECIMAL
   011E'   FE2E                 CPI     '.'             ;POINT FIRST
   0120'   2805                 JRZ     ..G1            ;OK TO START
   0122'   CD 0000:06           CALL    CARTYP          ;CHECK 1ST CHAR
   0125'   3065                 JRNC    FAIL            ;FAIL IF NO DIGIT
                        ;+
                        ; PARSE INTEGER PART. IF A DECIMAL POINT IS FOUND,
                        ; AN EXPONENT IS COMPUTED IN B. THIS EXPONENT IS
                        ; DECREMENTED!EVERY TIME A DIGIT AFTER THE DECIMAL
                        ; POINT IS FOUND
                        ;-
   0127'   CD 015A'     ..G1:   CALL    GETINT          ;PARSE INTEGER PART
   012A'   0600                 MVI     B,0             ;EXPONENT IS 0
   012C'   7E                   MOV     A,M             ;CHECK FOR DECIMAL
   012D'   FE2E                 CPI     '.'             ;POINT
   012F'   2004                 JRNZ    ..G2            ;ISNT ONE, TOO BAD
   0131'   23                   INX     H               ;HL->AFTER OP
   0132'   CD 0177'             CALL    GETI            ;PARSE FRACTION
                        ;+
                        ; IF AN EXPONENT IS FOUND, IT IS PARSED AND
                        ; ADDED TO THE CURRENT EXPONENT IN B
                        ;-
   0135'   CD 018E'     ..G2:   CALL    GETEXP          ;PARSE EXPONENT
   0138'   3E1C                 MVI     A,N.FLTD        ;CONVERT TO FLT
   013A'   CD 0000:09           CALL    DONCU
                        ;+
                        ; HERE B = BASE 10 EXPONENT AND THE STACK HAS
                        ; THE FLOATING NUMBER PARSED WITH THE DECIMAL
                        ; POINT TO THE FAR RIGHT. WE MULTIPLY/DIVIDE
                        ; THE NUMBER OF TIMES INDICATED BY THE EXPONENT
                        ; TO GET THE TRUE VALUE
                        ;-
   013D'                ..G3:   TST     B               ;EXPONENT 0?[
LL 013D'   78           +.IFN B         -A?, ?[ MOV     A,B                 ]
   013E'   B7           +               ORA     A]
   013F'   2816                 JRZ     ..EX            ;YES? CAN EXIT
   0141'   FA 014F'             JM      ..G5            ;NEGATIVE? DIVIDE
   0144'   CD 0000:18           CALL    PSHF10          ;MULTIPLY BY 10.0
   0147'   3E12                 MVI     A,N.FMUL
   0149'   05                   DCR     B               ;DEC EXPONENT
   014A'   CD 0000:09   ..G4:   CALL    DONCU           ;DO IT
   014D'   18EE                 JMPR    ..G3            ;GO AGAIN
   014F'   CD 0000:18   ..G5:   CALL    PSHF10          ;DIVIDE BY 10.0
   0152'   3E13                 MVI     A,N.FDIV
   0154'   04                   INR     B               ;BUMP EXPONENT
   0155'   18F3                 JMPR    ..G4
   0157'   3E06         ..EX:   MVI     A,$FVAL         ;RETURN TYPE FLT
   0159'   C9                   RET
                        ;++++
                        ;
```

```
                        ; GETINT, GETI
                        ; INTERNAL ROUTINES TO PARSE INTEGER AND MAINTAIN
                        ; A 32 BIT NUMBER ON NCU STACK. THE EXPONENT IN
                        ; B IS DECREMENTED EVERY TIME A DIGIT IS PARSED
                        ;
                        ; NEEDS:
                        ;   HL -> DIGIT
                        ;   NCU STACK HAS CURRENT NUMBER FOR GETI, GETINT
                        ;   WILL PUSH A 32 BIT 0
                        ;
                        ; RETURNS:
                        ;   FAIL:
                        ;       C BIT SET
                        ;       FAILS IF NO DIGIT FOUND
                        ;   SUCCEED:
                        ;       HL -> NON-DIGIT
                        ;       NCU STACK HAS 32 BIT INTEGER PARSED
                        ;
                        ; CALL:
                        ;   CARTYP - CHECHK CHARACTER TYPE
                        ;   DONCU - DO NCU OPERATION
                        ;   IPSH0 - PUSH INTEGER 0
                        ;   IPUSH - PUSH INTEGER
                        ;
                        ;----
   015A'   CD 0000:11   GETINT: CALL    IPSH0          ;PUSH 32 BIT ZERO
   015D'   CD 0000:11           CALL    IPSH0
   0160'   CD 0000:06           CALL    CARTYP         ;CHECK DIGIT
   0163'   3027                 JRNC    FAIL           ;FAIL RETURN
   0165'   7E           GI:     MOV     A,M            ;A=DIGIT
   0166'   23                   INX     H              ;HL->AFTER DIGIT
   0167'   D630                 SUI     '0'            ;GET IN BINARY
                                CLR     D              ;PUT IT IN DE[
LL 0169'   1600         +.IFN D          -A?, ?[ MVI    D                ,0][
                        XRA     A]]
   016B'   5F                   MOV     E,A
   016C'   CD 0000:12           CALL    IPUSH          ;PUSH ON NCU STACK
   016F'   CD 0000:11           CALL    IPSH0          ;PAD HI ORDER
   0172'   3E2C                 MVI     A,N.DADD       ;ADD TO RESULT
   0174'   CD 0000:09           CALL    DONCU
   0177'   CD 0000:06   GETI:   CALL    CARTYP         ;CHECK NEXT DIGIT
   017A'   D0                   RNC                    ;RETURN IF NOT DIG
   017B'   11 000A              LXI     D,10           ;PUSH INTEGER 10
   017E'   CD 0000:12           CALL    IPUSH          ;ON NCU STACK
   0181'   CD 0000:11           CALL    IPSH0
   0184'   3E2E                 MVI     A,N.DMUL       ;MULTIPLY OUR NUM
   0186'   CD 0000:09           CALL    DONCU          ;BY 10
   0189'   05                   DCR     B              ;DEC EXPONENT
   018A'   18D9                 JMPR    GI             ;CONTINUE
   018C'   37           FAIL:   STC                    ;SET CARRY
   018D'   C9                   RET
                        ;++++
                        ;
                        ; GETEXP
                        ; PARSE EXPONENT IN STRING AND RETURN IN B
```

```
                        ;
                        ; NEEDS:
                        ;   HL -> POSSIBLE EXPONENT IN STRING
                        ;  B = OLD EXPONENT OR 0
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER EXPONENT (IF ANY)
                        ;   B = OLD EXPONENT PLUS NEW EXPONENT
                        ;
                        ; CALLS:
                        ;   GETINT - PARSE INTEGER
                        ;   DONCU - DO NCU OPERATION
                        ;   IPOP - POP INTEGER
                        ;
                        ;----
018E'   FE45            GETEXP: CPI     'E'             ;IS EXPONENT?
0190'   C0                      RNZ                     ;NO? SKIP IT
0191'   23                      INX     H
0192'   C5              GETSKP: PUSH    B               ;SAVE OLD EXP
0193'   7E                      MOV     A,M             ;A = NEXT THING AFTER E
0194'   FE2D                    CPI     '-'             ;CHECK SIGN
0196'   F5                      PUSH    PSW             ;SAVE CCS
0197'   2001                    JRNZ    ..G1            ;NO SIGN?
0199'   23                      INX     H               ;HL->AFTER SIGN
019A'   CD 015A'       ..G1:   CALL    GETINT          ;GET EXPONENT
                                CERROR  ER.NUM          ;MUST BE NUMBER[
019D'   3004            +               JRNC ..0001
019F'   CD 0000:0B      +               CALL ERRPGM
01A2'   2B              +               .BYTE ER.NUM
01A3'                   +..0001:]
01A3'   CD 0000:10              CALL    IPOP            ;POP HI PART
01A6'   CD 0000:10              CALL    IPOP            ;DE=EXPONENT
01A9'   F1                      POP     PSW             ;WAS THERE A SIGN?
01AA'   7B                      MOV     A,E             ;A=EXPONENT PARSED
01AB'   2002                    JRNZ    ..G2            ;NO SIGN
01AD'   ED44                    NEG                     ;NEGATE IT
01AF'   C1              ..G2:   POP     B               ;B=OLD EXPONENT
01B0'   80                      ADD     B               ;ADD TO NEW ONE
01B1'   47                      MOV     B,A             ;SAVE IN B
01B2'   C9                      RET                     ;AND RETURN
                        ;++++
                        ;
                        ; POWER
                        ;
                        ;----
                                M.CMD[POWER,..POWX,0,PLAY,..PARG,..POW][
01B3'   18              +POWER: .BYTE $CMDADR
01B4'   03              +       .BYTE (..POWX-POWER)/16+1
01B5'   00              +       .BYTE 0
01B6'   0000:16         +       .WORD PLAY
01B8'   01C3'           +       .WORD ..POW
01BA'   01C6'           +       .WORD ..PARG
01BC'   504F57455200+           .ASCIZ /POWER/
                        +]
01C2'   00                      .BYTE   0
```

```
01C3'   CD 0000:05      ..POW:  CALL    ARG
01C6'   060600          ..PARG: .BYTE   $FVAL,$FVAL,$TAF
01C9'   CD 0000:13              CALL    IYNCU           ;GET 1ST ARG
01CC'   CD 0000:0F              CALL    INCSIY          ;POINT AT 2ND ONE
01CF'   CD 0000:13              CALL    IYNCU           ;AND 2ND ARG
01D2'   3E0B                    MVI     A,N.PWR         ;POWER FUNCTION
01D4'   181B                    JMPR    FRET            ;AND RETURN
01D6'                   ..POWX:
                        ;++++
                        ;
                        ; SIN
                        ;
                        ;----
                                M.CMD[SINE,SINX,0,SQRT,FNARG,..SIN1]I
01D6'   18              +SINE:  .BYTE $CMDADR
01D7'   03              +       .BYTE (SINX-SINE)/16+1
01D8'   00              +       .BYTE 0
01D9'   027E'           +       .WORD SQRT
01DB'   01E5'           +       .WORD ..SIN1
01DD'   01EB'           +       .WORD FNARG
01DF'   53494E4500      +       .ASCIZ /SINE/
                        +]
01E4'   00                      .BYTE   0
01E5'                   ..SIN1:
01E5'   3E02                    MVI     A,N.SIN         ;SINE
01E7'   F5              FFUN:   PUSH    PSW             ;SAVE OPERATION
01E8'   CD 0000:05              CALL    ARG             ;GET ARGUMENT
01EB'   0600            FNARG:  .BYTE   $FVAL,$TAF
01ED'   CD 0000:13              CALL    IYNCU           ;ONTO NCU FROM IY
01F0'   F1                      POP     PSW             ;A = FUNCTION
01F1'   CD 0000:09      FRET:   CALL    DONCU           ;RESULT ON NCU
01F4'   3E06                    MVI     A,$FVAL         ;FLOATING VALUE
01F6'   C3 0000:1B              JMP     RETNONE         ;RETURN
01F9'                   $INX:
                        ;++++
                        ;
                        ; COS
                        ;
                        ;----
                                M.CMD[COSINE,..COSX,0,AFIX[C],FNARG,..COS1]I
01F9'   18              +COSINE: .BYTE $CMDADR
01FA'   02              +       .BYTE (..COSX-COSINE)/16+1
01FB'   00              +       .BYTE 0
01FC'   630C            +       .WORD AFIX[C][("C"-101Q)*16+FSTINT]
01FE'   020A'           +       .WORD ..COS1
0200'   01EB'           +       .WORD FNARG
0202'   434F53494E45+           .ASCIZ /COSINE/
                        +]
0209'   00                      .BYTE   0
020A'   3E03            ..COS1: MVI     A,N.COS ;COSINE
020C'   18D9                    JMPR    FFUN
020E'                   ..COSX:
                        ;++++
                        ;
                        ; TAN
```

```
                        ;
                        ;----
                                M.CMD[TANGENT,..TANX,0,AFIX[T],FNARG,..TAN][
020E'   18          +TANGENT:       .BYTE $CMDADR
020F'   02          +               .BYTE (..TANX-TANGENT)/16+1
0210'   00          +               .BYTE 0
0211'   641C        +               .WORD AFIX[T][("T"-101Q)*16+FSTINT]
0213'   0220'        +               .WORD ..TAN
0215'   01EB'        +               .WORD FNARG
0217'   54414E47454E+               .ASCIZ /TANGENT/
                    +]
021F'   00                      .BYTE   0
0220'   3E04          ..TAN:  MVI     A,N.TAN         ;TANGENT
0222'   18C3                  JMPR    FFUN
0224'                   ..TANX:
                        ;++++
                        ;
                        ; INT
                        ;
                        ;----
                                M.CMD[INT,..INTX,0,AFIX[I],..INTA,..INT][
0224'   18          +INT:       .BYTE $CMDADR
0225'   02          +           .BYTE (..INTX-INT)/16+1
0226'   00          +           .BYTE 0
0227'   636C        +           .WORD AFIX[I][("I"-101Q)*16+FSTINT]
0229'   0232'        +           .WORD ..INT
022B'   0235'        +           .WORD ..INTA
022D'   494E5400    +           .ASCIZ /INT/
                    +]
0231'   00                      .BYTE   0
0232'   CD 0000:05    ..INT:  CALL    ARG
0235'   0400            ..INTA: .BYTE   $IVAL,$TAF
0237'   CD 0000:0E              CALL    GETOPND         ;GET OPERAND
023A'   C3 0000:1B              JMP     RETNONE         ;RETURN INTEGER
023D'                   ..INTX:
                        ;++++
                        ;
                        ; ASIN
                        ;
                        ;----
                                M.CMD[ARCSIN,..ACSX,0,AFIX[A],FNARG,..ASIN][
023D'   18          +ARCSIN:    .BYTE $CMDADR
023E'   02          +           .BYTE (..ACSX-ARCSIN)/16+1
023F'   00          +           .BYTE 0
0240'   62EC        +           .WORD AFIX[A][("A"-101Q)*16+FSTINT]
0242'   024E'        +           .WORD ..ASIN
0244'   01EB'        +           .WORD FNARG
0246'   41524353494E+           .ASCIZ /ARCSIN/
                    +]
024D'   00                      .BYTE   0
024E'   3E05          ..ASIN: MVI     A,N.ASIN        ;ARC SINE
0250'   1895                  JMPR    FFUN
0252'                   ..ACSX:
                        ;++++
                        ;
```

```
                        ; ACOS
                        ;
                        ;----
                                    M.CMD[ARCCOS,..ACCX,0,ARCSIN,FNARG,..ACOS][
0252'   18          +ARCCOS:  .BYTE $CMDADR
0253'   02          +         .BYTE (..ACCX-ARCCOS)/16+1
0254'   00          +         .BYTE 0
0255'   023D'       +         .WORD ARCSIN
0257'   0263'       +         .WORD ..ACOS
0259'   01EB'       +         .WORD FNARG
025B'   415243434F53+         .ASCIZ /ARCCOS/
                    +]
0262'   00                    .BYTE   0
0263'   3E06          ..ACOS: MVI     A,N.ACOS        ;ARC COSINE
0265'   C3 01E7'              JMP     FFUN
0268'                   ..ACCX:
                        ;++++
                        ;
                        ; ATAN
                        ;
                        ;----
                                    M.CMD[ARCTAN,..ATNX,0,ARCCOS,FNARG,..ATAN][
0268'   18          +ARCTAN:  .BYTE $CMDADR
0269'   02          +         .BYTE (..ATNX-ARCTAN)/16+1
026A'   00          +         .BYTE 0
026B'   0252'       +         .WORD ARCCOS
026D'   0279'       +         .WORD ..ATAN
026F'   01EB'       +         .WORD FNARG
0271'   41524354414E+         .ASCIZ /ARCTAN/
                    +]
0278'   00                    .BYTE   0
0279'   3E07          ..ATAN: MVI     A,N.ATAN        ;ARC TANGENT
027B'   C3 01E7'              JMP     FFUN
027E'                   ..ATNX:
                        ;++++
                        ;
                        ; SQRT
                        ;
                        ;----
                                    M.CMD[SQRT,..SQRX,0,SUBSTR,FNARG,..SQRT][
027E'   18          +SQRT:    .BYTE $CMDADR
027F'   02          +         .BYTE (..SQRX-SQRT)/16+1
0280'   00          +         .BYTE 0
0281'   0000:1C      +         .WORD SUBSTR
0283'   028D'       +         .WORD ..SQRT
0285'   01EB'       +         .WORD FNARG
0287'   5351525400  +         .ASCIZ /SQRT/
                    +]
028C'   00                    .BYTE   0
028D'   3E01          ..SQRT: MVI     A,N.SQRT        ;SQUARE ROOT
028F'   C3 01E7'              JMP     FFUN
0292'                   ..SQRX:
                        ;++++
                        ;
                        ; LOG
```

```
                        ;
                        ;----
                                M.CMD[LOG,..LOGX,0,LPAD,FNARG,..LOG][
0292'   18              +LOG:   .BYTE $CMDADR
0293'   02              +       .BYTE (..LOGX-LOG)/16+1
0294'   00              +       .BYTE 0
0295'   0000:14         +       .WORD LPAD
0297'   02A0'           +       .WORD ..LOG
0299'   01EB'           +       .WORD FNARG
029B'   4C4F4700        +       .ASCIZ /LOG/
                        +]
029F'   00                      .BYTE   0
02A0'   3E08             ..LOG: MVI     A,N.LOG          ;BASE 10 LOG
02A2'   C3 01E7'                JMP     FFUN
02A5'                    ..LOGX:
                        ;++++
                        ;
                        ; LN
                        ;
                        ;----
                                M.CMD[LN,..LNX,0,LOG,FNARG,..LN][
02A5'   18              +LN:    .BYTE $CMDADR
02A6'   02              +       .BYTE (..LNX-LN)/16+1
02A7'   00              +       .BYTE 0
02A8'   0292'           +       .WORD LOG
02AA'   02B2'           +       .WORD ..LN
02AC'   01EB'           +       .WORD FNARG
02AE'   4C4E00          +       .ASCIZ /LN/
                        +]
02B1'   00                      .BYTE   0
02B2'   3E09             ..LN:  MVI     A,N.LN           ;BASE E LOG
02B4'   C3 01E7'                JMP     FFUN
02B7'                    ..LNX:
                        ;++++
                        ;
                        ; EXP
                        ;
                        ;----
                                M.CMD[EXP,..EXPX,0,AFIX[E],FNARG,..EXP][
02B7'   18              +EXP:   .BYTE $CMDADR
02B8'   02              +       .BYTE (..EXPX-EXP)/16+1
02B9'   00              +       .BYTE 0
02BA'   632C            +       .WORD AFIX[E][("E"-101Q)*16+FSTINT]
02BC'   02C5'           +       .WORD ..EXP
02BE'   01EB'           +       .WORD FNARG
02C0'   45585000        +       .ASCIZ /EXP/
                        +]
02C4'   00                      .BYTE   0
02C5'   3E0A             ..EXP: MVI     A,N.EXP          ;RAISE TO E POWER
02C7'   C3 01E7'                JMP     FFUN
02CA'                    ..EXPX:
                        ;++++
                        ;
                        ; PI
                        ;
```

```
                        ;----
                                M.CMD[PI,..PIX,0,AFIX[P],0,..PI][
02CA'   18              +PI:    .BYTE $CMDADR
02CB'   02              +       .BYTE (..PIX-PI)/16+1
02CC'   00              +       .BYTE 0
02CD'   63DC            +       .WORD AFIX[P][("P"-101Q)*16+FSTINT]
02CF'   02D7'           +       .WORD ..PI
02D1'   0000            +       .WORD 0
02D3'   504900          +       .ASCIZ /PI/
                        +]
02D6'   00                      .BYTE   0
02D7'   3E1A            ..PI:   MVI     A,N.PUPI        ;RETURN PI
02D9'   C3 01F1'                JMP     FRET
02DC'                   ..PIX:
                        ;++++
                        ;
                        ; FORMAT
                        ; FORMATS A FLOATING POINT NUMBER INTO AN INTEGER AND FR
                        ACTIONAL
                        ; PART (ASCII STRING) WITH N DIGITS AFTER THE DECIMAL PO
                        INT
                        ;
                        ; NEEDS:
                        ;   FLOATING NUMBER TO FORMAT
                        ;   INTEGER GIVING NUMBER OF DIGITS AFTER DECIMAL POINT
                        ;       (MAY BE BETWEEN 0 AND 6)
                        ;
                        ; RETURNS:
                        ;   STRING WITH FORMATTED NUMBER
                        ;
                        ; CALLS:
                        ;   ARG - FETCH ARGUMENTS
                        ;   ALSTS - ALLOCATE STRING
                        ;   GETOPND - FETCH OPERAND
                        ;   INC5IY - BUMP IY TO NEXT ARG
                        ;   MSGN - GET SIGN
                        ;   NORMLIZ - NORMALIZ NUMBER
                        ;   NUMIT - CONVERT TO ASCII
                        ;
                        ;----
                                M.CMD[FORMAT,..FMTX,0,AFIX[F],..FMARG,..FMT][
02DC'   18              +FORMAT: .BYTE $CMDADR
02DD'   09              +       .BYTE (..FMTX-FORMAT)/16+1
02DE'   00              +       .BYTE 0
02DF'   633C            +       .WORD AFIX[F][("F"-101Q)*16+FSTINT]
02E1'   02EC'           +       .WORD ..FMT
02E3'   02EF'           +       .WORD ..FMARG
02E5'   464F524D4154+   +       .ASCIZ /FORMAT/
                        +]
02EC'   CD 0000:05      ..FMT:  CALL    ARG
02EF'   060400          ..FMARG:.BYTE   $FVAL,$IVAL,0
02F2'   E5                      PUSH    H
02F3'   3E01                    MVI     A,($SASCII+19)/16
02F5'   CD 0000:04              CALL    ALSTS           ;ALLOCATE STRING
02F8'   E5                      PUSH    H               ;SAVE THE PTR
```

