```
*H FLOATING POINT
BINARY FLOATING POINT REPRESENTATION

        24 BIT SIGN-MAGNITUDE NORMALIZED FRACTION.
        8 BIT EXCESS-128 EXPONENT (BASE 2)
        SIGN BIT REPLACES MOST SIGNIFICANT FRACTION BIT (IMPLIED "1")
        EXPONENT FIELD OF ZERO IMPLIES VALUE IS ZERO

        7F FF FF FF = 16777215/16777216*2↑127 = 1.7014117*10↑38
        00 00 00 81 = 1/2*2↑1 = 1.0000000
        00 00 00 01 = 1/2*2↑(-127) = 2.9387359*10↑(-39)
        XX XX XX 00 = 0
        FF FF FF FF = -16777215/16777216*2↑127 = -1.7014117*10↑38

        ALL RESULTS ARE ROUNDED TO 24 BIT FRACTIONS.

DECIMAL FLOATING POINT REPRESENTATION:

        13 BCD DIGIT SIGN-MAGNITUDE NORMALIZED FRACTION
        8 BIT EXCESS-128 EXPONENT (BASE 10)
        SIGN BIT IS MOST SIGNIFICANT DIGIT
        EXPONENT FIELD OF ZERO IMPLIES VALUE IS ZERO

        09 99 99 99 99 99 99 FF = .99999 99999 999 * 10↑127
        01 00 00 00 00 00 00 81 = 1.0
        01 00 00 00 00 00 00 01 = .1 * 10↑(-127)
        XX XX XX XX XX XX XX 00 = 0
        89 99 99 99 99 99 99 FF = -.99999 99999 999 * 10↑127

        ALL RESULTS ARE TRUNCATED TO 13 DIGIT FRACTIONS.

CALLING SEQUENCE:

        IX POINTS TO FIRST BYTE OF FIRST ARGUMENT
        IY POINTS TO SECOND ARGUMENT (IF ANY)
        RESULT REPLACES FIRST ARGUMENT, IX IS UNCHANGED
        VALUE OF SECOND ARGUMENT IS UNCHANGED
        NO OTHER REGISTERS PRESERVED (INCL. ALTERNATE BANK)

STANDARD FUNCTIONS:

        SUB     X=X-Y
        ADD     X=X+Y
        MULT    X=X*Y
        DIVI    X=X/Y
        RND     X=RND  PSUEDO-RANDOM NUMBER 0<=X<1
        RANDMZ         RANDOMIZES RND SEQUENCE
        INT     X=INT(X) FLOOR FUNCTION
        ABS     X=ABS(X) ABSOLUTE VALUE
        SGN     X=IF X=0 THEN 0 ELSE IF X>0 THEN 1 ELSE -1
        NEG     X=-X
        POW     X=X↑Y
        SQR     X=SQR(X)
        EXP     X=EXP(X)
        LOG     X=LOG(X)  NATURAL LOG
        TAN     X=TAN(X) X IN RADIANS
        COS     X=COS(X) X IN RADIANS
        SIN     X=SIN(X) X IN RADIANS
        ATN     X=ATN(X)  -PI/2 <= RESULT <= PI/2
```

```
RELATIONAL FUNCTIONS (RETURN 1 FOR TRUE, 0 FOR FALSE):

        EQ      X=(X=Y)
        GE      X=(X>=Y)
        GT      X=(X>Y)
        LE      X=(X<=Y)
        LT      X=(X<Y)
        NE      X=(X<>Y)

BOOLEAN FUNCTIONS (READ NONZERO AS TRUE, ZERO AS FALSE):

        AND     X=X.AND.Y
        OR      X=X.OR.Y
        NOT     X=.NOT.X

SPECIAL FUNCTIONS:

        RSUB    X=Y)-X
        REC     X=1/X
        BRK     X=X-INT(X)
        ONE     X=1
        ZERO    X=0
        INF     X=IF X>=0 THEN (LARGEST POS. #) ELSE (LARGEST NEG. #)
*H INTEGERS
INTEGER REPRESENTATION:

        16 BIT 2'S COMPLEMENT
        REVERSED BYTES (LOW ORDER BYTE AT LOWER ADDRESS)

        FF 7F = 32767
        01 00 = 1
        00 00 = 0
        FF FF = -1
        00 80 = -32768

CALLING SEQUENCE:

        IX POINTS TO FIRST BYTE OF FIRST ARGUMENT
        IY POINTS TO SECOND ARGUMENT (IF ANY)
        RESULT REPLACES FIRST ARGUMENT
        IX IS UNCHANGED
        NO OTHER REGISTERS PRESERVED

STANDARD FUNCTIONS:

        IADD    X=X+Y
        ISUB    X=X-Y
        IMUL    X=X*Y
        IDIV    X=X/Y
        IABS    X=ABS(X)
        ISGN    X=IF X=0 THEN 0 ELSE IF X<0 THEN -1 ELSE 1

RELATIONAL AND BOOLEAN FUNCTIONS (SEE FLOATING POINT DEFINITIONS):

        IEQ,IGE,IGT,ILE,ILT,INE
        IAND,IOR,INOT

SPECIAL FUNCTIONS (SEE FLOATING POINT DEFINITIONS):

        IONE,IZERO,INEG
```

```
*H CONVERSION
CONVERSION FUNCTIONS:

        FIX         CONVERTS X FROM FLOATING POINT TO INTEGER FORMAT
        FLOAT       CONVERTS X FROM INTEGER TO FLOATING POINT FORMAT
                         (NOTE THAT 4 OR 8 BYTE RESULT REPLACES 2 BYTE ARG)
        DTOB        CONVERTS ASCII CHARS TO FLOATING POINT NUMBER
                         (NOTE SEE LISTING FOR CALLING SEQUENCE)
        BTOD        CONVERTS FLOATING POINT NUMBER TO ASCII STRING
                         (NOTE SEE LISTING FOR CALLING SEQUENCE)

MODULES:

        BMATH0,BMATH1,BMATH2,BMATHM
        DMATH0,DMATH1,DMATH2,DMATHM
                          FILE RECORD RECORD FILE  STARTING DATE OF   DATE OF
 FILENAME          DRIVE  TYPE COUNT  LENGTH PROPS ADDRESS  CREATION  LAST MOD.

 DMATH0.S            2     A   182     0080                 770801    780718
 DMATH2.S            2     A   147     0080                 770801    780830
 DMATH1.S            2     A   127     0080                 770919    780718
 DMATH0.OBJ          2     B    28     0080                 770801    780718
 DMATH1.OBJ          2     B    18     0080                 770801    780718
 DMATH2.OBJ          2     B    19     0080                 770801    780830
 DMATHM.S            2     A     8     0080                 770914    780718
 DMATHM.OBJ          2     B     3     0080                 770914    780718
 BMATH0.OBJ          2     B    22     0080                 770802    780718
 BMATH1.OBJ          2     B    19     0080                 770802    780718
 BMATH2.OBJ          2     B    22     0080                 770802    780718
 MATHLIB.S           2     A    35     0080                 770802    780718
 BNSRCH.S            2     A     8     0080                 770912    780801
 BMATH1.S            2     A   156     0080                 770802    780718
 BMATH0.S            2     A   147     0080                 770802    780718
 BMATH2.S            2     A   134     0080                 770802    780718
 MATH.DOC            2     A    26     0080                 ......    780718
 DIV3.S              2     A    14     0080                 770815    780828
 BMATHM.S            2     A     8     0080                 770914    780719
 BMATHM.OBJ          2     B     3     0080                 770920    780719
 CAL.S               2     A   120     0080                 780718    780828
 ADM.S               2     A     6     0080                 780719    780719
 BEEHIVE.S           2     A     6     0080                 780719    780719
 CAL.CMD             2     A     1     0080                 770920    780719
 INFOTON.S           2     A     6     0080                 780719    780719

    26 FILES EXAMINED
    25 FILES LISTED
  1290 TOTAL SECTORS FOR LISTED FILES
```

```
*H ADD AND SUBTRACT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT REVERSE SUBTRACT ROUTINE
;    NEGATE X AND ADD
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
RSUB    LD   A,(IX)
        XOR  80H
        LD   (IX),A      ; SET NEW SIGN
        LD   H,A
        LD   D,(IY)
        JP   AD05
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT SUBTRACT ROUTINE
;    NEGATE Y AND ADD
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SUB     LD   A,(IY)      ; GET SIGN OF Y
        XOR  80H         ; CHANGE IT
        LD   D,A
        LD   H,(IX)
        JP   AD05
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT ADD ROUTINE
;
;    INPUT:   IX - FIRST ARG & RESULT
;             IY - SECOND ARG
;    OUTPUT:  IX - SUM
;    ERRORS:  OVERFLOW, UNDERFLOW
;    REGISTERS FOR ADDITION:
;        H'L'HL  ARG WITH LARGER EXP
;        D'E'DE  2ND ARG
;        B'      BIT 7 IS SIGN OF RESULT
;        C'      LARGER EXPONENT
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; GET MS BYTES
;
ADD     LD   H,(IX)
        LD   D,(IY)
;
; CHECK FOR ZERO ARGUMENTS
;
AD05    XOR  A           ; CLEAR A
        CP   (IX+3)
        JP   NZ,AD06     ; X IS NONZERO
        CP   (IY+3)
        RET  Z           ; Y IS ALSO ZERO
        LD   A,D         ; GET Y MS BYTE
        PUSH IY
        POP  HL
        CALL LD          ; COPY Y TO X
        LD   (IX),A      ; STORE MS BYTE
        RET
AD06    LD   (IX),H      ; STORE NEW X SIGN
        CP   (IY+3)      ;    IN CASE Y IS ZERO
```

```
        RET  Z          ; Y IS ZERO, RETURN X
;
; LOAD ARGUMENTS, TEST SIGNS
;       H'L'H IS X FRACTION
;       D'E'D IS Y FRACTION
;       L     IS X EXPONENT
;       E     IS Y EXPONENT
;       B'    IS X SIGN
;       C'    IS Y SIGN
;
AD07    LD   A,H        ; EXTRACT SIGN
        AND  80H
        LD   B,A
        SET  7,H        ; RESTORE IMPLIED 1
        LD   A,D
        AND  80H
        LD   C,A
        SET  7,D
        XOR  B
        EX   AF,AF'     ; SAVE XOR OF SIGNS
        LD   L,(IX+1)
        LD   E,(IY+1)
        EXX             ; MAIN BANK
        LD   H,(IX+2)
        LD   D,(IY+2)
        LD   L,(IX+3)
        LD   E,(IY+3)
;
; PUT LARGER EXP IN HL
; SHIFT DE RIGHT TO MAKE EXP'S EQUAL
;
AD10    LD   A,L
        CP   E
        JR   NC,AD20    ; NO SWAP NEEDED
        EX   DE,HL      ; SWAP X AND Y
        EXX             ; ALT BANK
        EX   DE,HL
        LD   B,C        ; B' IS SIGN OF LARGER
        EXX             ; MAIN BANK
AD20    CALL ALIGN      ; ALIGN FRACTIONS
;
; DECIDE WHETHER TO ADD OR SUBTRACT
;
        EX  AF,AF'      ; RETRIEVE XOR OF SIGNS
        JP   M,SB20     ; SIGNS ARE DIFFERENT
;
; ADD
;
        ADD  HL,DE      ; ADD FRACTIONS
        LD   DE,80H     ; ROUNDING CONSTANT
        EXX             ; ALT BANK
        ADC  HL,DE
;
; NORMALIZE (AT MOST 1 SHIFT RIGHT)
;
        JP   NC,FROUND  ; ALREADY NORMALIZED
        INC  C          ; INCR EXPONENT
        JP   Z,OVERB    ; OVERFLOW
        RR   H          ; SHIFT RESULT RIGHT
        RR   L
```

```
        EXX             ; MAIN BANK
        RR   H
        RR   L
        EXX             ; ALT BANK
        JP   FROUND
;
; SUBTRACT
;
SB20    AND  A
        SBC  HL,DE      ; SUBTRACT FRACTIONS
        LD   DE,80H     ; PREPARE FOR ROUNDING
        EXX             ; ALT BANK
        SBC  HL,DE
        JP   NC,SB30    ; X WAS > Y
;
; IF RESULT IS NEGATIVE, NEGATE HL
;
        LD   A,80H      ; CHANGE SIGN
        XOR  B
        LD   B,A
        EXX             ; MAIN BANK
        EX   DE,HL
        LD   HL,0
        AND  A
        SBC  HL,DE      ; FORM NEGATIVE
        LD   DE,80H     ; PREPARE FOR ROUND
        EXX             ; ALT BANK
        EX   DE,HL
        LD   HL,0
        SBC  HL,DE
;
; POST NORMALIZE BY SHIFTING LEFT
;
SB30    JP   M,FROUND   ; ALREADY NORMALIZED
        EXX             ; MAIN BANK
        LD   B,24       ; PREPARE FOR POST-SHIFT
SB40    ADD  HL,HL      ; SHIFT RESULT LEFT
        EXX             ; ALT BANK
        DEC  C          ; DECR EXPONENT
        JP   Z,UNDER    ; UNDERFLOW
        ADC  HL,HL
        JP   M,FROUND   ; NORMALIZED
        EXX             ; MAIN BANK
        DJNZ SB40       ; LOOP NOT DONE
        JP   ZERO
;
; ALIGN ARGUMENT IN DE
;
;   SHIFT RIGHT UNTIL EXPONENTS EQUAL
;   RETURN X EXP IN C'
;       DE IS ALIGNED FRACTION
;       L IS ZERO
;
ALIGN   LD   A,L        ; GET X EXP
        LD   L,0        ; CLEAR LSBYTE OF X
        EXX             ; ALT BANK
        LD   C,A        ; PUT IN C'
        EXX             ; MAIN BANK
        SUB  E          ; CALC SHIFT COUNT
        LD   E,L        ; CLEAR LSBYTE OF Y
```

```
        RET  Z          ; SHIFT COUNT IS ZERO
        LD   B,25       ; MAX COUNT
AL10    EXX             ; ALT BANK
        SRL  D          ; SHIFT RIGHT
        RR   E
        EXX             ; MAIN BANK
        RR   D
        RR   E
        DEC  A          ; DECR SHIFT COUNT
        RET  Z          ; COUNT IS ZERO
        DJNZ AL10       ; CONTINUE SHIFTING
        RET
*H MULTIPLY
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT MULTIPLY ROUTINE
;   PRODUCES ROUNDED RESULT
;   INPUT:   IX POINTS TO MULTIPLIER
;            IY POINTS TO MULTIPLICAND
;   OUTPUT:  IX POINTS TO NORMALIZED PRODUCT
;   ERRORS:  EXPONENT OVERFLOW OR UNDERFLOW
;   MAIN LOOP REGISTER ASSIGNMENTS:
;       AHL - 24 BIT ACCUM, INITIALLY ZERO
;       H'L'D' - 24 BIT RIGHT EXTENSION
;                OF ACCUM, INIT MULTIPLIER
;       CDE - 24 BIT MULTIPLICAND
;       B   - LOOP COUNTER
;       A'  - BIT 7 IS SIGN OF RESULT
;   TIMING:  APPROX 2531 (1943 INNER LOOP)
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CHECK FOR ZERO OPERANDS
;
MULT    XOR  A
        CP   (IX+3)
        JP   Z,ZERO     ; X IS ZERO
        CP   (IY+3)
        JP   Z,ZERO     ; Y IS ZERO
;
; LOAD OPERANDS, INITIALIZE
;
        LD   H,(IX)     ; X IN H'L'D'
        LD   L,(IX+1)
        LD   D,(IX+2)
        LD   A,H        ; SAVE SIGN
        SET  7,H        ; RESTORE IMPLIED 1
        EXX             ; MAIN BANK
        CALL LOADY      ; Y IN CDE & FINAL SIGN IN A
        EX   AF,AF'     ; SAVE SIGN
        XOR  A          ; ZERO ACCUM AHL
        LD   H,A
        LD   L,A
        LD   B,24       ; LOOP COUNTER
        JP   ML1
;
; MAIN MULTIPLY LOOP
;   PRODUCES 48 BIT RESULT IN AHLH'L'D'
;
MLOOP   RRA             ; SHIFT ACCUM RIGHT
```

```
        RR   H
        RR   L
ML1     EXX             ; ALT BANK
        RR   H          ; SHIFT MPLIER RIGHT
        RR   L
        RR   D
        EXX             ; MAIN BANK
        JR   NC,ML2     ; MPLIER BIT IS 0
        ADD  HL,DE      ; ADD MPLICAND
        ADC  A,C        ; MAY CAUSE CARRY
ML2     DJNZ MLOOP      ; COUNTER NOT DONE
;
; NORMALIZE (AT MOST 1 SHIFT RIGHT)
;
        JR   NC,MROUND  ; ALREADY NORMALIZED
        RRA             ; SHIFT RESULT RIGHT
        RR   H
        RR   L          ; ROUNDING BIT IN CY
        JP   MROUN1
;
; ROUND
;
MROUND  EXX             ; ALT BANK
        RL   H          ; GET ROUNDING BIT
        EXX             ; MAIN BANK
        DEC  B          ; SET B TO -1
MROUN1  CALL RSTO       ; ROUND AND STORE RESULT
;
; ADD EXP'S AND SHIFT COUNT
;   0 OR 2 OVERFLOWS OK, 1 IS ERROR
;
        ADD  A,C        ; ADD OTHER EXPONENT
MCHK    JP   PE,M10     ; OVERFLOW
        ADD  A,B        ; ADD SHIFT COUNT
        JP   PE,MERR    ; OVERFLOW
        JP   M20
M10     ADD  A,B        ; ADD SHIFT COUNT
        JP   PO,MERR    ; NO OVERFLOW
M20     ADD  A,80H      ; ADD BIAS
        JP   Z,UNDER    ; UNDERFLOW
        LD   (IX+3),A   ; STORE EXP
        RET
*H DIVIDE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT DIVIDE ROUTINE
;   PRODUCES ROUNDED RESULT
;   INPUT:   IX POINTS TO DIVIDEND
;            IY POINTS TO DIVISOR
;   OUTPUT:  IX POINTS TO QUOTIENT
;   ERRORS:  DIVISOR IS ZERO
;            EXPONENT OVERFLOW OR UNDERFLOW
;   REGISTER ASSIGNMENT FOR MAIN LOOP:
;       AHL - 24 BIT ACCUM, INIT DIVIDEND
;       CDE - 24 BIT DIVISOR
;       D'E'H'L' - RESULT REGISTER
;       B   - LOOP COUNTER
;       A'  - BIT 7 IS SIGN OF RESULT
;   TIMING:  APPROX 3595 (3022 INNER LOOP)
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CHECK FOR ZERO OPERANDS
;
DIVI    XOR  A
        CP   (IY+3)
        JP   Z,DIVO       ; Y IS ZERO
        CP   (IX+3)
        JP   Z,ZERO       ; X IS ZERO
;
; LOAD OPERANDS, CALCULATE SIGN
;
        LD   B,(IX)       ; LOAD X INTO BHL
        LD   H,(IX+1)
        LD   L,(IX+2)
        LD   A,B          ; GET SIGN IN A
        CALL LOADY        ; LOAD Y INTO CDE
        EX   AF,AF'       ; SIGN BIT IN A'
        LD   A,B          ; X IN AHL
        OR   80H          ; RESTORE IMPLIED 1
        LD   B,26         ; LOOP COUNTER
        JP   DVSUB
;
; MAIN DIVIDE LOOP
;   PRODUCES 26 BIT RESULT IN (BIT 0 OF)D'E'H'L'CY
;
DVOV    AND  A
        SBC  HL,DE        ; SUBTRACT DIVISOR
        SBC  A,C
DVRES1  SCF               ; SET CARRY
        DEC  B            ; DEC LOOP COUNTER
        JR   Z,DVNORM     ; LOOP DONE
DVLOOP  EXX               ; ALT BANK
        ADC   HL,HL       ; SHIFT RESULT INTO D'E'H'L'
        RL   E
        RL   D
        EXX               ; MAIN BANK
        ADD  HL,HL        ; SHIFT ACCUM LEFT
        ADC  A,A
        JR   C,DVOV       ; OVERFLOW ON SHIFT
DVSUB   SBC  HL,DE        ; SUBTRACT DIVISOR
        SBC  A,C
        JR   NC,DVRES1    ; POSITIVE RESULT
        ADD  HL,DE        ; RESTORE ACCUM
        ADC  A,C
        AND  A            ; TURN OFF CARRY
        DJNZ DVLOOP       ; LOOP NOT DONE
;
; NORMALIZE RESULT
;
DVNORM  EXX               ; ALT BANK
        LD   B,0          ; CLEAR SHIFT COUNT
        LD   A,E          ; RESULT NOW IN D'AH'L'
        BIT  0,D          ; CHECK FIRST RESULT BIT
        JR   Z,DROUND     ; MSBIT IS ZERO
        INC  B            ; INC SHIFT COUNT
        SCF               ; SHIFT RESULT RIGHT
        RRA
        RR   H
        RR   L            ; ROUNDING BIT IN CY
```

```
DROUND  CALL RSTO       ; ROUND AND STORE FRACTION
;