```
02F9'   EB                      XCHG                    ;TO THE TOP
02FA'   CD 0000:0E              CALL    GETOPND         ;GET FLOATING VALUE
02FD'   CD 0000:0F              CALL    INC5IY          ;POINT TO NEXT ARG
                        ;+
                        ; GET # OF DIGITS AFTER DP IN DE
                        ; IT MUST BE POSITIVE AND LESS THAN 6
                        ;-
0300'   CD 0000:0E              CALL    GETOPND         ;GET # DIGITS AFTER DP
0303'   7B                      MOV     A,E             ;A = # DIGITS AFTER
                                TST     A               ;MUST BE > 0[
0304'   B7              +       ORA     A]
0305'   FA 030C'              JM      ..ERR           ;ERROR
0308'   FE07                   CPI     6+1             ;MUST BE < 6
030A'   3804                   JRC     ..NN            ;OR ERROR
030C'                   ..ERR:  ERROR   ER.DP[
030C'   CD 0000:0B     +       CALL ERRPGM
030F'   37              +       .BYTE ER.DP
                        +]
                        ;+
                        ; PRINT INITIAL SIGN AND NORMALIZE NUMBER
                        ;-
0310'   CD 0042'        ..NN:  CALL    MSGN            ;GET THE SIGN
0313'   D5                     PUSH    D               ;SAVE DIGIT COUNT
0314'   CD 0050'               CALL    NORMLIZ         ;NORMALIZE NUMBER
                        ;+
                        ; IF THE NUMBER IS GREATER THAN 10**10, IT IS
                        ; TOO BIG TO DISPLAY. IF IT IS GREATER THAN 10**6,
                        ; TRAILING ZEROS ARE ADDED
                        ;-
0317'   79                      MOV     A,C             ;A=EXPONENT
0318'   FE07                   CPI     6+1             ;0 < EXP < 6?
031A'   381C                   JRC     ..N1            ;YES?
031C'   FE0B                   CPI     10+1            ;6 < EXP < 10?
031E'   3808                   JRC     ..N             ;YES?
0320'   FEF0                   CPI     -16             ;-16 < EXP < 10
0322'   3014                   JRNC    ..N1            ;YES?
                                ERROR   ER.FMT          ;ELSE ERROR[
0324'   CD 0000:0B     +       CALL ERRPGM
0327'   38              +       .BYTE ER.FMT
                        +]
0328'   DE05             ..N:   SBI     5               ;SUBTRACT 5
032A'   4F                     MOV     C,A             ;NEW EXPONENT
032B'   CD 00E8'               CALL    FLTOUT
032E'   3630             ..NW:  MVI     M,'0'           ;TRAILING 0S
0330'   23                     INX     H
0331'   0D                     DCR     C               ;UNTIL COUNTUP
0332'   20FA                   JRNZ    ..NW
0334'   362E                   MVI     M,'.'           ;PUT DP IN
0336'   180C                   JMPR    ..N5
                        ;+
                        ; CONVERT FLOATING NUMBER TO ASCII AND GET THE
                        ; NUMBER OF DIGITS REQUESTED AFTER THE DECIMAL POINT
                        ;-
0338'   E5               ..N1:  PUSH    H               ;SAVE PTR
0339'   CD 002C'               CALL    NUMIT           ;GET ASCII NUMBER
```

```
033C'   E1                      POP     H               ;HL -> FIRST DIGIT
                                ;+
                                ; WE SEARCH THE DIGIT STRING FOR THE DECIMAL POINT
                                ;-
033D'   3E2E                    MVI     A,'.'           ;DP TO COMPARE
033F'   BE              ..N2:   CMP     M               ;IS IT POINT?
0340'   23                      INX     H               ;POINT AFTER
0341'   20FC                    JRNZ    ..N2            ;SCAN FOR IT
0343'   2B                      DCX     H               ;HL -> AT DP
                                ;+
                                ; DECIMAL POINT FOUND, WE COUNT OUT THE DIGITS
                                ; UNTIL DECIMAL POINT COUNTER EXHAUSTED OR END
                                ; OF STRING IS REACHED
                                ; E = NUMBER OF DIGITS ALLOWED AFTER DP
                                ;-
0344'   D1              ..N5:   POP     D
                                TST     E               ;IF 0, EXIT[
0345'   7B             +.IFN E        -A, [   MOV     A,E             ]
0346'   B7             +       ORA     A]
0347'   2812                    JRZ     ..OUT
0349'   23                      INX     H               ;HL -> AFTER DP
034A'   1D              ..N3:   DCR     E               ;ONE LESS DIGIT
034B'   FA 035B'                JM      ..OUT           ;ALL GONE? EXIT
                                TST     M               ;FOUND THE NULL?[
LL 034E'   7E           +.IFN M        -A?, ?[ MOV     A,M             ]
   034F'   B7           +       ORA     A]
0350'   23                      INX     H               ;HL -> AT NEXT CHAR
0351'   20F7                    JRNZ    ..N3            ;NO? CONTINUE
                                ;+
                                ; END OF STRING REACHED, NUMBER MUST BE PADDED WITH
                                ; TRAILING ZEROS UNTIL THE DECIMAL POINT COUNTER
                                ; IN E BECOMES MINUS
                                ;-
0353'   2B                      DCX     H               ;HL -> AT NULL
0354'   3630          ..N4:   MVI     M,'0'           ;TRAILING 0
0356'   23                      INX     H
0357'   1D                      DCR     E               ;ONE LESS DIGIT
0358'   F2 0354'                JP      ..N4
                                ;+
                                ; EXIT POINT
                                ; PUT IN TRAILING NULL, POP OFF STRING ADDRS & EXIT
                                ;-
035B'   D1              ..OUT:  POP     D               ;DE -> TOP OF STRING
                                CLR     M               ;ENDING NULL[
035C'   3600           +.IFN M         -A, [   MVI     M               ,0][
                        XRA     A]]
035E'   E1                      POP     H               ;RESTORE OTHERS
035F'   3E08                    MVI     A,$STRADR       ;RETURN TYPE STRING
0361'   C3 0000:1B             JMP     RETNONE         ;RETURN
0364'                   ..FMTX:
0364'   CCCC    7DCC    1:      .WORD   0CCCCH,07DCCH   ;.1
0368'   2400    14F4    N6:     .WORD   02400H,014F4H   ;10**6
                        .END
```


EF -

+++++ SYMBOL TABLE +++++

```
AASN    005F          ALSTS   0000:04 X     ARCCOS  0252          ARGSIN  0222
ARCTAN  0268'   I     ARG     0000:05 X     ARGEND  6252          ARGSTK  60CC
BACKGR  65CC          BLANK   0020          BOTRAM  6000          BOTTOM  64A9
CARTYP  0000:06 X     CHARSL  65DD          CLEARS  60CA          CNTRL   000C
CNTRLC  65CD          CNTRLO  65D9          CNTRLU  0015          CNTRLZ  65E4
CNVFS   0001'   I     COSINE  01F9'   I     CPLFLG  6540          CPLIN   681C
CPLTMP  672C          CPREF   691C          CP.ARG  0001          CP.MXL  0080
CP.MXR  0014          CP.NO   0002          CR      000D          CSBLOK  668D
CSFLAG  6260          CURCX   6813          CURCY   6815          CURLIN  6934
CURREN  64A5          CVDSTR  0000:07 X     CVFSTR  0000:08 X     C.CO    0011
C.C1    0012          C.CX    000B          C.CY    000D          C.DP    0013
C.ST    0002          C.X     0003          C.XF    000F          C.XS    0007
C.Y     0005          C.YF    0010          C.YS    0009          DDTON   65CE
DEVBL   6589          DEVCL0  6579          DEVCL1  657B          DEVCL2  657D
DEVCL3  657F          DEVCL4  6581          DEVCL5  6583          DEVCL6  6585
```

```
 .INSERT A:S.ASM
@.REMARK /
@                       *
@                     * * *
@                       ***
@                     *******
@                         *
@                       *
@                      *
@                     *
@                    *
@                    *******
@
@      "WHEN YOU CARE ENOUGH TO PROGRAM
@                 THE VERY BEST"
@
@              ZGRASS V3.00000000
@BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
@                   (C) 1978
@
@/
 .LINK
 .IDENT ES
 .PREL
 .INSERT A:MAC.ASM
@
 .INSERT A:ZRAM.ASM
 .INSERT A:NCUEQU.ASM
 .INSERT A:CPLEQU.ASM
 ;+
 ;  INTERNALS
 ;-
 .INTERN AND                ;LOGICAL AND
 .INTERN CPAREN             ;CLOSE PAREN
 .INTERN DOPTAB             ;DOUBLE CHAR OPERATORS
 .INTERN EQ                 ;EQUAL
 .INTERN FADD               ;FLOATING ADD
 .INTERN FDIV               ;FLOATING DIVIDE
 .INTERN FMOD               ;FLOATING REMAINDER
 .INTERN FMUL               ;FLOATING MULTIPLY
 .INTERN FRND               ;FLOATING RANDOM
 .INTERN FSUB               ;FLOATING SUBTRACT
 .INTERN GE                 ;GREATER OR EQUAL
 .INTERN GT                 ;GREATER
 .INTERN LE                 ;LESS OR EQUAL
 .INTER  LT                 ;LESS
 .INTERN NE                 ;NOT EQUAL
 .INTERN OPAREN             ;OPEN PAREN
 .INTERN OR                 ;LOGICAL OR
 .INTERN RMDR               ;COMPUTE REMAINDER
 .INTERN SOPTAB             ;SINGLE CHAR OPERATORS
 .INTERN UOPTAB             ;UNARY OPERATORS
 ;+
 ;  EXTERNALS
 ;-
```

```
                        .EXTERN CAT             ;STRING CONCATENATION
                        .EXTERN CONVAB          ;CONVERT BOTH
                        .EXTERN CONVB           ;CONVERT OPERAND B
                        .EXTERN DEC5IY          ;DEC IY BY 5
                        .EXTERN DONCU           ;PERFORM NCU OPERATION
                        .EXTERN ERRPGM          ;ERROR TRAP
                        .EXTERN FPOP            ;POP FLT FROM NCU STACK
                        .EXTERN FPUSH           ;PUSH FLT ON NCU STACK
                        .EXTERN GETOPND         ;GET OPERAND
                        .EXTERN IGET            ;GET INT FROM NCU STACK
                        .EXTERN IND             ;INDIRECTION
                        .EXTERN IPOP            ;POP FROM NCU STACK
                        .EXTERN IPSH0           ;PUSH 0 ON NCU STACK
                        .EXTERN IPUSH           ;PUSH TO NCU STACK
                        .EXTERN IYNCU           ;IY STACK -> NCU
                        .EXTERN NCUIY           ;NCU -> IY STACK
                        .EXTERN POPOPND         ;POP OPERAND
                        .EXTERN PSHOPND         ;PUSH OPERAND
                        .EXTERN PUTOPND         ;PUT OPERAND
                        .EXTERN SCMP            ;STRING COMARE ROUTINE
                        ;++++
                        ;
                        ; FSUB
                        ; SUBTRACTS 2 FLOATING POINT OPERANDS
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B
                        ;        OPERAND A
                        ;
                        ; RETURNS:
                        ;   IY->(OPERAND A - OPERAND B)
                        ;
                        ;----
0001'  06               FSUB:   .BYTE   $FVAL
0002'  CD 0000:14               CALL    POPOPND         ;GET OPERAND B
0005'  CD 0000:14               CALL    POPOPND         ;GET OPERAND A
0008'  3E19                     MVI     A,N.XCHF        ;EXCHANGE THEM
000A'  CD 0000:08               CALL    DONCU           ;NCU->B,A
000D'  3E11                     MVI     A,N.FSUB        ;COMPUTE A-B
000F'  CD 0000:08       FRET:   CALL    DONCU           ;DO LAST OPERATION
0012'  3E06                     MVI     A,$FVAL         ;RETURN TYPE FLT
0014'  C3 0000:15               JMP     PSHOPND         ;PUSH THE RESULT
                        ;++++
                        ;
                        ; FADD
                        ; ADDS 2 FLOATING POINT OPERANDS
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B
                        ;        OPERAND A
                        ;
                        ; RETURNS:
                        ;   IY->(OPERAND A + OPERAND B)
                        ;
                        ;----
```

ES -

FADD,FSUB - FLOATING POINT ADD,SUBTRACT

```
0017'   06              FADD:   .BYTE   $FVAL
0018'   CD 0000:14              CALL   POPOPND          ;GET OPERAND B
001B'   CD 0000:14              CALL   POPOPND          ;GET OPERAND A
001E'   3E10                    MVI    A,N.FADD         ;COMPUTE A+B
0020'   18ED                    JMPR   FRET             ;PUSH A+B
                        ;++++
                        ;
                        ; MOD
                        ; DIVIDES 2 DOUBLE PRECISION INTEGERS AND
                        ; RETURNS THE REMAINDER
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B
                        ;        OPERAND A
                        ;
                        ; RETURNS:
                        ;   IY->OPERAND A - (OPERAND A / OPERAND B) * OPERAND B
                        ;
                        ;----
0022'   00              FMOD:   .BYTE   $DVAL
0023'   CD 0000:14              CALL   POPOPND          ;GET OPERAND B
0026'   CD 0000:14              CALL   POPOPND          ;GET OPERAND A
0029'   3E39                    MVI    A,N.XCHD         ;EXCHANGE THEM
002B'   CD 0000:08              CALL   DONCU
002E'   CD 003B'                CALL   RMDR             ;GET REMAINDER
0031'   3E1C                    MVI    A,N.FLTD         ;CONVERT TO FLT
0033'   CD 0000:08              CALL   DONCU
0036'   3E06                    MVI    A,$FVAL          ;RETURN TYPE FLT
0038'   C3 0000:15              JMP    PSHOPND
                        ;++++
                        ;
                        ; RMDR
                        ; COMPUTES THE REMAINDER OF 2 DOUBLE PRECISION
                        ; INTEGERS AND THE QUOTIENT, TOO!
                        ;
                        ; NEEDS:
                        ;  NCU->OPERAND B
                        ;       OPERAND A
                        ;
                        ; RETURNS:
                        ;   NCU STACK HAS REMAINDER AS TOS
                        ;   QUOTIENT AS NOS
                        ;
                        ;----
003B'   FDE5            RMDR:   PUSH   Y
003D'   FD21 6538               LXI    Y,RMDTMP         ;IY->TEMP AREA
0041'   3E00                    MVI    A,$DVAL          ;SAVE OPND B
0043'   CD 0000:16              CALL   PUTOPND          ;ON IY STACK
                        ;+
                        ; FROM HEREON IN, THE COMMENT REFLECT THE CONTENTS
                        ; OF THE NCU STACK. FOR EXAMPLE: B,A DENOTES
                        ; B AS TOS, A AS NOS
                        ;-
0046'   3E37                    MVI    A,N.PTOD         ;A, A
0048'   CD 0000:08              CALL   DONCU
```

```
004B'   CD 0000:12          CALL    IYNCU           ;B, A, A
004E'   3E2F                MVI     A,N.DDIV        ;A/B, A
0050'   CD 0000:08          CALL    DONCU
0053'   F5                  PUSH    PSW             ;SAVE QUOTIENT STATUS
0054'   3E37                MVI     A,N.PTOD        ;A/B, A/B, A
0056'   CD 0000:08          CALL    DONCU
0059'   CD 0000:12          CALL    IYNCU           ;B, A/B, A/B, A
005C'   3E2E                MVI     A,N.DMUL        ;B*A/B, A/B, A
005E'   CD 0000:08          CALL    DONCU
0061'   CD 0000:13          CALL    NCUIY           ;A/B, A
0064'   3E39                MVI     A,N.XCHD        ;A, A/B
0066'   CD 0000:08          CALL    DONCU
0069'   CD 0000:12          CALL    IYNCU           ;B*A/B, A, A/B
006C'   3E2D                MVI     A,N.DSUB        ;A-B*A/B, A/B
006E'   CD 0000:08          CALL    DONCU
0071'   F1                  POP     PSW             ;RESTORE CCS
0072'   FDE1                POP     Y
0074'   C9                  RET
                    ;++++
                    ;
                    ; FDIV
                    ; DIVIDES 2 FLOATING POINT OPERANDS
                    ;
                    ; NEEDS:
                    ;   IY->OPERAND B
                    ;       OPERAND A
                    ;
                    ; RETURNS:
                    ;   IY->(OPERAND A / OPERAND B)
                    ;
                    ;----
0075'   06          FDIV:   .BYTE   $FVAL
0076'   CD 0000:14          CALL    POPOPND         ;GET OPERAND B
0079'   CD 0000:14          CALL    POPOPND         ;GET OPERAND A
007C'   3E19                MVI     A,N.XCHF        ;EXCHANGE THEM
007E'   CD 0000:08          CALL    DONCU           ;NCU->B,A
0081'   3E13                MVI     A,N.FDIV        ;DIVIDE THEM
0083'   188A                JMPR    FRET            ;PUSH & RETURN
                    ;++++
                    ;
                    ; FMUL
                    ; MULTIPLIES 2 FLOATING POINT OPERANDS
                    ;
                    ; NEEDS:
                    ;   IY->OPERAND B
                    ;       OPERAND A
                    ;
                    ; RETURNS:
                    ;   IY -> OPERAND A * OPERAND B
                    ;
                    ;----
0085'   06          FMUL:   .BYTE   $FVAL
0086'   CD 0000:14          CALL    POPOPND         ;GET OPERAND B
0089'   CD 0000:14          CALL    POPOPND         ;GET OPERAND A
008C'   3E12                MVI     A,N.FMUL        ;MULTIPLY
```

```
008E'   C3 000F'              JMP     FRET            ;PUSH & RETURN
                        ;++++
                        ;
                        ; FRND
                        ; PRODUCES A RANDOM FLOATING POINT NUMBER IN THE
                        ; GIVEN RANGE (OPERAND A - OPERAND B)
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B
                        ;       OPERAND A
                        ;
                        ; RETURNS:
                        ;    IY->RANDOM NUMBER BETWEEN OPERAND A
                        ;       AND OPERAND B INCLUSIVE
                        ;
                        ;----
0091'   06              FRND:   .BYTE   $FVAL
0092'   CD 0000:0C             CALL    GETOPND         ;NCU->B
0095'   CD 0000:14             CALL    POPOPND         ;NCU->B, B
0098'   CD 0000:0C             CALL    GETOPND         ;NCU-> A, B, B
                        ;+
                        ; HERE WE ATTEMPT TO FIND WHICH ARGUMENT WAS BIGGER
                        ; WE MUST RESOLVE THE TWO ARGUMENTS INTO A POSITIVE
                        ; DIFFERENCE AND A LOWER BOUND
                        ;-
009B'   3E11                    MVI     A,N.FSUB        ;NCU->B-A, B
009D'   CD 0000:08              CALL    DONCU
                                TST     A               ;CHECK SIGNI
00A0'   B7              +       ORA     A]
00A1'   F2 00A8'                JP      ..R1            ;NO? >0
00A4'   3E15                    MVI     A,N.CHSF        ;NCU->A-B, B
00A6'   1805                    JMPR    ..R2
00A8'   CD 0000:0C      ..R1:   CALL    GETOPND         ;NCU->A, B-A, B
00AB'   3E19                    MVI     A,N.XCHF        ;NCU->B-A, A, B
00AD'   CD 0000:08      ..R2:   CALL    DONCU           ;WHATEVER
                        ;+
                        ; SINCE THE RANDOM NUMBER WE GET WILL BE BETWEEN
                        ; 0 AND .5, WE MULTIPLY THE RANGE BY 2 TO COMPENSATE
                        ; SINCE THE CALCULATIONS NEED A 0 TO 1 RANGE
                        ;-
00B0'   3E17                    MVI     A,N.PTOF        ;NCU->RANGE,RANGE,LO
00B2'   CD 0000:08              CALL    DONCU
00B5'   3E10                    MVI     A,N.FADD        ;NCU->RANGE*2,LO
00B7'   CD 0000:08              CALL    DONCU
00BA'   CD 00E7'                CALL    RND             ;NCU->RND, RANGE, LO
00BD'   EB                      XCHG                    ;DE GETS RAND NUM
00BE'   CD 0000:11              CALL    IPUSH           ;PUSH ON NCU STACK
00C1'   CD 00E7'                CALL    RND             ;GET 'NUTHER 16 BITS
00C4'   EB                      XCHG                    ;DE GETS RAND NUM
                        ;+
                        ; THE EXPONENT IS ZEROED, THE HI ORDER BIT OF
                        ; THE MANTISSA SET TO NORMALIZE. THIS GIVES A
                        ; RANDOM NUMBER BETWEEN .5 AND 1
                        ;-
                                CLR     D               ;CLEAR EXPI
```

ES -
RND - RANDOM NUMBER STUFF

```
LL 00C5'   1600          +.IFN D          -A?, ?[ MVI     D               ,0][
                          XRA     A]]
   00C7'   CBFB                   SET     7,E             ;SET HI BIT
   00C9'   CD 0000:11             CALL    IPUSH           ;RND IS .5-1
   00CC'   CD 0000:10             CALL    IPSHO           ;NOW PUSH .5
                          ;+
                          ; WE SUBTRACT .5 TO GET A NUMBER BETWEEN 0 AND
                          ; .5, THE RANGE WAS MULTIPLIED BY 2 EARLIER,
                          ; WE EFFECTIVELY HAVE A NUMBER BETWEEN 0 AND 1
                          ;-
   00CF'   11 0080                LXI     D,080H          ;ONTO NCU STACK
   00D2'   CD 0000:11             CALL    IPUSH
   00D5'   3E11                   MVI     A,N.FSUB        ;SUBTRACT TO MAKE
   00D7'   CD 0000:08             CALL    DONCU           ;RANGE 0-.5
                          ;+
                          ; NOW ALL WE DO IS MULTIPLY THE THE RANGE AND
                          ; ADD THE LOWER BOUND!
                          ;-
   00DA'   3E12                   MVI     A,N.FMUL        ;NCU->RANGE*RND, LO
   00DC'   CD 0000:08             CALL    DONCU
   00DF'   CD 0000:07             CALL    DEC5IY
   00E2'   3E10                   MVI     A,N.FADD        ;NCU->LO+RANGE*RMD
   00E4'   C3 000F'               JMP     FRET
                          ;++++
                          ;
                          ; RND
                          ; RANDOM BITSTRING ROUTINE - RETURNS 16 BITS OF
                          ; MADNESS IN HL. TAKEN FROM HVGSYS ORIGINALLY
                          ; WRITTEN BY JEFF FREDRICKSON
                          ;
                          ; RETURNS:
                          ;   HL = RANDOM NUMBER (-32767 TO 32767)
                          ;
                          ; DESTROYS: EVERYTHING (ENTIRE 8080 REGISTER SET)
                          ;
                          ;----
   00E7'   CD 00F2'       RND:    CALL    ..RND1          ;GET HI ORDER
   00EA'   7C                     MOV     A,H
   00EB'   F5                     PUSH    PSW
   00EC'   CD 00F2'               CALL    ..RND1          ;GET LO ORDER
   00EF'   F1                     POP     PSW
   00F0'   6F                     MOV     L,A
   00F1'   C9                     RET
   00F2'   2A 655F        ..RND1: LHLD    RANSHT
   00F5'   CD 010C'               CALL    ..SHFT
   00F8'   01 0017                LXI     B,23
   00FB'   09                     DAD     B
   00FC'   8A                     ADC     D
   00FD'   22 655F                SHLD    RANSHT
   0100'   2A 6561                LHLD    RANSHT+2
   0103'   5F                     MOV     E,A
   0104'   CD 010C'               CALL    ..SHFT
   0107'   19                     DAD     D
   0108'   22 6561                SHLD    RANSHT+2
   010B'   C9                     RET
```