; SUBTRACT EXPONENTS
;
        SUB  C
        JP   MCHK
;
; RECIPROCAL
;
REC     CALL STO1       ; SAVE X
        LD   HL,C1
        CALL LD         ; GET A 1
        LD   IY,TEMP1
        CALL DIVI
        RET
*H MULT, DIVI UTILITIES
;
; RSTO  ROUND, STORE FRACTION, LOAD EXPONENTS
;   INPUT:   ROUNDING BIT IN CARRY
;            NORMALIZED FRACTION IN AHL
;            FINAL SIGN BIT IN A'
;   OUTPUT:  UPDATED SHIFT COUNT IN B
;            UNBIASED EXP'S IN A AND C
;   TIMING:  APPROX 186
;
RSTO    LD   DE,0
        ADC  HL,DE      ; ADD ROUNDING BIT
        ADC  A,0
        JR   NC,$+3     ; NO OVERFLOW
        INC  B          ; INC SHIFT COUNT
; NOTE:  DUE TO IMPLIED 1, ACTUAL SHIFTING UNNECESSARY
        LD   (IX+1),H   ; STORE FRACTION
        LD   (IX+2),L
        AND  7FH        ; TURN OFF IMPLIED 1
        LD   H,A        ; SAVE REST OF BYTE
        EX   AF,AF'     ; GET FINAL SIGN
        OR   H          ; COMBINE
        LD   (IX),A     ; STORE MSBYTE
        LD   A,(IY+3)   ; GET EXP
        SUB  80H        ; REMOVE BIAS
        LD   C,A
        LD   A,(IX+3)   ; GET OTHER EXP
        SUB  80H        ; REMOVE BIAS
        RET
;
; LOADY  LOADS FRACTION PART OF (IY) INTO CDE
;   INPUT:   MSBYTE OF X IN A
;   OUTPUT:  SIGN OF RESULT IN BIT 7 OF A
;   TIMING:  86
;
LOADY   LD   C,(IY)     ; LOAD FRACTION
        LD   D,(IY+1)
        LD   E,(IY+2)
        XOR  C          ; CALC SIGN BIT
        AND  80H
        SET  7,C        ; RESTORE IMPLIED 1
        RET
;
; OVER OR UNDERFLOW ERRORS
;
```

```
MERR    JP   P,UNDER     ; UNDERFLOW
        EX   AF,AF'      ; GET SIGN
        LD   (IX),A      ; SET SIGN OF RESULT
        JP   OVER
;
; DIVISION BY ZERO ERROR
;
DIVO    LD   HL,ERROR
        SET  4,(HL)
        JP   INF
*H FIX
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FIX CONVERTS FLOATING POINT NUMBER TO
;       16 BIT 2'S COMPLEMENT
;       RESULT IS ROUNDED
;
;   INPUT:  IX  PTS TO FL PT NUMBER
;   OUTPUT: IX  PTS TO 2'S COMP INTEGER
;           HL  CONTAINS 2'S COMP INTEGER
;   ERRORS: OVERFLOW IF INPUT CANNOT BE REPRESENTED IN
;       16 BITS.  MAX OUTPUT IS 32767, MIN IS -32768
;   ALGORITHM: SHIFT COUNT IS DERIVED FROM EXPONENT
;       FRACTION PART IS SHIFTED RIGHT AND CONVERTED
;       TO 2'S COMPLEMENT.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
FIX     LD   IY,C_5
        CALL ADD         ; ADD 1/2 TO ROUND
        CALL FIXA        ; CONVERT NUMBER
        JR   NC,FX60     ; NO OVERFLOW
;
; INTEGER OVERFLOW
;
        LD   HL,ERROR
        SET  6,(HL)      ; FLAG OVERFLOW
        LD   HL,7FFFH    ; LARGEST INTEGER
        BIT  7,(IX)
        JR   Z,FX60      ; SIGN POSITIVE
        INC  HL          ; SMALLEST INTEGER
;
; STORE RESULT
;
FX60    LD   (IX),L
        LD   (IX+1),H
        RET
;
; FIXA MAIN 16 BIT FIX OPERATION
;   INPUT:  IX PTS TO FL PT NUMBER
;   OUTPUT: HL IS 16 BIT 2'S COMPLEMENT
;           CY IS SET IF HL IS INVALID (OVERFLOW)
;
FIXA    XOR  A
        LD   H,A         ; CLEAR ACCUM
        LD   L,A
        CP   (IX+3)
        RET  Z           ; INPUT WAS ZERO
        LD   H,(IX)      ; LOAD MS 16 BITS
        SET  7,H         ; RESTORE IMPLIED 1
        LD   L,(IX+1)
```

```
        LD   A,16+80H
        SUB  (IX+3)      ; CALC SHIFT COUNT
        RET  C           ; X >= 2^16
        JR   Z,FX25      ; COUNT IS ZERO
;
; INITIALIZE FOR LOOP
;
        LD   B,16        ; SET MAX COUNT
        CP   B
        JR   NC,FX10     ; COUNT >= 16
        LD   B,A         ; SET COUNT
FX10    XOR  A           ; CLEAR A
        CP   (IX+2)
        RLA              ; A IS 1 IF (IX+2)>0
;
; SHIFT RIGHT UNTIL COUNT=0
;
FX20    SRL  H
        RR   L
        ADC  A,0         ; ACCUM BITS SHFTD OUT
        DJNZ FX20
;
; CONVERT TO 2'S COMPLEMENT
;
FX25    BIT  7,(IX)
        JR   Z,FX40      ; POSITIVE
        AND  A
        JR   Z,FX30      ; EXACT INTEGER
        INC  HL
FX30    EX   DE,HL
        LD   HL,0
        AND  A
        SBC  HL,DE       ; NEGATE
        JP   P,FX50      ; OVERFLOW
        AND  A           ; CLEAR CY
        RET
FX40    AND  A           ; CLEAR CY
        BIT  7,H
        RET  Z           ; NO OVERFLOW
;
; OVERFLOW
;
FX50    SCF              ; SET CY
        RET
*H FLOAT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOAT CONVERTS 16 BIT 2'S COMPLEMENT INTEGER
;       TO FLOATING POINT FORMAT
;   (FLOAT1 ASSUMES INPUT IS IN HL)
;
;   INPUT:  IX  PTS TO 16 BIT 2'S COMP INTEGER
;   OUTPUT: IX  PTS TO FLOATING PT NUMBER
;   ERRORS: NONE
;   ALGORITHM:  CONVERT TO SIGN MAGNITUDE AND NORMALIZE
;   REGISTERS:  HL IS 16 BIT ACCUM
;               A  IS SIGN
;               C  IS EXPONENT
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;
; TEST INPUT
;
FLOAT   LD   L,(IX)      ; LOAD 16 BIT INTEGER
        LD   H,(IX+1)
FLOAT1  LD   A,H         ; TEST FOR ZERO
        OR   L
        JP   Z,ZERO      ; INPUT WAS ZERO
;
; CONVERT TO SIGN MAGNITUDE, SIGN IN A
;
        LD   A,H         ; GET SIGN BIT
        AND  80H
        JR   Z,FL10      ; POSITIVE
        EX   DE,HL
        LD   HL,0
        SBC  HL,DE       ; NEGATE
;
; NORMALIZE
;
FL10    LD   C,16+80H    ; INITIAL EXPONENT
        JP   M,FL30      ; ALREADY NORMALIZED
        AND  A           ; CLEAR CY
FL20    DEC  C           ; DEC EXP
        ADC  HL,HL       ; SHIFT LEFT
        JP   P,FL20      ; NOT NORMALIZED YET
;
; STORE RESULT
;
FL30    RES  7,H         ; CLEAR IMPLIED 1
        OR   H           ; ADD SIGN
        LD   (IX),A
        LD   (IX+1),L
        LD   (IX+2),D
        LD   (IX+3),C
        RET
*H BRK
;
; BRK SEPARATES INTEGER AND FRACTION PARTS
;
;   INPUT:  IX  PTS TO X
;   OUTPUT: IX  PTS TO FRACTION 0 <= X < 1
;           HL  IS LOW ORDER 16 BITS OF 2'S COMP FIX32(X)
;           CY  IS SET IF INPUT > 2^24
;   ALGORITHM: USES 32 BIT FIX AND FLOAT, AND FL PT SUB
;       ALWAYS RETURNS VALID FRACTION PART
;
BRK     CALL STO1        ; SAVE COPY OF X
        CALL FIX32       ; GET 32 BIT INTEGER
        JR   C,BRK20     ; X > 2^24
        EXX              ; ALT BANK
        BIT  7,B
        EXX              ; MAIN BANK
        PUSH HL          ; SAVE HL AND FLAGS
        PUSH AF
        CALL FLO32       ; CONVERT TO FLOATING
        LD   IY,TEMP1
        CALL RSUB        ; CALC X - INT(X)
        POP  AF          ; RESTORE REGS
        POP  HL
```

```
        JR   Z,BRK1O     ; POSITIVE SIGN
        EX   DE,HL
        LD   HL,0
        AND  A
        SBC  HL,DE       ; NEGATE HL
BRK1O   AND  A           ; CLEAR CY
        RET
BRK2O   CALL ZERO        ; FORCE ZERO FRACTION
        SCF              ; SET CY
        RET
*H CP
;
; COMPARE ROUTINE
;   RETURNS CY SET IF X < Y
;   RETURNS Z  SET IF X = Y
;
CP      LD   A,(IX)      ; GET MS BYTE OF X
        AND  80H
        JR   NZ,CP1O     ; X IS NEGATIVE
        BIT  7,(IY)
        RET  NZ          ; Y IS NEGATIVE
        LD   A,(IX+3)    ; TEST EXP'S
        CP   (IY+3)
        RET  NZ
        LD   A,(IX)      ; TEST MS BYTE
        CP   (IY)
        RET  NZ
        LD   A,(IX+1)
        CP   (IY+1)
        RET  NZ
        LD   A,(IX+2)
        CP   (IY+2)
        RET
CP1O    XOR  (IY)        ; TEST SIGN AND SET NZ
        RLA              ; PUT SIGN IN CY
        RET  C           ; SIGNS DIFFERENT
        LD   A,(IY+3)
        CP   (IX+3)
        RET  NZ
        LD   A,(IY)
        CP   (IX)
        RET  NZ
        LD   A,(IY+1)
        CP   (IX+1)
        RET  NZ
        LD   A,(IY+2)
        CP   (IX+2)
        RET
*H FLO32
;
; FLO32   CONVERTS 32 BIT INTEGER TO FLOATING POINT
;   BY NORMALIZING AND ROUNDING TO 24 BITS
;   ENTRY TO FLO321 WITH DE ZERO AVOIDS ROUNDING
;   INPUT:   H'L'HL IS 32 BIT INTEGER
;            DE IS ROUNDING VALUE FOR FLO321
;            B' BIT 7 IS SIGN
;   OUTPUT:  IX POINTS TO FL PT NUMBER
;
;
; TEST FOR ZERO
```

```
;
FLO32   LD   DE,80H      ; ROUNDING BIT
FLO321  LD   A,L         ; OR ALL 4 BYTES
        OR   H
        EXX              ; ALT BANK
        OR   L
        OR   H
        JP   Z,ZERO      ; INPUT IS ZERO
;
; NORMALIZE ACCUM, COUNT EXP IN C'
;
FNORM   LD   C,0A0H      ; INIT EXP TO +32
        BIT  7,H
        JR   NZ,FROUND   ; ALREADY NORMALIZED
FNORL   DEC  C           ; DEC EXP
        EXX              ; MAIN BANK
        ADD  HL,HL       ; SHIFT ACC LEFT
        EXX              ; ALT BANK
        ADC  HL,HL
        JP   P,FNORL     ; NOT NORMALIZED YET
;
; ROUND  FRACTION TO 24 BITS
;   NOTE: ROUNDING OVERFLOW IMPLIES FRACTION
;         IS A POWER OF 2, SO HIGHEST BIT NEED
;         NOT BE PUT IN BEFORE STORING
;
FROUND  EXX              ; MAIN BANK
        ADD  HL,DE       ; ADD ROUNDING BIT
        EXX              ; ALT BANK
        LD   DE,0
        ADC  HL,DE
        JR   NC,FSTO     ; NO OVERFLOW
        INC  C           ; INC EXP
        JP   Z,OVERB     ; OVERFLOW
;
; STORE RESULTS  (ENTER IN ALT BANK)
;
FSTO    LD   (IX+3),C    ; BINARY EXPONENT
        LD   A,H         ; PUT IN SIGN BIT
        AND  7FH
        OR   B
        LD   (IX),A      ; MSBYTE
        LD   (IX+1),L    ; 2ND BYTE
        EXX              ; MAIN BANK
        LD   (IX+2),H    ; 2RD BYTE
        RET
;
; LOAD FRACTION PART (CALLED IN ALT BANK)
;
FLD     LD   H,0         ; H'L'HL IS ACCUM
        LD   L,(IX)
        LD   A,L         ; GET SIGN BIT
        AND  80H
        LD   B,A         ; PUT IN B'
        SET  7,L         ; RESTORE IMPLIED 1
        EXX              ; MAIN BANK
        LD   H,(IX+1)
        LD   L,(IX+2)
        RET
*H FIX32
```

```
;
; FIX32  CONVERTS FL PT NUMBER TO 32 BIT INTEGER
;   INPUT:   IX POINTS TO FL PT NUMBER
;   OUTPUT:  H'L'HL - 32 BIT INTEGER
;            B'     - BIT 7 IS SIGN BIT
;            A      - ZERO IF INPUT WAS INTEGER
;            CY     - SET IF INPUT >= 2^32
;
FIX32   XOR  A           ; TEST FOR ZERO
        CP   (IX+3)
        JP   NZ,FXA      ; NOT ZERO
        LD   H,A         ; CLEAR H'L'
        LD   L,A
        LD   B,A         ; CLEAR SIGN
        EXX              ; MAIN BANK
        LD   H,A         ; CLEAR HL
        LD   L,A
        RET              ; A IS ZERO, CY OFF
;
; CALCULATE SHIFT COUNT
;
FXA     CALL FLD         ; LOAD FRACTION TO L'HL
        LD   A,98H       ; CALC SHIFT COUNT
        SUB  (IX+3)
        RET  C           ; INPUT WAS TOO BIG
        RET  Z           ; SHIFT COUNT IS ZERO
        LD   B,24        ; MAX COUNT IS 24
        CP   B
        JR   NC,FX1      ; COUNT >= 24
        LD   B,A         ; PUT COUNT IN B
FX1     XOR  A           ; CLEAR A
;
; SHIFT LOOP
;
FXL     EXX              ; ALT BANK
        SRL  H           ; SHIFT ACCUM RIGHT
        RR   L
        EXX              ; MAIN BANK
        RR   H
        RR   L
        ADC  A,0         ; COUNT 1'S SHFTD OUT
        DJNZ FXL         ; COUNT NOT DONE
;
; USE CEILING FUNCTION FOR NEGATIVE NUMBERS
;
        RET  Z           ; NO 1'S SHIFTED OUT
        EXX              ; ALT BANK
        BIT  7,B
        EXX              ; MAIN BANK
        RET  Z           ; POSITIVE SIGN
        LD   DE,1        ; INCREMENT ACCUM
        ADD  HL,DE
        RET  NC          ; NO CY FROM  INCR
        EXX              ; ALT BANK
        INC  HL          ; ADD CY
        EXX              ; MAIN BANK
        AND  A           ; TURN OFF CY
        RET
*H UTILITIES
;
```

```
; UTILITIES
;
STO1    LD   DE,TEMP1    ; DEST IS TEMP1
        JP   STO
STO2    LD   DE,TEMP2    ; DEST IS TEMP2
        JP   STO
STO3    LD   DE,TEMP3    ; DEST IS TEMP3
STO     PUSH IX          ; SOURCE IS ACCUM
        POP  HL
MOV     LD   BC,4        ; COUNT IS 4
        LDIR
        RET
LD      PUSH IX          ; DEST IS ACCUM
        POP  DE
        JP   MOV
*H BOOLEAN
*H BOOLEAN
;
; LOGICAL OPERATORS
;
IOR     LD   A,(IX)
        OR   (IX+1)
        OR   (IY)
        OR   (IY+1)
        JP   INE1
IAND    LD   A,(IX)
        OR   (IX+1)
        JP   Z,IZERO     ; X IS ZERO
        LD   A,(IY)
        OR   (IY+1)
        JP   INE1
INOT    LD   A,(IX)
        OR   (IX+1)
        JR   IEQ1
OR      LD   A,(IX+3)
        OR   (IY+3)
        JP   NE1
AND     LD   A,(IX+3)
        AND  A
        JP   Z,ZERO      ; X IS ZERO
        LD   A,(IY+3)
        AND  A
        JR   NE1
NOT     LD   A,(IX+3)
        AND  A
        JR   EQ1
;
; RELATIONAL OPERATORS
;
SGT     CALL SCP
        JR   IGT1
SLE     CALL SCP
        JR   ILE1
SLT     CALL SCP
        JR   ILT1
SGE     CALL SCP
        JR   IGE1
SEQ     CALL SCP
        JR   IEQ1
SNE     CALL SCP
```

```
            JR    INE1
IGT         CALL  ICP
IGT1        JR    Z,IZERO
            JR    C,IZERO
            JR    IONE
ILE         CALL  ICP
ILE1        JR    Z,IONE
            JR    C,IONE
            JR    IZERO
ILT         CALL  ICP
ILT1        JR    C,IONE
            JR    IZERO
IGE         CALL  ICP
IGE1        JR    C,IZERO
            JR    IONE
IEQ         CALL  ICP
IEQ1        JR    Z,IONE
            JR    IZERO
INE         CALL  ICP
INE1        JR    Z,IZERO
            JR    IONE
GT          CALL  CP
GT1         JR    Z,ZERO
            JR    C,ZERO
            JR    ONE
LE          CALL  CP
LE1         JR    Z,ONE
            JR    C,ONE
            JR    ZERO
LT          CALL  CP
LT1         JR    C,ONE
            JR    ZERO
GE          CALL  CP
GE1         JR    C,ZERO
            JR    ONE
EQ          CALL  CP
EQ1         JR    Z,ONE
            JR    ZERO
NE          CALL  CP
NE1         JR    Z,ZERO
            JR    ONE
*H CONSTANT FUNCTIONS
;
; PRODUCE UNITY RESULT
;
IONE        LD    (IX),1
            LD    (IX+1),0
            RET
ONE         LD   (IX),0
            LD   (IX+1),0
            LD   (IX+2),0
            LD   (IX+3),81H
            RET
;
; PRODUCE ZERO RESULT
;
IZERO       LD    (IX),0
            LD    (IX+1),0
            RET
ZERO        LD    (IX),0
```

```
        LD    (IX+1),0
        LD    (IX+2),0
        LD    (IX+3),0
        RET
;
; PRODUCE INFINITE RESULT
;
INF     RL    (IX)        ; GET OLD SIGN
        LD    (IX),0FEH
        RR    (IX)        ; REPLACE SIGN
        LD    (IX+1),0FFH
        LD    (IX+2),0FFH
        LD    (IX+3),0FFH
        RET
;
; FLAG OVERFLOW ERROR, STORE INFINITY
;
OVERB   EXX               ; ALT BANK
        LD    (IX),B      ; SET SIGN BIT
OVER    LD    HL,ERROR
        SET   6,(HL)      ; FLAG OVERFLOW
        JP    INF
;
; FLAG UNDERFLOW ERROR, STORE ZERO
;
UNDER   LD    HL,ERROR
        SET   5,(HL)      ; FLAG UNDERFLOW
        JP    ZERO
*H CONSTANTS AND WORK AREAS
C1      DEFW 0            ; 1.0
        DEFW 8100H
C_5     DEFW 0            ; 0.5
        DEFW 8000H
SEED    DEFW 0
        DEFW 0
TEMP1   DEFS 4
TEMP2   DEFS 4
TEMP3   DEFS 4
;
; REFERENCES
;
        EXTERNAL          ERROR,ICP,SCP
        GLOBAL            ADD,SUB,RSUB,MULT,DIVI,REC
        GLOBAL            FIX,FIXA,FLOAT,BRK,FIX32,FLO32,FLO321
        GLOBAL            CP,ST01,ST02,ST03,TEMP1,TEMP2
        GLOBAL            TEMP3,SEED,LD,FLD,FLOAT1
        GLOBAL            OVER,UNDER,ZERO,ONE,INF
        GLOBAL            GT,LT,GE,LE,EQ,NE,AND,OR,NOT
        GLOBAL            IGT,ILT,IGE,ILE,IEQ,INE,IAND,IOR,INOT
        GLOBAL            SGT,SLT,SGE,SLE,SEQ,SNE
        GLOBAL            C1,C_5
```

```
*H DECIMAL TO BINARY
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; DECIMAL TO BINARY CONVERSION
;   INPUT:  IX POINTS TO OUTPUT AREA
;   OUTPUT: IX POINTS TO FL PT NUMBER
;   ERRORS: NO DIGITS IN FRACTION PART
;           NO DIGITS IN EXPONENT
;           EXPONENT OVERFLOW
;           EXPONENT UNDERFLOW
;   REGISTER ASSIGNMENT IN MAIN LOOP:
;       H'L'HL - 32 BIT ACCUM, INIT ZERO
;       B      - DECIMAL EXPONENT, INIT 0
;       C      - FLAGS: BIT 6 - DIGITS FOUND
;                       BIT 7 - FRACTION OVRFLW
;       A      - NEW DIGIT
;       B'     - SIGN  IN BIT 7
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DTOB    CALL LMARK      ; MARK SCAN HERE
        CALL DBINIT     ; INITIALIZE REGISTERS
        CALL LIN        ; GET FIRST CHAR
        CP   '+'
        JR   Z,D10      ; SIGN IS +
        CP   '-'
        JR   NZ,D20     ; NO SIGN
        EXX             ; ALT BANK
        SET  7,B        ; FLAG NEGATIVE SIGN
        EXX             ; MAIN BANK
;
; LOOP TO PROCESS CHARS BEFORE DECIMAL POINT
;
D10     CALL LIN        ; GET NEXT CHAR
D20     CP   '.'