```
010C'                  ..SHFT: MVD     B,H[
010C'   44            +        MOV     B,H
010D'   4D            +        MOV     B+1,H+1]
                               CLR     A[
010E'   AF            +.IFN A-A, [     MVI     A,0][   XRA     A]]
010F'   1607                   MVI     D,7
0111'   29             ..SH1:  DAD     H
0112'   17                     RAL
0113'   15                     DCR     D
0114'   20FB                   JRNZ    ..SH1
0116'   09                     DAD     B
0117'   8A                     ADC     D
0118'   C9                     RET
                       ;++++
                       ;
                       ; CMP
                       ; COMPARISON ROUTINE FOR RELATIONAL OPERATORS
                       ;
                       ; NEEDS:
                       ;    IY->OPERAND B
                       ;         OPERAND A
                       ;
                       ; RETURNS:
                       ;    BOTH OPERANDS POPPED OFF
                       ;    CC'S SET FOR COMPARISON
                       ;
                       ;
                       ; CALLS:
                       ;    SCMP - STRING COMPARE
                       ;    FCMP - FLOATING COMAPARE
                       ;
                       ;----
0119'   FD4600         CMP:    MOV     B,E.TYP(Y)          ;B=TYPE OF OPNDB
011C'   FD7EFB                MOV     A,E.TYP-E.SIZ(Y) ;A=TYPE OF OPNDA
011F'   B8                     CMP     B                   ;TYPES EQUAL?
0120'   2807                   JRZ     ..C2                ;DISPATCH, NO CONV
0122'   3801                   JRC     ..C1                ;CONVERT TO TYPE OF B
0124'   47                     MOV     B,A                 ;CONVERT TO TYPE OF A
0125'   CD 0000:05     ..C1:   CALL    CONVAB              ;BOTH OPERANDS
0128'   78                     MOV     A,B                 ;A=CONVERTED TYPE
0129'   FE08           ..C2:   CPI     $STRADR             ;STRINGS?
012B'   CA 0000:17             JZ      SCMP                ;STRING COMPARE
012E'   FE06                   CPI     $FVAL               ;FLOATING NUMS?
0130'   280F                   JRZ     FCMP                ;FLT COMPARE
                       ;+++
                       ;
                       ; ICMP
                       ; COMPARES 2 SINGLE PRECISION INTEGERS
                       ; (CAN ALSO BE USED FOR POINTERS)
                       ;
                       ;---
0132'   CD 0000:14     ICMP:   CALL    POPOPND             ;GET OPND B
0135'   EB                     XCHG                        ;HL=OPND B
0136'   CD 0000:14             CALL    POPOPND             ;GET OPND A
0139'   EB                     XCHG                        ;HL=A,DE=B
```

```
                                CLC[
013A'   B7              +       ORA     A]
013B'   ED52                    DSBC    D               ;COMPUTE A-B
013D'   11 0000         CRET:   LXI     D,0
0140'   C9                      RET
                        ;++++
                        ;
                        ; FCMP
                        ; COMPARES 2 FLOATING POINT OPERANDS
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B
                        ;       OPERAND A
                        ;
                        ; RETURNS:
                        ;   DE = 0 (FALSE)
                        ;   CONDITION CODES SET FOR (OPERAND A - OPERAND B)
                        ;
                        ;----
0141'                   FCMP:
0141'   CD 0000:14             CALL    POPOPND         ;GET OPERAND B
0144'   CD 0000:14             CALL    POPOPND         ;GET OPERAND A
0147'   3E19                   MVI     A,N.XCHF        ;EXCHANGE THEM
0149'   CD 0000:08              CALL    DONCU
014C'   3E11                   MVI     A,N.FSUB        ;SUBTRACT THEM
014E'   CD 0000:08              CALL    DONCU
                                TST     A               ;CHECK CCS[
0151'   B7              +       ORA     A]
0152'   18E9                    JMPR    CRET            ;RETURN
                        ;++++
                        ;
                        ; RELATIONAL OPERATORS LT, GT, GE, LE, NE, EQ
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B
                        ;       OPERAND A
                        ;
                        ; RETURNS:
                        ;   IF THE RELATIONAL OPERATOR WAS SATISFIED BY THE
                        ;   RESULT OF (OPERAND B - OPERAND A)
                        ;   IY->1 (TRUE)
                        ;   ELSE
                        ;   IY->0 (FALSE)
                        ;
                        ;----
0154'   FE              GE:     .BYTE   $ANYVAL
0155'   CD 0119'                CALL    CMP             ;COMPARE A-B
0158'   F2 016F'                JP      TRUE            ;A >= B?
015B'   1813                    JMPR    FALSE           ;A < B
015D'   FE              LE:     .BYTE   $ANYVAL
015E'   CD 0119'                CALL    CMP             ;COMPARE A-B
0161'   280C                   JRZ     TRUE            ;A = B?
0163'   FA 016F'                JM      TRUE            ;A < B?
0166'   1808                    JMPR    FALSE           ;A > B
0168'   FE              LT:     .BYTE   $ANYVAL
```

```
0169'  CD 0119'            CALL    CMP             ;COMPARE A-B
016C'  F2 0170'            JP      FALSE           ;A >= B?
016F'  13           TRUE:  INX     D               ;DE=1 (TRUE)
0170'  CD 0000:11   FALSE: CALL    IPUSH           ;PUSH DE
0173'  3E1D                MVI     A,N.FLTS        ;CONVERT TO FLT
0175'  C3 000F'            JMP     FRET
0178'  FE           GT:    .BYTE   $ANYVAL
0179'  CD 0119'            CALL    CMP             ;COMPARE A-B
017C'  28F2                JRZ     FALSE           ;A = B?
017E'  FA 0170'            JM      FALSE           ;A < B?
0181'  18EC                JMPR    TRUE            ;A > B
0183'  FE           EQ:    .BYTE   $ANYVAL
0184'  CD 0119'            CALL    CMP             ;COMPARE A-B
0187'  20E7                JRNZ    FALSE           ;A != B?
0189'  18E4                JMPR    TRUE            ;A = B
018B'  FE           NE:    .BYTE   $ANYVAL
018C'  CD 0119'            CALL    CMP             ;COMPARE A-B
018F'  28DF                JRZ     FALSE           ;A = B?
0191'  18DC                JMPR    TRUE            ;A != B
                   ;++++
                   ;
                   ; CHKTF
                   ; INTERNAL FUNCTION TO RETURN 0 OR 1 BASED ON
                   ; WHETHER AN OPERAND IS 0 OR NOT.
                   ;
                   ; NEEDS:
                   ;  IY-> FLOATING OPERAND
                   ;
                   ; RETURNS:
                   ;   NCU STACK HAS FLOATING OPERAND
                   ;   D HAS 0 IF OPERAND WAS 0, OTHERWISE 1
                   ;
                   ; CALLS:
                   ;   GETOPND - GET OPERAND
                   ;   DONCU - DO NCU OPERATION
                   ;
                   ;----
0193'  CD 0000:0C  CHKTF: CALL    GETOPND         ;GET OPERAND
0196'  3E17               MVI     A,N.PTOF        ;DO NCU OP TO
0198'  CD 0000:08         CALL    DONCU           ;SIGN OF OPERAND
                          TST     A[
019B'  B7          +      ORA     A]
                          CLR     D               ;ASSUME 0[
019C'  1600        +.IFN D        -A?, ?[ MVI     D               ,0][
                   XRA     A]]
019E'  C8                 RZ                      ;0? RETURN FALSE
019F'  14                 INR     D               ;RETURN TRUE (1)
01A0'  C9                 RET
                   ;++++
                   ;
                   ; AND
                   ; LOGICAL AND OPERATION
                   ;
                   ; NEEDS:
                   ;  IY-> FLOATING OPERAND B
```

```
                        ;       FLOATING OPERAND A
                        ;
                        ; RETURNS:
                        ;  IY-> FLOATING 1 IF A AND B BOTH 1,
                        ;       ELSE FLOATING 0
                        ;
                        ; CALLS:
                        ;   CHKTF - INTERNAL CHECK FOR TRUE, FALSE
                        ;
                        ;----
01A1'  06               AND:    .BYTE   $FVAL
01A2'  CD 0193'                 CALL    CHKTF           ;CHECK SIGN
01A5'  5A                       MOV     E,D             ;SAVE OPND B SIGN
01A6'  CD 0193'                 CALL    CHKTF           ;CHECK SIGN OF A
01A9'  7A                       MOV     A,D             ;A=SIGN OF B
01AA'  A3                       ANA     E               ;AND THEM
01AB'  11 0000                  LXI     D,0             ;ASSUME FALSE
01AE'  CA 0170'                 JZ      FALSE
01B1'  C3 016F'                 JMP     TRUE            ;WAS TRUE!
                        ;++++
                        ;
                        ; OR
                        ; LOGICAL OR ROUTINE
                        ;
                        ; NEEDS:
                        ;  IY-> FLOATING OPERAND B
                        ;       FLOATING OPERAND A
                        ;
                        ; RETURNS:
                        ;  IY-> FLOATING 0 IF A AND B BOTH 0, ELSE
                        ;       FLOATING 1
                        ;
                        ; CALLS:
                        ;   CHKTF - INTERNAL ROUTINE TO CHECK TRUE, FALSE
                        ;
                        ;-----
01B4'  06               OR:     .BYTE   $FVAL
01B5'  CD 0193'                 CALL    CHKTF           ;GET SIGN OF B
01B8'  5A                       MOV     E,D             ;SAVE IN E
01B9'  CD 0193'                 CALL    CHKTF           ;GET SIGN OF A
01BC'  7A                       MOV     A,D             ;A=SIGN OF B
01BD'  B3                       ORA     E               ;DO THE OR
01BE'  11 0000                  LXI     D,0             ;ASSUME FALSE
01C1'  CA 0170'                 JZ      FALSE
01C4'  C3 016F'                 JMP     TRUE            ;WAS TRUE AFTER ALL
                        ;++++
                        ;
                        ; OPERATOR TABLES
                        ;   UOPTAB - UNARY OPERATORS
                        ;   SOPTAB - SINGLE CHAR BINARY OPERATORS
                        ;   DOPTAB - DOUBLE CHAR UNARY OPERATORS
                        ;
                        ; ENTRIES ARE AS FOLLOWS:
                        ; OFFSET         DESCRIPTION
                        ; ------         -----------
```

```
                        ; O.CHAR        CHARACTER DESCRIPTION OF OPERATOR
                        ;               THIS IS 2 BYTES, 2ND BYTE NULL IF
                        ;               IT IS A SINGLE CHAR OPERATOR
                        ; O.PREC        PRECEDENCE OF OPERATOR
                        ; O.TYP         TYPE OF VALUES OPERATOR WANTS
                        ; O.SUB         TOKEN FOR SUBROUTINE TO EVALUATE
                        ;-----
01C7'                   UOPTAB:
                                OPR     '-',,8,CP.FSUB  ;UNARY MINUS[
01C7'   2D              +       .BYTE   '-'
01C8'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01C9'   08              +       .BYTE   8
01CA'   0F              +       .BYTE   CP.FSUB ]
                                OPR     '+',,8,CP.FADD  ;UNARY PLUS[
01CB'   2B              +       .BYTE   '+'
01CC'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01CD'   08              +       .BYTE   8
01CE'   0E              +       .BYTE   CP.FADD ]
                                OPR     '@',,9,CP.IND   ;INDIRECTION[
01CF'   40              +       .BYTE   '@'
01D0'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01D1'   09              +       .BYTE   9
01D2'   15              +       .BYTE   CP.IND  ]
01D3'                   OPAREN: OPR     '(',,0,0                ;OPEN PAREN[
01D3'   28              +       .BYTE   '('
01D4'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01D5'   00              +       .BYTE   0
01D6'   00              +       .BYTE   0               ]
01D7'                   CPAREN: OPR     ')',,0,0                ;CLOSE PAREN[
01D7'   29              +       .BYTE   ')'
01D8'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01D9'   00              +       .BYTE   0
01DA'   00              +       .BYTE   0               ]
01DB'                   SOPTAB:
                                OPR     '!',,2,CP.OR    ;LOGICAL OR[
01DB'   7C              +       .BYTE   '!'
01DC'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01DD'   02              +       .BYTE   2
01DE'   1D              +       .BYTE   CP.OR   ]
                                OPR     '^',,3,CP.AND   ;LOGICAL AND[
01DF'   5E              +       .BYTE   '^'
01E0'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01E1'   03              +       .BYTE   3
01E2'   1C              +       .BYTE   CP.AND  ]
                                OPR     '-',,5,CP.FSUB  ;SUBTRACTION[
01E3'   2D              +       .BYTE   '-'
01E4'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01E5'   05              +       .BYTE   5
01E6'   0F              +       .BYTE   CP.FSUB ]
                                OPR     '+',,5,CP.FADD  ;ADDITION[
01E7'   2B              +       .BYTE   '+'
01E8'   00              +.IFB [], [.BYTE 0] [.BYTE ]
01E9'   05              +       .BYTE   5
01EA'   0E              +       .BYTE   CP.FADD ]
                                OPR     '&',,5,CP.CAT   ;STRING CONCATENATION[
```

```
01EB'   26          +         .BYTE   '&'
01EC'   00          +.IFB [], [.BYTE 0] [.BYTE ]
01ED'   05          +         .BYTE   5
01EE'   14          +         .BYTE   CP.CAT  ]
                              OPR     '*',,6,CP.FMUL  ;MULTIPLICATION[
01EF'   2A          +         .BYTE   '*'
01F0'   00          +.IFB [], [.BYTE 0] [.BYTE ]
01F1'   06          +         .BYTE   6
01F2'   10          +         .BYTE   CP.FMUL ]
                              OPR     '/',,6,CP.FDIV  ;DIVISION[
01F3'   2F          +         .BYTE   '/'
01F4'   00          +.IFB [], [.BYTE 0] [.BYTE ]
01F5'   06          +         .BYTE   6
01F6'   11          +         .BYTE   CP.FDIV ]
                              OPR     '\',,6,CP.FMOD  ;REMAINDER[
01F7'   5C          +         .BYTE   '\'
01F8'   00          +.IFB [], [.BYTE 0] [.BYTE ]
01F9'   06          +         .BYTE   6
01FA'   12          +         .BYTE   CP.FMOD ]
                              OPR     '%',,7,CP.FRND  ;RANDOM[
01FB'   25          +         .BYTE   '%'
01FC'   00          +.IFB [], [.BYTE 0] [.BYTE ]
01FD'   07          +         .BYTE   7
01FE'   13          +         .BYTE   CP.FRND ]
                              OPR     '<',,4,CP.LT    ;LESS THAN[
01FF'   3C          +         .BYTE   '<'
0200'   00          +.IFB [], [.BYTE 0] [.BYTE ]
0201'   04          +         .BYTE   4
0202'   16          +         .BYTE   CP.LT   ]
                              OPR     '>',,4,CP.GT    ;GREATER THAN[
0203'   3E          +         .BYTE   '>'
0204'   00          +.IFB [], [.BYTE 0] [.BYTE ]
0205'   04          +         .BYTE   4
0206'   17          +         .BYTE   CP.GT   ]
                              OPR     '=',,4,CP.EQ    ;EQUAL[
0207'   3D          +         .BYTE   '='
0208'   00          +.IFB [], [.BYTE 0] [.BYTE ]
0209'   04          +         .BYTE   4
020A'   18          +         .BYTE   CP.EQ   ]
                              OPR     '#',,4,CP.NE    ;NOT EQUAL[
020B'   23          +         .BYTE   '#'
020C'   00          +.IFB [], [.BYTE 0] [.BYTE ]
020D'   04          +         .BYTE   4
020E'   19          +         .BYTE   CP.NE   ]
                              OPR     AASN,,1,CP.ASS  ;ASSIGNMENT OPER[
020F'   5F          +         .BYTE   AASN
0210'   00          +.IFB [], [.BYTE 0] [.BYTE ]
0211'   01          +         .BYTE   1
0212'   03          +         .BYTE   CP.ASS  ]
0213'                DOPTAB:
                              OPR     '<','=',4,CP.LE ;LESS OR EQUAL[
0213'   3C          +         .BYTE   '<'
0214'   3D          +.IFB ['='], [.BYTE      0] [.BYTE '=']
0215'   04          +         .BYTE   4
0216'   1A          +         .BYTE   CP.LE   ]
```

```
                                OPR     '>','=',4,CP.GE ;GREATER OR EQUALL
0217'   3E              +       .BYTE   '>'
0218'   3D              +.IFB ['='], [.BYTE      0] [.BYTE '=']
0219'   04              +       .BYTE   4
021A'   1B              +       .BYTE   CP.GE   ]
021B'   00                      .BYTE   0
                         .END
```

ES -
+++++ SYMBOL TABLE +++++

AASN	005F	AND	01A1' I	ARGEND	6252	ARGSTK	60CC
BACKGR	65CC	BLANK	0020	BOTRAM	6000	BOTTOM	64A9
CAT	0000:04 X	CHARSL	65DD	CHKTF	0193'	CLEARS	60CA
CMP	0119'	CNTRL	000C	CNTRLC	65CD	CNTRLO	65D9
CNTRLU	0015	CNTRLZ	65E4	CONVAB	0000:05 X	CONVB	0000:06 X
CPAREN	01D7' I	CPLFLG	6540	CPLIN	681C	CPLTMP	672C
CPREF	691C	CP.AND	001C	CP.ARG	0001	CP.ASS	0003

```
 .INSERT A:S.ASM
@.REMARK /
@                        *
@                      * * *
@                       ***
@                     *******
@                         *
@                        *
@                       *
@                      *
@                     *******
@
@      "WHEN YOU CARE ENOUGH TO PROGRAM
@                THE VERY BEST"
@
@              ZGRASS V3.00000000
@BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
@                   (C) 1978
@
@/
 .INSERT A:MAC.ASM
@
 .INSERT A:NCUEQU.ASM
 .INSERT A:CPLEQU.ASM
 .INSERT A:ZRAM.ASM
 .LINK
 .IDENT  EVAL
 .PREL
 ;+
 ; INTERNALS
 ;-
 .INTERN CONVERT          ;CONVERT VALUE
 .INTERN EVAL             ;SAME AS EVALX
 .INTERN EVALARG          ;EVALUATE TO A VALUE
 .INTERN EVALC            ;COMPILE EXPRESSION
 .INTERN GETOPND          ;FETCH OPERAND
 .INTERN GETVAR           ;GET VARIABLE
 .INTERN POPOPND          ;POP OPERAND
 .INTERN PSHOPND          ;PUSH OPERAND
 .INTERN PUTOPND          ;SAVE OPERAND
 .INTERN SRCHOPR          ;FIND OPERATOR IN TABLE
 ;+
 ;
 ; EXTERNALS
 ;-
 .EXTERN CARTYP           ;FIND CHARACTER TYPE
 .EXTERN CONVI            ;INTERNAL CONVERT ROUTINE
 .EXTERN CPAREN           ;CLOSE PAREN OPERATOR
 .EXTERN DEC5IY           ;DEC IY BY 3
 .EXTERN DONCU            ;DO NCU OPERATION
 .EXTERN DOPTAB           ;DOUBLE CHAR OPERATORS
 .EXTERN ERRPGM           ;ERROR TRAP
 .EXTERN EVLCPL           ;EVALUATE COMPILED CODE
 .EXTERN FNDSPC           ;PARSE SPECIAL VAR
```

```
                        .EXTERN FPUSH           ;PUSH FLT ONTO NCU STACK
                        .EXTERN FREE            ;FREE MEMORY
                        .EXTERN FUNCTION        ;GENERATE FUNCTION CALL
                        .EXTERN GENCOD          ;GENERATE 1 BYTE OF CODE
                        .EXTERN GETNUM          ;PARSE FLOATING NUM
                        .EXTERN GETSTR          ;PARSE LITERAL STRING
                        .EXTERN GETSUB          ;GET OPERATOR SUB
                        .EXTERN GETVAL          ;GET VALUE OF NAME
                        .EXTERN INC5IY          ;BUMP IY BY 3
                        .EXTERN IPUSH           ;PUSH DE ON NCU STACK
                        .EXTERN IYNCU           ;IY STACK -> NCU
                        .EXTERN NAMGET          ;LOOK UP A NAME
                        .EXTERN NAMBLD          ;CREATE A NAME
                        .EXTERN NAMSET          ;PARSE A NAME
                        .EXTERN NCUIY           ;NCU -> IY STACK
                        .EXTERN OPAREN          ;OPEN PAREN OPERATOR
                        .EXTERN OPUSH           ;PUSH OPERAND
                        .EXTERN SOPTAB          ;SINGLE CHAR OPERATORS
                        .EXTERN UOPTAB          ;UNARY OPERATORS
                        ;++++
                        ;
                        ; EVALC
                        ; PARSES ZGRASS EXPRESSIONS AND PRODUCES 'CODE' TO
                        ; EVALUATE THEM
                        ;
                        ; NEEDS:
                        ;   HL -> 1ST CHAR OF ASCII EXPRESSION
                        ;   PCNT = FLAG TELLING WHETHER THIS IS A
                        ;       RECURSIVE CALL OR NOT. IF PCNT=0, THIS IS
                        ;       THE FIRST ENTRY AND THE OPERATOR STACK IS SET UP

                        ;       OTHERWISE, IT IS ASSUMED TO BE A RECURSIVE CALL
                        ;       AND STACK IS NOT SET
                        ;   IY -> NEXT BYTE IN CODE AREA
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER LAST CHAR OF EXPRESSION
                        ;   IY -> AFTER TERMINATOR IN CODE AREA
                        ;
                        ;----
0001'                   EVALC:
0001'  DDE5                     PUSH    X
0003'  D5                       PUSH    D
0004'  C5                       PUSH    B
                        ;+
                        ; INITIALIZATION
                        ; CHECK ENTRY FLAG
                        ; GET STACK POINTERS SET UP
                        ; PUSH INITIAL PAREN
                        ;-
0005'  3A 655E                  LDA     PCNT            ;CHECK PAREN COUNT
                                TST     A               ;IF ZERO, FIRST ENTRY[
0008'  B7               +       ORA     A]
0009'  F5                       PUSH    PSW             ;SAVE ON STACK
000A'  CC 0189'                 CZ      SETSTK          ;SET UP STACK AREA
```

```
                                CLR     A               ;CLEAR PAREN COUNT[
000D'   AF              +.IFN A         -A, [   MVI     A                 ,0][
                         XRA    A]]
000E'   32 655E                 STA     PCNT
0011'   11 0000:1C              LXI     D,OPAREN        ;DE -> OPEN PAREN ENTRY
                                MVD     X,D             ;IX -> O.P. TABLE ENT[
0014'   D5              +.IFIDN [X] [X], [      PUSH    D
0015'   DDE1            +               POP     X
                        +]
0017'   1811                   JMPR    ..EPAR          ;SIMULATE (
                        ;+
                        ; POSSIBLE UNAR OPERATOR MAY PRECEDE OPERAND
                        ; IF CURRENT CHAR IS DELIM CAN BE ( ) OR UNARY OP
                        ;-
0019'                   ..EVTOP:
0019'   CD 0000:04              CALL    CARTYP          ;CHECK CURRENT CHAR
001C'   2029                    JRNZ    ..EFST          ;NOT DELIM, 1ST OPND
001E'   CD 022D'                CALL    UNOPR           ;UNARY OPERATOR?
0021'   3824                    JRC     ..EFST          ;NO, 1ST OPERAND
                        ;+
                        ; CHECK FOR OPEN PAREN (
                        ; PCNT = PAREN COUNT, IS BUMPED EVERY TIME AN
                        ; OPEN PAREN SEEN, DEC'D EVERY TIME CLOSE FOUND
                        ; DEC PAREN COUNT AND PUSH ( ON OPERATOR STACK
                        ; AND BACK TO THE TOP AGAIN
                        ;-
0023'   DD7E00                  MOV     A,O.CHAR(X)     ;CHECK IF CURRENT OP
0026'   FE28                    CPI     '('             ;IS AN OPEN PAREN
0028'   200C                    JRNZ    ..ECLOS         ;NO? TRY CLOSE
002A'   3A 655E        ..EPAR: LDA     PCNT            ;BUMP PAREN COUNT
002D'   3C                      INR     A
002E'   32 655E                 STA     PCNT
0031'   CD 012B'                CALL    PSHOPR          ;PUSH THE PAREN
0034'   18E3                    JMPR    ..EVTOP         ;BACK TO TOP
                        ;+
                        ; CHECK FOR CLOSE PAREN )
                        ; ALSO COME HERE WHEN END DELIMITER FOUND
                        ; THE LINE PTR IS BACKED UP TO POINT AT THE
                        ; CLOSE PAREN OR DELIMITER FOR LATER EXAMINATION
                        ;-
0036'   FE29            ..ECLOS:CPI     ')'             ;CLOSE PAREN
0038'   2003                    JRNZ    ..EUN           ;UNARY OP THEN
003A'   2B              ..EDLM: DCX     H               ;BACK UP PTR
003B'   184B                    JMPR    ..ECLX          ;PROCESS IT
                        ;+
                        ; A UNARY OPERATOR WAS FOUND
                        ; A FAKE OPERAND (0) IS PUSHED ON THE OPERAND
                        ; STACK AND THE OPERATOR IS CHECKED
                        ;-
003D'   11 0000         ..EUN:  LXI     D,0             ;PUSH A 0
0040'   3E02                    MVI     A,$NULL         ;TYPE IS 0
0042'   CD 0000:1D              CALL    OPUSH           ;GENERATE A PUSH
0045'   1815                    JMPR    ..EPREC         ;PARSE 2ND
                        ;+
                        ; ACTUAL OPERAND PARSED HERE
```

```
                        ; IT IS PUSHED ON THE OPERAND STACK AND AN ATTEMPT
                        ; IS MADE TO GET A BINARY OPERATOR. IF THERE IS NO
                        ; OPERATOR THE EXPRESSION IS ENDED AND WE EXIT
                        ; OTHERWISE THE OPERATOR IS CHECKED AGAINST STACK
                        ;-
0047'  CD 019D'         ..EFST: CALL    POPND           ;GET AN OPERAND
004A'  3803                     JRC     ..EOP           ;FUNCTION CALL
004C'  CD 0000:1D               CALL    OPUSH           ;GENERATE A PUSH
004F'  CD 01E3'         ..EOP:  CALL    GETOPR          ;GET OPERATOR
0052'  3008                     JRNC    ..EPREC         ;NO? EXIT THEN
0054'  11 0000:06               LXI     D,CPAREN        ;SIMULATE A
                                MVD     X,D             ;CLOSE PAREN[
0057'  D5               +.IFIDN [X] [X], [      PUSH    D
0058'  DDE1             +               POP     X
                        +]
005A'  182C                     JMPR    ..ECLX          ;FOR END DELIM
                        ;+
                        ; OPERAND HAS BEEN PUSHED ONTO OPERAND
                        ; STACK. HERE WE DETERMINE WHETHER OR NOT TO PUSH
                        ; THE OPERATOR. THE STACK IS EVALUATED UNTIL THE
                        ; PRECEDENCE OF THE CURRENT OPERATOR IS GREATER
                        ; THAN THE PREDECENCE OF THE OPERATOR NN THE TOP
                        ; OF THE STACK. THEN THE CURRENT OPERATOR IS PUSHED
                        ; AND WE GO TO THE TOP TO PARSE ANOTHER OPERAND
                        ;-
005C'  CD 016C'         ..EPREC:CALL    FETOPR          ;GET TOP OPERATOR
005F'  78                       MOV     A,B             ;A = PRECEDENCE OF TOP
0060'  DDBE02                   CMP     O.PREC(X)       ;COMPARE WITH CURRENT
0063'  F2 006B'                 JP      ..EPOP          ;LESS? POP & EVAL
0066'  CD 012B'                 CALL    PSHOPR          ;PUSH & CONTINUE
0069'  18AE                     JMPR    ..EVTOP         ;BACK TO TOP
                        ;+
                        ; IX -> CURRENT OPERATOR ENTRY
                        ; HERE WE EVALUATE THE 2 OPERANDS ON THE TOP OF THE
                        ; STACK WITH THE CURRENT OPERATOR
                        ; IF IT WAS NOT A CLOSE PAREN OR DELIMITER WE LOOP
                        ; TO SEE IF THE OPERATOR CAN BE PUSHED NOW THAT
                        ; WE HAVE EVALUATED
                        ;-
006B'  DDE5             ..EPOP: PUSH    X               ;SAVE OUR X
006D'  CD 014A'                 CALL    POPOPR          ;POP OPERATOR
0070'  3A 6540                  LDA     CPLFLG          ;IN COMPILE MODE?
                                TST     A[
0073'  B7               +       ORA     A]
0074'  E5                       PUSH    H
0075'  DD7E03                   MOV     A,O.SUB(X)      ;A=OPERATOR TOKEN
0078'  C4 0000:10               CNZ     GENCOD          ;GENERATE THE CODE
007B'  CC 0000:13               CZ      GETSUB          ;ELSE DO IT
007E'  E1                       POP     H
007F'  DDE1                     POP     X
0081'  DD7E00                   MOV     A,O.CHAR(X)     ;A = OPERATOR
0084'  FE29                     CPI     ')'             ;CLOSE PAREN?
0086'  20D4                     JRNZ    ..EPREC         ;DONT LOOP THEN
                        ;+
                        ; IF THE OPERATOR IS A CLOSE PAREN OR DELIMITER
```

```
                        ; WE CONTINUE POPPING UNTIL AN OPEN PAREN IS FOUND
                        ; THEN THE PAREN COUNT IS DECREMENTED. IF IT IS 0,
                        ; WE CAN EXIT. IF NOT, AND THE CURRENT CHAR IS
                        ; NOT AN OPEN PAREN WE KEEP EVALUATING, IT IS
                        ; THE END DELIMITER
                        ;-
0088'  CD 016C'         ..ECLX: CALL    FETOPR          ;GET TOP OPER
008B'  FE28                     CPI     '('             ;OPEN? NO, EVAL AGAIN
008D'  20DC                     JRNZ    ..EPOP
008F'  DDE5                     PUSH    X               ;SAVE CURRENT ONE
0091'  CD 014A'                 CALL    POPOPR          ;POP THE PAREN
0094'  DDE1                     POP     X
0096'  3A 655E                  LDA     PCNT            ;DEC PAREN COUNT
0099'  3D                       DCR     A
009A'  32 655E                  STA     PCNT
009D'  2808                     JRZ     ..EVEX          ;0? EXIT THEN
009F'  7E                       MOV     A,M             ;A = CURRENT CHAR
00A0'  FE29                     CPI     ')'             ;TRULY A CLOSE?
00A2'  20E4                     JRNZ    ..ECLX          ;END DELIM? EVAL ALL
00A4'  23                       INX     H               ;HL -> AFTER PAREN
00A5'  18A8                     JMPR    ..EOP           ;ELSE CONTINUE PARSE
                        ;+
                        ; EVAL EXIT POINT
                        ;-
00A7'  F1               ..EVEX: POP     PSW             ;A = FORMER PCNT
00A8'  32 655E                  STA     PCNT            ;RESTORE IT
00AB'  C1                       POP     B
00AC'  D1                       POP     D
00AD'  DDE1                     POP     X
00AF'  3A 6540                  LDA     CPLFLG          ;IN COMPILE MODE?
                                TST     A[
00B2'  B7               +       ORA     A]
00B3'  C8                       RZ                      ;NO? RETURN
                                CLR     A               ;TERMINATOR[
00B4'  AF               +.IFN A         -A, [   MVI     A                ,0][
                         XRA    A]]
00B5'  C3 0000:10               JMP     GENCOD
                        ;++++
                        ;
                        ; EVAL, EVALARG
                        ; EVALUATE ARITHMETIC EXPRESSIONS
                        ;
                        ; ENTRIES:
                        ;   EVALARG - RETURNS ANY VALUE, WILL NOT RETURN
                        ;       A NAME
                        ;   EVAL - RETURNS THE TYPE OF EXPRESSION GIVEN
                        ;       A = TYPE OF THING WANTED. IN ADDITION TO
                        ;           THE NORMAL TYPES, 2 ADDITIONAL ONES ARE
                        ;           PROVIDED:
                        ;               $ANYVAL - ANY VALUE
                        ;               $ANYNAM - ANY NAME (ASSIGNABLE THING)
                        ;
                        ; NEEDS:
                        ;   HL -> 1ST CHAR OF ASCII EXPRESSION
                        ;   IY -> STACK AREA
```

```
                        ;    PCNT - AS FOR EVALC
                        ;
                        ; RETURNS:
                        ;    IY -> VALUE OF THING RETURNED
                        ;    HL -> AFTER LAST BYTE OF EXPRESSION PARSED
                        ;    BC DESTROYED
                        ;
                        ; CALLS:
                        ;    EVALC - EVALUATE EXPRESSION
                        ;    CONVERT - CONVERT IF NECESSARY
                        ;
                        ;----
00B8'                   EVALARG:
00B8'  3EFE                     MVI     A,$ANYVAL       ;ANY VALUE
00BA'  D5               EVAL:   PUSH    D
00BB'  F5                       PUSH    PSW
00BC'  CD 0001'                 CALL    EVALC           ;EVALUATE IT
00BF'  C1                       POP     B               ;B=TYPE TO CONVERT TO
00C0'  CD 00C5'                 CALL    CONVERT         ;CONVERT IF NEEDED
00C3'  D1                       POP     D
00C4'  C9                       RET
                        ;++++
                        ;
                        ; CONVERT
                        ; CONVERTS THE OPERAND ON THE TOP OF THE IY STACK
                        ; TO THE GIVEN TYPE (WILL HANDLE NAMES TOO)
                        ;
                        ; NEEDS:
                        ;    B = TYPE TO CONVERT TO
                        ;    IY -> VALUE OF THING TO CONVERT
                        ;
                        ; RETURNS:
                        ;    IY -> RESULTING VALUE AFTER CONVERSION
                        ;
                        ; CALLS:
                        ;    CONVI - CONVERT VALUE
                        ;    ANYNAM - CONVERT NAME
                        ;
                        ;----
00C5'  FD7E00           CONVERT:MOV     A,E.TYP(Y)      ;A=TYPE WE HAVE
00C8'  B8                       CMP     B               ;THE SAME?
00C9'  C8                       RZ                      ;EXIT THEN
00CA'  78                       MOV     A,B             ;A=TYPE WE WANT
00CB'  FEFC                     CPI     $ANYNAM         ;WANT A NAME?
00CD'  CC 00D7'                 CZ      ANYNAM          ;CHECK IT OUT
00D0'  C8                       RZ                      ;ZERO? RETURN
00D1'  E5                       PUSH    H
00D2'  CD 0000:05               CALL    CONVI           ;CONVERT THEN
00D5'  E1                       POP     H               ;RESTORE PTR
00D6'  C9                       RET
                        ;++++
                        ;
                        ; ANYNAM
                        ; INTERNAL ROUTINE FOR EVAL WHICH CHECKS THE
                        ; TYPE GIVEN AND SETS THE Z BIT IF IT IS A
```

```
                        ; NAME (OTHERWISE ERROR)
                        ;
                        ; NEEDS:
                        ;   A = TYPE TO CHECK
                        ;
                        ; RETURNS:
                        ;   SETS Z BIT IF TYPE IS ONE OF THE FOLLOWING:
                        ;       $ADDRI, $ADDRF, $ADDRS, $NAME
                        ;
                        ;----
00D7'  FE0A             ANYNAM: CPI     $NAME           ;NAME?
00D9'  C8                       RZ
00DA'  FE05                     CPI     $ADDRI          ;INTEGER ADDR?
00DC'  C8                       RZ
00DD'  FE07                     CPI     $ADDRF          ;FLOATING ADDR?
00DF'  C8                       RZ
00E0'  FE09                     CPI     $ADDRS          ;STRING ADDR?
00E2'  C8                       RZ
                                ERROR   ER.ASN[
00E3'  CD 0000:0A       +               CALL ERRPGM
00E6'  15               +               .BYTE ER.ASN
                        +]
                        ;++++
                        ;
                        ; GETOPND
                        ; FETCH OPERAND FROM IY STACK
                        ;
                        ; RETURNS:
                        ;   A = TYPE OF OPERAND
                        ;   DE = OPERAND FETCHED
                        ;   TOS OF NCU STACK GETS OPERAND IF FLOATING
                        ;
                        ; ERRORS:
                        ;   ER.STK - STACK UNDERFLOW
                        ;
                        ; CALLS:
                        ;   IYNCU - GET FLOATING OPERAND
                        ;
                        ;----
00E7'                   GETOPND:
00E7'  FD7E00                   MOV     A,E.TYP(Y)      ;A=TYPE
00EA'  FE06                     CPI     $FVAL           ;FLOATING?
00EC'  CA 0000:17               JZ      IYNCU           ;GET ONTO NCU
00EF'  FE00                     CPI     $DVAL           ;DOUBLE?
00F1'  CA 0000:17               JZ      IYNCU           ;GET ONTO NCU
00F4'  FEFF                     CPI     -1              ;END OF STACK?
                                ZERROR  ER.STK[
00F6'  2004             +               JRNZ ..0001
00F8'  CD 0000:0A       +               CALL ERRPGM
00FB'  04               +               .BYTE ER.STK
00FC'                   +..0001:]
00FC'  FD5E01                   MOV     E,E.LVAL(Y)     ;GET IN DE
00FF'  FD5602                   MOV     D,E.HVAL(Y)
0102'  C9                       RET
                        ;++++
```

```
                        ;
                        ; PUTOPND
                        ; STORE OPERAND ON IY STACK
                        ;
                        ; NEEDS:
                        ;   A = TYPE OF OPERAND
                        ;   DE = OPERAND TO STORE
                        ;   TOS OF NCU STACK HAS OPERAND IF FLOATING
                        ;
                        ; CALLS:
                        ;   NCUIY - GET FLOAT FROM NCU
                        ;
                        ;----
0103'                   PUTOPND:
0103'   FD7700                 MOV     E.TYP(Y),A
0106'   FE06                   CPI     $FVAL           ;FLOATING?
0108'   CA 0000:1B            JZ      NCUIY           ;GET FROM NCU
010B'   FE00                   CPI     $DVAL           ;DOUBLE?
010D'   CA 0000:1B            JZ      NCUIY           ;GET ROM NCU
0110'   FD7301                 MOV     E.LVAL(Y),E     ;SAVE DE
0113'   FD7202                 MOV     E.HVAL(Y),D
0116'   FD360300               MVI     E.VAL+2(Y),0
011A'   FD360400               MVI     E.VAL+3(Y),0
011E'   C9                     RET
                        ;++++
                        ; PSHOPND
                        ; PUSHES ONTO THE OPERAND STACK
                        ;
                        ; NEEDS:
                        ;   A = TYPE OF OPERANE
                        ;   DE = OPERAND TO PUSH
                        ;   TOS OF NCU STACK HAS OPERAND IF FLOATING
                        ;   THESE ARE NOT DESTROYED BY PSHOPND
                        ;
                        ; CALLS:
                        ;   DEC5IY - POINT TO NEXT SPOT
                        ;   PUTOPND - STORE ON IY STACK
                        ;
                        ;----
011F'                   PSHOPND:
011F'   CD 0000:15              CALL    INC5IY          ;IY->NEXT SPOT
0122'   C3 0103'               JMP     PUTOPND         ;STORE IT
                        ;++++
                        ;
                        ; POPOPND
                        ; POPS OFF THE OPERAND STACK
                        ;
                        ; RETURNS:
                        ;   A = TYPE OF OPERAND
                        ;   DE = OPERAND POPPED
                        ;   TOS OF NCU STACK GETS OPERAND IF FLOATING
                        ;
                        ; ERRORS:
                        ;   ER.STK - TRIED TO POP TOO MANY
                        ;       (THIS OCCURS IF THE MARKER IS FOUND)
```

```
                        ;
                        ; CALLS:
                        ;   DEC5IY - POINT TO PREVIOUS SPOT
                        ;   GETOPND - FETCH OPERAND
                        ;
                        ;----
0125'                   POPOPND:
0125'  CD 00E7'                CALL    GETOPND         ;GET THE OPERAND
0128'  C3 0000:07               JMP     DEC5IY          ;IY->PREVIOUS ENTRY
                        ;++++
                        ;
                        ; PSHOPR
                        ; PUSHES ONTO THE OPERATOR STACK
                        ;
                        ; NEEDS:
                        ;   IX = OERATOR OFFSET O PUSH
                        ;
                        ; MAINTAINS:
                        ;   OPRSP - OPERATOR STACK POINTER
                        ;   OPRSZ - SIZE OF OPERATOR STACK
                        ;
                        ; ERRORS:
                        ;   ER.IY - OUT OF STACK SPACE
                        ;
                        ;----
012B'  E5               PSHOPR: PUSH    H
012C'  D5                       PUSH    D
012D'  3A 655D                  LDA     OPRSZ           ;A = SIZE OF STACK
0130'  3D                       DCR     A               ;IF 0, NO MORE SPACE
                                ZERROR  ER.IY[
0131'  2004             +       JRNZ    ..0002
0133'  CD 0000:0A       +       CALL    ERRPGM
0136'  43               +       .BYTE   ER.IY
0137'                   +..0002:]
0137'  32 655D                  STA     OPRSZ           ;SAVE NEW SIZE
013A'  2A 655B                  LHLD    OPRSP           ;HL -> NEXT SPOT
                                MVD     D,X             ;DE = STUFF TO STORE[
013D'  DDE5             +.IFIDN [X                ] [X], [        PUSH    X
013F'  D1               +       POP     D
                        +]
0140'  72                       MOV     M,D             ;SAVE DE ON STACK
0141'  23                       INX     H
0142'  73                       MOV     M,E
0143'  23                       INX     H
0144'  22 655B                  SHLD    OPRSP           ;SAVE NEW SP
0147'  D1                       POP     D
0148'  E1                       POP     H
0149'  C9                       RET
                        ;++++
                        ;
                        ; POPOPR
                        ; POPS OFF THE OPERATOR STACK
                        ;
                        ; RETURNS:
                        ;   IX = OPERATOR OFFSET POPPED
```

```
                        ;
                        ; MAINTAINS:
                        ;   OPRSP - OPERATOR STACK POINTER
                        ;   OPRSZ - SIZE OF OPERATOR STACK
                        ;
                        ; ERRORS:
                        ;   ER.STK - TRIED TO POP TOO MANY
                        ;       (THIS OCCURS WHEN MARKER FOUND)
                        ;
                        ;----
014A'  E5               POPOPR: PUSH    H
014B'  D5                       PUSH    D
014C'  3A 655D                  LDA     OPRSZ           ;A=SIZE OF STACK
014F'  3C                       INR     A               ;BUMP IT
0150'  32 655D                  STA     OPRSZ           ;AND SAVE IT
0153'  2A 655B                  LHLD    OPRSP           ;HL->THING TO POP
0156'  2B                       DCX     H
0157'  5E                       MOV     E,M             ;POP INTO DE
0158'  2B                       DCX     H
0159'  22 655B                  SHLD    OPRSP           ;SAVE NEW SP
015C'  56                       MOV     D,M
015D'  3EFF                     MVI     A,-1            ;CHECK MARKER
015F'  BA                       CMP     D               ;AGAINST HI PART
                                ZERROR  ER.STK          ;IF -1, ERROR[
0160'  2004             +       JRNZ    ..0003
0162'  CD 0000:0A       +       CALL    ERRPGM
0165'  04               +       .BYTE   ER.STK
0166'                   +..0003:]
                                MVD     X,D             ;RETURN IN X[
0166'  D5               +.IFIDN [X] [X], [      PUSH    D
0167'  DDE1             +       POP     X
                        +]
0169'  D1                       POP     D
016A'  E1                       POP     H
016B'  C9                       RET
                        ;++++
                        ;
                        ; FETOPR
                        ; GETS THE CURRENT OPERATOR FROM OPERATOR STACK
                        ;
                        ; RETURNS:
                        ;   A = LO-ORDER CHARACTER OF OPERATOR
                        ;   B = PRECEDENCE OF OPERATOR
                        ;   C = TYPE OF OPERATOR
                        ;
                        ; ERRORS:
                        ;   ER.STK - ALREADY TRIED TO POP TOO MANY
                        ;       (OCCURS WHEN MARKER FOUND)
                        ;
                        ;----
016C'  D5               FETOPR: PUSH    D
016D'  DDE5                     PUSH    X
016F'  DD2A 655B                LIXD    OPRSP           ;GET STACK PTR
0173'  DD5EFF                   MOV     E,-1(X)         ;INTO DE
0176'  DD56FE                   MOV     D,-2(X)
```