        JR   Z,D40      ; DECML PT FOUND
        CALL DIGIT?
        JR   C,D50      ; NOT A DIGIT
        SET  6,C        ; FLAG DIGIT FOUND
        BIT  7,C
        JR   NZ,D30     ; OVERFLOW
        CALL MAD        ; CONVERT THIS DIGIT
        JP   D10
D30     INC  B          ; INC DECML EXP
        JP   D10
;
; LOOP TO PROCESS CHARS AFTER DECIMAL POINT
;
D40     CALL LIN        ; GET NEXT CHAR
        CALL DIGIT?
        JR   C,D50      ; NOT A DIGIT
        SET  6,C        ; FLAG DIGIT FOUND
        BIT  7,C
        JR   NZ,D40     ; OVERFLOW
        DEC  B          ; DEC DECML EXP
        CALL MAD        ; CONVERT THIS DIGIT
        JP   D40
;
; CHANGE FRACTION TO FLOATING POINT FORMAT
;
D50     BIT  6,C
```

```
        JP   Z,LREST     ; NO DIGITS FOUND
        PUSH AF          ; SAVE THIS CHAR
        CALL FL032
        POP  AF
;
; PROCESS EXPONENT
;   REGISTERS:
;       C - BIT 7 IS SIGN
;           BIT 6 IS DIGIT-FOUND FLAG
;           BIT 5 IS DIGIT-OVERFLOW FLAG
;       D - IS ACCUM FOR D-TO-B CONVERSION
;
        LD   D,0         ; INITIALIZE REGISTERS
        LD   C,D
        CP   'E'-'0'
        JR   NZ,D90      ; NOT AN 'E'
        CALL LIN
        CP   '+'
        JR   Z,D60       ; SIGN IS +
        CP   '-'
        JR   NZ,D70      ; NO SIGN
        SET  7,C         ; FLAG NEG SIGN
D60     CALL LIN
D70     CALL DIGIT?
        JR   C,D80       ; NOT A DIGIT
        SET  6,C         ; FLAG DIGIT FOUND
        BIT  5,C
        JR   NZ,D100     ; OVER OR UNDERFLOW
        CALL MAD1        ; CONVERT THIS DIGIT
        JP   D60
D80     BIT  6,C
        JR   Z,D120      ; NO DIGITS FOUND
        BIT  7,D
        JR   NZ,EXPERR   ; OVER OR UNDERFLOW
D90     CALL DSCALE      ; SCALE BY DECML EXP
        JP   RTN
;
; FINISH SCANNING DIGITS
;
D100    CALL LIN         ; FINISH SCANNING DIGITS
        CALL DIGIT?
        JR   NC,D100     ; DIGIT IS FOUND
;
; DETERMINE FROM SIGN BIT WHETHER OVER OR UNDER
;
EXPERR  BIT  7,C
        JR   NZ,D110     ; SIGN IS NEGATIVE
        CALL OVER        ; FLAG ERROR
        JP   RTN
D110    CALL UNDER       ; FLAG ERROR
        JP   RTN
;
; ILLEGAL NUMBER
;
D120    LD   HL,ERROR
        SET  7,(HL)
        RET
*EJECT
RTN     CALL LBACK       ; BACKSPACE SCAN PTR
        JP   RC2
```

```
;
; DIGIT? ROUTINE CONVERTS ASCII CHAR TO 0-9 VALUE
;    RETURNS CY ON IF NOT A DIGIT
;
DIGIT?  SUB   '0'
        RET   C           ; WAS < 30H
        CP    10
        CCF               ; CY SET IF > 9
        RET
;
; DBINIT CLEARS REGISTERS H'L',B',HL,BC,A
;
DBINIT  XOR   A           ; CLEAR A
        LD    H,A
        LD    L,A
        LD    B,A
        EXX               ; MAIN BANK
        LD    H,A
        LD    L,A
        LD    B,A
        LD    C,A
        RET
;
; MAD  ROUTINE TO MULTIPLY ACCUM BY 10 AND ADD NEW DIGIT
;
MAD     CALL MUL10        ; MULTIPLY BY 10
        EXX               ; MAIN BANK
        LD    D,0         ; ADD A TO ACCUM
        LD    E,A
        ADD   HL,DE
        EXX               ; ALT BANK
        JR    NC,$+3      ; NO CARRY FROM LOW ORDER
        INC   HL          ; ADD CARRY
        LD    A,H         ; CHECK OVERFLOW
        AND   0F0H
        EXX               ; MAIN BANK
        RET   Z           ; NO OVERFLOW
        SET   7,C         ; FLAG OVERFLOW
        RET
;
; MUL10   MULTIPLIES 32 BIT ACCUM BY 10
;   ACCUM IS H'L'HL
;   USES D'E'DE
;   CALLED IN MAIN BANK, RETURNS ALT BANK
;
MUL10   ADD   HL,HL       ; SHIFT ACC LEFT
        LD    E,L         ; COPY INTO DE
        LD    D,H
        EXX               ; ALT BANK
        ADC   HL,HL
        LD    E,L
        LD    D,H
        EXX               ; MAIN BANK
        ADD   HL,HL       ; SHIFT ACC LEFT
        EXX               ; ALT BANK
        ADC   HL,HL
        EXX               ; MAIN BANK
        ADD   HL,HL       ; SHIFT ACC LEFT
        EXX               ; ALT BANK
        ADC   HL,HL
```

```
        EXX             ; MAIN BANK
        ADD  HL,DE      ; ADD DE TO ACC
        EXX             ; ALT BANK
        ADC  HL,DE
        RET
;
; MAD1   MULTIPLY 8 BIT ACC BY 10, ADD A
;   SAVES RESULT IN D
;
MAD1    PUSH AF         ; SAVE A
        LD   A,D        ; GET OLD RESULT
        ADD  A,A        ; SHIFT LEFT
        ADD  A,A        ; SHIFT LEFT
        ADD  A,D        ; ADD OLD VALUE
        ADD  A,A        ; SHIFT LEFT
        POP  DE         ; NEW DIGIT IN D
        ADD  A,D
        LD   D,A        ; STORE IN D
        AND  0F0H
        RET  Z          ; NO DIGIT OVERFLOW
        SET  5,C        ; FLAG OVERFLOW
        RET
*EJECT
;
; DSCALE  ROUTINE TO COMBINE DECIMAL EXPONENTS
;         AND MULTIPLY BY POWER OF 10 FROM TABLE
;   INPUT:   C - BIT 7 IS SIGN OF DECML EXP
;            D - IS MAGNITUDE OF DECML EXP
;            B - IS 2'S COMPL EXP FROM FRACTION
;   OUTPUT:  IX POINTS TO FL PT NUMBER
;   ERRORS:  EXPONENT OVER OR UNDERFLOW
;   REGISTER ASSIGNMENT:
;       DE- POINTER INCREMENT
;       C - EXPONENT MAGNITUDE
;       B - LOOP COUNTER
;       IY- TABLE POINTER
;
;
; INITIALIZE
;
DSCALE  LD   IY,TOPTAB  ; TABLE POINTER
        LD   A,D        ; MOVE EXP MAGN TO A
        LD   DE,-5      ; POINTER INCREMENT
;
; CONVERT SIGN-MAGNITUDE TO BINARY AND ADD
;
        BIT  7,C
        JR   Z,DSC10    ; POSITIVE SIGN
        NEG
DSC10   LD   C,0        ; CLEAR SIGN BIT
        ADD  A,B        ; ADD BOTH EXP'S
        RET  Z          ; ZERO EXPONENT
        JP   P,DSC40    ; RESULT IS POSITIVE
        JP   PE,OVER    ; OVERFLOW
        NEG             ; GET MAGNITUDE
        CP   100
        JP   NC,UNDER   ; MAGNITUDE >= 100
        LD   C,A        ; SAVE MAGN IN C
        LD   B,LENTAB   ; LOOP COUNTER
;
```

```
; LINEAR SEARCH THRU TABLE OF POWERS OF 10
;
DSC20   CP   (IY+4)
        JP   C,DSC30    ; EXP < TABLE EXP
        PUSH BC         ; SAVE REGS
        CALL DIVI       ; DIVIDE BY TABLE ENTRY
        POP  BC
        LD   DE,-5      ; RESTORE DE
        LD   A,C
        SUB  (IY+4)     ; SUB TABLE EXP
        LD   C,A
DSC30   ADD  IY,DE      ; DECR POINTER
        DJNZ DSC20      ; LOOP NOT DONE
        RET
;
; EXPONENT SIGN IS POSITIVE
;
DSC40   JP   PE,UNDER   ; UNDERFLOW
        CP   100
        JP   NC,OVER    ; EXP >= 100
        LD   C,A        ; INIT FOR LOOP
        LD   B,LENTAB
;
; LINEAR SEARCH THRU TABLE OF POWERS OF 10
;
DSC50   CP   (IY+4)
        JP   C,DSC60    ; EXP < TABLE EXP
        PUSH BC         ; SAVE REGS
        CALL MULT
        POP  BC
        LD   DE,-5      ; RESTORE DE
        LD   A,C
        SUB  (IY+4)     ; SUB TABLE EXP
        LD   C,A
DSC60   ADD  IY,DE      ; DECR POINTER
        DJNZ DSC50      ; LOOP NOT DONE
        RET
*H BINARY TO DECIMAL
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT BINARY TO DECIMAL CONVERSION
;   RESULT IS ROUNDED TO 7 DECIMAL DIGITS
;   INPUT:   HL POINTS TO OUTPUT AREA
;            IX POINTS TO FL PT NUMBER
;   OUTPUT:  ASCII STRING IN OUTPUT AREA
;   OUTPUT:  FL PT NUMBER IS MODIFIED
;            A IS 2'S COMPLEMENT EXPONENT (-128<=A<=126)
;            OUTPUT AREA IS 12 BYTES:
;               1  SIGN OF FRACTION (SPACE OR -)
;               7  DIGITS OF FRACTION
;               1  "E"
;               1  SIGN OF EXPONENT (+ OR -)
;               2  DIGITS OF EXPONENT
;            EXPONENT IS FOR DECIMAL PT AFTER FIRST DIGIT
;   ERRORS:  NONE
;   REGISTER ASSIGNMENT FOR INNER LOOP:
;       H'L'HL - 32 BIT ACCUM, INIT FRACTION PART
;                WITH 4 BITS TO LEFT OF POINT
;       C      - DECIMAL EXPONENT
;       B      - LOOP COUNTER
```

```
;        IY      - OUTPUT POINTER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CONVERT SIGN, TEST FOR ZERO
;
BTOD     BIT  7,(IX)      ; TEST SIGN BIT
         LD   (HL),' '    ; DEFAULT SIGN
         JR   Z,BD02      ; POSITIVE SIGN
         LD   (HL),'-'
BD02     INC  HL
         PUSH HL          ; SAVE OUTPUT POINTER
         LD   A,(IX+3)    ; CHECK FOR ZERO
         OR   A
         JP   NZ,BD05     ; X IS NOT ZERO
         LD   HL,0        ; CLEAR ACCUM
         EXX              ; MAIN BANK
         LD   HL,0
         LD   C,0         ; CLEAR DECML EXP
         POP  IY          ; OUTPUT PTR IN IY
         JP   BD50
;
; SCALE X, LOAD FRACTION
;
BD05     CALL BSCALE      ; SCALE BY POWER OF 10
         EXX              ; ALT BANK
         CALL FLD         ; LOAD FRACTION
         POP  IY          ; PTR IN IY
;
; SHIFT TO GET ZERO BINARY EXP
;   AND ALIGN BIN POINT BETWEEN BITS 3 AND 4
;   OF H'
;   INITIALLY, 7D <= EXP <= 80
;
         LD   A,(IX+3)    ; GET BINARY EXP
         SUB  7CH         ; REMOVE BIAS, ADD 4
BD20     ADD  HL,HL       ; SHIFT LEFT
         EXX              ; ALT BANK
         ADC  HL,HL
         EXX              ; MAIN BANK
         DEC  A           ; DECR EXPONENT
         JR   NZ,BD20     ; EXP NOT ZERO YET
;
; NORMALIZE TO: .1 < ACCUM < 1
;    .1 IS .199999 (HEX)
;
         EXX              ; ALT BANK
         LD   A,1         ; FIRST HEX DIGIT
         CP   H
         JP   NZ,BD40     ; ACCUM <> .1
         LD   A,99H       ; ALL OTHER HEX DIGITS
         CP   L
         JP   NZ,BD40     ; ACCUM <> .1
         EXX              ; MAIN BANK
         CP   H
         JP   NZ,BD45     ; ACCUM <> .1
         CP   L
         JP   C,BD50      ; ACCUM > .1
BD30     CALL MUL10       ; MPY BY 10
         EXX              ; MAIN BANK
```

```
        DEC   C           ; DECR DECML EXP
        JP    BD50
BD40    EXX               ; MAIN BANK
BD45    JR    NC,BD30     ; ACCUM < .1
;
; ROUND BY ADDING 1/2 LAST DECIMAL POSITION
;
BD50    LD    B,7         ; LOOP COUNTER
        LD    DE,0DH      ; .5E-7
        ADD   HL,DE       ; ADD ROUNDING VALUE
        EXX               ; ALT BANK
        JR    NC,$+3      ; NO CY FROM LOW BYTES
        INC   HL          ; ADD IN CARRY
        LD    A,H         ; GET MSBYTE
        AND   0F0H
        JR    Z,BD55      ; NO ROUNDING OVERFLW
        INC   C           ; INCR DECML EXP
        JP    BD65        ; USE THIS AS 1ST DIGIT
BD55    EXX               ; MAIN BANK
;
; LOOP TO CONVERT FRACTION
;
BD60    CALL  MUL10       ; MPY BY 10
BD65    LD    A,H         ; GET MSBYTE
        LD    D,H
        AND   0FH         ; CLEAR TOP 4 BITS
        LD    H,A
        LD    A,D         ; GET MSBYTE AGAIN
        EXX               ; MAIN BANK
        RRA               ; SHIFT RIGHT 4
        RRA
        RRA
        RRA
        AND   0FH         ; CONVERT TO ASCII
        OR    '0'
        LD    (IY),A      ; OUTPUT CHAR
        INC   IY
        DJNZ  BD60        ; LOOP NOT DONE
;
; CONVERT EXPONENT
;
        LD    (IY),'E'
        LD    (IY+1),'+'
        LD    A,C         ; GET 2'S COMP EXP
        PUSH  AF          ; SAVE IT
        DEC   A           ; SUB ONE
        JP    P,BD70      ; SIGN IS +
        LD    (IY+1),'-'
        NEG
BD70    LD    B,0FFH      ; QUOTIENT
        LD    C,10        ; DIVISOR
BD80    SUB   C           ; LOOP DIVIDES BY 10
        INC   B           ; INCR QUOTIENT
        JR    NC,BD80     ; STILL POSITIVE
        ADD   A,C         ; RESTORE REMAINDER
        OR    '0'         ; CONVERT TO ASCII
        LD    (IY+3),A    ; STORE 2ND DIGIT
        LD    A,B
        OR    '0'         ; CONVERT TO ASCII
        LD    (IY+2),A    ; STORE 1ST DIGIT
```

```
        POP  AF          ; GET BINARY EXP AGAIN
        RET
*EJECT
;
; BSCALE   TRY TO REDUCE BINARY EXP TO ZERO
;         SCALING BY POWER OF 10
;   INPUT:   IX POINTS TO FL PT NUMBER
;   OUTPUT:  IX POINTS TO SCALED FL PT NUMBER
;            .0625 < X < 1
;            C CONTAINS DECIMAL EXP OF SCALE FACTOR
;   ERRORS:  NONE
;   REGISTER ASSIGNMENT FOR LOOP:
;       A  - IS COMPARISON EXPONENT
;       B  - IS LOOP COUNTER
;       C  - IS DECIMAL EXPONENT
;       DE - IS POINTER INCREMENT
;       IY - IS TABLE POINTER
;
BSCALE  LD   IY,TOPTAB   ; PT TO TOP OF TABLE
        LD   DE,-5       ; 5 BYTES PER ENTRY
        LD   B,LENTAB    ; LOOP COUNTER
        LD   C,0         ; CLEAR DEC EXP
        LD   A,(IX+3)    ; GET BINARY EXP
        CP   81H
        JR   NC,BSC30    ; EXP IS POSITIVE
        NEG              ; NEGATE EXP
;
; SCAN TABLE OF POWERS OF 10
;
BSC10   CP   (IY+3)
        JP   C,BSC20     ; EXP < TABLE EXP
        PUSH BC
        CALL MULT        ; MULTIPLY BY TABLE ENTRY
        POP  BC
        LD   DE,-5       ; POINTER INCREMENT
        LD   A,C         ; GET DECML EXP
        SUB  (IY+4)      ; SUB LOG(TABLE ENTRY)
        LD   C,A         ; RPLACE DECML EXP
        XOR  A
        SUB  (IX+3)      ; GET NEW BINRY EXP
BSC20   ADD  IY,DE       ; DECR POINTER
        DJNZ BSC10       ; LOOP NOT DONE
        RET
;
; SCAN TABLE OF POWERS OF 10
;
BSC30   ADD  A,3         ; ADD 3 TO BINRY EXP
        JR   C,BSC50     ; OVERFLW, USE LARGEST SCALE
BSC40   CP   (IY+3)
        JP   C,BSC60     ; EXP+3 < TABLE EXP
BSC50   PUSH BC
        CALL DIVI        ; DIVIDE BY TABLE ENTRY
        POP  BC
        LD   DE,-5       ; POINTER INCREMENT
        LD   A,C         ; GET DECML EXP
        ADD  A,(IY+4)    ; ADD LOG(TABLE ENTRY)
        LD   C,A
        LD   A,(IX+3)    ; GET NEW BINRY EXP
        ADD  A,3         ; ADD 3
BSC60   ADD  IY,DE
```

```
        DJNZ BSC40        ; LOOP NOT DONE
        RET
*H RND
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; RANDOM NUMBER GENERATOR
;    PRODUCES 32 BIT RANDOM NUMBERS:
;      X(I+1) = 3141592621 * X(I) + 2718281829
;    NORMALIZED AND TRUNCATED TO 24 BIT
;      FLOATING POINT, 0 <= X < 1
;   INPUT:    IX POINTS TO OUTPUT AREA
;             SEED,SEED+2 CONTAIN LAST 32 BIT RESULT
;   OUTPUT:   IX POINTS TO RANDOM FL PT NUMBER
;             SEED,SEED+2 CONTAIN NEW RESULT
;   ERRORS:   NONE
;   REGISTER ASSIGNMENT FOR MAIN LOOP:
;        H'L'HL - IS 32 BIT ACCUM
;        D'E'DE - IS 32 BIT REGISTER
;        B'C'CA - IS 32 BIT RIGHT EXTENSION
;        B       - IS LOOP COUNTER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; INITIALIZE REGISTERS
;
RND     LD   HL,0         ; CLEAR ACCUM
        LD   DE,(SEED)    ; GET X(I)
        LD   BC,0BB40H    ; MS HALF OF 3141592621
        EXX               ; MAIN BANK
        LD   HL,0
        LD   DE,(SEED+2)
        LD   C,0E6H       ; 3RD BYTE OF 3141592621
        LD   A,2DH        ; 4TH BYTE OF 3141592621
        LD   B,33         ; LOOP COUNTER
        JP   ML2
;
; 32 BIT MULTIPLY LOOP
;
MLOOP   JR   NC,ML1       ; MPLIER BIT IS ZERO
        ADD  HL,DE        ; ADD MPLICAND TO ACCUM
        EXX               ; ALT BANK
        ADC  HL,DE
        EXX               ; MAIN BANK
;
; ROTATE ACCUM RIGHT
;
ML1     EXX               ; ALT BANK
        RR   H
        RR   L
        EXX               ; MAIN BANK
        RR   H
        RR   L
;
; ROTATE RIGHT EXTENSION
;
ML2     EXX               ; ALT BANK
        RR   B
        RR   C
        EXX               ; MAIN BANK
        RR   C
```

```
        RRA             ; MPLIER BIT IN CY
        DJNZ MLOOP      ; LOOP COUNT NOT DONE
;
; MOVE LOW ORDER 32 BITS TO ACCUM
;
        EXX             ; ALT BANK
        LD   H,B
        LD   L,C
        EXX             ; MAIN BANK
        LD   H,C
        LD   L,A
;
; ADD 2718281829
;
        LD   DE,0B065H  ; LS HALF OF 2718281829
        ADD  HL,DE
        EXX             ; ALT BANK
        LD   DE,0A205H  ; MS HALF OF 2718281829
        ADC  HL,DE
;
; STORE NEW RESULT
;
        LD   (SEED),HL  ; MS HALF