```
                                MVD     X,D             ;IX -> OPERATOR ENTRY[
0179'   D5              +.IFIDN [X] [X], [      PUSH    D
017A'   DDE1            +       POP     X
                        +]
017C'   DD7E00                  MOV     A,O.CHAR(X)     ;CHARACTER IN A
017F'   DD4602                  MOV     B,O.PREC(X)     ;PRECEDENCE IN B
0182'   DD4E03                  MOV     C,O.TYP(X)      ;TYPE IN C
0185'   DDE1                    POP     X               ;RESTORE IX
0187'   D1                      POP     D
0188'   C9                      RET
                        ;++++
                        ;
                        ; SETSTK
                        ; SETS UP THE OPERATOR STACK
                        ;
                        ; RETURNS:
                        ;   OPRSP - OPERATOR STACK POINTER
                        ;   OPRSZ - OPERATOR STACK SIZE (BYTES)
                        ;
                        ;----
0189'   E5              SETSTK: PUSH    H
018A'   21 6547                 LXI     H,OPRSTK        ;OPERATOR STACK
018D'   3EFF                    MVI     A,-1            ;MARKER AS FIRST
018F'   77                      MOV     M,A             ;ELEMENT OF STACK
0190'   23                      INX     H
0191'   77                      MOV     M,A
0192'   23                      INX     H
0193'   22 655B                 SHLD    OPRSP
0196'   3E0A                    MVI     A,OPRL/2        ;LENGTH OF STACK
0198'   32 655D                 STA     OPRSZ
019B'   E1                      POP     H
019C'   C9                      RET
                        ;++++
                        ;
                        ; POPND
                        ; PARSES AN OPERAND AND RETURNS ITS TYPE AND VALUE
                        ;
                        ; NEEDS:
                        ;   HL -> OPERAND
                        ;
                        ; RETURNS:
                        ;   A = TYPE OF OPERAND
                        ;   HL -> AFTER OPERAND
                        ;   DE = VALUE OF OPERAND
                        ;   OR POINTER TO NAME BLOCK
                        ;
                        ; ERRORS:
                        ;   ER.OPN - OPERAND EXPECTED BUT NONE FOUND
                        ;
                        ; CALLS:
                        ;   CARTYP - CHECK 1ST CHAR
                        ;   FNDSPC - GET SPECIAL VAR
                        ;   GETVAR - GET VARIABLE NAME
                        ;   GETNUM - GET NUMBER VALUE
                        ;   GETSTR - GET LITERAL STRING
```

```
                        ;
                        ;-----
019D'                   POPND:
019D'  CD 0000:0C               CALL    FNDSPC          ;CHECK SPECIALS FIRST
01A0'  D0                       RNC                     ;RETURN IF SO
01A1'  CD 0000:04               CALL    CARTYP          ;CHECK 1ST CHAR
01A4'  3807                     JRC     ..PNUM          ;DIGIT? NUMBER
01A6'  C2 01B5'                 JNZ     GETVAR          ;VARIABLE?
01A9'  CD 0000:12               CALL    GETSTR          ;STRING?
01AC'  D0                       RNC                     ;YES? RETURN
01AD'  CD 0000:11       ..PNUM: CALL    GETNUM          ;TRY NUMBER
01B0'  D0                       RNC                     ;YES? RETURN
                                ERROR   ER.OPN          ;OPERAND EXPECTED HERE!
01B1'  CD 0000:0A       +       CALL ERRPGM
01B4'  14               +       .BYTE ER.OPN
                        +]
                        ;++++
                        ;
                        ; GETVAR
                        ; PARSES A VARIABLE NAME AND RETURNS POINTER
                        ; TO NAME BLOCK
                        ;
                        ; NEEDS:
                        ;   HL -> VARIABLE NAME (1ST LETTER)
                        ;
                        ; RETURNS:
                        ;   VARIABLE:
                        ;       A = TYPE NAME ($NAME)
                        ;       HL -> AFTER VARIABLE NAME
                        ;       DE -> TOP OF NAME BLOCK
                        ;       CARRY BIT CLEAR FOR VARIABLE
                        ;   FUNCTION CALL:
                        ;       HL -> AFTER ARGUMENT LIST
                        ;       IY -> AFTER FUNCTION CALL CODE
                        ;       DE DESTROYED
                        ;       CARRY BIT SET FOR FUNCTION CALL
                        ;
                        ; CALLS:
                        ;   NAMSET - PARSE NAME
                        ;   NAMGET - LOOK UP NAME
                        ;   NAMBLD - BUILD NAME IF NOT THERE
                        ;   FUNCTION - GENERATE FUNCTION CALL
                        ;
                        ;-----
01B5'                   GETVAR:
01B5'  CD 0000:1A               CALL    NAMSET          ;PARSE NAME
01B8'  7E                       MOV     A,M             ;A=CHAR AFTER NAME
01B9'  FE28                     CPI     '('             ;PAREN? IS FUNCTION
01BB'  280A                     JRZ     ..GFUN          ;GENERATE FUNCTION CALL
01BD'  FE20                     CPI     ' '             ;BLANK AFTER NAME?
01BF'  C2 0000:18               JNZ     NAMGET          ;NO, JUST A VAR
01C2'  CD 0000:0F               CALL    FUNCTION        ;GENERATE FUNCTION
01C5'  180D                     JMPR    ..GF            ;AND EXIT
                        ;+
                        ; COME HERE WHEN A FUNCTION NAME IS FOUND
```

```
                        ; HL -> OPEN PAREN AFTER NAME
                        ; DE -> NAME BLOCK OF FUNCTION NAME
                        ; WE LOOK UP NAME AND GENERATE CODE TO EVALUATE
                        ; ALL FUNCTION ARGUMENTS
                        ;-
01C7'   CD 0000:0F      ..GFUN: CALL    FUNCTION                ;GENERATE FUNCTI
                        ON CALL
01CA'   7E                      MOV     A,M             ;CHECK DELIMITER
01CB'   FE29                    CPI     ')'             ;FOUND PAREN
                                NZERROR ER.PAR          ;NO? NEED ONE![
01CD'   2804            +       JRZ     ..0004
01CF'   CD 0000:0A      +       CALL    ERRPGM
01D2'   2A              +       .BYTE ER.PAR
01D3'                   +..0004:]
01D3'   23                      INX     H               ;HL-> AFTER PAREN
01D4'   37              ..GF:   STC                     ;SET CARRY
01D5'   C9                      RET
                        ;++++
                        ;
                        ; CHKAZ
                        ; CHECKS FOR INTEGER VAR NAME
                        ;
                        ; NEEDS:
                        ;   HL -> ALPHABETIC
                        ;
                        ; RETURNS:
                        ;   CARRY SET - NOT ALPHABETIC
                        ;   Z BIT SET - A-Z FOLLOWED BY DELIMITER
                        ;   ELSE A-Z FOLLOWED BY ALPHANUMERIC
                        ;
                        ; CALLS:
                        ;   CARTYP - CHECK TYPE OF CHAR
                        ;
                        ;----
01D6'                   CHKAZ:
01D6'   CD 0000:04              CALL    CARTYP          ;ALPHABETIC
01D9'   D8                      RC                      ;NO? FAIL THEN
01DA'   37                      STC                     ;SET FAIL RC
01DB'   C8                      RZ                      ;FAIL IF DELIM TOO
01DC'   23                      INX     H               ;HL->AFTER A-Z
01DD'   CD 0000:04              CALL    CARTYP          ;CHECK FOR DELIM
01E0'   2B                      DCX     H               ;RESTORE PTR
                                CLC                     ;SUCCEED[
01E1'   B7              +       ORA     A]
01E2'   C9                      RET
                        ;++++
                        ;
                        ; GETOPR
                        ; PARSES OPERATOR AND RETURNS PTR TO EVALUATION
                        ; ROUTINE (IN IX)
                        ;
                        ; NEEDS:
                        ;   HL -> OPERATOR (+,- CURRENTLY)
                        ;
                        ; RETURNS:
```

```
                        ;    SUCCEED:
                        ;        HL -> AFTER OPERATOR
                        ;        IX -> ROUTINE TO DO THAT OPERATION
                        ;        CARRY BIT CLEAR
                        ;    FAIL:
                        ;        CARRY BIT SET
                        ;        FAILS IF THE OPERATOR IS NOT VALID
                        ;
                        ;----
01E3'                   GETOPR:
01E3'   CD 0000:04              CALL    CARTYP          ;IS A DELIM?
01E6'   2802                    JRZ     ..G1            ;YES? CONTINUE
01E8'   37                      STC                     ;FAIL
01E9'   C9                      RET
01EA'   C5              ..G1:   PUSH    B
01EB'   D5                      PUSH    D
                        ;+
                        ; HERE WE ATTEMPT TO GET A DOUBLE CHARACTER OPERATOR
                        ; INTO BC. IF THE 2ND CHAR IS NOT A DELIMITER IT IS
                        ; ASSUMED TO BE A SINGLE CHARACTER OPERATOR
                        ;-
01EC'   4E                      MOV     C,M             ;C=1ST CHAR
01ED'   23                      INX     H               ;HL->AFTER 1ST BYTE
01EE'   CD 0000:04              CALL    CARTYP          ;CHECK 2ND CHAR
01F1'   2011                    JRNZ    GSIN            ;ALPHANUM? SINGLE THEN
01F3'   46                      MOV     B,M             ;BC=DOUBLE CHAR OP
01F4'   11 0000:09              LXI     D,DOPTAB        ;DE->DOUBLE OP TABLE
01F7'   CD 0211'                CALL    FNDOPR          ;SEARCH IT
01FA'   3808                    JRC     GSIN            ;NOT FOUND? TRY SINGLE
01FC'   23                      INX     H               ;HL->AFTER OPERATOR
                                CLC                     ;SUCCEED[
01FD'   B7              +       ORA     A]
01FE'                    GEX:   MVD     X,D             ;GET OFFSET IN X[
01FE'   D5              +.IFIDN [X] [X], [      PUSH    D
01FF'   DDE1            +       POP     X
                        +]
0201'   D1                      POP     D
0202'   C1                      POP     B
0203'   C9                      RET
                        ;+
                        ; HERE WE ATTEMPT TO FIND A SINGLE CHARACTER
                        ; OPERATOR IN A TABLE. WE KNOW IT IS NOT A DOUBLE
                        ; SO IF IT IS NOT A SINGLE WE FAIL RETURN
                        ; C = CHARACTER TO CHECK
                        ;-
0204'   11 0000:1E      GSIN:   LXI     D,SOPTAB        ;TRY SINGLE TABLE
                        ;+
                        ; SRCHOPR - ENTRY POINT FOR UNOPR
                        ;-
0207'                   SRCHOPR:CLR     B               ;2ND CHAR NULL[
LL 0207'   0600         +.IFN B         -A?, ?[ MVI     B               ,0][
                        XRA     A]]
0209'   CD 0211'              CALL    FNDOPR          ;SEARCH TABLE
020C'   30F0                  JRNC    GEX             ;IF FOUND, EXIT
020E'   2B                    DCX     H               ;RESTORE PTR
```

```
020F'   18ED                JMPR    GEX             ;FAIL EXIT
                    ;++++
                    ;
                    ; FNDOPR
                    ; SEARCHES THE GIVEN OPERATOR TABLE FOR A SPECIFIED
                    ; OPERATOR AND RETURNS ITS ENTRY OFFSET
                    ;
                    ; NEEDS:
                    ;   BC = OPERATOR TO SEARCH FOR (2 CHARS)
                    ;        B = HI ORDER CHAR (NULL FOR 1 CHAR OPS)
                    ;        C = LO ORDER CHAR
                    ;   DE -> TOP OF OPERATOR TABLE TO SEARCH
                    ;        UOPTAB - UNARY OPERATORS
                    ;        SOPTAB - SINGLE CHAR BINARY OPERATORS
                    ;        DOPTAB - DOUBLE CHAR BINARY OPERATORS
```

```
 .LINK
 .IDENT EA
 .PREL
 .INSERT A:S.ASM
@.REMARK /
@                       *
@                     * * *
@                       ***
@                     *******
@                         *
@                        *
@                       *
@                      *
@                     *
@                     *******
@
@     "WHEN YOU CARE ENOUGH TO PROGRAM
@                 THE VERY BEST"
@
@              ZGRASS V3.00000000
@BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
@                   (C) 1978
@
@/
 .INSERT A:MAC.ASM
@
 .INSERT A:ZRAM.ASM
 .INSERT A:NCUEQU.ASM
 ;+
 ; INTERNALS
 ;-
 .INTERN ASSIGN             ;ASSIGNMENT
 .INTERN ASSINP             ;ENTRY FOR INPUT
 .INTERN CNVFD              ;CONVERT FLT TO DOUBLE
 .INTERN CNVFI              ;CONVERT FLT TO INT
 .INTERN CNVID              ;CONVERT INT TO DOUBLE
 .INTERN CNVIF              ;CONVERT INT TO FLT
 .INTERN CVDSTR             ;CONVERT TO STRING
 .INTERN CVFSTR             ;CONVERT FRACTION
 .INTERN DONCU              ;PERFORM NCU OPERATION
 .INTERN FPOP               ;POP FLT FROM NCU
 .INTERN FPUSH              ;PUSH FLT ONTO NCU
 .INTERN GETMSB             ;GET MOST SIGNIF BYTE
 .INTERN IGET               ;GET INTEGER OFF NCU STACK
 .INTERN IPOP               ;POP INT OFF NCU STACK
 .INTERN IPSHO              ;PUSH 0 INT ON NCU STACK
 .INTERN IPUSH              ;PUSH INT ONTO NCU STACK
 .INTERN IYNCU              ;IY STACK -> NCU
 .INTERN NCUIY              ;NCU -> IY STACK
 .INTERN PSHF10             ;PUSH 10.0
 ;+
 ; EXTERNALS
 ;-
 .EXTERN ALSTR              ;ALLOCATE STRING
 .EXTERN CONV               ;CONVERSION
```

```
                        .EXTERN CONVI           ;CONVERSION
                        .EXTERN ENDSTR          ;CLEAN UP STRING ALLOC
                        .EXTERN ERRPGM          ;ERROR
                        .EXTERN GETLOC          ;GET LOCAL NAME
                        .EXTERN GETOPND         ;GET OPERAND
                        .EXTERN GETVAL          ;GET VALUE
                        .EXTERN INCUSE          ;BUMP USE COUNT
                        .EXTERN POPOPND         ;POP OPERAND
                        .EXTERN PSHOPND         ;PUSH OPERAND
                        .EXTERN PUTOPND         ;PUT OPERAND
                        .EXTERN RMDR            ;COMPUTE REMAINDER
                        .EXTERN SDEL            ;DELETE STRING
                        .EXTERN XCGVAL          ;ASSIGN VALUE
                        ;++++
                        ;
                        ; ASSIGN
                        ; ASSIGNS OPERAND B'S VALUE TO OPERAND A
                        ;
                        ; NEEDS:
                        ;   IY->OPERAND B (NAME OR VALUE)
                        ;       OPERAND A (NAME ONLY)
                        ;
                        ; RETURNS:
                        ;   IY->OPERAND A (NAME)
                        ;   DESTROYS DE, BC AND HL
                        ;
                        ; ERRORS:
                        ;   ER.ASN - CANNOT ASSIGN TO NON-NAMED THING
                        ;
                        ; CALLS:
                        ;   POPOPND - POP OPERAND
                        ;   XCGVAL - DELETE OLD VALUE, ASSIGN NEW
                        ;   SDEL - DELETE STRING
                        ;   CONV - CONVERT OPERAND
                        ;   FPOP - POP OFF NCU STACK
                        ;
                        ;----
0001'                   ASSIGN:
0001'  CD 0000:0B               CALL    GETVAL          ;GET VALUE OF B
0004'  FD7EFB                   MOV     A,E.TYP-E.SIZ(Y) ;A=TYPE OF OPNDA
0007'  FD66FD                   MOV     H,E.HVAL-E.SIZ(Y) ;HL->NAME OR ADDR
000A'  FD6EFC                   MOV     L,E.LVAL-E.SIZ(Y)
000D'  CD 0000:09       ASSINP: CALL    GETLOC          ;GET LOCAL NAME
0010'  FE0A                     CPI     $NAME           ;MUST BE A NAME
0012'  2006                     JRNZ    ..ADDR          ;TRY SPECIAL VAR
0014'  CD 0000:0D               CALL    POPOPND         ;POP IT OFF
0017'  C3 0000:12               JMP     XCGVAL          ;ASSIGN NEW VAL
001A'  E5               ..ADDR: PUSH    H               ;SAVE ADDR
001B'  0604                     MVI     B,$IVAL         ;ASSUME INT
001D'  FE05                     CPI     $ADDRI          ;INTEGER ADDR
001F'  2823                     JRZ     ..AINT          ;DO ASSIGNMENT
0021'  0606                     MVI     B,$FVAL         ;ASSUME FLT
0023'  FE07                     CPI     $ADDRF          ;FLOAT ADDR
0025'  2828                     JRZ     ..AFLT
0027'  0608                     MVI     B,$STRADR       ;TRY STRING
```

```
0029'  FE09                    CPI     $ADDRS          ;STRING ADDR
                               NZERROR ER.ASN          ;CANNOT DO IT[
002B'  2804            +       JRZ ..0001
002D'  CD 0000:08      +       CALL ERRPGM
0030'  15              +       .BYTE ER.ASN
0031'                  +..0001:]
0031'                   ..ASTR:
0031'  5E                      MOV     E,M             ;HL->OLD STRING
0032'  23                      INX     H
0033'  56                      MOV     D,M
0034'  D5                      PUSH    D               ;SAVE OLD STRING
0035'  CD 0000:05              CALL    CONV            ;CONVERT TO STRING
0038'  CD 0000:0D              CALL    POPOPND         ;POP IT OFF
003B'  CD 0000:0C              CALL    INCUSE          ;BUMP USE COUNT
003E'  E1                      POP     H               ;HL->OLD STRING
003F'  CD 0000:11              CALL    SDEL            ;DELETE IT
0042'  1806                    JMPR    ..AEX           ;EXIT
0044'  CD 0000:05      ..AINT: CALL    CONV
0047'  CD 0000:0D              CALL    POPOPND
004A'  E1              ..AEX:  POP     H
004B'  73                      MOV     M,E             ;PUT VALUE IN
004C'  23                      INX     H
004D'  72                      MOV     M,D
004E'  C9                      RET
004F'  CD 0000:05      ..AFLT: CALL    CONV            ;CONVERT TO FLT
0052'  CD 0000:0D              CALL    POPOPND         ;GET FLT VAL
0055'  E1                      POP     H               ;HL -> WHERE TO STORE
0056'  C3 0097'                JMP     FPOP            ;STORE VALUE
                       ;++++
                       ;
                       ; NCUIY
                       ; POPS FLOATING POINT OPERAND OFF NCU STACK
                       ; AND STORES ON IY STACK (DOES NOT PUSH)
                       ;
                       ; NEEDS:
                       ;   IY-> WHERE TO STORE OPERAND
                       ;
                       ;----
0059'  DB3E            NCUIY:  IN      NCUDAT          ;MOST SIGNIFICANT
005B'  FD7704                  MOV     E.VAL+3(Y),A    ;BYTE IS STORED
005E'  DB3E                    IN      NCUDAT          ;FIRST, E.VAL+3
0060'  FD7703                  MOV     E.VAL+2(Y),A
0063'  DB3E                    IN      NCUDAT
0065'  FD7702                  MOV     E.VAL+1(Y),A
0068'  DB3E                    IN      NCUDAT
006A'  FD7701                  MOV     E.VAL(Y),A
006D'  C9                      RET
                       ;++++
                       ;
                       ; IYNCU
                       ; PUSHES FLOATING POINT OPERAND ONTO NCU STACK
                       ; FROM IY STACK (DOES NOT POP)
                       ;
                       ; NEEDS:
                       ;   IY->OPERAND TO PUSH
```

```
                        ;
                        ;-----
006E'  FD7E01           IYNCU:  MOV     A,E.VAL(Y)      ;LEAST SIGNIFICANT
0071'  D33E                     OUT     NCUDAT          ;BYTE IS PUSHED
0073'  FD7E02                   MOV     A,E.VAL+1(Y)    ;FIRST, E.VAL
0076'  D33E                     OUT     NCUDAT
0078'  FD7E03                   MOV     A,E.VAL+2(Y)
007B'  D33E                     OUT     NCUDAT
007D'  FD7E04                   MOV     A,E.VAL+3(Y)
0080'  D33E                     OUT     NCUDAT
0082'  FD7E00                   MOV     A,E.TYP(Y)
0085'  C9                       RET
                        ;++++
                        ;
                        ; GETMSB
                        ; GETS THE MOST SIGNIFICANT BYTE FROM THE TOP OF
                        ; THE NCU STACK AND PUTS IT IN ACCUMULATOR
                        ;
                        ; RETURNS:
                        ;   A = MOST SIGNIFICANT BYTE OF TOS
                        ;   (EXPONENT FOR FLOATING NUMBERS)
                        ;
                        ;-----
0086'  DB3E             GETMSB: IN      NCUDAT          ;GET MSB
0088'  D33E                     OUT     NCUDAT          ;LEAVE ON STACK
                                TST     A               ;SET SIGN BITS[
008A'  B7               +       ORA     A]
008B'  C9                       RET
                        ;++++
                        ;
                        ; FPUSH
                        ; PUSHES A FLOATING POINT NUMBER FROM MEMORY
                        ; ONTO THE NCU STACK. THE LEAST SIGNIFICANT
                        ; BYTE IS ASSUMED TO BE AT THE LOWEST ADDRESS
                        ;
                        ; NEEDS:
                        ;   HL -> FLOATING POINT NUMBER TO PUSH
                        ;       (LEAST SIG BYTE)
                        ;   DESTROYS HL
                        ;
                        ; CALLS:
                        ;   IYNCU - PUSH FLT NUMBER
                        ;
                        ;-----
008C'  2B               FPUSH:  DCX     H               ;FAKE OUT TYPE
008D'  E5                       PUSH    H
008E'  FDE3                     XTIY                    ;IY->NUMBER TO PUSH
0090'  CD 006E'                 CALL    IYNCU           ;PUSH ON NCU STACK
0093'  FDE3                     XTIY
0095'  E1                       POP     H
0096'  C9                       RET
                        ;++++
                        ;
                        ; FPOP
                        ; POPS A FLOATING POINT NUMBER OFF THE NCU STACK
```