        LD   B,0        ; CLEAR SIGN BIT
        EXX             ; MAIN BANK
        LD   (SEED+2),HL; LS HALF
        LD   DE,0       ; SET FOR NO ROUNDING
        CALL FL0321     ; FLOAT INTEGER
        LD   A,(IX+3)   ; DIVIDE BY 2↑32
        SUB  32
        LD   (IX+3),A
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; RANDOMIZE  USES REFRESH REGISTER TO MODIFY SEED
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
RANDMZ  LD   A,R        ; GET REFRESH COUNTER
        LD   (SEED),A   ; MODIFY SEED
        RET
*H SHORT FUNCTIONS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; INTEGER FUNCTION -- FLOOR FUNCTION
;   INPUT,OUTPUT: IX POINTS TO FL PT NUMBER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INT     CALL FIX32      ; CONVERT TO INTEGER
        RET  C          ; X >= 2↑24
        CALL FL032      ; CONVERT TO FLOATING POINT
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; ABSOLUTE VALUE FUNCTION
;   INPUT,OUTPUT: IX POINTS TO FL PT NUMBER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ABS     RES  7,(IX)     ; TURN OFF SIGN BIT
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;
; SIGN FUNCTION
;   INPUT,OUTPUT: IX POINTS TO FL PT NUMBER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SGN     XOR  A          ; CLEAR A
        CP   (IX+3)
        RET  Z          ; ZERO INPUT IS NOP
        LD   A,(IX)     ; GET SIGN
        AND  80H
        CALL ONE        ; OUTPUT 1
        LD   (IX),A     ; SIGN OF INPUT
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; NEGATE FUNCTION
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
NEG     LD   A,(IX)
        XOR  80H
        LD   (IX),A
        RET
*H INTEGER ADD AND SUB
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; IADD, ISUB  INTEGER ADD AND SUBTRACT
;
;   INPUT:  IX  PTS TO X
;           IY  PTS TO Y
;   OUTPUT: IX  PTS TO SUM OR DIFFERENCE
;   ERRORS: OVERFLOW (POSITIVE OR NEGATIVE)
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IADD    CALL ASLOAD     ; LOAD ARGUMENTS
        ADC  HL,DE
        JP   PO,ASSTO   ; NO OVERFLOW
ASOV    JP   P,NINF     ; OVFLW SIGN POS
        JP   PINF
ISUB    CALL ASLOAD     ; LOAD ARGUMENTS
        SBC  HL,DE
        JP   PO,ASSTO   ; NO OVERFLOW
        JR   ASOV
;
; ASLOAD  LOADS ARGS TO HL, DE, RESETS CY
;
ASLOAD  LD   H,(IX+1)
        LD   L,(IX)
        LD   D,(IY+1)
        LD   E,(IY)
        AND  A
        RET
;
; ASSTO  STORES HL AS RESULT
;
ASSTO   LD   (IX),L
        LD   (IX+1),H
        RET
*H INTEGER MULTIPLY
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
; IMUL  INTEGER MULTIPLY
;
;   INPUT:  IX  PTS TO X
;           IY  PTS TO Y
;   OUTPUT: IX  PTS TO PRODUCT
;   ERRORS: OVERFLOW (POS OR NEG)
;   REGISTERS:  DE IS MULTIPLICAND
;               AC IS MULTIPLIER AND ACCUM EXTENSION
;               HL IS ACCUM
;               B  IS LOOP COUNTER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IMUL    CALL MDLOAD     ; LOAD ARGS
        LD   A,B        ; X IN AC
;
; 16 BIT BY 16 BIT UNSIGNED MULTIPLY
;
        LD   HL,0       ; CLEAR ACCUM
        LD   B,16       ; SET COUNTER
        RRA             ; INITIALIZE CARRY
        RR   C
IM10    JR   NC,IM20    ; MPLIER BIT IS ZERO
        ADD  HL,DE      ; ADD MULTIPLICAND
IM20    RR   H          ; ROT ACC-MPLICAND INTO CY
        RR   L
        RRA
        RR   C
        DJNZ IM10       ; COUNTER NOT 0
;
; TEST FOR > 16 BIT PRODUCT
;
        LD   B,A        ; PRODUCT IN BC
        LD   A,H
        OR   L
        JR   NZ,INF?    ; OVERFLOW
;
; TEST FOR >15 BITS, AND STORE RESULTS
;
MDSTO   EX   AF,AF'     ; GET SIGN OF RESULT
        AND  80H
        JR   Z,MDS20    ; POS RESULT
;
; NEGATE BC
;
        LD   HL,0
        SBC  HL,BC
        JR   Z,MDS10    ; RESULT WAS ZERO
        JP   P,NINF     ; NEGATIVE OVERFLOW
MDS10   LD   (IX),L     ; STORE RESULT
        LD   (IX+1),H
        RET
;
; TEST FOR > 15 BITS
;
MDS20   BIT  7,B
        JR   NZ,PINF    ; POSITIVE OVERFLOW
        LD   (IX),C
        LD   (IX+1),B
        RET
;
```

```
; INF? CHECKS SIGN BIT IN A'
; PINF PRODUCES POS OVERFLOW
; NINF PRODUCES NEG OVERFLOW
;
IDV0    LD   HL,ERROR
        SET  4,(HL)      ; FLAG ERROR
INF?    EX   AF,AF'      ; GET SIGN BIT
        AND  80H
        JR   NZ,NINF     ; NEG SIGN
PINF    LD   (IX),0FFH   ; RESULT IS 7FFFH
        LD   (IX+1),7FH
        JR   FINF
NINF    LD   (IX),0      ; RESULT IS 8000H
        LD   (IX+1),80H
FINF    LD   HL,ERROR
        SET  6,(HL)      ; FLAG ERROR
        RET
*H INTEGER DIVIDE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; IDIV  INTEGER DIVIDE
;
;   INPUT:  IX  PTS TO X
;           IY  PTS TO Y
;   OUTPUT: IX  PTS TO X/Y
;   ERRORS: Y IS ZERO
;   ALGORITHM: NON-RESTORING DIVISION, 16 BY 15
;   REGISTERS:  AC IS DIVIDEND, ACCUM EXT
;               DE IS DIVISOR
;               HL IS ACCUM, REMAINDER AT END
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IDIV    CALL MDLOAD      ; LOAD ARGS
        LD   A,D
        OR   E
        JR   Z,IDV0      ; DIVISOR IS ZERO
        LD   A,B         ; X IN AC
;
; 16 BIT BY 15 BIT UNSIGNED DIVIDE
;
        LD   HL,0        ; CLEAR ACCUM
        LD   B,16        ; SET COUNTER
IDV10                    ; 0 <= HL < DE
        RL   C           ; ROT ACC-RESULT LEFT
        RLA
        ADC  HL,HL       ; NO CY POSSIBLE
        SBC  HL,DE       ; SUB DIVISOR
IDV20   CCF              ; CALC RESULT BIT
        JR   NC,IDV50    ; ACCUM WENT NEGATIVE
IDV30   DJNZ IDV10       ; COUNTER NOT 0
        JP   IDV60
IDV40                    ; -DE <= HL < 0
        RL   C           ; R0 ACC-RESULT LEFT
        RLA
        ADC  HL,HL
        AND  A           ; TURN OFF CY
        ADC  HL,DE       ; ADD DIVISOR
        JR   C,IDV30     ; ACCUM WENT POSITIVE
        JR   Z,IDV20     ; ACCUM BECAME ZERO
IDV50   DJNZ IDV40       ; COUNTER NOT 0
```

```
IDV60   RL   C           ; SHIFT IN LAST RESULT BIT
        RLA
        LD   B,A        ; QUOTIENT IN BC
        JP   MDSTO      ; STORE RESULT
;
; MDLOAD  LOADS X INTO BC, Y INTO DE
;   TAKES ABS OF EACH, STORES XOR OF SIGNS
;   IN A'
;
MDLOAD  LD   B,(IX+1)   ; LOAD X
        LD   C,(IX)
        LD   A,B
        AND  80H
        JR   Z,MDL10    ; X IS POSITIVE
;
; NEGATE X
;
        LD   HL,0
        SBC  HL,BC
        LD   B,H
        LD   C,L
;
; LOAD Y
;
MDL10   LD   D,(IY+1)
        LD   E,(IY)
        XOR  D          ; FORM SIGN OF RESULT
        EX   AF,AF'     ; SAVE IT
        BIT  7,D
        RET  Z          ; Y IS POSITIVE
;
; NEGATE Y
;
        LD   HL,0
        SBC  HL,DE
        EX   DE,HL
        RET
*H IABS AND ISGN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; IABS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IABS    BIT  7,(IX+1)
        RET  Z          ; X IS POSITIVE
INEG    LD   D,(IX+1)   ; LOAD X
        LD   E,(IX)
        LD   HL,0       ; NEGATE X
        AND  A
        SBC  HL,DE
        JP   PE,NINF    ; NEG OVERFLOW
        LD   (IX+1),H   ; STORE X
        LD   (IX),L
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; ISGN
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ISGN    LD   A,(IX+1)   ; GET SIGN
```

```
        BIT   7,A
        JR    Z,ISG10     ; X IS NONNEG
        LD    (IX),0FFH   ; STORE -1
        LD    (IX+1),0FFH
        RET
ISG10   OR    (IX)
        RET   Z           ; X IS ZERO
        LD    (IX+1),1    ; STORE +1
        LD    (IX),0
        RET
*H ICP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; ICP  INTEGER COMPARE
;
;   RETURNS CY SET IF X < Y
;   RETURNS Z  SET IF X = Y
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ICP     LD    B,(IX+1)    ; GET SIGN OF X
        LD    A,B
        AND   80H         ; TEST SIGN, CLEAR CY
        JR    NZ,ICP10    ; X IS NEG
        BIT   7,(IY+1)
        RET   NZ          ; Y IS NEG
        LD    A,B
        CP    (IY+1)      ; SIGNS ARE BOTH POS
        RET   NZ
        LD    A,(IX)
        CP    (IY)
        RET
ICP10   XOR   (IY+1)      ; TEST SIGN, CLEAR 7
        RLA               ; SIGN BIT INTO CY
        RET   C           ; SIGNS DIFFERENT
        LD    A,B
        CP    (IY+1)      ; BOTH SIGNS NEG
        RET   NZ
        LD    A,(IX)
        CP    (IY)
        RET
*H TABLE
;
; TABLE OF POWERS OF 10
;
LENTAB  EQU   11
LOTAB   EQU   $-5
        DEFW  0020H       ; FIRST 7 VALUES ARE INTEGERS
        DEFW  8400H
        DEFB  1
        DEFW  0048H
        DEFW  8700H
        DEFB  2
        DEFW  007AH
        DEFW  8A00H
        DEFB  3
        DEFW  401CH
        DEFW  8E00H
        DEFB  4
        DEFW  5043H
        DEFW  9100H
```

```
          DEFB 5
          DEFW 2474H
          DEFW 9400H
          DEFB 6
          DEFW 9618H
          DEFW 9880H
          DEFB 7
HITAB     EQU  $-5
          DEFW 0BC3EH      ; THIS VALUE EXACT
          DEFW 9B20H
          DEFB 8
          DEFW 1B0EH       ; THESE 3 VALUES ROUNDED
          DEFW 0B6CAH      ; (1.000000027E16)
          DEFB 16
          DEFW 0C253H      ; (1.000000014E24)
          DEFW 0D01CH
          DEFB 24
TOPTAB    DEFW 0C51DH      ; (1.000000033E32)
          DEFW 0EBAEH
          DEFB 32
;
; REFERENCES
;
          EXTERNAL MULT,DIVI,ERROR,OVER,UNDER,SEED
          EXTERNAL ZERO,ONE,FL032,FL0321,FLD,FIX32
          EXTERNAL RC2,LMARK,LREST,LBACK,LIN
          GLOBAL  DT0B,BT0D
          GLOBAL RND,RANDMZ,INT,ABS,SGN,NEG
          GLOBAL IADD,ISUB,IMUL,IDIV,IABS,ISGN,ICP,INEG
*H POWER
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; X TO THE Y POWER
;
;    INPUT:  IX  PTS TO X
;            IY  PTS TO Y
;    OUTPUT: IX  PTS TO X^Y
;    ERRORS: NONE DIRECTLY
;    SPECIAL CASES: IF Y IS 0, RETURN 1
;    ALGORITHM:  FOR INTEGER Y, A SEQUENCE OF
;                SQUARING AND MULTIPLYING BY X
;                PRODUCES THE RESULT
;         FOR NONINTEGER Y, RESULT IS EXP(Y*LOG(X))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; DETERMINE IF Y IS INTEGER
;
POW       PUSH IX          ; SAVE IX
          PUSH IY          ; COPY IY TO IX
          POP  IX
          CALL FIX32       ; GET INTEGER PART
          POP  IX          ; RESTORE IX
          OR   A
          JR   Z,POW10     ; Y WAS AN INTEGER
;
; FOR Y A NONINTEGER, TAKE EXP(Y*LOG(X))
;
          PUSH IY          ; SAVE IY
          CALL LOG         ; TAKE LOG OF X
```

```
        POP  IY
        CALL MULT       ; Y*LOG(X)
        CALL EXP
        RET
;
; FOR Y AN INTEGER, MULTIPLY
;
POW10   XOR  A          ; CHECK FOR ZERO
        CP   (IY+3)
        JP   Z,ONE      ; X TO 0 POWER
        PUSH HL         ; SAVE HL
        EXX             ; ALT BANK
        PUSH HL         ; SAVE H'L'
        BIT  7,B
        CALL NZ,REC     ; NEGATIVE POWER
        CALL ST01       ; SAVE COPY OF X IN TEMP1
        POP  HL         ; RESTORE INTEGER
        EXX             ; MAIN BANK
        POP  HL
        LD   B,32       ; LOOP COUNTER
;
; LOOK FOR FIRST 1 BIT
;
POW20   PUSH BC         ; SAVE COUNTER
        ADD  HL,HL      ; SHIFT LEFT 1
        EXX             ; ALT BANK
        ADC  HL,HL
        EXX             ; MAIN BANK
        JR   C,POW40    ; 1 BIT FOUND
        POP  BC         ; GET COUNT
        DJNZ POW20      ; LOOP NOT DONE
        JP   ONE        ; THIS JUMP SHOULD NOT HAPPEN
;
; SQUARE FOR EACH ZERO BIT
; SQUARE AND MULTIPLY BY X FOR EACH 1
;
POW30   PUSH BC         ; SAVE COUNT
        ADD  HL,HL      ; SHIFT LEFT 1
        PUSH HL         ; SAVE HL
        EXX             ; ALT BANK
        ADC  HL,HL
        PUSH HL         ; SAVE H'L'
        PUSH AF         ; SAVE CY
        PUSH IX         ; COPY IX TO IY
        POP  IY
        CALL MULT       ; SQUARE
        POP  AF         ; GET CY BIT
        JR   NC,POW35   ; ZERO BIT WAS SHIFTED
        LD   IY,TEMP1
        CALL MULT       ; MULTIPLY BY X
POW35   POP  HL         ; RESTORE H'L'
        EXX             ; MAIN BANK
        POP  HL         ; RESTORE HL
POW40   POP  BC         ; GET COUNT
        DJNZ POW30      ; LOOP NOT DONE
        RET
;
; INTEGER VERSION, FLOATS ARGS FIRST, THEN
;   CALLS POW, FIXING RESULT.  USES TEMP2,
;   TEMP3 SINCE POW WILL FIND Y AN INTEGER
```

```
;
IPOW    PUSH IX ; SAVE ORIGINAL PTR
        LD   L,(IY)      ; GET Y
        LD   H,(IY+1)
        LD   IX,TEMP3
        CALL FLOAT1      ; FLOAT Y VALUE IN TEMP3
        POP  IX ; GET ORIGINAL PTR
        PUSH IX
        LD   L,(IX)      ; GET X
        LD   H,(IX+1)
        LD   IX,TEMP2
        CALL FLOAT1      ; FLOAT X VALUE IN TEMP2
        LD   IY,TEMP3    ; PT TO Y
        CALL POW         ; USE REAL ROUTINE
        CALL FIX         ; FIX RESULT
        POP  IX          ; GET ORIGINAL IX
        LD   (IX),L      ; STORE RESULT
        LD   (IX+1),H
        RET
*H SQUARE ROOT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; SQUARE ROOT FUNCTION
;
;   INPUT:  IX  PTS TO X
;   OUTPUT: IX  PTS TO SQR(X)
;   ERRORS: X IS NEGATIVE, RETURNS SQR(ABS(X))
;   SPECIAL CASES: X IS 0
;   ALGORITHM:  INPUT IS SCALED TO .25 <= X < 1
;       BY REMOVING AN EVEN EXPONENT.  HALF THE EVEN
;       EXPONENT IS ADDED TO THE RESULT EXPONENT
;       AN INITIAL APPROXIMATION IS GIVEN BY
;       A 1ST ORDER POLYNOMIAL, AND IMPROVED TO
;       REQUIRED ACCURACY IN 3 ITERATIONS OF
;       NEWTON'S METHOD.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CHECK VALID ARGUMENTS
;
SQR     LD   A,(IX+3)    ; GET EXP
        OR   A
        JP   Z,ZERO      ; ZERO INPUT
        BIT  7,(IX)
        CALL NZ,SQERR    ; NEGATIVE ARG
;
; EXP OF RESULT IS ROUND(ORIGINAL EXP/2)
; GIVE X EITHER 0 OR -1 EXPONENT
;
        SUB  80H         ; REMOVE BIAS
        LD   B,A         ; SAVE UNBIASED EXP
        SRA  A           ; DIVIDE BY 2
        ADC  A,0         ; ROUND
        PUSH AF          ; SAVE EXP OF RESULT
        ADD  A,A         ; MULT BY 2
        SUB  B           ; SUB ORIGINAL EXP
        NEG
        ADD  A,80H       ; RESTORE BIAS
;
; CALC INITIAL APPROX
```

```
;
        DEC   A           ; DIVIDE BY 2
        LD    (IX+3),A
        CALL  ST01        ; SAVE X/2 IN TEMP1
        LD    IY,PSQ00
        CALL  ADD
;
; PERFORM 3 ITERATIONS
;
        LD    B,3
SQ10    PUSH  BC          ; SAVE COUNTER
        CALL  ST02        ; SAVE LAST APPROX IN TEMP2
        LD    HL,TEMP1
        CALL  LD          ; GET X/2
        LD    IY,TEMP2
        CALL  DIVI        ; DIVIDE BY LAST APPROX
        DEC   (IY+3)      ; DIVIDE LAST APPROX BY 2
        CALL  ADD         ; NEW APPROX
        POP   BC          ; GET COUNTER
        DJNZ  SQ10        ; LOOP NOT DONE
;
; STORE EXPONENT
;
        POP   AF          ; GET SAVED EXP
        ADD   A,80H       ; ADD BIAS
        LD    (IX+3),A
        RET
;
; SQUARE ROOT OF NEGATIVE NUMBER
;
SQERR   LD    HL,ERROR
        SET   3,(HL)      ; FLAG ERROR
        RES   7,(IX)      ; TAKE ABS VALUE
        RET
*H EXP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; EXPONENTIATION ROUTINE
;
;   INPUT:  IX  PTS TO X
;   OUTPUT: IX  PTS TO E↑X
;   ERRORS: OVER/UNDERFLOW IN HANDLING EXPONENT PART
;   SPECIAL CASES:  INTERNALLY, FRACTION PART IS ZERO
;   ALGORITHM:  MULTIPLY BY LOG BASE 2 OF E AT BEGINNING
;       THEN CALCULATE 2↑X.  INTEGER PART OF X IS ADDED
;       TO EXPONENT OF RESULT.  FRACTION PART IS SCALED
;       TO 0 <= X < .25  BY REMOVING HIGH ORDER 2 BITS
;       AND SCALING BY 1, 2↑.75, 2↑.5, OR 2↑.25 AT END.
;       APPROXIMATION TO 2↑X IS GIVEN BY:
;               (Q(X↑2) + X*P(X↑2)) / (Q(X↑2) - X*P(X↑2))
;       WHERE Q IS 1ST ORDER POLYNOMIAL, P IS 0TH ORDER.
;       REFERENCE IS "COMPUTER APPROXIMATIONS", HART, JOHN F.
;          ET AL, FUNCTION NUMBER 1080
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CONVERT E↑X TO 2↑X
;
EXP     LD    IY,LGE4     ; MULT BY 4*LOG2(E)
        CALL  MULT
```

```
;
; GET 8 BIT 2'S COMPL INTEGER PART IN A
; GET HIGH 2 BITS OF FRACTION PART IN DE
;
        CALL FIXA       ; GET 16 BIT INTEGER PART
        JP   C,EXERR1   ; INTEGER OVERFLOW
        PUSH HL         ; SAVE INTEGER
        CALL BRK        ; CALC FRACTION PART
        POP  HL         ; RESTORE INTEGER
        SRA  H          ; DIVIDE INTEGER BY 4
        RR   L
        RRA             ; SAVE 2 BITS IN A
        SRA  H
        RR   L
        RLA
        RLA             ; 2 BITS NOW IN A
        RLA             ; MULT BY 4
        RLA
        AND  0CH
        LD   E,A
        LD   D,0        ; DE IS NOW TABLE DISPLACEMENT
        LD   A,H        ; CHECK SIZE OF INTEGERR
        RLC  L          ; SIGN BIT IN CY
        ADC  A,0
        JR   NZ,EXERR2  ; INTEGER WAS > 8 BIT 2'S COMP
        LD   A,L
        RRCA            ; RESTORE 8 BIT INTEGER
        PUSH AF         ; SAVE INTEGER PART
        PUSH DE         ; SAVE NEXT TWO BITS
        XOR  A
        CP   (IX+3)
        JR   Z,EX10     ; FRACTION PART IS ZERO
        DEC  (IX+3)     ; SCALE TO 0 <= X < 1/4
        DEC  (IX+3)     ; UNDERFLOW CANNOT OCCUR HERE
;
; EVALUATE RATIONAL APPROX. TO 2^X
;
EX10    CALL ST01       ; SAVE A COPY OF X IN TEMP1
        LD   IY,PEX00
        CALL MULT
        CALL ST02       ; SAVE X*P(X^2) IN TEMP2
        LD   HL,TEMP1
        CALL LD         ; GET ANOTHER COPY OF X
        LD   IY,TEMP1
        CALL MULT       ; CALC X^2
        LD   IY,QEX00
        CALL ADD
        CALL ST01       ; SAVE Q(X^2) IN TEMP1
        LD   IY,TEMP2
        CALL SUB
        CALL ST03       ; SAVE Q-X*P IN TEMP3
        LD   HL,TEMP1
        CALL LD         ; GET Q
        LD   IY,TEMP2
        CALL ADD        ; CALC Q+X*P
        LD   IY,TEMP3
        CALL DIVI       ; FINAL RESULT
;
; ADD IN FRACTIONAL AND INTEGER POWERS OF 2
;
```

```
        POP   DE          ; GET TABLE DISPL
        LD    IY,EXTAB
        ADD   IY,DE       ; POINT TO FRACTIONAL POWER
        LD    A,E
        OR    A
        CALL  NZ,MULT     ; MULTIPLY BY TAB ENTRY
;
; ADD INTEGER PART TO EXPONENT OF RESULT
;
        POP   BC          ; INTEGER PART IN B
        LD    A,(IX+3)    ; GET EXP
        SUB   80H         ; REMOVE BIAS
        ADD   A,B
        JP    PE,EXERR3   ; OVER/UNDERFLOW
        ADD   A,80H       ; RESTORE BIAS
        JP    Z,UNDER     ; UNDERFLOW
        LD    (IX+3),A
        RET
;
; EXP ERRORS
;
EXERR1  LD    H,(IX)      ; GET ORIGINAL SIGN
EXERR2  BIT   7,H         ; TEST SIGN
        JP    NZ,UNDER    ; NEGATIVE SIGN
        JP    OVER
EXERR3  JP    P,UNDER     ; POS SIGN AFTER OVERFLOW
        JP    OVER
*H LOGARITHM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; LOG ROUTINE
;
;    INPUT:  IX  PTS TO X
;    OUTPUT: IX  PTS TO NATURAL LOG OF X
;    ERRORS: X IS NEGATIVE, RETURN LOG(ABS(X))
;            X IS ZERO, RETURN NEGATIVE OVERFLOW
;    SPECIAL CASES: X IS 1, RETURN ZERO
;    ALGORITHM: EXPONENT IS STIPPED OFF, FLOATED, AND
;        ADDED TO RESULT AT END.  REMAINING FRACTION
;        PART IS MULTIPLIED BY SQR(2) TO GIVE RANGE
;        OF SQR(2)/2 <= X < SQR(X).  LOG FUNCTION
;        IS APPROXIMATED BY:
;                Z*P(Z^2)
;        WHERE Z = (X-1) / (X+1)  AND P IS 2ND ORDER.
;        REFERENCE IS FUNCTION 2661.
;        LOG(SQR(2)) IS SUBTRACTED FROM RESULT TO CORRECT
;        FOR SCALING.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CHECK FOR VALID ARGUMENTS
;
LOG     BIT   7,(IX)
        CALL  NZ,LGERR1   ; NEGATIVE ARG
        LD    IY,UNITY
        CALL  CP
        JP    Z,ZERO      ; LOG(1)
        LD    A,(IX+3)
        OR    A
        JP    Z,LGERR2    ; ZERO ARG
```

```
;
; MAKE EXP INTO 16 BIT 2'S COMPLEMENT
;
        SUB  80H        ; REMOVE BIAS
        LD   L,A        ; PUT IN HL
        LD   H,0
        JR   NC,LO10    ; SIGN IS POSITIVE
        LD   H,0FFH     ; MAKE NEGATIVE
;
; CONVERT EXP TO LOG BASE E, SAVE IN TEMP3
;
LO10    PUSH IX         ; SAVE ACC PTR
        LD   IX,TEMP3
        CALL FLOAT1     ; FLOAT HL
        LD   IY,LN2
        CALL MULT       ; MULT BY LOG(2)
        POP  IX
;
; SCALE TO 1/SQR(2) <= X < SQR(2)
;
        LD   (IX+3),80H ; 1/2 <= X < 1
        LD   IY,SQR2
        CALL MULT       ; MULT BY SQR(2)
;
; APPROXIMATION FOR LOGARITHM
;
        CALL STO1       ; SAVE X IN TEMP1
        PUSH IX         ; SAVE PTR TO ACCUM
        LD   IX,TEMP1
        LD   IY,UNITY
        CALL ADD        ; X+1 IN TEMP1
        POP  IX
        CALL SUB        ; X-1 IN ACCUM
        LD   IY,TEMP1
        CALL DIVI
        CALL STO1       ; (X-1)/(X+1) IN TEMP1
        CALL MULT
        CALL STO2       ; Z^2 IN TEMP2
        LD   IY,PL002   ; CALC P(Z^2)
        CALL MULT
        LD   IY,PL001
        CALL ADD
        LD   IY,TEMP2
        CALL MULT
        LD   IY,PL000
        CALL ADD
        LD   IY,TEMP1   ; CALC Z * P(Z^2)
        CALL MULT
        LD   IY,LNSQ2
        CALL SUB        ; SUB LOG(SQR(2))
        LD   IY,TEMP3
        CALL ADD        ; ADD EXP PART
        RET
;
; ERRORS
;
LGERR1  LD   HL,ERROR
        SET  2,(HL)     ; FLAG LOG OF NEG
        RES  7,(HL)     ; TAKE ABS VALUE
        RET
```

```
LGERR2  SET  7,(IX)      ; SET NEGATIVE SIGN
        JP   OVER        ; LOG OF ZERO
*H TANGENT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; TANGENT ROUTINE
;
;   INPUT:  IX  PTS TO X (IN RADIANS)
;   OUTPUT: IX  PTS TO TAN(X)
;   ERRORS: NONE DIRECTLY
;   ALGORITHM:  RANGE REDUCTION TO 0 <= X <= PI/4.
;       MULTIPLY BY 4/PI.  LOW 3 BITS OF INTEGER
;       PART GIVE THE OCTANT NUMBER, AND IS USED
;       AS A FLAG.  TAKE FRACTIONAL PART, IGNORE
;       INTEGER PART (EXCEPT FLAG BITS).
;       OCTANTS 1,3,5,7 FORM 1 - FRACTION PART.
;       APPROXIMATION IS GIVEN BY:
;               X*P(X↑2) / Q(X↑2)
;       WHERE P IS 1ST ORDER AND Q IS 2ND ORDER POLY.
;       REFERENCE FUNCTION IS 4282.
;       OCTANTS 1,2,5,6 FORM RECIPROCAL OF RESULT.
;       OCTANTS 2,3,6,7 RETURN NEGATIVE SIGN.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; SCALE BY 4/PI
;
TAN     LD   IY,FOVRPI
        CALL MULT
        CALL BRK         ; BREAK INTEGER AND FRACTION
        LD   A,L
        PUSH AF          ; SAVE FLAG BITS
        AND  1           ; TEST FLAG
        JR   Z,TA10      ; THEY WERE 0 OR 2
        LD   IY,UNITY
        CALL RSUB        ; FORM 1 - FRACTION PART
;
; TAN APPROX
;
TA10    CALL ST03        ; SAVE X IN TEMP3
        LD   IY,TEMP3
        CALL MULT
        CALL ST02        ; SAVE X↑2 IN TEMP2
        LD   IY,QTA01
        CALL ADD
        LD   IY,TEMP2
        CALL MULT        ; MULT BY X↑2
        LD   IY,QTA00
        CALL ADD
        CALL ST01        ; SAVE Q(X↑2) IN TEMP1
        LD   HL,TEMP2
        CALL LD          ; GET X↑2
        LD   IY,PTA01
        CALL MULT
        LD   IY,PTA00
        CALL ADD
        LD   IY,TEMP3
        CALL MULT        ; X * P(X↑2) IN ACCUM
        POP  AF          ; GET 2 BIT FLAG
        PUSH AF
```

```
        AND  3
        JP   PE,TA20     ; THEY WERE 0 OR 3
        CALL ST02        ; SAVE X*P IN TEMP2
        LD   HL,TEMP1
        CALL LD          ; GET Q
        LD   IY,TEMP2    ; SET FOR Q/XP
        JP   TA30
TA20    LD   IY,TEMP1    ; SET FOR XP/Q
TA30    CALL DIVI
;
; GET SIGN OF RESULT
;
        POP  AF          ; GET 2 BIT FLAG
        AND  2
        RET  Z           ; ON 0 OR 1
        SET  7,(IX)      ; MAKE RESULT NEG
        RET
*H COSINE, SINE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; COSINE ROUTINE
;   EFFECTIVELY ADDS PI/2 AND CALLS SINE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
COS     LD   IY,FOVRPI
        CALL MULT         ; SCALE BY 4/PI
        CALL BRK         ; SEPARATE INTEGER FROM FRACTION
        LD   A,L
        ADD  A,2         ; ADD EFFECTIVE PI/2
        JP   SI10
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; SINE ROUTINE
;
;   INPUT:  IX  PTS TO X (IN RADIANS)
;   OUTPUT: IX  PTS TO SIN(X)
;   ERRORS: NONE
;   ALGORITHM:  RANGE REDUCTION TO 0 <= X <= PI/4
;       BY MULTIPLYING BY 4/PI AND TAKING FRACTION
;       PART.  LOW 3 BITS OF INTEGER PART GIVE
;       OCTANT NUMBER.
;       OCTANTS 1,3,5,7 FORM 1 - FRACTION PART
;       OCTANTS 0,3,4,7 USE SINE APPROXIMATION:
;               X*P(X↑2) / Q(X↑2)
;       WHERE P IS 2ND ORDER AND Q IS 1ST ORDER POLY.
;       REFERENCE FUNCTION IS 3060.
;       OCTANTS 1,2,5,6 USE COSINE APPROXIMATION:
;               P(X↑2) / Q(X↑2)
;       WHERE P IS 2ND ORDER AND Q IS 1ST ORDER POLY.
;       REFERENCE FUNCTION IS 3840.
;       OCTANTS 4,5,6,7 RETURN NEGATIVE SIGN.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SIN     LD   IY,FOVRPI
        CALL MULT        ; SCALE BY 4/PI
        CALL BRK         ; GET INTEGER AND FRACTION
        LD   A,L
SI10    PUSH AF          ; SAVE 3 BIT FLAG
        AND  1
        JR   Z,SI20      ; 0,2,4, OR 6
```

```
        LD   IY,UNITY
        CALL RSUB        ; CALC 1 - FRACTION PART
SI20    POP  AF          ; GET 3 BIT FLAG
        PUSH AF
        AND  3
        JP   PO,SI30     ; 1,2,5 OR 6
;
; SIN APPROX
;
        CALL ST03        ; SAVE X IN TEMP3
        LD   IY,TEMP3
        CALL MULT        ; SQUARE X
        CALL ST02        ; SAVE X↑2 IN TEMP2
        LD   IY,QSI00
        CALL ADD
        CALL ST01        ; SAVE Q(X↑2) IN TEMP1
        LD   HL,TEMP2
        CALL LD          ; GET X↑2
        LD   IY,PSI02
        CALL MULT
        LD   IY,PSI01
        CALL ADD
        LD   IY,TEMP2
        CALL MULT
        LD   IY,PSI00
        CALL ADD
        LD   IY,TEMP1    ; PT TO Q
        CALL DIVI        ; P/Q
        LD   IY,TEMP3
        CALL MULT
        JP   SI40
;
; COS APPROX
;
SI30    PUSH IX          ; COPY IX TO IY
        POP  IY
        CALL MULT
        CALL ST02        ; SAVE X↑2 IN TEMP2
        LD   IY,QC000
        CALL ADD
        CALL ST01        ; SAVE Q(X↑2) IN TEMP1
        LD   HL,TEMP2
        CALL LD          ; GET X↑2
        LD   IY,PC002
        CALL MULT
        LD   IY,PC001
        CALL ADD
        LD  IY,TEMP2
        CALL MULT
        LD   IY,PC000
        CALL ADD
        LD   IY,TEMP1
        CALL DIVI
SI40    POP  AF          ; GET 3 BIT FLAG
        AND  4
        RET  Z           ; 4,5,6 OR 7
        SET  7,(IX)      ; NEGATE RESULT
        RET
*H ARCTANGENT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;
; ARCTANGENT ROUTINE
;
;   INPUT:  IX  PTS TO X
;   OUTPUT: IX  PTS TO ATN(X)  (IN RADIANS)
;   ERRORS: NONE
;   ALGORITHM:  RESULT WILL HAVE SIGN OF INPUT.
;       RANGE REDUCTION BY TAKING ABS VALUE AND
;       TESTING FOR 1 OF 3 REGIONS:
;       1)      X < TAN(PI/8)  USE APPROX. DIRECTLY.
;       2)      X < TAN(3*PI/8)  REPLACE X BY
;               (X-1) / (X+1) AND ADD PI/4 AT END
;       3)      REPLACE X BY 1/X AND SUB FROM PI/2
;               AT END.
;       APPROXIMATION FOR 0 <= X <= TAN(PI/8):
;               X*P(X↑2) / Q(X↑2)
;       WHERE P IS 1ST ORDER AND Q 2ND ORDER POLY.