```
                        ; INTO MEMORY (LEAST SIGNIFICANT BYTE FIRST)
                        ;
                        ; NEEDS:
                        ;   HL -> WHERE TO STORE NUMBER
                        ;       (LEAST SIGNIFICANT BYTE)
                        ;   DESTROYS HL
                        ;
                        ; CALLS:
                        ;   NCUIY - POP OFF FLT NUMBER
                        ;
                        ;----
0097'  2B               FPOP:   DCX     H               ;FAKE OUT TYPE
0098'  E5                       PUSH    H
0099'  FDE3                     XTIY                    ;IY->WHERE TO STORE
009B'  CD 0059'                 CALL    NCUIY           ;POP OFF NCU STACK
009E'  FDE3                     XTIY
00A0'  E1                       POP     H
00A1'  C9                       RET
                        ;++++
                        ;
                        ; PSHF10
                        ; PUSHES A FLOATING POINT 10 ONTO THE STACK OF
                        ; THE NUC
                        ;
                        ; DESTROYS DE
                        ;
                        ; CALLS:
                        ;   IPUSH - PUSHES SINGLE INTEGER
                        ;
                        ;----
00A2'  CD 00E2'         PSHF10: CALL    IPSH0   ;LO ORDER IS 0
00A5'  11 04A0                  LXI     D,04A0H         ;HI ORDER
00A8'  C3 00E5'                 JMP     IPUSH
                        ;++++
                        ;
                        ; DONCU
                        ; DOES A SPECIFIC NCU OPERATION
                        ; WILL WAIT FOR COMPLETION
                        ;
                        ; NEEDS:
                        ;   A = WHICH NCU OPERATION
                        ;
                        ; RETURNS:
                        ;   TOP OF NCU STACK HAS RESULT, OPERANDS WHICH
                        ;   WERE ON NCU STACK BEFORE OPERATION MAY BE POPPED
                        ;
                        ;----
00AB'  D33F             DONCU:  OUT     NCUCOM          ;TELL WHICH OPERATION
00AD'  DB3F             ..WAIT: IN      NCUCOM          ;CHECK STATUS
00AF'  A7                       ANA     A
00B0'  FA 00AD'                 JM      ..WAIT          ;UNTIL SIGN BIT OFF
00B3'  CB4F                     BIT     N.OFLO,A        ;ARITHMETIC OVERFLOW?
                                NZERROR ER.OFLOI
00B5'  2804             +       JRZ     ..0002
00B7'  CD 0000:08       +       CALL    ERRPGM
```

```
00BA'   17          +         .BYTE ER.OFLO
00BB'               +..0002:]
00BB'   CB57                  BIT     N.UFLO,A        ;ARITHMETIC UNDERFLOW?
                              NZERROR ER.UFLO[
00BD'   2804        +         JRZ ..0003
00BF'   CD 0000:08  +         CALL ERRPGM
00C2'   33          +         .BYTE ER.UFLO
00C3'               +..0003:]
00C3'   CB5F                  BIT     N.ARG,A         ;ILLEGAL ARGUMENT?
                              NZERROR ER.ARG[
00C5'   2804        +         JRZ ..0004
00C7'   CD 0000:08  +         CALL ERRPGM
00CA'   34          +         .BYTE ER.ARG
00CB'               +..0004:]
00CB'   CB67                  BIT     N.DIV,A         ;DIVIDE BY 0?
                              NZERROR ER.DIV[
00CD'   2804        +         JRZ ..0005
00CF'   CD 0000:08  +         CALL ERRPGM
00D2'   18          +         .BYTE ER.DIV
00D3'               +..0005:]
00D3'   E660                  ANI     N.SIGN+N.ZERO   ;CLEAR ERROR BITS
00D5'   2003                  JRNZ    ..D1            ;NO POS? TRY NEG,0
00D7'   3E01                  MVI     A,1             ;RETURN PLUS
00D9'   C9                    RET
00DA'   FE40          ..D1:   CPI     N.SIGN          ;NEGATIVE?
00DC'   3EFF                  MVI     A,-1            ;LETS ASSUME SO
00DE'   C8                    RZ                      ;YES, IT WAS
                              CLR     A               ;MUST BE 0[
00DF'   AF          +.IFN A           -A, [   MVI     A               ,0][
                     XRA      A]]
00E0'   C9                    RET
00E1'   C9                    RET
                     ;++++
                     ;
                     ; IPUSH, IPSH0
                     ; PUSHES A SINGLE-PRECISION FIXED POINT INTEGER ONTO
                     ; THE TOP OF THE NCU STACK.  IPSH0 PUSHES A 0
                     ;
                     ; NEEDS:
                     ;   DE = INTEGER TO PUSH
                     ;
                     ;----
00E2'   11 0000      IPSH0:   LXI     D,0             ;WE DO THIS A LOT
00E5'   7B           IPUSH:   MOV     A,E             ;LO ORDER
00E6'   D33E                  OUT     NCUDAT          ;TO STACK
00E8'   7A                    MOV     A,D             ;HI ORDER
00E9'   D33E                  OUT     NCUDAT          ;TO STACK
00EB'   C9                    RET
                     ;++++
                     ;
                     ; IGET, IPOP
                     ; FETCHES A SINGLE-PRECISION FIXED POINT INTEGER
                     ; FROM THE NCU STACK.  IPOP POPS IT OFF, IGET
                     ; LEAVES STACK ASIS
                     ;
```

```
                        ; RETURNS:
                        ;   DE = INTEGER FROM TOP OF NCU STACK
                        ;
                        ;----
00EC'  3E77             IGET:   MVI     A,N.PTOS
00EE'  D33F                     OUT     NCUCOM
00F0'  00                       NOP                     ;TIME FOR NCU
00F1'  00                       NOP                     ;TO DO IT
00F2'  DB3E             IPOP:   IN      NCUDAT
00F4'  57                       MOV     D,A             ;HI ORDER
00F5'  DB3E                     IN      NCUDAT
00F7'  5F                       MOV     E,A             ;LO ORDER
00F8'  C9                       RET
                        ;++++
                        ;
                        ; CNVIF
                        ; CONVERTS INTEGER OPERAND TO FLOATING
                        ;
                        ; NEEDS:
                        ;  IY-> INTEGER OPERAND
                        ;
                        ; RETURNS:
                        ;  IY-> FLOATING OPERAND
                        ;
                        ; CALLS:
                        ;   GETOPND - GET INT OPND
                        ;   IPUSH - PUSH ON NCU STACK
                        ;   PUTOPND - PUT FLT OPND ON IY STACK
                        ;   DONCU - DO CONVERSION
                        ;
                        ;----
00F9'  CD 0000:0A       CNVIF:  CALL    GETOPND         ;GET INT
00FC'  CD 00E5'                 CALL    IPUSH           ;PUSH ON NCU STACK
00FF'  3E1D                     MVI     A,N.FLTS        ;CONVERT TO FLT
0101'  CD 00AB'                 CALL    DONCU
0104'  3E06                     MVI     A,$FVAL         ;RETURN TYPE FLT
0106'  C3 0000:0F               JMP     PUTOPND         ;SAVE IT
                        ;++++
                        ;
                        ; CNVFI
                        ; CONVERT FLOATING OPERAND TO INTEGER
                        ;
                        ; NEEDS:
                        ;  IY-> FLOATING OPERAND
                        ;
                        ; RETURNS:
                        ;  IY-> INTEGER OPERAND
                        ;
                        ; CALLS:
                        ;   GETOPND - GET FLT OPND
                        ;   PUTOPND - SAVE INT OPND ON IY STACK
                        ;   DONCU - DO CONVERSION
                        ;
                        ;----
0109'  CD 0000:0A       CNVFI:  CALL    GETOPND         ;GET FLT OPND
```

FLOATING TO INTEGER CONVERSION

```
010C'   3E1F                MVI     A,N.FIXS        ;CONVERT TO INT
010E'   CD 00AB'            CALL    DONCU
0111'   CD 00F2'            CALL    IPOP            ;GET IN DE
0114'   3E04                MVI     A,$IVAL         ;RETURN TYPE INT
0116'   C3 0000:0F          JMP     PUTOPND         ;SAVE IT
                    ;++++
                    ;
                    ; CNVID
                    ; CONVERT INTEGER OPERAND TO DOUBLE PRECISION (32 BIT)
                    ;  INTEGER
                    ;
                    ; NEEDS:
                    ;   IY-> SINGLE PRECISION INTEGER
                    ;
                    ; RETURNS:
                    ;   IY-> DOUBLE PRECISION INTEGER
                    ;
                    ;----
0119'   FD7E02      CNVID:  MOV     A,E.HVAL(Y)     ;GET HI BYTE OF INT
                            TST     A               ;CHECK SIGN[
011C'   B7          +       ORA     A]
011D'   3EFF                MVI     A,-1            ;ASSUME NEGATIVE
011F'   FA 0123'            JM      ..C1
                            CLR     A               ;WAS POSITIVE[
0122'   AF          +.IFN A         -A, [   MVI     A                ,0][
                     XRA    A]]
0123'   FD7703        ..C1:   MOV     E.VAL+2(Y),A    ;PAD HI BYTES
0126'   FD7704                MOV     E.VAL+3(Y),A
0129'   FD360000              MVI     E.TYP(Y),$DVAL
012D'   C9                  RET
                    ;++++
                    ;
                    ; CNVFD
                    ; CONVERT FLOATING POINT OPERAND TO DOUBLE
                    ; PRECISION INTEGER OPERAND
                    ;
                    ; NEEDS:
                    ;   IY-> FLOATING POINT OPERAND
                    ;
                    ; RETURNS:
                    ;   IY-> DOUBLE PRECISION INTEGER OPERAND
                    ;
                    ;----
012E'   CD 0000:0A  CNVFD:  CALL    GETOPND         ;GET FLT OP
0131'   3E1E                MVI     A,N.FIXD        ;CONVERT TO DOUBLE
0133'   CD 00AB'            CALL    DONCU
0136'   3E00                MVI     A,$DVAL         ;TYPE IS DOUBLE
0138'   C3 0000:0F          JMP     PUTOPND         ;PUT IT BACK
                    ;++++
                    ;
                    ; CVDSTR, CVFSTR
                    ; CONVERT DOUBLE INTEGER ON NCU STACK INTO ASCII AND
                    ; STORE IN GIVEN AREA, CVDSTR WILL SUPPRESS LEADING
                    ; ZEROS, CVFSTR WILL SUPPRESS SIGN (ASSUMES POSITIVE)
                    ;
```

```
                        ; NEEDS:
                        ;   HL -> WHERE TO PUT FIRST DIGIT
                        ;   NCU->DOUBLE PRECISION INTEGER
                        ;   B = FIELD WIDTH, USED ONLY FOR LEADING ZEROS
                        ;   DESTROYS DE, B
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER LAST DIGIT
                        ;   ALL ENTRIES ON NCU STACK DESTROYED!
                        ;
                        ; CALLS:
                        ;   GETMSB - GET MOST SIGNIFICANT BYTE
                        ;   DONCU - DO NCU OPERATION
                        ;   IPUSH, IPOP - PUSH, POP SINGLE INTS
                        ;   RMDR - FIND REMAINDER
                        ;
                        ;----
   013B'                CVDSTR: CLR     B               ;NO FIELD[
LL 013B'   0600        +.IFN B         -A?, ?[ MVI     B               ,0][
                        XRA     A]]
   013D'   CD 0086'             CALL    GETMSB          ;GET HI ORDER BYTE
   0140'   F2 014B'             JP      CVFSTR          ;>0? LEAVE IT BE
   0143'   362D                 MVI     M,'-'           ;LEADING MINUS
   0145'   23                   INX     H
   0146'   3E34                 MVI     A,N.CHSD        ;CHANGE SIGN
   0148'   CD 00AB'             CALL    DONCU
                        ;+
                        ; THE STACK IS USED TO SAVE THE DIGITS AS THEY ARE
                        ; COMPUTED FROM RIGHT TO LEFT. THE BOTTOM OF THE STACK
                        ; IS MARKED WITH A NULL. WHEN ALL THE DIGITS ARE
                        ; COMPUTED, THEY ARE POPPED OFF IN REVERSE ORDER AND
                        ; GO INTO THE STRING LEFT TO RIGHT. A DIGIT IS FOUND
                        ; BY DIVIDING THE CURRENT INTEGER BY 10. THE REMAINDER
                        ; IS THE NEXT DIGIT, THE QUOTIENT IS THE NEXT INTEGER.
                        ; WHEN THE QUOTIENT BECOMES 0, ALL DIGITS ARE ON THE
                        ; STACK
                        ;-
   014B'                CVFSTR: CLR     A               ;MARK END OF DIGS[
   014B'   AF           +.IFN A         -A, [   MVI     A               ,0][
                        XRA     A]]
   014C'   F5                   PUSH    PSW             ;ON THE STACK
   014D'   11 000A      ..C2:   LXI     D,10            ;SAVE DOUBLE 10
   0150'   CD 00E5'             CALL    IPUSH
   0153'   CD 00E2'             CALL    IPSHO
   0156'   CD 0000:10           CALL    RMDR            ;REMAINDER
   0159'   F5                   PUSH    PSW             ;SAVE QUOTIENT SIGN
   015A'   CD 00F2'             CALL    IPOP            ;IGNORE HI PART
   015D'   CD 00F2'             CALL    IPOP            ;DE = LO ORDER DIGITS
   0160'   7B                   MOV     A,E             ;NEW DIGIT
   0161'   C630                 ADI     '0'             ;MAKE IT ASCII
   0163'   D1                   POP     D               ;D=SIGN OF QUOTIENT
   0164'   F5                   PUSH    PSW             ;SAVE DIGIT
   0165'   05                   DCR     B               ;COUNT THIS DIGIT
                                TST     D[
LL 0166'   7A           +.IFN D-A?, ?[  MOV     A,D]
```

```
   0167'   B7          +         ORA     A]
   0168'   20E3                  JRNZ    ..C2            ;NOT 0? GO AGAIN
                                 TST     B               ;PUT IN LEADING ZEROS?[
LL 016A'   78          +.IFN B           -A?, ?[ MOV     A,B             ]
   016B'   B7          +         ORA     A]
   016C'   F2 014D'              JP      ..C2            ;YES? CONTINUE
                      ;+
                      ; ALL DIGITS HAVE BEEN COMPUTED. STACK HAS:
                      ; SP -> ASCII DIGIT
                      ;       ...
                      ;       ASCII DIGIT
                      ;       0 MARKER
                      ; WE COPY ALL DIGITS INTO STRING
                      ;-
   016F'   F1          ..C5:     POP     PSW             ;A=ASCII DIGIT
   0170'   77                    MOV     M,A             ;SAVE DIGIT
   0171'   23                    INX     H               ;HL->NEXT SPACE
                                 TST     A               ;WAS THE NULL?[
   0172'   B7          +         ORA     A]
   0173'   20FA                  JRNZ    ..C5            ;NO? COPY MORE
   0175'   2B                    DCX     H               ;POINT AT NULL
   0176'   C9                    RET
                       .END
```

++++++ SYMBOL TABLE ++++++

AASN	005F		ALSTR	0000:04	X	ARGEND	6252		ARGSTK	60CC	
ASSIGN	0001'	I	ASSINP	000D'	I	BACKGR	65CC		BLANK	0020	
BOTRAM	6000		BOTTOM	64A9		CHARSL	65DD		CLEARS	60CA	
CNTRL	000C		CNTRLC	65CD		CNTRLO	65D9		CNTRLU	0015	
CNTRLZ	65E4		CNVFD	012E'	I	CNVFI	0109'	I	CNVID	0119'	I
CNVIF	00F9'	I	CONV	0000:05	X	CONVI	0000:06	X	CPLFLG	6540	
CPLIN	681C		CPLTMP	672C		CPREF	691C		CP.ARG	0001	
CP.MXL	0080		CP.MXR	0014		CP.NO	0002		CR	000D	

```
 .INSERT A:MAC.ASM
@
 .INSERT A:S.ASM
@.REMARK /
@                          *
@                        * * *
@                         ***
@                       *******
@                           *
@                          *
@                         *
@                        *
@                       *
@                      *******
@
@      "WHEN YOU CARE ENOUGH TO PROGRAM
@                 THE VERY BEST"
@
@               ZGRASS V3.00000000
@BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
@                    (C) 1978
@
@/
 .INSERT A:ZRAM.ASM
 .INSERT A:NCUEQU.ASM
 .INSERT A:CPLEQU.ASM
 .LINK
 .IDENT  CPL
 .PREL
 ;+
 ; THESE INTERNALS ARE ONLY HERE BECAUSE OF DEBUGGING
 ; THEY CAN BE KILLED WHEN COMPILE WORKS
 ;-
          .INTERN LINCPL
          .INTERN SAVREF
          .INTERN GETLAB
          .INTERN GETREF
          .INTERN RESOLV
 ;+
 ; INTERNALS
 ;-
 .INTERN CFOR              ;COMPILED FOR
 .INTERN CGO               ;COMPILED GOTO
 .INTERN CIF               ;COMPILED IF
 .INTERN COMPILE           ;COMPILE COMMAND
 .INTERN CNXT              ;COMPILED NEXT
 .INTERN CSKP              ;COMPILED SKIP
 .INTERN XBR               ;XEQ BRANCH
 .INTERN XIF               ;XEQ IF
 .INTERN XINC              ;XEQ INCRMENT (NEXT)
 ;+
 ; EXTERNALS
 ;-
 .EXTERN ALCPL             ;ALLOCATE MEMORY
 .EXTERN ALLOCD            ;RETURN MEMORY FRAGMENT
```

```
.EXTERN ARG             ;PARSE ARGUMENTS
.EXTERN ASSIGN          ;ASSIGN VALUE
.EXTERN BLANX           ;PARSE ARG DELIMS
.EXTERN CARTYP          ;FIND CHAR TYPE
.EXTERN CONVI           ;CONVERT
.EXTERN CORE            ;LINK FOR CODE
.EXTERN DEC5IY          ;DEC IY BY 5
.EXTERN DONCU           ;DO NCU OPERATION
.EXTERN ERRPGM          ;ERROR MESSAGE
.EXTERN EVALC           ;CREATE CODE BLOCK
.EXTERN EVLCPL          ;EVAL COMPILED CODE
.EXTERN FPUSH           ;PUSH FLOATING
.EXTERN FREEALL         ;FREE LOCAL NAMES
.EXTERN FUNCTION        ;COMPILE FUNCTION CALL
.EXTERN GENCOD          ;GENERATE 1 BYTE OF CODE
.EXTERN GETNUM          ;PARSE FLOATING NUM
.EXTERN GETOPND         ;GET OPERAND
.EXTERN GETSKP          ;PARSE NUMBER
.EXTERN GETVAL          ;GET VALUE OF NAME
.EXTERN INC5IY          ;BUMP IY TO NEXT ARG
.EXTERN IPOP            ;POP FROM NCU
.EXTERN IPUSH           ;PUSH ON NCU
.EXTERN KISNL           ;CHECK END DELIM
.EXTERN KLB             ;SKIP BLANKS
.EXTERN NAMADR          ;LOOK UP NAME
.EXTERN NAMBLD          ;CREATE NAME
.EXTERN NAMSET          ;PARSE NAME
.EXTERN NENAME          ;LOOK UP LOCAL NAME
.EXTERN NXTVAL          ;GET NAME VALUE
.EXTERN OPUSH           ;GENERATE PUSH
.EXTERN POPOPND         ;PARSE OPERAND
.EXTERN PSHOPND         ;PUSH OPERAND
.EXTERN RETNONE         ;RETURN FOR CODE
.EXTERN XCGVAL          ;ASSIGN VALUE
;++++
;
; COMPILE
; COMPILES AN ASCII STRING INTO INTERNAL FAST CODE
; AND REPLACES THE STRING
;
; SYNTAX:
;       COMPILE NAME
;   <NAME> - NAME OF AN ASCII STRING TO COMPILE
;
; RETURNS:
;   DE -> NAME PASSED (NOW HAS CODE)
;   A = TYPE NAME ($NAME)
;   NAME IS NOW ASSIGNED TO A COMPILED CODE BLOCK
;
; CALLS:
;   ALCPL - ALLOCATE MEMORY FOR COMPILE BLOCK
;   LINCPL - COMPILE A LINE
;   RESOLV - RESOLVE REFERENCES
;   FREEALL - FREE LOCAL NAMES, LABELS
;   GETOPND - GET STRING NAME
```

```
                        ;    XCGVAL - ASSIGN CODE TO NAME
                        ;    NXTVAL - GET STRING VALUE
                        ;    GENCOD - GENERATE COMPILE CODE
                        ;
                        ; COMPILER BLOCK LOOKS LIKE:
                        ;
                        ;         ********************
                        ;         *     TYPE         *  $CPLADR
                        ;         ********************
                        ;         *     LENGTH       *  LENGTH IN BLOCKS
                        ;         ********************
                        ;         *  FLAGS           * 0
                        ;         ********************
                        ;         *  TEMP LIST       *
                        ;         *  LINKAGE         *
                        ;         ********************
                        ;         *  USE COUNT       *
                        ;         ********************
                        ;         * # OF LOCALS     *  CP.LOC
                        ;         ********************
                        ;         * START OF CODE   *  CP.SIZ
                        ;         *     ....         *
                        ;         *                  *
                        ;         ********************
                        ;
                        ;-----
                                  M.CMD[COMPILE,..CDX,0,CORE,0,..CODE][
0001'   18              +COMPILE:         .BYTE $CMDADR
0002'   0B              +         .BYTE (..CDX-COMPILE)/16+1
0003'   00              +         .BYTE 0
0004'   0000:0B         +         .WORD CORE
0006'   0012'           +         .WORD ..CODE
0008'   0000            +         .WORD 0
000A'   434F4D50494C+    +         .ASCIZ /COMPILE/
                        +]
0012'   FDE5             ..CODE: PUSH    Y               ;IY->STACK AREA
0014'   E5                        PUSH    H               ;HL->EXPRESSION
                        ;+
                        ; HERE WE GET A POINTER TO THE FIRST CHARACTER OF
                        ; THE STRING TO COMPILE. WE LOOK UP THE NAME
                        ; AND GET ITS STRING VALUE. IF NOT A STRING
                        ; OR IF NULL, IT IS AN ERROR
                        ;-
0015'   CD 0000:06                CALL    ARG             ;GET THE NAME
0018'   0A00                      .BYTE   $NAME,$TAF
001A'   CD 0000:16                CALL    GETOPND         ;DE->NAMEBLOCK
001D'   EB                        XCHG                    ;HL->NAMEBLOCK
001E'   E5                        PUSH    H               ;SAVE NAME PTR
001F'   CD 0000:22                CALL    NXTVAL          ;HL->VALUE
                                  ZERROR  ER.CPL          ;0? CANNOT COMPILE[
0022'   2004            +         JRNZ ..0001
0024'   CD 0000:0E       +         CALL ERRPGM
0027'   3B              +         .BYTE ER.CPL
0028'                   +..0001:]
0028'   3E08                      MVI     A,$STRADR       ;IS A STRING?
```