;       REFERENCE FUNCTION IS 5071.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; TAKE ABS VALUE, DETERMINE RANGE
;
ATN     LD   A,(IX)     ; GET SIGN
        AND  80H
        PUSH AF         ; SAVE SIGN
        RES  7,(IX)     ; MAKE X POSITIVE
        LD   IY,TAPI8
        CALL CP         ; COMPARE TO 1ST LIMIT
        LD   B,0        ; SET FLAG TO 0
        JR   C,AT20     ; < TAN(PI/8)
        LD   IY,TA3PI8
        CALL CP         ; COMPARE TO 2ND LIMIT
        JR   C,AT10     ; < TAN(3*PI/8)
;
; TAN(3*PI/8) <= X
;    TAKE 1/X NOW, SUB FROM PI/2 AT END
;
        CALL REC        ; TAKE RECIPROCAL
        LD   B,2        ; SET FLAG TO 2
        JP   AT20
;
; TAN(PI/8) <= X < TAN(3*PI/8)
;   CALC (X-1)/(X+1) NOW,
;   ADD PI/4 AT END
;
AT10    CALL STO1       ; SAVE X IN TEMP1
        PUSH IX         ; SAVE PTR TO ACCUM
        LD   IX,TEMP1
        LD   IY,UNITY
        CALL ADD        ; X+1 IN TEMP1
        POP  IX
        CALL SUB        ; X-1 IN ACCUM
        LD   IY,TEMP1
        CALL DIVI
        LD   B,1        ; SET FLAG TO 1
;
; APPROX FOR ARCTAN
;
AT20    PUSH BC         ; SAVE FLAG
```

```
        CALL STO3       ; SAVE X IN TEMP3
        LD   IY,TEMP3
        CALL MULT
        CALL STO2       ; SAVE X↑2 IN TEMP2
        LD   IY,QAT01
        CALL ADD
        LD   IY,TEMP2
        CALL MULT
        LD   IY,QAT00
        CALL ADD
        CALL STO1       ; SAVE Q(X↑2) IN TEMP1
        LD   HL,TEMP2
        CALL LD         ; GET X↑2
        LD   IY,PAT01
        CALL MULT
        LD   IY,PAT00
        CALL ADD
        LD   IY,TEMP1
        CALL DIVI
        LD   IY,TEMP3
        CALL MULT
        POP  BC         ; GET FLAG
        BIT  0,B
        JR   Z,AT30     ; 1ST OR 3RD INTERVAL
; SECOND INTERVAL
        LD   IY,PI4
        CALL ADD        ; ADD PI/4
        JP   AT40
AT30    BIT  1,B
        JR   Z,AT40     ; 1ST INTERVAL
; THIRD INTERVAL
        LD   IY,PI2
        CALL RSUB       ; SUB FROM PI/2
;
; GIVE RESULT SIGN OF ARG
;
AT40    POP  AF         ; GET ORIGINAL SIGN
        OR   (IX)
        LD   (IX),A
        RET
*H CONSTANTS
;
; CONSTANTS
;
UNITY   DEFW 0          ; FL PT 1.0
        DEFW 8100H
LGE4    DEFW 0AA38H     ; 4 * LOG BASE 2 OF E
        DEFW 833BH      ; 5.770780164
PEX00   DEFW 950AH      ; P COEFFICIENTS
        DEFW 84D8H      ; 8.6615859416262
QEX00   DEFW 0EF47H     ; Q COEFFICIENTS
        DEFW 85BFH      ; 24.9920639458976
                        ; QEX01 IS 1.0
EXTAB   EQU  $-4        ; FRACTIONAL POWERS OF 2
        DEFW 3718H      ; 2↑1/4
        DEFW 81F0H      ; 1.189207115
        DEFW 0435H      ; 2↑1/2
        DEFW 81F3H      ; 1.414213562
        DEFW 4457H      ; 2↑3/4
        DEFW 81FDH      ; 1.681792831
```

```
PSQ00   DEFW 0F655H     ; SQR INIT APPROX
        DEFW 7F1AH      ; .4178932188
LN2     DEFW 7231H      ; LOG BASE E OF 2
        DEFW 8018H      ; .6931471806
LNSQ2   DEFW 7231H      ; LOG BASE E OF SQR(2)
        DEFW 7F18H      ; .3465735903
SQR2    EQU  EXTAB+8    ; SQUARE ROOT OF 2
PL000   DEFW 0000H      ; P COEFFICIENTS
        DEFW 8204H      ; 2.0000008368
PL001   DEFW 982AH
        DEFW 80D1H      ; .6664400777
PL002   DEFW 9254H
        DEFW 7F22H      ; .41517739
F0VRPI  DEFW 0F922H     ; 4/PI
        DEFW 8183H      ; 1.273239545
PTA00   DEFW 6C54H      ; P COEFFICIENTS
        DEFW 88A9H      ; 212.42445758263
PTA01   DEFW 0DAC8H
        DEFW 844EH      ; -12.55329742424
QTA00   DEFW 3B07H      ; Q COEFFICIENTS
        DEFW 89CEH      ; 270.46722349399
QTA01   DEFW 318FH
        DEFW 872FH      ; -71.59606050466
; QTA02 IS 1.0
PSI00   DEFW 4653H      ; P COEFFICIENTS
        DEFW 863FH      ; 52.81860134812
PSI01   DEFW 0A294H
        DEFW 8335H      ; -4.644800481954
PSI02   DEFW 0AC31H
        DEFW 7D59H      ; .0867545069521
QSI00   DEFW 8006H      ; Q COEFFICIENTS
        DEFW 8760H      ; 67.250731777791
; QSI01 IS 1.0
PC000   DEFW 0BF3EH     ; P COEFFICIENTS
        DEFW 86CAH      ; 47.68729218663
PC001   DEFW 53DBH
        DEFW 84F8H      ; -13.70800004765
PC002   DEFW 4865H
        DEFW 7FF8H      ; .447822343969
QC000   DEFW 0BF3EH     ; Q COEFFICIENTS
        DEFW 86C9H      ; 47.68729082658
; QC001 IS 1.0
PAT00   DEFW 8F4AH      ; P COEFFICIENTS
        DEFW 844EH      ; 12.65998609915
PAT01   DEFW 0D04BH
        DEFW 8365H      ; 6.3691887127
QAT00   DEFW 8F4AH      ; Q COEFFICIENTS
        DEFW 844EH      ; 12.65998646243
QAT01   DEFW 6D29H
        DEFW 8400H      ; 10.5891113168
; QAT02 IS 1.0
TAPI8   DEFW 1354H      ; TAN(PI/8)
        DEFW 7FCDH      ; .4142135624
TA3PI8  DEFW 821AH      ; TAN(3*PI/8)
        DEFW 827AH      ; 2.414213562
PI2     DEFW 0F49H      ; PI / 2
        DEFW 81DBH      ; 1.570796327
PI4     DEFW 0F49H      ; PI / 4
        DEFW 80DBH      ; .7853981634
;
```

```
; REFERENCES
;
        GLOBAL  POW,IPOW,SQR,EXP,LOG,SIN,COS,TAN,ATN
        EXTERNAL        OVER,UNDER,ZERO,ERROR,ONE
        EXTERNAL        FIX32,FLO32,ADD,SUB,RSUB
        EXTERNAL        MULT,DIVI,REC,STO1,STO2
        EXTERNAL        STO3,TEMP1,TEMP2,TEMP3,LD
        EXTERNAL        CP,FIX,FIXA,FLOAT,BRK,FLOAT1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; MISCELLANEOUS ROUTINES REQUIRED AS EXTERNALS FOR MATH PACKAGE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
LMARK   LD   (POINT),HL ; STORE PTR FOR DTOB
        RET
LREST   RET             ; ENTRY FOR UNSUCCESSFUL DTOB (NO DIGITS FOUND)
LBACK   RET             ; DUMMY ROUTINE (BACK UP PTR FOR BASIC)
RC2     RET             ; NORMAL RETURN, IF SOME KIND OF NUMBER FOUND
LIN     PUSH HL         ; SAVE HL
        LD   HL,(POINT) ; GET PTR
LIN10   LD   A,(HL)     ; GET NEXT CHAR
        INC  HL         ; ADVANCE PTR
        CP   ' '
        JR   Z,LIN10    ; IGNORE BLANKS
        LD   (POINT),HL ; SAVE PTR
        POP  HL         ; RESTORE HL
        RET
SCP     RET             ; STRING COMPARE
POINT   DEFS 2          ; TEMP PTR STORAGE
ERROR   DEFB 0          ; ERROR FLAGS FOR MATH PACKAGE
ILLNUM  EQU  7          ; ILLEGAL NUMBER (DTOB)
OVRFLW  EQU  6          ; OVERFLOW
UNDFLW  EQU  5          ; UNDERFLOW
DVDZRO  EQU  4          ; DIVISION BY ZERO
SQRNGN  EQU  3          ; SQUARE ROOT OF NEGATIVE NUMBER
LOGNGN  EQU  2          ; LOG OF NEGATIVE NUMBER
;
; REFERENCES
;
        GLOBAL          ERROR,SCP,LMARK,LREST,LBACK,RC2,LIN
```

```
*H ADD AND SUBTRACT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT REVERSE SUBTRACT ROUTINE
;   NEGATE X AND ADD
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
RSUB    LD   A,(IX)      ; GET MS BYTE OF X
        XOR  80H         ; NEGATE SIGN
        LD   D,A         ; SAVE IT
        LD   E,(IY)      ; GET MS BYTE OF Y
        JP   AD05
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT SUBTRACT ROUTINE
;   NEGATE Y AND ADD
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SUB     LD   A,(IY)      ; GET MSBYTE OF Y
        XOR  80H         ; NEGATE SIGN
        LD   E,A         ; SAVE IT
        LD   D,(IX)      ; GET MSBYTE OF X
        JP   AD05
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT ADD ROUTINE
;
;   INPUT:  IX  PTS TO 1ST ARG
;           IY  PTS TO 2ND ARG
;   OUTPUT: IX  PTS TO SUM (X IS REPLACED BY SUM, Y IS UNCHANGED)
;   ERRORS: EXPONENT OVERFLOW AND UNDERFLOW
;   SPECIAL CASES: ZERO ARGUMENT(S)
;
;   INPUT FOR AD05: D CONTAINS MS BYTE OF X
;                   E CONTAINS MS BYTE OF Y
;   ALGORITHM:  ABSOLUTE VALUE MAGNITUDES ARE COMPARED,
;       THE LARGER IS LOADED TO AC1, THE SMALLER TO AC2.
;       AC2 IS SHIFTED TO ALIGN DECIMAL POINTS.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ADD     LD   D,(IX)      ; GET MS BYTES
        LD   E,(IY)
;
; CHECK FOR ZERO ARGUMENTS
;
AD05    XOR  A           ; CLEAR A
        CP   (IX+7)
        JP   NZ,AD07     ; X IS NONZERO
        CP   (IY+7)
        RET  Z           ; Y IS ALSO ZERO
        LD   A,E         ; GET Y MSBYTE
        PUSH IY
        POP  HL
        CALL LD          ; COPY Y TO X
        LD   (IX),A      ; STORE MSBYTE
        RET
AD07    LD   (IX),D      ; STRE NEW SIGN IN CASE Y IS 0
        CP   (IY+7)
        RET  Z           ; Y IS ZERO, RETURN X
;
```

```
; LOAD ARGUMENTS, LARGER MAGNITUDE IN AC1
;
        PUSH DE         ; SAVE MS BYTES
        RES  7,D        ; CLEAR SIGN BITS
        RES  7,E
        CALL CP1        ; COMPARE X TO Y IGNORING SIGN
        JR   C,AD10     ; X < Y
        CALL STA1       ; X TO AC1
        CALL STYA2      ; Y TO AC2
        POP  BC         ; GET SIGNS
        LD   A,B
        XOR  C
        LD   A,B        ; USE SIGN OF X FOR RESULT
        JP   AD15
AD10    CALL STYA1      ; Y TO AC1
        CALL STA2       ; X TO AC2
        POP  BC         ; GET SIGNS
        LD   A,C
        XOR  B
        LD   A,C        ; USE SIGN OF Y FOR RESULT
;
; PRENORMALIZE BY SHIFTING AC2 RIGHT
;
AD15    EX   AF,AF'     ; SAVE RESULT SIGN, XOR FLAGS
        LD   A,(AC1E)   ; GET AC1 EXPONENT
        LD   HL,AC2E
        SUB  (HL)       ; CALC SHIFT COUNT
        JR   Z,AD30     ; FRACTIONS ALREADY ALIGNED
        CP   13
        JP   NC,FST0    ; AC2 IS INSIGNIFICANT
        LD   B,A        ; SET COUNTER
AD20    XOR  A          ; CLEAR A
        LD   HL,AC2     ; PT TO ACCUM
        CALL SRD1       ; SHFT RT 1 DIGIT
        DJNZ AD20       ; COUNTER NOT 0
;
; INITIALIZE FOR ADD OR SUB
;
AD30    LD   B,7        ; COUNTER
        LD   HL,AC20    ; POINTERS
        LD   DE,AC10
        AND  A          ; CLEAR CY
        EX   AF,AF'     ; GET SIGN & FLAGS
        JP   M,SB10     ; SIGNS WERE DIFFERENT
;
; ADD
;
        EX   AF,AF'     ; SAVE SIGN OF RESULT
AD40    LD   A,(DE)
        ADC  A,(HL)
        DAA
        LD   (DE),A
        DEC  DE
        DEC  HL
        DJNZ AD40
;
; POST NORMALIZE, AT MOST 1 SHFT RIGHT
;
        LD   A,(AC1)    ; CHECK FOR OVERFLOW
        AND  0F0H
```

```
        JP   Z,FSTO      ; ALREADY NORMALIZED
        LD   HL,AC1E
        INC  (HL)        ; INCREMENT EXPONENT
        JP   Z,OVERA     ; OVERFLOW
        LD   HL,AC1      ; PT TO ACCUM
        CALL SRD1        ; SHIFT RIGHT
        JP   FSTO
;
; SUB
;
SB10    EX   AF,AF'      ; SAVE SIGN OF RESULT
SB20    LD   A,(DE)
        SBC  A,(HL)
        DAA
        LD   (DE),A
        DEC  DE
        DEC  HL
        DJNZ SB20
;
; POST-NORMALIZE BY SHIFTING LEFT
;
        LD   A,(AC1E)    ; PUT EXP IN C
        LD   C,A
        LD   HL,AC1
        CALL NORML2      ; NORMALIZE FRACTION
        JP   Z,ZERO      ; ZERO FRACTION RESULT
        JP   C,UNDER     ; EXPONENT UNDERFLOW
        LD   A,C         ; STORE EXP
        LD   (AC1E),A
        JP   FSTO
;
; COPY AC1 TO (IX) AND SET SIGN OF RESULT
;
FSTO    LD   HL,AC1      ; PT TO AC1
        CALL LD
        JP   RSTO        ; SET SIGN FROM A'
*H MULTIPLY
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT MULTIPLY ROUTINE
;
;   INPUT:  IX  PTS TO MULTIPLICAND
;           IY  PTS TO MULTIPLIER
;   OUTPUT: IX  PTS TO PRODUCT (PRODUCT REPLACES X, Y IS UNCHANGED)
;   ERRORS: EXPONENT OVERFLOW OR UNDERFLOW
;   SPECIAL CASES:  ARGUMENT(S) ZERO
;   ALGORITHM:  X, 2X, 4X, AND 8X ARE STORED IN TEMPORARY REGS.
;       Y IS PLACED IN AC1.
;       MULTIPLIER IS EXAMINED BIT AT A TIME WITHIN EACH DIGIT,
;       CORRESPONDING TEMP REGS ARE ADDED TO (IX), AND (IX)
;       IS SHIFTED RIGHT.
;
;   TIMING:  APPROX 19814 (16916 IN LOOP)
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CHECK FOR ZERO OPERANDS
;
MULT    XOR  A
        CP   (IX+7)
```

```
        JP    Z,ZERO
        CP    (IY+7)
        JP    Z,ZERO
;
; CALCULATE AND SAVE SIGN OF RESULT
;
        LD    A,(IX)      ; MS BYTE OF X
        XOR   (IY)
        EX    AF,AF'      ; SAVE IN A'
;
; STORE X, 2X, 4X, AND 8X IN TEMP REGISTERS
;
        PUSH IX
        POP  DE
        LD    HL,6
        ADD   HL,DE
        LD    (SAVE),HL   ; SAVE PTR TO LS BYTE
        EX    DE,HL
        CALL MDTL
;
; SAVE Y IN AC1 (IN CASE X AND Y ARE SAME)
; CLEAR (IX)
;
        CALL STYA1        ; STORE Y IN AC1
        LD    IY,AC1      ; PT TO Y
        CALL ZERO         ; CLEAR X
        LD    B,7         ; LOOP COUNTER
        JP    ML20        ; SKIP SHIFT OF AC1
;
; SHIFT PRODUCT RIGHT ONE DIGIT
; GET NEXT MULTIPLIER DIGIT
;
ML10    XOR   A           ; CLEAR A
        CALL SRD          ; SHFT (IX) RIGHT
        CALL MLDIG        ; MULTIPLY BY 2ND DIGIT
        XOR   A
        CALL SRD          ; SHFT (IX) RIGHT
ML20    LD    C,(IY+6)    ; GET 2 DIGITS
        DEC   IY          ; DECR PTR
        CALL MLDIG        ; MULTIPLY BY 1ST DIGIT
        DJNZ ML10         ; COUNTER NOT 0
;
; CHECK NORMALIZATION
;
        DEC   B           ; DEFAULT SHIFT COUNT IS -1
        LD    A,(IX)      ; GET MS BYTE
        AND   0F0H
        JR    Z,ML60      ; ALREADY NORMALIZED
        INC   B           ; SET SHIFT COUNT TO 0
        XOR   A
        CALL SRD          ; SHIFT 0 INTO (IX)
;
; CALCULATE EXPONENT OF RESULT
;
ML60    LD    A,(TEMP0E) ; EXP OF ORIGINAL X
        SUB   80H         ; REMOVE BIAS
        LD    C,A
        LD    A,(IY+14)   ; EXP OF ORIGINAL Y
        SUB   80H         ; REMOVE BIAS
        ADD   A,C         ; ADD TWO EXPONENTS
```

```
;
; CHECK EXPONENT CALCULATION FOR ERRORS
;
MCHK    JP   PO,M1O      ; NO OVERFLOW
        ADD  A,B         ; ADD SHIFT COUNT
        JP   PO,MERR     ; NO SECOND OVERFLOW
        JP   M2O
M1O     ADD  A,B         ; ADD SHIFT COUNT
        JP   PE,MERR     ; OVERFLOW
M2O     ADD  A,80H       ; RESTORE BIAS
        JP   Z,UNDER     ; UNDERFLOW
        LD   (IX+7),A    ; STORE EXP
        JP   RSTO
*H DIVIDE
;
; RECIPROCAL
;
REC     CALL ST04        ; SAVE X
        LD   HL,C1
        CALL LD          ; GET A 1
        LD   IY,TEMP4
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOATING POINT DIVISION ROUTINE
;
;   INPUT:  IX  PTS TO DIVIDEND
;           IY  PTS TO DIVISOR
;   OUTPUT: IX  PTS TO QUOTIENT (QUOT. REPLACES X, Y UNCHANGED)
;   ERRORS: DIVISOR ZERO
;           EXPONENT OVERFLOW OR UNDERFLOW
;   SPECIAL CASES:  DIVIDEND ZERO
;   ALGORITHM:  Y, 2Y,4Y,8Y ARE STORED IN TEMP. REGS.
;       X IS PLACED IN AC1.  AC1 IS COMPARED TO THE TEMP.