```
002A'   BE                      CMP     M
                                NZERROR ER.CPL          ;NO? CANT COMPILE[
002B'   2804            +       JRZ ..0002
002D'   CD 0000:0E      +       CALL ERRPGM
0030'   3B              +       .BYTE ER.CPL
0031'                   +..0002:]
0031'   11 000A                  LXI     D,$SASCII       ;GET STRING ADDR
0034'   19                      DAD     D               ;HL->1ST CHAR OF STRING
0035'   E5                      PUSH    H               ;SAVE ADDR
                        ;+
                        ; HERE WE ALLOCATE MEMORY FOR THE COMPILED CODE
                        ; BLOCK AND INITIALIZE THE POINTERS
                        ; ICPLIN - INDEX INTO TABLE OF POINTERS TO COMPILED
                        ;   CODE LINES
                        ; ICPREF - INDEX INTO TABLE OF LINE NUMBERS OF LINES
                        ;   WHICH HAVE UNRESOLVED REFERENCES
                        ; CURLIN - LINE NUMBER OF CURRENT LINE
                        ; LOCNUM - LOCAL VARIABLE COUNTER
                        ; CPLFLG - COMPILE MODE FLAG
                        ;-
0036'   CD 0000:04              CALL    ALCPL           ;ALLOCATE CODE BLOCK
0039'   E3                      XTHL                    ;(SP)->BLOCK,HL->STRING
003A'   E5                      PUSH    H               ;SAVE THE PTR
003B'   21 681C                 LXI     H,CPLIN         ;SET UP LINE PTR
003E'   22 6930                 SHLD    ICPLIN
0041'   21 691C                 LXI     H,CPREF         ;SET UF REF PTR
0044'   22 6932                 SHLD    ICPREF
                                CLR     A               ;CURRENT LINE #[
0047'   AF              +.IFN A         -A, [   MVI     A               ,0][
                         XRA    A]]
0048'   32 6934                 STA     CURLIN
004B'   32 6730                 STA     LOCNUM          ;LOCAL VAR COUNTER
004E'   3C                      INR     A               ;SET IT
004F'   32 6540                 STA     CPLFLG          ;COMPILE FLAG
0052'   E1                      POP     H               ;HL->STRING TO COMPILE
                        ;+
                        ; STACK HAS STATUS SO FAR:
                        ; SP->  ADDRESS OF COMPILED BLOCK
                        ;       ADDRESS OF NAME BLOCK TO ASSIGN TO
                        ;       OLD HL'
                        ;       OLD IY
                        ; WE NOW COMPILE ALL THE LINES IN THE STRING
                        ; HL -> 1ST CHARACTER OF STRING
                        ; IY -> 1ST BYTE OF COMPILED CODE AREA
                        ;-
0053'   CD 00C1'        ..CNXT: CALL    LINCPL          ;COMPILE A LINE
0056'   30FB                    JRNC    ..CNXT
                                CLR     A               ;RESET COMPILE FLAG[
0058'   AF              +.IFN A         -A, [   MVI     A               ,0][
                         XRA    A]]
0059'   32 6540                 STA     CPLFLG
005C'   CD 0000:14              CALL    GENCOD          ;TERMINATE CODE
                        ;+
                        ; NOW WE MUST RESOLVE REFERENCES (SKIPS, GOTOS)
                        ; DELETE ALL LOCAL NAMES AND LABELS
```

```
                        ; AND COPY THE LIST OF TEMPS (LITERAL STRINGS,
                        ; ETC) INTO THE HEADER OF THE COMPILED BLOCK.
                        ;-
005F'  CD 0150'                 CALL    RESOLV          ;RESOLVE REFS
0062'  2A 672E                  LHLD    LOCTAB          ;HL->1ST NAME
0065'  CD 0000:12               CALL    FREEALL         ;FREE THEM ALL
0068'  DDE1                     POP     X               ;IX->COMPILE BLOCK
006A'  2A 60CA                  LHLD    TEMPHDR         ;HL->TEMP LIST
006D'  DD7503                   MOV     $NLINK(X),L
0070'  DD7404                   MOV     $NLINK+1(X),H
0073'  3A 6730                  LDA     LOCNUM          ;SAVE # OF LOCS
0076'  DD7706                   MOV     CP.LOC(X),A     ;IN HEADER TOO
0079'  21 0000                  LXI     H,0             ;CLEAR TEMPS
007C'  22 60CA                  SHLD    TEMPHDR
007F'  22 672E                  SHLD    LOCTAB          ;AND LOCALS
                                MVD     H,X             ;HL->BLOCK HDR[
0082'  DDE5             +.IFIDN [X                ] [X], [        PUSH    X
0084'  E1               +       POP     H
                        +]
                                MVD     D,Y             ;DE->AFTER CODE[
0085'  FDE5             +.IFIDN [Y                ] [Y], [        PUSH    Y
0087'  D1               +       POP     D
                        +]
0088'  CD 0000:05              CALL    ALLOCD          ;RETURN FRAGMENT
008B'  EB                       XCHG                    ;DE->TOP OF BLOCK
008C'  3E12                     MVI     A,$CPLADR       ;A=TYPE OF BLOCK
                        ;+
                        ; NOW WE ASSIGN THE COMPILED BLOCK TO THE NAME AND
                        ; RETURN THE NAME AS THE VALUE TO THE CALLER
                        ;-
008E'  E1                       POP     H               ;HL->NAMEBLOCK
008F'  E5                       PUSH    H               ;SAVE AGAIN
0090'  CD 0000:27               CALL    XCGVAL          ;ASSIGN TO NAME
0093'  D1                       POP     D               ;DE->NAMEBLOCK
0094'  21 0000                  LXI     H,0             ;AND CPL WORD
0097'  22 672C                  SHLD    CPLTMP
009A'  3E0A                     MVI     A,$NAME         ;RETURN TYPE NAME
009C'  E1                       POP     H
009D'  FDE1                     POP     Y
009F'  C3 0000:26               JMP     RETNONE
00A2'                   ..CDX:
                        ;++++
                        ;
                        ; NEWLIN
                        ; CHANGES POINTERS AND CONTEXT FOR A NEW LINE
                        ; OF COMPILED CODE
                        ;
                        ; NEEDS:
                        ;   ICPLIN - INDEX INTO LINE POINTER TABLE
                        ;   CURLIN - LINE NUMBER OF CURRENT LINE
                        ;
                        ; RETURNS:
                        ;   CURLIN - BUMPED BY ONE
                        ;   ICPLIN - BUMPED BY 2
                        ;
```

```
                        ;----
00A2'                   NEWLIN:
                        ;+
                        ; THE CURRENT LINE NUMBER IS INCREMENTED, IF THERE
                        ; ARE ALREADY CP.MXL LINES, NO MORE WILL FIT, WE
                        ; PRINT AN ERROR
                        ;-
00A2'   3A 6934                 LDA     CURLIN          ;BUMP CURRENT LINE #
00A5'   3C                      INR     A               ;BUMP IT
00A6'   FE80                    CPI     CP.MXL          ;TOO MANY?
                                NCERROR ER.TML          ;YES? ERROR!
00A8'   3804            +       JRC     ..0003
00AA'   CD 0000:0E      +       CALL    ERRPGM
00AD'   3E              +       .BYTE   ER.TML
00AE'                   +..0003:]
00AE'   32 6934                 STA     CURLIN          ;SAVE NEW LINE #
                        ;+
                        ; THE ADDRESS OF THE COMPILED CODE FOR THIS LINE IS
                        ; SAVED IN THE TABLE CPLIN. ICPLIN IS THE INDEX INTO
                        ; THE TABLE. EACH ENTRY HAS THE ADDRESS OF THE CODE
                        ; COMPILED FOR THAT PARTICULAR LINE
                        ;-
00B1'   E5                      PUSH    H
00B2'   2A 6930                 LHLD    ICPLIN          ;HL->TABLE ENTRY
00B5'                   IFLIN:  MVD     D,Y             ;DE->CODE ADDR!
00B5'   FDE5            +.IFIDN [Y                      ] [Y], [        PUSH    Y
00B7'   D1              +       POP     D
                        +]
00B8'   73                      MOV     M,E             ;SAVE ADDR OF
00B9'   23                      INX     H               ;THIS NEXT LINE
00BA'   72                      MOV     M,D             ;OF CODE
00BB'   23                      INX     H               ;HL->NEXT ENTRY
00BC'   22 6930                 SHLD    ICPLIN          ;SAVE NEW PTR
00BF'   E1                      POP     H
00C0'   C9                      RET
                        ;++++
                        ;
                        ; LINCPL
                        ; COMPILES A SINGLE LINE OF A STRING INTO INTERNAL
                        ; CODE FORMAT
                        ;
                        ; NEEDS:
                        ;   HL -> START OF LINE TO COMPILE
                        ;   IY -> WHERE TO PUT COMPILED CODE
                        ;
                        ; RETURNS:
                        ;   C BIT SET IF NO MORE LINES LEFT
                        ;   HL -> AFTER LINE COMPILED (AT NEXT LINE
                        ;       OR NULL)
                        ;   IY -> AFTER COMPILED CODE
                        ;
                        ; USES:
                        ;   ICPLIN - CURRENT LINE PTR INDEX
                        ;   CURLIN - CURRENT LINE #
                        ;
```

```
                        ; CALLS:
                        ;   GETLAB - PARSE LABEL
                        ;   NEWLIN - SET UP FOR NEW LINE
                        ;   EVALC - COMPILE EXPRESSION
                        ;   KISNL - CHECK FOR TERMINATOR
                        ;   NAMSET - PARSE NAME
                        ;   CARTYP - CHECK CHARACTER TYPE
                        ;   FUNCTION - PARSE FUNCTION CALL
                        ;
                        ;----
   00C1'  CD 00A2'      LINCPL: CALL    NEWLIN
   00C4'  CD 0000:1D    LINXT:  CALL    KLB             ;SKIP BLANX
   00C7'  CD 0000:09            CALL    CARTYP          ;CHECK CHARACTER TYPE
   00CA'  381F                  JRC     ..LAB           ;DIGIT? LABEL THEN
   00CC'  200E                  JRNZ    ..NAM           ;LETTER? NAME!
                        ;+
                        ; IF THE LINE STARTS WITH A DOT '.' IT IS A COMMENT
                        ; COMMENTS ARE IGNORED (BUT STILL COUNTED AS LINES)
                        ;-
                             TST     M               ;IF NULL, SET CARRY[
LL 00CE'  7E            +.IFN M      -A?, ?[ MOV     A,M                ]
   00CF'  B7            +            ORA     A]
   00D0'  37                    STC
   00D1'  C8                    RZ
   00D2'  FE2E                  CPI     '.'             ;STARTS WITH DOT?
   00D4'  CA 025C'              JZ      LINEND          ;SKIP TO END
   00D7'  CD 0000:1C            CALL    KISNL           ;IS TERMY?
   00DA'  2829                  JRZ     LEND            ;LEAVE IT BE THEN
                        ;+
                        ; THE FIRST CHARACTER IS A LETTER, IT IS A COMMAND
                        ; IF IT IS FOLLOWED BY A TERMINATOR, IT IS A COMMAND
                        ; WITH NO ARGUMENTS. OTHERWISE WE TREAT IT AS AN
                        ; EXPRESSION
                        ;-
   00DC'  CD 0000:20    ..NAM:  CALL    NAMSET          ;PARSE THE NAME
   00DF'  CD 0000:1C            CALL    KISNL           ;END OF LINE?
   00E2'  EB                    XCHG                    ;HL->START OF NAME
   00E3'  201D                  JRNZ    ..EVL           ;NOT END, BACKUP
   00E5'  EB                    XCHG                    ;DE->START,HL->END
   00E6'  CD 0000:13            CALL    FUNCTION        ;FUNCTION CALL
   00E9'  181A                  JMPR    LEND            ;END OF LINE
                        ;+
                        ; LABELS ARE PARSED OVER AND PUT INTO THE NAMES TABLE
                        ; THE LINE NUMBER OF THE LINE WHICH CONTAINED THE
                        ; LABEL IS SAVED IN THE VALUE FIELD OF THE NAME
                        ;-
   00EB'  CD 0116'      ..LAB:  CALL    GETLAB          ;PARSE LABEL
   00EE'  E5                    PUSH    H               ;SAVE LINE PTR
   00EF'  21 0005               LXI     H,$NVALUE       ;OFFSET OF VALUE
   00F2'  19                    DAD     D               ;HL->VALUE FIELD
                             TST     M               ;IF NONZERO,[
LL 00F3'  7E            +.IFN M      -A?, ?[ MOV     A,M                ]
   00F4'  B7            +            ORA     A]
                                NZERROR ER.DUP          ;DUPLICATE LABEL[
   00F5'  2804          +            JRZ ..0004
```

```
00F7'  CD 0000:0E  +          CALL ERRPGM
00FA'  3C          +          .BYTE ER.DUP
00FB'              +..0004:]
00FB'  3A 6934               LDA     CURLIN          ;A=LINE #
00FE'  77                    MOV     M,A             ;SAVE IT
00FF'  E1                    POP     H               ;RESTORE PTR
0100'  18C2                  JMPR    LINXT           ;NOW DO LINE
                    ;+
                    ; THE CODE IS COMPILED - THE ONLY THING THAT REMAINS ON
                    ; THE LINE IS AN EXPRESSION (FUNCTION CALL, ETC.)
                    ; THEN THE TERMINATOR IS CHECKED AND WE EXIT
                    ;-
0102'  CD 0000:0F   ..EVL:  CALL    EVALC           ;COMPILE IT
0105'  CD 0000:1D   LEND:   CALL    KLB             ;SKIP BLANX
                            TST     M               ;CHECK FOR END[
LL 0108'  7E        +.IFN M         -A?, ?[ MOV     A,M             ]
0109'  B7           +          ORA     A]
010A'  C8                    RZ
010B'  23                    INX     H               ;HL->AFTER TERMY
010C'  FE3B                  CPI     ';'             ;SEMI? GO AGAIN
010E'  28B4                  JRZ     LINXT
0110'  FE0A                  CPI     NL              ;NEWLINE?
0112'  C8                    RZ                      ;END OF LINE
0113'  2B                    DCX     H               ;BACK IT UP
0114'  18AE                  JMPR    LINXT           ;IS FROM IF
                    ;++++
                    ;
                    ; GETLAB
                    ; PARSES A LABEL AND PUTS IT INTO THE NAMES TABLE
                    ; FOR THE COMPILER
                    ;
                    ; NEEDS:
                    ;   HL -> POSSIBLE LABEL
                    ;
                    ; RETURNS:
                    ;   CARRY SET IF NO LABEL, ELSE
                    ;   HL -> AFTER LABEL
                    ;   DE -> NAMEBLOCK ADDRESS
                    ;
                    ; CALLS:
                    ;   CARTYP - CHECK 1ST CHAR
                    ;   NAMSET - PARSE NAME
                    ;   NAMADR - LOOK UP NAME
                    ;   NAMBLD - CREATE NAME
                    ;   KLB - SKIP BLANX AFTER LABEL
                    ;
                    ;----
0116'  CD 0000:09   GETLAB: CALL    CARTYP          ;IF 1ST CHAR IS
0119'  3011                 JRNC    ..FAIL          ;NOT NUMBER, RET
011B'  CD 0000:20           CALL    NAMSET          ;PARSE LABEL
011E'  21 672B              LXI     H,LOCTAB-$NLINK ;NAMES LISTHEAD
0121'  CD 0000:21           CALL    NENAME          ;LOOK IT UP
0124'  C4 0000:1F           CNZ     NAMBLD          ;CREATE IT
0127'  CD 0000:1D           CALL    KLB             ;SKIP BLANKS
                            CLC                     ;CLEAR CARRY[
```

```
012A'   B7          +         ORA     A]
012B'   C9                    RET
012C'   37           ..FAIL:  STC                     ;SET CARRY
012D'   C9                    RET
                     ;++++
                     ;
                     ; SAVREF
                     ; SAVES THE LINE NUMBER OF THE CURRENT LINE IN THE
                     ; REFERENCE TABLE - AFTER THE FIRST PASS IS MADE OVER
                     ; ALL LINES, ALL REFERENCES WILL BE RESOLVED BY RESOLV
                     ;
                     ; NEEDS:
                     ;   ICPREF - INDEX INTO REFERENCE TABLE
                     ;   CURLIN - NUMBER OF CURRENT LINE
                     ;
                     ; RETURNS:
                     ;   ICPREF - BUMPED BY ONE
                     ;   LINE NUMBER SAVED IN REFERENCE TABLE CPREF
                     ;
                     ;----
012E'   E5           SAVREF:  PUSH    H
012F'   D5                    PUSH    D
0130'   2A 6932               LHLD    ICPREF           ;HL->TABLE ENTRY
                              MVD     D,Y              ;DE->CURRENT ADDR[
0133'   FDE5        +.IFIDN [Y                  ] [Y], [         PUSH    Y
0135'   D1          +         POP     D
                    +]
0136'   1B                    DCX     D                ;POINT AT TOKEN
0137'   73                    MOV     M,E              ;SAVE IT
0138'   23                    INX     H
0139'   72                    MOV     M,D
013A'   23                    INX     H
013B'   22 6932               SHLD    ICPREF           ;SAVE THE PTR
013E'   D1                    POP     D
013F'   E1                    POP     H
0140'   C9                    RET
                     ;++++
                     ;
                     ; GETREF
                     ; LOOKS UP THE ADDRESS OF THE GIVEN LINE IN THE LINE
                     ; TABLE CREATED AT COMPILE TIME
                     ;
                     ; NEEDS:
                     ;   A = NUMBER OF LINE TO FIND
                     ;
                     ; RETURNS:
                     ;   DE -> 1ST BYTE OF CODE COMPILED FOR THAT LINE
                     ;
                     ; USES:
                     ;   CPLIN - TABLE OF POINTERS TO COMPILED CODE FOR
                     ;       LINES OF THE CURRENT MACRO
                     ;
                     ;----
0141'   E5           GETREF:  PUSH    H
0142'   11 681C              LXI     D,CPLIN          ;DE->TABLE OF LINES
```

```
   0145'   3D                   DCR     A               ;MAKE 0 BASED
   0146'   87                   ADD     A               ;WORD OFFSET
   0147'   6F                   MOV     L,A             ;HL=LINE NUMBER
                                CLR     H[
LL 0148'   2600          +.IFN H-A?, ?[  MVI     H,0][   XRA     A]]
   014A'   19                   DAD     D               ;HL->TABLE ENTRY
   014B'   5E                   MOV     E,M             ;DE=ADDRESS OF
   014C'   23                   INX     H               ;COMPILED CODE
   014D'   56                   MOV     D,M             ;FOR THAT LINE
   014E'   E1                   POP     H
   014F'   C9                   RET
                         ;++++
                         ;
                         ; RESOLV
                         ; RESOLVES REFERENCES TO LABELS AND LINES MADE BY SKIPS
                         ; AND GOTOS IN COMPILED CODE
                         ;
                         ; NEEDS:
                         ;   CPLIN - TABLE OF POINTERS TO EACH LINE OF
                         ;       COMPILED CODE
                         ;   CPREF - TABLE OF LINE NUMBERS OF LINES WHICH
                         ;       CONTAIN UNRESOLVED REFERENCES
                         ;
                         ; RETURNS:
                         ;   RESOLVES ALL SKIP,GOTO REFERENCES TO BRANCHES
                         ;   TO SPECIFIC ADDRESSES IN THE COMPILED CODE
                         ;
                         ; CALLS:
                         ;   GETREF - GET LABEL REFERENCE NAME
                         ;   NXTVAL - GET VALUE OF NAME
                         ;
                         ; NOTES:
                         ; GOTO REFERENCES LOOK LIKE:
                         ;       CP.GO
                         ;       PTR TO NAMEBLOCK
                         ;       0
                         ; SKIP REFERENCES LOOK LIKE:
                         ;       CP.SKP
                         ;       LINE # TO SKIP TO
                         ;
                         ; THEY ARE RESOLVED INTO:
                         ;       CP.BR
                         ;       ADDR IN COMPILED CODE TO GO TO
                         ;
                         ;----
   0150'   2A 6932       RESOLV: LHLD    ICPREF          ;HL->NEXT REF SPOT
                                CLR     A[
   0153'   AF            +.IFN A-A, [    MVI     A,0][   XRA     A]]
   0154'   77                   MOV     M,A             ;ZERO LAST ONE
   0155'   23                   INX     H
   0156'   77                   MOV     M,A
   0157'   21 691C              LXI     H,CPREF         ;HL->REF TABLE
   015A'   E5                   PUSH    H               ;SAVE REF PTR
                         ;+
                         ; WE GET THE ADDRESS OF THE NEXT REFERENCE AND
```

```
                        ; FIND THE ADDRESS OF THE COMPILED CODE WITH CONTAIN
                        ; THE REFERENCE. THE TYPE OF REFERENCE (CP.SKP, CP.GO)
                        ; IS REPLACED BY A TRANSFER OF CONTROL CODE CP.BR
                        ;-
   015B'   E1           ..NXT:  POP     H               ;HL->NEXT REFERENCE
   015C'   5E                   MOV     E,M             ;DE->LINE WITH REF
   015D'   23                   INX     H
   015E'   56                   MOV     D,M
   015F'   23                   INX     H
                                TSTD    D               ;IF 0, NO MORE[
   0160'   7A           +       MOV     A,D
   0161'   B3           +       ORA     D               +1]
   0162'   C8                   RZ
   0163'   E5                   PUSH    H               ;SAVE REF PTR
   0164'   EB                   XCHG                    ;HL->LINE WITH REF
   0165'   7E                   MOV     A,M             ;A=TYPE (SKIP,GOTO)
   0166'   23                   INX     H               ;HL->REF INFO
                        ;+
                        ; IF IT IS A GOTO (CP.GO) THE NEXT 2 BYTES HAVE THE
                        ; ADDRESS OF THE NAME WHICH CONTAINS THE LABEL TO
                        ; GO TO. THE VALUE FIELD HAS THE LINE NUMBER OF
                        ; THE LINE WHOSE ADDRESS WE WANT
                        ;-
   0167'   FE08                 CPI     CP.GO           ;GOTO?
   0169'   2019                 JRNZ    ..SKP           ;NO? TRY SKIP
   016B'   E5                   PUSH    H
   016C'   5E                   MOV     E,M             ;DE=ADDR OF NAME
   016D'   23                   INX     H
   016E'   56                   MOV     D,M
   016F'   EB                   XCHG                    ;HL->NAME NODE
   0170'   CD 0000:22           CALL    NXTVAL          ;HL=LABEL NUMBER
                                TST     L               ;CHECK LINE #[
LL 0173'   7D           +.IFN L          -A?, ?[ MOV    A,L                 ]
   0174'   B7           +       ORA     A]
                                ZERROR  ER.LAB          ;0? WAS NO LABEL[
   0175'   2004         +       JRNZ ..0005
   0177'   CD 0000:0E   +       CALL ERRPGM
   017A'   25           +       .BYTE ER.LAB
   017B'                +..0005:]
   017B'   E1                   POP     H               ;HL->WHERE TO PUT ADDR
                        ;+
                        ; HERE WE HAVE THE LINE NUMBER OF A LINE TO GO TO
                        ; WE GET THE ADDRESS OF THE CORRESPONDING COMPILED
                        ; CODE LINE AND REPLACE THE NAMEBLOCK ADDRESS (GOTO)
                        ; OR LINE NUMBER (SKIP) WITH THE CODE LINE ADDRESS
                        ;-
   017C'   CD 0141'     ..GOX:  CALL    GETREF          ;DE->CODE LINE
   017F'   73                   MOV     M,E             ;SAVE ADDRESS
   0180'   23                   INX     H
   0181'   72                   MOV     M,D
   0182'   18D7                JMPR    ..NXT           ;TRY NEXT LINE
                        ;+
                        ; IF THE REFERENCE IS TO A SKIP, WE GET THE LINE
                        ; NUMBER TO SKIP TO. IF IT IS LARGER THAN THE
                        ; LAST LINE COMPILED, WE JUST SET IT TO THE END
```

```
                        ; OF THE CODE (SO SK 99 WORKS EVEN IN A 5 LINE PROGRAM)
                        ; IF NOT A SKIP OR GOTO, IS IMPOSSIBLE ERROR
                        ;-
0184'  FE07             ..SKP:  CPI     CP.SKP          ;SKIP REF?
0186'  2808                     JRZ     ..SK1
0188'  FE04                     CPI     CP.IF           ;IF IS LIKE SKIP
                                NZERROR ER.IMP          ;NO! CANT HAPPEN[
018A'  2804             +       JRZ ..0006
018C'  CD 0000:0E       +       CALL ERRPGM
018F'  03               +       .BYTE ER.IMP
0190'                   +..0006:]
0190'  3A 6934          ..SK1:  LDA     CURLIN          ;A=LAST LINE #
0193'  BE                       CMP     M               ;IS LARGER?
0194'  FA 017C'                 JM      ..GOX           ;SKIP TO LAST
0197'  7E                       MOV     A,M             ;A=LINE # TO SKIP
0198'  18E2                     JMPR    ..GOX
                        ;++++
                        ;
                        ; CGO
                        ; COMPILE-TIME GOTO PROCESSOR, SAVES THE REFERENCE
                        ; AND STORES THE ADDRESS OF THE LABLE NAME IN
                        ; THE COMPILED CODE BLOCK
                        ;
                        ; RETURNS:
                        ;       CP.GO
                        ;       ADDR OF NAMEBLOCK
                        ;
                        ;----
019A'  CD 012E'         CGO:    CALL    SAVREF          ;SAVE REF LINE #
019D'  CD 0116'                 CALL    GETLAB          ;DE->NAME OF LABEL
                                CERROR  ER.LBS          ;ILLEGAL LABEL[
01A0'  3004             +       JRNC ..0007
01A2'  CD 0000:0E       +       CALL ERRPGM
01A5'  3F               +       .BYTE ER.LBS
01A6'                   +..0007:]
                        ;++++
                        ;
                        ; STUFDE
                        ; SAVES THE CONTENTS OF DE IN COMPILED CODE
                        ;
                        ; NEEDS:
                        ;   IY -> WHERE TO SAVE CODE
                        ;   DE = 2 BYTES TO SAVE (E FIRST, THEN D)
                        ;
                        ; RETURNS:
                        ;   IY -> AFTER LAST BYTE SAVED (D)
                        ;
                        ; CALLS:
                        ;   GENCOD - SAVE ONE BYTE
                        ;
                        ;----
01A6'  7B               STUFDE: MOV     A,E             ;STORE LABEL NAME
01A7'  CD 0000:14               CALL    GENCOD          ;ADDR IN THE CODE
01AA'  7A                       MOV     A,D
01AB'  C3 0000:14               JMP     GENCOD
```

```
                        ;++++
                        ;
                        ; CSKP
                        ; COMPILE-TIME SKIP PROCESSOR, SAVE THE REFERENCE LINE
                        ; AND STORES THE LINE NUMBER TO SKIP TO IN THE COMPILED
                        ; CODE
                        ;
                        ;       SKIP <INTEGER>
                        ;
                        ;   <INTEGER> - ANY POSITIVE OR NEGATIVE INTEGER GIVING
                        ;       THE NUMBER OF LINES TO SKIP (FORWARD,BACKWARD)
                        ;
                        ; GENERATES:
                        ;       CP.SKP
                        ;       LINE # TO (2 BYTES) SKIP TO
                        ;       0
                        ; CLOBBERS DE,BC
                        ;
                        ;----
   01AE'                CSKP:   CLR     B[
LL 01AE'   0600         +.IFN B-A?, ?[  MVI     B,0][   XRA     A]]
   01B0'   CD 0000:17           CALL    GETSKP          ;GET NUMBER
   01B3'   58                   MOV     E,B             ;E = SKIP NUM
   01B4'   CD 012E'     IFSKP:  CALL    SAVREF          ;SAVE REFERENCE
   01B7'   3A 6934              LDA     CURLIN          ;A=LINE # OF LINE
                                CLR     D[
LL 01BA'   1600         +.IFN D-A?, ?[  MVI     D,0][   XRA     A]]
   01BC'   83                   ADD     E               ;A=LINE TO SKIP TO
   01BD'   5F                   MOV     E,A             ;D=LINE TO GOTO
   01BE'   F2 01A6'             JP      STUFDE          ;>0? OK THEN
   01C1'   1E01                 MVI     E,1             ;IF <0, DEFAULT TO 1
   01C3'   18E1                 JMPR    STUFDE          ;AND EXIT
                        ;++++
                        ;
                        ; CIF
                        ; COMPILE-TIME IF PROCESSOR, GENERATES THE CODE TO TEST
                        ; ARGUMENT TO IF AND A SKIP AROUND THE LINE AFTER THE IF

                        ;
                        ;
                        ;       IF <CONDITION>,ZGRASS STATEMENTT
                        ;
                        ;   <CONDITION> - ANY ZGRASS EXPRESSION, PREFERABLY ONE
                        WHICH
                        ;       EVALUATES TO TRUE (1) OR FALSE (0) AND CONTAINS
                        ;       A COMPARISON
                        ;
                        ; GENERATES:
                        ;       CP.IF
                        ;       CODE TO EVALUATE <CONDITION>
                        ;       0
                        ;       CP.SKP
                        ;       1
                        ;       0
                        ;
```

```
                        ; CALLS:
                        ;   IFSKP - GENERATE SKIP ADDR
                        ;   EVALC - COMPILE CONDITION CODE
                        ;   BLANX - SKIP TO NEXT ARG
                        ;
                        ;----
01C5'  1E01             CIF:    MVI     E,1
01C7'  CD 01B4'                 CALL    IFSKP           ;GENERATE SKIP ADDR
01CA'  CD 0000:0F               CALL    EVALC           ;AND TEST CASE
01CD'  C3 0000:08               JMP     BLANX           ;SKIP COMMA
                        ;++++
                        ;
                        ; XBR
                        ; EXECUTION TIME BRANCH ROUTINE, BRANCHES TO THE
                        ; GIVEN ADDRESS IN COMPILED CODE
                        ;
                        ; NEEDS:
                        ;       CP.BR
                        ;       ADDR OF LINE TO GO TO
                        ;
                        ; RETURNS:
                        ;   IX -> NEW PLACE IN COMPILED CODE
                        ;
                        ;----
01D0'  DD5E00           XBR:    MOV     E,0(X)          ;DE->LINE TO GOTO
01D3'  DD23                     INX     X
01D5'  DD5600                   MOV     D,0(X)
                                MVD     X,D             ;IX->NEW LINE[
01D8'  D5               +.IFIDN [X] [X], [      PUSH    D
01D9'  DDE1             +       POP     X
                        +]
01DB'  C9                       RET
                        ;++++
                        ;
                        ; XIF
                        ; EXECUTION TIME IF PROCESSOR, EVALUATE THE EXPRESSION
                        ; AND IF IT IS TRUE EXECUTES THE LINE AFTER THE IF
                        ;
                        ; NEEDS:
                        ;       CP.IF
                        ;       CODE FOR EXPRESSION
                        ;       0
                        ;       CP.BR
                        ;       ADDR TO SKIP TO
                        ;
                        ; RETURNS:
                        ;   IX -> THE BRANCH CODE AFTER THE IF (FALSE)
                        ;       THE CODE AFTER THE IF (AFTER THE BRANCH, TRUE)
                        ;
                        ; CALLS:
                        ;   EVLCPL - EVALUATE COMPILED CODE
                        ;   POPOPND - POP OPERAND
                        ;   CONVI - CONVERT VALUE
                        ;
                        ;----
```

```
01DC'   DD5E00      XIF:    MOV     E,0(X)          ;GET BRANCH ADDR
01DF'   DD23                INX     X
01E1'   DD5600              MOV     D,0(X)
01E4'   DD23                INX     X
01E6'   D5                  PUSH    D               ;SAVE IT
01E7'   CD 0000:10          CALL    EVLCPL          ;STACK HAS RESULT
01EA'   3E04                MVI     B,$IVAL         ;CONVERT TO INTEGER
01EC'   CD 0000:0A          CALL    CONVI
01EF'   CD 0000:24          CALL    POPOPND         ;DE=RESULT
                            TSTD    D               ;IF 0, RETURN[
01F2'   7A          +       MOV     A,D
01F3'   B3          +       ORA     D               +1]
01F4'   D1                  POP     D               ;DE = BRANCH ADDR
01F5'   C0                  RNZ                     ;WAS FALSE
                            MVD     X,D             ;GO TO IT[
01F6'   D5          +.IFIDN [X] [X], [      PUSH    D
01F7'   DDE1        +       POP     X
                    +]
01F9'   C9                  RET
```

```
;++++
;
; CFOR
; COMPILE-TIME FOR PROCESSOR, GENERATES THE CODE
; FOR THE 'FOR' STATEMENT:
;
;       FOR <ASSIGNMENT>,<CONDITION>,<INCREMENT>
;       BODY OF LOOP (ANY STATEMENTS)
;       NEXT <NAME>
;
;   <ASSIGNMENT> - ANY ZGRASS ASSIGNMENT STATEMENT,
;       THE <NAME> PART HERE MUST BE THE SAME AS THE
;       <NAME> IN THE NEXT IN ORDER FOR IT TO WORK
;   <CONDITION> - ANY ZGRASS EXPRESSION, PREFERABLY
;       IT DOES A COMPARISON AND EVALUATES TO TRUE
;       (1) OR FALSE (0)
;   <INCREMENT> - ANY ZGRASS EXPRESSION WHICH EVALUATES
;       TO A FLOATING POINT NUMBER, IT WILL BE ADDED
;       TO THE <NAME> IN THE NEXT STATEMENT
;   <NAME> - ANY NAME, ARRAY ELEMENT, SPECIAL VAR
;       (ANYTHING WHICH MAY BE ASSIGNED INTO)
;
; RETURNS:
;       CODE FOR <ASSIGNMENT>
;       0
;       CP.IF (IF STATEMENT)
;       LINE # OF LINE AFTER NEXT (2 BYTES)
;       CODE FOR <CONDITION>
;       0
;       CODE FOR STATEMENTS IN BODY
;       OF FOR LOOP (EACH NULL TERMINATED)
;       CODE TO PUSH <NAME> IN NEXT STATEMENT
;       CODE TO EVALUATE <INCREMENT>
;       CP.INC (INCREMENT)
;       CP.BR (BRANCH)
;       ADDRESS OF IF STATEMENT (2 BYTES)
```

```
                        ;       O
                        ;
                        ; NOTES:
                        ;   IF THE <NAME> USED IN THE NEXT AND ASSIGNMENT
                        ;   DO NOT MATCH IT'S TUFF TITS BECAUSE THERE IS NO WAY
                        ;   TO CHECK AT COMPILE TIME. (CONSIDER THE CASE OF AN
                        ;   ARRAY ELEMENT WHICH IS ONLY KNOWN AT EXECUTION
                        ;   TIME) NESTED FOR'S WILL WORK, ERROR MESSAGE ARE
                        ;   ISSUED IF THE NEXT'S AND FOR'S ARE NOT MATCHED
                        ;
                        ; CALLS:
                        ;   CLRIY - CLEAR IND ON IY & BACKUP
                        ;   CIF - COMPILE IF CODE
                        ;   GENCOD - GENERATE 1 BYTE OF CODE
                        ;   BLANX - GET NEXT ARG
                        ;   EVALC - GENERATE COMPILED CODE
                        ;   KISNL - CHECK END OF LINE
                        ;   LINCPL - COMPILE ONE LINE
                        ;   LINEND - PROCESS END OF LINE
                        ;   IPUSH,IPOP - PUSH POP OFF NCU
                        ;   OPUSH - GENERATE PUSH OF OPERAND
                        ;   STUFDE - SAVE DE IN COMPILED CODE
                        ;   FPUSH - PUSH FLOATING NUMBER ON NCU
                        ;
                        ; USES:
                        ;   CURLIN - LINE # OF CURRENT LINE
                        ;
                        ;-----
01FA'                   CFOR:
                        ;+
                        ; WE BACK UP OVER THE CP.FOR AND GENERATE THE CODE
                        ; FOR THE <ASSIGNMENT> PART. THEN THE CODE FOR THE IF
                        ; <CONDITION> IS GENERATED
                        ;-
01FA'  CD 0268'                 CALL    CLRIY           ;DEC IY, CLEAR
01FD'  CD 0000:0F               CALL    EVALC           ;GENERATE ASSIGN CODE
0200'  CD 0000:08               CALL    BLANX           ;HL->CONDITION
0203'  FDE5                     PUSH    Y               ;SAVE ADDR OF IF
0205'  3E04                     MVI     A,CP.IF         ;GENERATE IF
0207'  CD 0000:14               CALL    GENCOD          ;SAVE IT
020A'  CD 01C5'                 CALL    CIF             ;FOR CONDITION
020D'  E5                       PUSH    H               ;SAVE INC ADDR
020E'  CD 025C'                 CALL    LINEND          ;END OF LINE
0211'  FD23                     INX     Y               ;END OF EXPRESSION
                        ;+
                        ; WE PARSE LINES UNTIL THE NEXT STATEMENT HAS BEEN FOUND

                        ; IF THERE IS A NEXT, THE LAST TOKEN WILL BE CP.NXT
                        ; (BEFORE THE ENDING NULL OF THE LINE) ANY OTHER FOR/NEX
                        T
                        ; PAIRS WILL HAVE BEEN PARSED OVER BY NOW
                        ;-
0213'  CD 00C1'         ..NXT:  CALL    LINCPL          ;DO A STATEMENT
                                CERROR  ER.NXT          ;MISSING NEXT![
0216'  3004             +       JRNC    ..0008
```

```
0218'  CD 0000:0E  +          CALL ERRPGM
021B'  44          +          .BYTE ER.NXT
021C'              +..0008:]
021C'  FD7EFE                 MOV     A,-2(Y)          ;CHECK FOR NEXT
021F'  FE06                   CPI     CP.NXT           ;IF NOT, JUST
0221'  20F0                   JRNZ    ..NXT            ;CONTINUE
0223'  FD2B                   DCX     Y                ;IY->NULL
0225'  CD 0268'               CALL    CLRIY            ;DEC IY, CLEAR
                       ;+
                       ; THE NEXT HAS BEEN FOUND, IT HAS ALREADY GENERATED
                       ; THE CODE TO PUSH THE <NAME> PART. WE MUST GET
                       ; THE POINTER TO THE <INCREMENT> EXPRESSION AND
                       ; GENERATE THE CODE FOR IT NOW. IF THERE IS NONE,
                       ; A DEFAULT VALUE OF 1.0 IS PUSHED
                       ;-
0228'  E3                     XTHL                     ;HL->INCR EXPR
0229'  CD 0000:1C             CALL    KISNL            ;IS THERE ONE?
022C'  200D                   JRNZ    ..INC            ;YES? PARSE IT
022E'  21 029B'               LXI     H,FONE           ;HL->1.0
0231'  CD 0000:11             CALL    FPUSH            ;PUSH ON NCU
0234'  3E06                   MVI     A,$FVAL          ;FLOATING TYPE
0236'  CD 0000:23             CALL    OPUSH            ;GEN PUSH CODE
0239'  1805                   JMPR    ..BR             ;GENERATE BRANCH
023B'  CD 0000:0F  ..INC:     CALL    EVALC            ;GEN INC CODE
023E'  FD2B                   DCX     Y                ;SKIP THE NULL
0240'  E1          ..BR:      POP     H                ;RESTORE HL
0241'  3E06                   MVI     A,CP.INC         ;AND INC IT
0243'  CD 0000:14             CALL    GENCOD
                       ;+
                       ; NOW THE BRANCH BACK TO THE IF STATEMENT IS GENERATED
                       ; ON THE STACK WE SAVED THE ADDRESS OF THE CP.IF, WE
                       ; STUFF THIS IN THE BRANCH ADDRESS AND STUFF THE LINE
                       ; NUMBER OF THE LINE AFTER THE NEXT INTO THE IF'S
                       ; DISPATCH FIELD (2 BYTES AFTER THE CP.IF, SEE CIF)
                       ;-
0246'  3E05                   MVI     A,CP.BR          ;BRANCH TO
0248'  CD 0000:14             CALL    GENCOD           ;THE CONDITION
024B'  D1                     POP     D                ;DE->IF TEST
024C'  CD 01A6'               CALL    STUFDE           ;SAE IT
024F'  13                     INX     D                ;DE->IF LINE #
0250'  FD23                   INX     Y                ;LEAVE A NULL
0252'  3A 6934                LDA     CURLIN           ;A=CURRENT LINE #
0255'  3C                     INR     A                ;NEXT LINE #
0256'  12                     STAX    D                ;SAVE FOR IF
                              TST     M                ;IF NULL, EXIT[
LL 0257'  7E          +.IFN M          -A?, ?[ MOV      A,M                ]
0258'  B7             +        ORA     A]
0259'  C8                     RZ
025A'  2B                     DCX     H                ;BACK OVER NEWLINE
025B'  C9                     RET
                       ;++++
                       ;
                       ; LINEND
                       ; SKIPS TO THE BEGINNING OF THE NEXT LINE
                       ;
```

```
                        ; NEEDS:
                        ;   HL -> ANYTHING
                        ;
                        ; RETURNS:
                        ;   HL -> START OF NEXT LINE OR NULL AT THE
                        ;       END OF THE STRING IF NO NEXT LINE
                        ;
                        ;----
  025C'                 LINEND: TST     M               ;IF NULL, RETURN[
LL 025C'  7E           +.IFN M          -A?, ?[ MOV     A,M             ]
  025D'  B7             +         ORA     A]
  025E'  C8                       RZ
  025F'  23                       INX     H               ;SKIP TO NEXT ONE
  0260'  FE0A                     CPI     NL              ;NEWLINE?
  0262'  C8                       RZ                      ;RETURN THEN
  0263'  FE3B                     CPI     ';'             ;SEMI?
  0265'  C8                       RZ                      ;RETURN THEN
  0266'  18F4                     JMPR    LINEND          ;KEEP CHECKIN'
                        ;++++
                        ;
                        ; CLRIY
                        ; LITTLE ROUTINE TO BACK UP IY ONE BYTE ANDCLEAR WHERE
                        ;  IT POINTS, THIS IS SO MANY INSTRUCTIONS THAT I MADE
                        ;  IT A SUBROUTINE, ONLY USED IN COMPILER CODE
                        ;
                        ;----
  0268'  FD36FF00       CLRIY:  MVI     -1(Y),0         ;CLEAR BACK 1
  026C'  FD2B                     DCX     Y               ;DEC PTR
  026E'  C9                       RET
                        ;++++
                        ;
                        ; CNXT
                        ; COMPILE-TIME NEXT PROCESSOR, GENERATES THE CODE TO PUS
                        H
                        ;  THE VARIABLE NAME OF THE NEXT FOLLOWED BY A CP.NXT TOK
                        EN
                        ;  TO TELL CFOR THE NEXT HAS BEEN FOUND
                        ;
                        ;       NEXT <NAME>
                        ;
                        ;   <NAME> - ANY ZGRASS NAME, SOMETHING WHICH MAY BE
                        ;       ASSIGNED TO
                        ;
                        ; GENERATES:
                        ;       CODE TO PUSH <NAME>
                        ;       CP.NXT (NEXT TOKEN)
                        ;       0
                        ;
                        ; CALLS:
                        ;   EVALC - GENERATE COMPILED CODE
                        ;   CLRIY - CLEAR IND ON IY & BACKUP
                        ;
                        ;----
  026F'  CD 0268'       CNXT:   CALL    CLRIY           ;DEC IY, CLEAR
  0272'  CD 0000:OF             CALL    EVALC           ;GENERATE VAR PUSH
```

```
0275'  FD36FF06            MVI     -1(Y),CP.NXT    ;NEXT FOR FOR
0279'  C9                  RET
                    ;+++++
                    ;
                    ; XINC
                    ; EXECUTION-TIME INCREMENT ROUTINE - USED ONLY BY
                    ; FOR/NEXT TO ADD THE INCREMENT TO THE NEXT VARIABLE
                    ;
                    ; NEEDS:
                    ;  IY-> INCREMENT TO ADD
                    ;       NAME TO ADD IT TO
                    ;
                    ; RETURNS:
                    ;  IY-> NAME WITH INCREMENT ADDED
                    ;
                    ; CALLS:
                    ;   PSHOPND,POPOPND - PUSH,POP OPERAND
                    ;   GETOPND - GET OPERAND
                    ;   GETVAL - GET VALUE OF NAME
                    ;   DONUC - DO NCU OPERATION
                    ;   ASSIGN - ASSIGN VALUE TO NAME
                    ;
                    ;-----
027A'  0606         XINC:   MVI     B,$FVAL         ;MAKE SURE IT'S
027C'  CD 0000:0A           CALL    CONVI           ;FLOATING POINT
027F'  CD 0000:24           CALL    POPOPND         ;GET INC ON NCU
0282'  CD 0000:16           CALL    GETOPND         ;FETCH THE NAME
0285'  CD 0000:25           CALL    PSHOPND         ;PUSH THE NAME
0288'  CD 0000:18           CALL    GETVAL          ;CONVERT TO VAL
028B'  CD 0000:24           CALL    POPOPND         ;VALUE ON NCU
028E'  3E10                 MVI     A,N.FADD        ;ADD THEM
0290'  CD 0000:0D           CALL    DONCU
0293'  3E06                 MVI     A,$FVAL         ;PUSH RESULT
0295'  CD 0000:25           CALL    PSHOPND
0298'  C3 0000:07           JMP     ASSIGN          ;ASSIGN TO NAME
029B'  0000    0180   NE:   .WORD   0H,0180H       ;FLOATING 1.0
                    .END
```