;       REGS. A BIT AT A TIME WITHIN EACH DIGIT, SUBTRACTING
;       WHEN TEMP <= AC1.  RESULT BITS ARE ACCUMULATED IN C,
;       AND STORED A BYTE AT A TIME INTO (IX)
;   TIMING:  APPROX. 22938 (20108 LOOP)
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CHECK FOR ZERO OPERANDS
;
DIVI    XOR  A
        CP   (IY+7)
        JP   Z,DIVO      ; Y IS ZERO
        CP   (IX+7)
        JP   Z,ZERO      ; X IS ZERO
;
; CALCULATE AND SAVE SIGN OF RESULT
;
        LD   A,(IX)      ; MS BYTE OF X
        XOR  (IY)
        EX   AF,AF'      ; SAVE IN A'
;
; STORE Y, 2Y, 4Y, AND 8Y IN TEMP REGISTERS
;
        PUSH IY
        POP  HL
        CALL MDTL
;
```

```
; INITIALIZE AC1
;
        PUSH IX          ; SAVE IX
        CALL STA1        ; STORE X IN AC1
        LD   BC,701H     ; COUNTER IS 7, SHFT COUNT 1
        PUSH BC          ; SAVE COUNTERS
        LD   C,0FH       ; INIT C
        LD   HL,TEMP0
        CALL DVCP        ; COMPARE X TO Y (FRACTIONS)
        JR   NC,DV30     ; X >= Y
        POP  BC
        DEC  C           ; DECR SHIFT COUNT
        PUSH BC
        JP   DV20        ; BEGIN WITH SHIFT DIGIT
;
; DIVIDE LOOP
;
DV10    PUSH BC          ; SAVE COUNTERS
        LD   HL,AC1
        XOR  A
        CALL SLD1        ; SHIFT AC1 LEFT 1 DIGIT
        CALL DVDIG       ; GET 1 DIGIT RESULT
DV20    LD   HL,AC1
        XOR  A
        CALL SLD1        ; SHIFT AC1 LEFT A DIGIT
DV30    CALL DVDIG       ; GET 2ND DIGIT RESULT
        LD   A,C         ; COMPL RESULT IN A
        CPL              ; TRUE RESULT
        LD   (IX),A      ; STORE PAIR OF QUOTIENT DIGITS
        INC  IX
        POP  BC
        DJNZ DV10        ; COUNTER NOT 0
;
; CALCULATE EXPONENT OF RESULT
;
        LD   B,C         ; PUT SHIFT COUNT IN B
        LD   A,(TEMP0E)  ; EXP OF ORIGINAL Y
        SUB  80H
        LD   C,A         ; SAVE UNBIASED EXP
        LD   A,(AC1E)    ; EXP OF ORIGINAL X
        SUB  80H         ; REMOVE BIAS
        SUB  C           ; SUB TWO EXPONENTS
        POP  IX          ; RESTORE IX
        JP   MCHK        ; CHECK FOR ERRORS
*H MULT, DIVI UTILITIES
;
; MLDIG  MULTIPLY DIGIT
;   INPUT:  C   CONTAINS DIGIT IN LOW 4 BITS
;           (SAVE) IS PTR TO LS BYTE OF ACCUM
;   OUTPUT: C   IS ROTATED 4 BITS
;           PRODUCT OF MPLICAND WITH DIGIT IS
;           ADDED TO ACCUM
;   TIMING 950
;
MLDIG   LD   HL,TEMP00   ; PT TO MPLICAND
        RR   C
        CALL C,MLADD     ; ADD X
        LD   HL,TEMP10
        RR   C
        CALL C,MLADD     ; ADD 2X
```

```
        LD   HL,TEMP20
        RR   C
        CALL C,MLADD     ; ADD 4X
        LD   HL,TEMP30
        RR   C
        CALL C,MLADD     ; ADD 8X
        RET
;
; MLADD   ADD UTILITY FOR MULTIPLY
;
;   INPUT:  HL  PTS TO NUMBER
;           (SAVE) PTS TO LS BYTE OF ACCUM
;   OUTPUT: 14 DIGIT NUMBER IS ADDED TO ACCUM
;   TIMING 407
;
MLADD   LD   DE,(SAVE)   ; PT TO LS BYTE
        AND  A           ; CLEAR CY
        PUSH BC          ; SAVE BC
        LD   B,7         ; SET COUNTER
ML40    LD   A,(DE)
        ADC  A,(HL)
        DAA
        LD   (DE),A
        DEC  DE
        DEC  HL
        DJNZ ML40
        POP  BC
        RET
;
; RSTO  PRODUCES SIGN OF RESULT
;   INPUT:  IX  PTS TO RESULT
;           A'  BIT 7 IS SIGN BIT
;   OUTPUT: IX  PTS TO RESULT
;   TIMING 74
;
RSTO    LD   A,(IX)      ; GET MS BYTE
        AND  0FH         ; CLEAR UPPER DIGIT
        LD   B,A
        EX   AF,AF'      ; GET NEW SIGN BIT
        AND  80H
        OR   B
        LD   (IX),A      ; REPLACE MS BYTE
        RET
;
; DVDIG  ROUTINE TO DIVIDE 1 DIGIT
;   INPUT:  AC1 IS CURRENT REMAINDER (DIVIDEND)
;           TEMP REGS CONTAIN 1,2,4,8*DIVISOR
;   OUTPUT: C   CONTAINS QUOTIENT DIGIT IN LOW 4 BITS
;               (C ROTATED LEFT)
;           AC1 CONTAINS REMAINDER (< DIVISOR)
;   TIMING 1162
;
DVDIG   LD   HL,TEMP3    ; PT TO 8Y
        CALL DVCP
        JR   C,DVD10     ; AC1 < 8Y
        LD   HL,TEMP30
        CALL DVSUB       ; 8Y <= AC1
DVD10   RL   C           ; COMPL RESULT BIT IN C
        LD   HL,TEMP2    ; PT TO 4Y
        CALL DVCP
```

```
          JR    C,DVD20     ; AC1 < 4Y
          LD    HL,TEMP20
          CALL  DVSUB       ; 4Y < AC1
DVD20     RL    C           ; COMPL RESULT BIT IN C
          LD    HL,TEMP1    ; PT TO 2Y
          CALL  DVCP
          JR    C,DVD30     ; AC1 < 2Y
          LD    HL,TEMP10
          CALL  DVSUB       ; 2Y < AC1
DVD30     RL    C           ; COMPL RESULT BIT IN C
          LD    HL,TEMP0    ; PT TO Y
          CALL  DVCP        ; COMPARE TO AC1
          JR    C,DVD40     ; AC1 < Y
          LD    HL,TEMP00
          CALL  DVSUB       ; Y < AC1
DVD40     RL    C           ; COMPL RESULT BIT IN C
          RET
;
; DVCP  ROUTINE TO COMPARE (DE) TO AC1
;   RETURNS CY SET IFF (DE) < AC1
;   RETURNS Z  SET IFF (DE) = AC1
;   TIMING APPROX 42
;
DVCP      LD    DE,AC1      ; PT TO AC1
          LD    B,7         ; SET COUNTER
DVC10     LD    A,(DE)      ; GET BYTE
          CP    (HL)
          RET   NZ          ; < OR >
          INC   HL          ; INC PTRS
          INC   DE
          DJNZ  DVC10       ; COUNTER NOT 0
          RET
;
; DVSUB  ROUTINE TO SUBTRACT (DE) FROM AC1
;   ASSUMES 14 DIGIT OPERANDS
;   TIMING 376
;
DVSUB     LD    DE,AC10     ; PT TO ACCUM
          AND   A           ; CLEAR CY
          LD    B,7
DV40      LD    A,(DE)
          SBC   A,(HL)
          DAA
          LD    (DE),A
          DEC   DE
          DEC   HL
          DJNZ  DV40
          RET
;
; ROUTINE TO LOAD TEMP REGISTERS
;   TIMING 2014
;
MDTL      LD    DE,TEMP0    ; FIRST DEST
          CALL  MOV         ; LOAD TEMP0
          LD    HL,TEMP0    ; PT TO SRC
          RES   7,(HL)      ; CLEAR SIGN BIT
          LD    B,3         ; SET COUNTER
;
; LOOP TO COPY TEMPN TO TEMPN+1 & SHFT LFT 1 BIT
;
```

```
MDTL1   LD   A,B         ; SAVE COUNTER
        CALL MOV         ; MOVE TO NEXT TEMP
        LD   B,A         ; RESTORE COUNTER
        AND  A           ; CLEAR CY
        CALL SLB1        ; SHFT LEFT (HL ALREADY OK)
        INC  HL          ; PT TO MSBYTE JUST SHIFTED
        DJNZ MDTL1       ; COUNTER NOT 0
        RET
;
; DETERMINE OVER OR UNDER
;
MERR    JP   P,UNDER     ; UNDERFLOW
        JP   OVERA
;
; DIVISION BY ZERO ERROR
;
DIV0    LD   HL,ERROR
        SET  4,(HL)
        JP   INF
*H FIX
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FIX CONVERTS FLOATING POINT NUMBER TO
;       16 BIT 2'S COMPLEMENT INTEGER
;       (ROUNDING)
;
;   INPUT:  IX  PTS TO FL PT NUMBER
;   OUTPUT: IX  PTS TO INTEGER
;   ERRORS:  OVERFLOW IF X > 32767 OR X < -32768
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
FIX     LD   IY,C_5
        CALL ADD         ; ADD .5
        CALL FIXA        ; CONVERT NUMBER
        JR   NC,FX10     ; NO OVERFLOW
        LD   HL,ERROR
        SET  6,(HL)      ; FLAG OVERFLOW
        LD   HL,7FFFH    ; 32767
        BIT  7,(IX)
        JR   Z,FX10      ; POSITIVE SIGN
        LD   HL,8000H    ; -32768
FX10    LD   (IX),L      ; STORE INTEGER
        LD   (IX+1),H
        RET
;
; MAIN CONVERSION FOR FIX
;
;   RETURNS CY SET IF OVERFLOW
;   RETURNS Z  SET IF EXACT INTEGER
;   HL IS LOW 16 BITS OF INT(X) FOR ALL X < 10^13
;
FIXA    XOR  A
        LD   H,A         ; CLEAR ACCUM
        LD   L,A
        LD   (SAVE),A    ; CLEAR FLAG
;
; CALCULATE # DIGITS LEFT AND RIGHT OF DECML PT
;
        LD   A,(IX+7)    ; GET EXP
        SUB  80H         ; SUBTRACT BIAS
```

```
            JR    C,FXA100    ; X < .1
            JR    Z,FXA100    ; X < 1
            LD    B,A         ; SAVE # LEFT DIGITS
            LD    A,13
            SUB   B
            JR    C,FXA90     ; X >= 10↑13
            LD    C,A         ; SAVE # RIGHT DIGITS
;
; CONVERT LEFT DIGITS TO BINARY
;
            PUSH  IX          ; SET POINTER
            POP   IY
            LD    A,(IY)      ; GET FIRST BYTE
            INC   IY
            JP    FXA20
FXA10       CALL  NZ,FXA110   ; OVERFLOW
            LD    A,(IY)      ; GET NEXT TWO DIGITS
            INC   IY
            PUSH  AF          ; SAVE THIS BYTE
            RRA               ; GET LEFT DIGIT
            RRA
            RRA
            RRA
            CALL  MAD         ; CONVERT LEFT DIGIT
            DJNZ  FXA15       ; COUNTER NOT 0
            POP   AF          ; GET RIGHT DIGIT
            AND   0FH
            JR    FXA30
FXA15       CALL  NZ,FXA110   ; OVERFLOW
            POP   AF          ; GET SAVED BYTE
FXA20       CALL  MAD         ; CONVERT RIGHT DIGIT
            DJNZ  FXA10       ; COUNTER NOT 0
;
; CALCULATE OR OF ALL DIGITS TO RIGHT OF DECML PT
;
            XOR   A           ; INITIALIZE A
FXA30       LD    B,C         ; USE OTHER COUNTER
            SRL   B           ; DIVIDE BY 2
            JR    Z,FXA50     ; COUNT IS 0
FXA40       OR    (IY)        ; OR 2 DIGITS
            INC   IY
            DJNZ  FXA40       ; COUNTER NOT 0
;
; CONVERT TO 2'S COMPLEMENT
;
FXA50       BIT   7,(IX)
            JR    Z,FXA80     ; POSITIVE SIGN
            EX    DE,HL
            LD    HL,0
            AND   A
            JR    Z,FXA60     ; EXACT INTEGER
            INC   DE          ; CORRECT MAGNITUDE
FXA60       SBC   HL,DE       ; SUBTRACT FROM 0
            JP    P,FXA90     ; OVERFLOW
FXA70       AND   A           ; SET Z OR NZ
            LD    A,(SAVE)    ; GET FLAG
            RLA               ; PUT IN CY
            RET
FXA80       BIT   7,H
            JR    Z,FXA70     ; NO OVERFLOW
```

```
FXA90   AND  A           ; SET Z OR NZ
        SCF              ; SET CY
        RET
FXA100  LD   A,(IX+7)    ; A IS 0 IFF FRACTION 0
        JP   FXA50
FXA110  LD   A,(SAVE)    ; GET FLAG
        OR   80H         ; TURN IT ON
        LD   (SAVE),A
        RET
;
; MAD ROUTINE TO MULTIPLY ACCUM BY 10 AND ADD NEW DIGIT
;
MAD     LD   D,H         ; SAVE COPY OF HL
        LD   E,L
        ADD  HL,HL       ; SHIFT LEFT 2
        ADD  HL,HL
        ADD  HL,DE       ; ADD COPY
        ADD  HL,HL       ; SHIFT LEFT 1
        AND  0FH         ; PUT NEW DIGIT IN DE
        LD   E,A
        LD   D,0
        ADD  HL,DE       ; ADD NEW DIGIT
        LD   A,H         ; TEST HIGH BITS
        AND  0F0H
        RET
*H FLOAT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; FLOAT CONVERTS 16 BIT 2'S COMPLEMENT INTEGER TO
;       FLOATING POINT FORMAT
;   (FLOAT1 ASSUMES INPUT IS IN HL)
;
;   INPUT:  IX  PTS TO 16 BIT INTEGER
;   OUTPUT: IX  PTS TO FLOATING PT NUMBER
;   ERRORS: NONE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; TEST INPUT
;
FLOAT   LD   L,(IX)      ; LOAD 16 BIT INTEGER
        LD   H,(IX+1)
FLOAT1  PUSH HL          ; SAVE INTEGER
        CALL ZERO        ; CLEAR OUTPUT AREA
        POP  HL
        LD   A,H         ; TEST FOR ZERO
        OR   L
        RET  Z           ; INPUT WAS ZERO
;
; CONVERT TO SIGN MAGNITUDE, SIGN IN A'
;
        LD   A,H         ; GET SIGN BIT
        AND  80H
        JR   Z,FL10      ; POSITIVE
        EX   DE,HL
        LD   HL,0
        SBC  HL,DE       ; NEGATE
;
; CONVERT BINARY TO DECIMAL
;
```

```
FL10    EX   AF,AF'      ; SAVE SIGN BIT
        EX   DE,HL       ; SAVE ACCUM
        PUSH IX
        POP  HL          ; PT TO OUTPUT AREA
        LD   BC,10000
        CALL FLDIV       ; DIVIDE BY 10000
        INC  HL          ; ADVANCE OUTPUT PTR
        LD   BC,1000
        CALL FLDIV       ; DIVIDE BY 1000
        LD   BC,100
        CALL FLDIV       ; DIVIDE BY 100
        INC  HL          ; ADVANCE OUTPUT PTR
        LD   BC,10
        CALL FLDIV       ; DIVIDE BY 10
        LD   A,E         ; GET REMAINDER
        RLD              ; ROT INTO OUTPUT
;
; NORMALIZE RESULT, STORE SIGN
;
        LD   C,85H       ; EXPONENT
        CALL NORML1
        JP   RSTO        ; STORE SIGN
;
; ROUTINE TO DIVIDE DE BY BC
;   4 BIT QUOTIENT IS ROTATED INTO (HL)
;
FLDIV   EX   DE,HL       ; PUT ACCUM IN HL
        XOR  A           ; CLEAR A, CY
        DEC  A           ; INITIALLY A IS -1
FLDIV1  SBC  HL,BC       ; SUBTRACT DIVISOR
        INC  A           ; INCR QUOTIENT
        JR   NC,FLDIV1   ; RESULT NOT NEGATIVE
        ADD  HL,BC       ; RESTORE REMAINDER
        EX   DE,HL       ; SAVE ACCUM, GET PTR
        RLD              ; STORE QUOTIENT
        RET
*H CP
;
; CP  FLOATING POINT COMPARE ROUTINE
;
;   RETURNS CY SET IF X < Y
;   RETURNS Z  SET IF X = Y
;   CP1 ASSUMES MS BYTES FOR X,Y IN D,E
;
CP      LD   D,(IX)      ; LOAD MS BYTES
        LD   E,(IY)
CP1     LD   A,D         ; GET SIGN OF X
        AND  80H         ; (CY IS OFF)
        JR   NZ,CP10     ; X IS NEGATIVE
        BIT  7,E
        RET  NZ          ; Y IS NEGATIVE
;
; X AND Y ARE BOTH POSITIVE, COMPARE EXPONENTS
;
        LD   A,(IX+7)
        CP   (IY+7)
        RET  NZ          ; EXP'S NOT =
;
; COMPARE X FRACTION TO Y FRACTION
;
```

```
        LD   A,D        ; MS BYTE
        CP   E          ; (SIGNS ARE EQUAL)
        RET  NZ
        PUSH IX         ; SET UP PTRS
        PUSH IY
        JR   CP20
;
; X IS NEGATIVE, CHECK SIGN OF Y
;
CP10    XOR  E          ; TEST SIGN, SET NZ
        RLA             ; SIGN BIT IN CY
        RET  C          ; Y WAS POSITIVE
;
; BOTH SIGNS NEGATIVE, TEST EXPONENTS
;
        LD   A,(IY+7)
        CP   (IX+7)
        RET  NZ
;
; COMPARE Y FRACTION TO X FRACTION
;
        LD   A,E
        CP   D
        RET  NZ
        PUSH IY         ; SET UP PTRS
        PUSH IX
;
; LOOP TO COMPARE 6 FRACTION BYTES
;
CP20    POP  HL         ; GET PTRS
        POP  DE
        INC  HL
        INC  DE
        LD   B,6        ; COUNTER
CP30    LD   A,(DE)
        CP   (HL)
        RET  NZ
        INC  HL
        INC  DE
        DJNZ CP30
        RET
*H NORMALIZE
;
; NORMLZ  NORMALIZES AN ACCUM BY SHIFTING LEFT
;
;   INPUT:  IX  PTS TO ACCUM (NORMLZ,NORML1)
;           HL  PTS TO ACCUM (NORML2)
;           C   CONTAINS BIASED EXP (NORML1,NORML2)
;   OUTPUT: NORMAL RETURN - NZ & NC
;           ZERO FRACTION -  Z & NC
;           EXP UNDERFLOW - NZ &  C
;           C   CONTAINS UPDATED EXPONENT
;           (IX+7) CONTAINS UPDATED EXPONENT
;
NORMLZ  LD   C,(IX+7)   ; GET EXPONENT
NORML1  PUSH IX         ; PT TO ACCUM
        POP  HL
NORML2  LD   A,(HL)     ; GET MS BYTE
        AND  A
        JR   NZ,NOR15   ; ALREADY NORMALIZED
```

```
        LD   B,12        ; MAX COUNT
        XOR  A           ; CLEAR A
NOR10   DEC  C           ; DECR EXPONENT
        JR   Z,NOR20     ; UNDERFLOW
        CALL SLD1        ; SHIFT LEFT
        INC  HL          ; PT TO 1ST BYTE
        LD   A,(HL)      ; GET MS BYTE
        AND  A
        JR   NZ,NOR15    ; NORMALIZED
        DJNZ NOR10       ; COUNTER NOT 0
NOR15   LD   (IX+7),C    ; STORE EXPONENT
        RET
;
; UNDERFLOW.  HL PTS TO MS BYTE, A CONTAINS MS BYTE
;
NOR20   INC  B           ; CALC BYTES LEFT
        SRL  B
NOR30   INC  HL          ; ADVANCE PTR
        OR   (HL)        ; COLLECT REMAINING BITS
        DJNZ NOR30       ; COUNT NOT 0
        RET  Z           ; FRACTION WAS ZERO
        SCF
        RET              ; (UNDERFLOW)
*H HALF
;
; HALF  DIVIDES A FLOATING POINT NUMBER BY 2
;   INPUT MUST BE POSITIVE (0 IN SIGN DIGIT)
;
;   INPUT:  IX  PTS TO X (HALF)
;   OUTPUT: INPUT REPLACED BY X/2
;           FLAGS SET ACCORDING TO NORMLZ
;
HALF    AND  A           ; CLEAR CY
        CALL SRB         ; DIVIDE FRACTION BY 2
        CALL NORMLZ      ; NORMALIZE RESULT
        RET
*H SHIFT UTILITIES
;
; SLD SHIFT LEFT DIGIT
;
;   INPUT:  IX  PTS TO ACCUM (SLD)
;           HL  PTS TO ACCUM (SLD1)
;           A   CONTAINS DIGIT TO SHIFT IN
;   OUTPUT: A   CONTAINS DIGIT SHIFTED OUT
;           HL  PTS TO 1 LESS THAN MS BYTE
;
SLD     PUSH IX          ; SET PTR TO ACCUM
        POP  HL
SLD1    PUSH BC          ; SAVE BC
        LD   BC,6
        ADD  HL,BC       ; PT TO RIGHT END
        LD   B,7         ; LOOP COUNTER
SLD10   RLD              ; ROT LEFT
        DEC  HL          ; PT TO NEXT BYTE
        DJNZ SLD10       ; COUNTER NOT 0
        POP  BC          ; RESTORE BC
        RET
;
; SRD  SHIFT RIGHT DIGIT
;
```

```
;   INPUT:  IX  PTS TO ACCUM (SRD)
;           HL  PTS TO ACCUM (SRD1)
;           A   CONTAINS DIGIT TO SHIFT IN
;   OUTPUT: A   CONTAINS DIGIT SHIFTED OUT
;           HL  PTS TO 1 MORE THAN LS BYTE
;
SRD     PUSH IX         ; SET PTR TO ACCUM
        POP  HL
SRD1    PUSH BC         ; SAVE BC
        LD   B,7        ; COUNTER
SRD10   RRD             ; ROT RIGHT
        INC  HL
        DJNZ SRD10      ; COUNTER NOT 0
        POP  BC         ; RESTORE BC
        RET
;
; SLB SHIFT LEFT BIT
;
;   INPUT:  IX  PTS TO ACCUM (SLB)
;           HL  PTS TO ACCUM (SLB1)
;           CY  CONTAINS BIT TO SHIFT IN
;   OUTPUT: CY  CONTAINS BIT SHIFTED OUT
;           HL  PTS TO 1 LESS THAN MS BYTE
;
SLB     PUSH IX         ; SET PTR TO ACCUM
        POP  HL
SLB1    PUSH BC         ; SAVE BC
        LD   BC,6
        ADD  HL,BC      ; PT TO RIGHT END
        LD   B,7        ; LOOP COUNTER
SLB10   LD   A,(HL)     ; GET BYTE
        ADC  A,A        ; SHIFT LEFT
        DAA
        LD   (HL),A     ; REPLACE
        DEC  HL         ; PT TO NEXT BYTE
        DJNZ SLB10      ; COUNTER NOT 0
        POP  BC         ; RESTORE BC
        RET
;
; SRB  SHIFT RIGHT BIT
;
;   INPUT:  IX  PTS TO ACCUM (SRB)
;           HL  PTS TO ACCUM (SRB1)
;           CY  CONTAINS BIT TO SHIFT IN
;   OUTPUT: CY  CONTAINS BIT SHIFTED OUT
;           HL  PTS TO 1 MORE THAN LS BYTE
;
SRB     PUSH IX         ; SET PTR TO ACCUM
        POP  HL
SRB1    PUSH BC         ; SAVE BC
        LD   B,7        ; COUNTER
SRB10   LD   A,(HL)     ; GET BYTE
        RRA             ; SHIFT
        RR   C          ; SAVE CY IN C
        BIT  7,A
        JR   Z,SRB20    ; 0 SHIFTED INTO UPPER
        SUB  30H        ; CORRECTION
SRB20   BIT  3,A
        JR   Z,SRB30    ; 0 SHIFTED INTO LOWER
        SUB  3          ; CORRECTION
```

```
SRB30   RL    C           ; GET CY
        LD    (HL),A      ; STORE BYTE
        INC   HL
        DJNZ  SRB10       ; COUNTER NOT 0
        POP   BC          ; RESTORE BC
        RET
*H MOVE UTILITIES
;
; MOVE ROUTINES
;
STO4    LD    DE,TEMP4
        JP    STO
STO5    LD    DE,TEMP5
        JP    STO
STO6    LD    DE,TEMP6
STO     PUSH  IX
        POP   HL          ; SET SRC PTR
MOV     LD    BC,8
        LDIR
        RET
LD      PUSH  IX
        POP   DE          ; SET DEST PTR
        JP    MOV
;
; SPECIALIZED LOAD ROUTINES, LOAD AC1 AND AC2 WITH ABS VALUE
;
STA1    PUSH  IX          ; SOURCE WILL BE (IX)
        JR    ST1
STA2    PUSH  IX          ; SOURCE WILL BE (IX)
        JR    ST2
STYA1   PUSH  IY          ; SOURCE WILL BE (IY)
ST1     POP   HL
MV1     LD    DE,AC1      ; DEST IS AC1
        CALL  MOV
        LD    HL,AC1
        LD    D,(HL)      ; SAVE MS BYTE IN D
        RES   7,(HL)      ; CLEAR SIGN BIT
        RET
STYA2   PUSH  IY          ; SOURCE WILL BE (IY)
ST2     POP   HL
MV2     LD    DE,AC2      ; DEST IS AC2
        CALL  MOV
        LD    HL,AC2
        LD    E,(HL)      ; SAVE MS BYTE IN E
        RES   7,(HL)      ; CLEAR SIGN BIT
        RET
*H BOOLEAN
;
; LOGICAL OPERATORS
;
IOR     LD    A,(IX)
        OR    (IX+1)
        OR    (IY)
        OR    (IY+1)
        JP    INE1
IAND    LD    A,(IX)
        OR    (IX+1)
        JP    Z,IZERO     ; X IS ZERO
        LD    A,(IY)
        OR    (IY+1)
```

```
          JP    INE1
INOT      LD    A,(IX)
          OR    (IX+1)
          JR    IEQ1
OR        LD    A,(IX+7)
          OR    (IY+7)
          JP    NE1
AND       LD    A,(IX+7)
          AND   A
          JP    Z,ZERO      ; X IS ZERO
          LD    A,(IY+7)
          AND   A
          JR    NE1
NOT       LD    A,(IX+7)
          AND   A
          JR    EQ1
;
; RELATIONAL OPERATORS
;
SGT       CALL  SCP
          JR    IGT1
SLE       CALL  SCP
          JR    ILE1
SLT       CALL  SCP
          JR    ILT1
SGE       CALL  SCP
          JR    IGE1
SEQ       CALL  SCP
          JR    IEQ1
SNE       CALL  SCP
          JR    INE1
IGT       CALL  ICP
IGT1      JR    Z,IZERO
          JR    C,IZERO
          JR    IONE
ILE       CALL  ICP
ILE1      JR    Z,IONE
          JR    C,IONE
          JR    IZERO
ILT       CALL  ICP
ILT1      JR    C,IONE
          JR    IZERO
IGE       CALL  ICP
IGE1      JR    C,IZERO
          JR    IONE
IEQ       CALL  ICP
IEQ1      JR    Z,IONE
          JR    IZERO
INE       CALL  ICP
INE1      JR    Z,IZERO
          JR    IONE
GT        CALL  CP
GT1       JR    Z,ZERO
          JR    C,ZERO
          JR    ONE
LE        CALL  CP
LE1       JR    Z,ONE
          JR    C,ONE
          JR    ZERO
LT        CALL  CP
```

```
LT1     JR   C,ONE
        JR   ZERO
GE      CALL CP
GE1     JR   C,ZERO
        JR   ONE
EQ      CALL CP
EQ1     JR   Z,ONE
        JR   ZERO
NE      CALL CP
NE1     JR   Z,ZERO
        JR   ONE
*H CONSTANT FUNCTIONS
;
; PRODUCE UNITY RESULT
;
IONE    LD   (IX),1
        LD   (IX+1),0
        RET
ONE     LD   B,7
        CALL ZERO1
        LD   (HL),81H
        LD   (IX),1
        RET
;
; PRODUCE ZERO RESULT
;
IZERO   LD   (IX),0
        LD   (IX+1),0
        RET
ZERO    LD   B,8
ZERO1   PUSH IX          ; PT TO RESULT
        POP  HL
ZER10   LD   (HL),0
        INC  HL
        DJNZ ZER10
        RET
;
; PRODUCE INFINITE RESULT
;
INF     PUSH IX          ; SET PTR
        POP  HL
        RL   (HL)        ; GET OLD SIGN
        LD   (HL),18
        RR   (HL)        ; REPLACE SIGN
        INC  HL
        LD   A,99H
        LD   B,6
INF10   LD   (HL),A
        INC  HL
        DJNZ INF10
        LD   (HL),0FFH   ; HIGHEST EXP
        RET
;
; FLAG OVERFLOW ERROR, STORE INFINITY
;
OVERA   EX   AF,AF'      ; GET SIGN OF RESULT
        LD   (IX),A      ; STORE IT
OVER    LD   HL,ERROR
        SET  6,(HL)      ; FLAG OVERFLOW
        JP   INF
```

```
;
; FLAG UNDERFLOW ERROR, STORE ZERO
;
UNDER   LD    HL,ERROR
        SET   5,(HL)      ; FLAG UNDERFLOW
        JP    ZERO
*H CONSTANTS AND WORK AREAS
C1      DEFW 1
        DEFW 0
        DEFW 0
        DEFW 8100H
C0      DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0
C_5     DEFW 5            ; .5
        DEFW 0
        DEFW 0
        DEFW 8000H
SEED    DEFW 0            ; 0
        DEFW 0
        DEFW 0
        DEFW 0
SEED10  EQU  SEED+6
CRNDA   DEFW 1403H        ; 31415 92653 581
        DEFW 9215H
        DEFW 3565H
        DEFW 8081H
CRNDC   DEFW 7102H        ; 27182 81828 459
        DEFW 8182H
        DEFW 8482H
        DEFW 8059H
CRNDC0  EQU  CRNDC+6
SAVE    DEFS 2
AC1     DEFS 8
AC10    EQU  AC1+6
AC1E    EQU  AC1+7        ; EXPONENT BYTE
AC2     DEFS 8
AC20    EQU  AC2+6
AC2E    EQU  AC2+7        ; EXPONENT BYTE
TEMP0   DEFS 8
TEMP00  EQU  TEMP0+6
TEMP0E  EQU  TEMP0+7
TEMP1   DEFS 8
TEMP10  EQU  TEMP1+6
TEMP2   DEFS 8
TEMP20  EQU  TEMP2+6
TEMP3   DEFS 8
TEMP30  EQU  TEMP3+6
TEMP4   DEFS 8
TEMP5   DEFS 8
TEMP6   DEFS 8
;
; REFERENCES
;
        EXTERNAL        ERROR,ICP,SCP
        GLOBAL          ADD,SUB,RSUB,MULT,REC,DIVI
        GLOBAL          MLDIG,MDTL
        GLOBAL          FIX,FIXA,FLOAT,FLOAT1,CP,MAD
        GLOBAL          GT,LE,LT,GE,EQ,NE,OR,AND,NOT
```

```
        GLOBAL          IGT,ILE,ILT,IGE,IEQ,INE,IOR,IAND,INOT
        GLOBAL          SGT,SLE,SLT,SGE,SEQ,SNE
        GLOBAL          SEED,SEED10,CRNDA,CRNDCO,C_5,SAVE,C1,C0
        GLOBAL          OVER,UNDER,ZERO,ONE,INF
        GLOBAL          NORMLZ,NORML1,NORML2,SLD,SRB,HALF
        GLOBAL          ST04,ST05,ST06,ST0,LD,SRD
        GLOBAL          TEMP4,TEMP5,TEMP6,AC1,AC10,AC2,AC20
*H DECIMAL TO BINARY
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; DECIMAL TO BINARY CONVERSION
;   INPUT:  IX POINTS TO OUTPUT AREA
;   OUTPUT: IX POINTS TO FL PT NUMBER
;   ERRORS: NO DIGITS IN FRACTION PART
;           NO DIGITS IN EXPONENT
;           EXPONENT OVERFLOW
;           EXPONENT UNDERFLOW
;   REGISTER ASSIGNMENT IN MAIN LOOP:
;       C       - DECIMAL EXPONENT
;       B       - FLAGS: BIT 6 - DIGITS FOUND
;                        BIT 7 - FRACTION OVRFLW
;       A       - NEW DIGIT
;       A'      - SIGN  IN BIT 7
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DTOB    CALL LMARK      ; MARK BEGINNING OF SCAN
        CALL ZERO       ; CLEAR RESULT
        XOR  A          ; CLEAR A
        LD   B,A        ; CLEAR FLAGS
        LD   C,80H+13   ; INIT EXPONENT
        EX   AF,AF'     ; PUT AWAY SIGN
        CALL LIN        ; GET FIRST CHAR
        CP   '+'
        JR   Z,D10      ; SIGN IS +
        CP   '-'
        JR   NZ,D20     ; NO SIGN
        EX   AF,AF'
        OR   80H        ; FLAG NEGATIVE SIGN
        EX   AF,AF'
;
; LOOP TO PROCESS CHARS BEFORE DECIMAL POINT
;
D10     CALL LIN        ; GET NEXT CHAR
D20     CP   '.'
        JR   Z,D40      ; DECML PT FOUND
        CALL DIGIT?
        JR   C,D50      ; NOT A DIGIT
        SET  6,B        ; FLAG DIGIT FOUND
        BIT  7,B
        JR   NZ,D30     ; OVERFLOW
        CALL CNV        ; CONVERT THIS DIGIT
        JP   D10
D30     INC  C          ; INC DECML EXP
        JP   D10
;
; LOOP TO PROCESS CHARS AFTER DECIMAL POINT
;
D40     CALL LIN        ; GET NEXT CHAR
        CALL DIGIT?
        JR   C,D50      ; NOT A DIGIT
```

```
        SET  6,B        ; FLAG DIGIT FOUND
        BIT  7,B
        JR   NZ,D40     ; OVERFLOW
        DEC  C          ; DEC DECML EXP
        CALL CNV        ; CONVERT THIS DIGIT
        JP   D40
;
; CHANGE FRACTION TO FLOATING POINT FORMAT
;
D50     BIT  6,B
        JP   Z,LREST    ; NO DIGITS FOUND
        LD   (SAVE),A   ; SAVE THIS CHAR
        CALL NORML1
        JR   Z,D130     ; ZERO FRACTION PART
        LD   A,(SAVE)   ; GET CHAR AGAIN
;
; PROCESS EXPONENT
;   REGISTERS:
;       B - BIT 7 IS SIGN
;           BIT 6 IS DIGIT-FOUND FLAG
;           BIT 5 IS DIGIT-OVERFLOW FLAG
;       C - IS EXP FROM FRACTION PART
;       HL - IS ACCUM FOR D-TO-B CONVERSION
;
        LD   HL,0       ; INITIALIZE REGISTERS
        LD   B,L
        CP   'E'-'0'
        JR   NZ,D90     ; NOT AN 'E'
;
; CONVERT SIGN
;
        CALL LIN
        CP   '+'
        JR   Z,D60      ; SIGN IS +
        CP   '-'
        JR   NZ,D70     ; NO SIGN
        SET  7,B        ; FLAG NEG SIGN
;
; CONVERT DIGITS
;
D60     CALL LIN
D70     CALL DIGIT?
        JR   C,D80      ; NOT A DIGIT
        SET  6,B        ; FLAG DIGIT FOUND
        BIT  5,B
        JR   NZ,D100    ; OVER OR UNDERFLOW
        CALL MAD        ; CONVERT THIS DIGIT
        LD   A,H        ; CHECK FOR OVERFLOW
        AND  A
        JP   Z,D60      ; HL < 256
        SET  5,B        ; SET FLAG
        JP   D60
;
; TEST FOR ILLEGAL EXPONENT PART
;
D80     BIT  6,B
        JR   Z,D120     ; NO DIGITS FOUND
;
; CONVERT HL ACCUM TO 2'S COMPLEMENT
;
```

```
D90     BIT  7,B
        JR   Z,D92       ; SIGN IS +
        EX   DE,HL
        LD   HL,0
        AND  A
        SBC  HL,DE
;
; ADD PREVIOUS EXPONENT (C)
; CHECK FOR OVERFLOW
;
D92     LD   B,0
        AND  A
        ADC  HL,BC       ; ADD TWO EXPONENTS
        JP   Z,UNDER     ; UNDERFLOW
        LD   A,H
        AND  A
        JR   NZ,D96      ; EXP < -128 OR > 127
;
; STORE SIGN AND EXPONENT
;
        EX   AF,AF'      ; GET SIGN
        OR   (IX)
        LD   (IX),A      ; STORE SIGN
        LD   (IX+7),L    ; STORE EXP
        JP   RTN
;
; DETERMINE SIGN OF EXP
;
D96     BIT  7,H
        JR   D105
;
; FINISH SCANNING DIGITS
;
D100    CALL LIN         ; FINISH SCANNING DIGITS
        CALL DIGIT?
        JR   NC,D100     ; DIGIT IS FOUND
;
; DETERMINE FROM SIGN BIT WHETHER OVER OR UNDER
;
        BIT  7,B
D105    JR   NZ,D110     ; SIGN IS NEGATIVE
        CALL OVER        ; FLAG ERROR
        JP   RTN
D110    CALL UNDER       ; FLAG ERROR
        JP   RTN
;
; ILLEGAL NUMBER
;
D120    LD   HL,ERROR
        SET  7,(HL)
        JP   RTN
;
; ZERO
;
D130    CALL ZERO
        JP   RTN
*EJECT
;
; SCANNER ROUTINES
;
```

```
RTN     CALL LBACK      ; BACK UP POINTER BY 1
        JP   RC2        ; GO TO SPECIAL RETURN
;
; DIGIT? ROUTINE CONVERTS ASCII CHAR TO 0-9 VALUE
;    RETURNS CY ON IF NOT A DIGIT
;
DIGIT?  SUB  '0'
        RET  C          ; WAS < 30H
        CP   10
        CCF             ; CY SET IF > 9
        RET
;
; CNV ROUTINE TO MULTIPLY DECIMAL ACCUM 1 BY 10 AND ADD
;
CNV     CALL SLD        ; SHIFT A INTO ACCUM
        LD   A,(IX)     ; GET HIGH BYTE
        AND  A
        RET  Z          ; < 13 DIGITS
        SET  7,B        ; FLAG DIGIT OVERFLOW
        RET
*H BINARY TO DECIMAL
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; BCD TO DECIMAL CONVERSION
;
;   INPUT:  IX PTS TO FL PT NUMBER
;           HL PTS TO OUTPUT AREA
;   OUTPUT: FL PT NUMBER IS MODIFIED
;           OUTPUT AREA IS 19 BYTES:
;               1 SIGN (SPACE OR -)
;               13 DIGITS OF FRACTION
;               1  "E"
;               1  SIGN OF EXPONENT (+ OR -)
;               3  DIGITS OF EXPONENT (LEADING ZEROS)
;           EXPONENT IS FOR DECIMAL PT AFTER FIRST DIGIT
;           A IS 2'S COMPLEMENT EXPONENT (-128<=A<=126)
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CONVERT SIGN
;
BTOD    BIT  7,(IX)     ; TEST SIGN
        LD   (HL),' '   ; DEFAULT SIGN
        JR   Z,BD10     ; + SIGN
        LD   (HL),'-'   ; - SIGN
BD10    INC  HL
;
; UNPACK FRACTION PART
;
        EX   DE,HL      ; OUTPUT PTR IN DE
        CALL SLD        ; SHIFT OUT SIGN BYTE
        LD   B,13       ; COUNTER
        LD   A,'0'
BD20    CALL SLD        ; GET A DIGIT
        LD   (DE),A     ; OUTPUT IT
        INC  DE
        DJNZ BD20       ; COUNTER NOT 0
;
; CONVERT EXPONENT SIGN, GET SIGN MAGNITUDE EXP
;
```

```
        EX   DE,HL       ; OUTPUT PTR IN HL
        LD   (HL),'E'
        INC  HL
        LD   (HL),'+'    ; DEFAULT SIGN
        LD   A,(IX+7)    ; GET BIASED EXP
        AND  A
        JR   Z,BD30      ; ZERO EXPON
        SUB  81H         ; REMOVE BIAS, DECR BY 1
        JR   NC,BD30     ; POSITIVE EXP
        LD   (HL),'-'    ; - SIGN
        NEG              ; CALC MAGNITUDE
;
; CONVERT HUNDREDS
;
BD30    INC  HL
        LD   (HL),'0'    ; DEFAULT DIGIT
        CP   100
        JP   C,BD40      ; EXP < 100
        LD   (HL),'1'
        SUB  100
;
; DIVIDE BY 10
;
BD40    INC  HL
        LD   B,2FH       ; QUOTIENT COUNTER
        LD   C,10        ; DIVISOR
BD50    SUB  C           ; SUBTRACT 10
        INC  B           ; INC QUOTIENT
        JR   NC,BD50     ; RESULT NOT NEGATIVE
        ADD  A,C         ; RESTORE REMAINDER
        LD   (HL),B      ; STORE QUOTIENT DIGIT
        INC  HL
        OR   '0'
        LD   (HL),A      ; STORE REMAINDER DIGIT
        LD   A,(IX+7)    ; GET EXPONENT
        SUB  80H         ; REMOVE BIAS
        RET
*H RND
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; RANDOM NUMBER GENERATOR
;
;   INPUT:  IX  PTS TO OUTPUT/WORK AREA
;           SEED CONTAINS PREVIOUS RANDOM NUMBER
;   OUTPUT: IX  PTS TO RANDOM NUMBER
;           SEED CONTAINS NEW NUMBER
;   ERRORS: NONE
;   ALGORITHM:  ARITHMETIC DONE MODULO 10↑13,
;       X := 3141592653581 * X + 2718281828459
;       RESULT IS DIVIDED BY 10↑13, AND NORMALIZED
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; INITIALIZE REGISTERS
;
RND     LD   IY,CRNDA    ; PT TO MPLIER
        PUSH IX
        POP  DE
        LD   HL,6
        ADD  HL,DE       ; PT TO LS BYTE OF (IX)
```

```
        LD    (SAVE),HL   ; SAVE FOR MLDIG
        LD    HL,SEED     ; PT TO LAST NUMBER
        CALL  MDTL        ; SAVE MULTIPLES IN TEMPO-3
        CALL  ZERO        ; CLEAR WORK AREA
        LD    B,7         ; SET MULTIPLY LOOP COUNTER
        JP    RN20
;
; MULTIPLICATION, USING (IX) AS ACCUM, (IY) AS
;   MULTIPLIER, (SEED) (SEED=IY-8) AS PRODUCT
;   MULTIPLY BY 31415 92653 581  (CRNDA)
;
RN10    PUSH  AF          ; SAVE DIGIT SHIFTED OUT
        CALL  MLDIG       ; MULTIPLY BY 2ND DIGIT
        XOR   A
        CALL  SRD         ; SHIFT ACCUM RIGHT
        RLA               ; PUT DIGIT IN HIGH 4 BITS
        RLA
        RLA
        RLA
        AND   0F0H
        LD    C,A         ; SAVE UPPER DIGIT
        POP   AF          ; GET LAST DIGIT
        OR    C           ; COMBINE
        LD    (IY-1),A    ; STORE IN SEED
RN20    LD    C,(IY+6)    ; GET NEXT 2 MPLIER DIGITS
        DEC   IY          ; DECR PTR
        CALL  MLDIG       ; MULTIPLY BY 1ST DIGIT
        XOR   A
        CALL  SRD         ; SHIFT ACCUM RIGHT
        DJNZ  RN10
        LD    (IY-1),A    ; STORE LAST DIGIT
;
; ADD 27182 81828 459  (CRNDC) TO (SEED)
;
        LD    B,7         ; LOOP COUNTER
        LD    HL,CRNDC0   ; PT TO CONSTANT
        LD    DE,SEED10   ; PT TO LS BYTE OF SEED
        AND   A           ; CLEAR CY
;
; ADD LOOP
;
RN30    LD    A,(DE)      ; GET 2 DIGITS
        ADC   A,(HL)      ; ADD 2 DIGITS
        DAA
        LD    (DE),A      ; STORE SUM
        DEC   DE
        DEC   HL
        DJNZ  RN30        ; COUNTER NOT 0
;
; SCALE AND NORMALIZE RESULT
;
        INC   DE          ; PT TO MS BYTE
        AND   0FH         ; CLEAR SIGN DIGIT
        LD    (DE),A
        EX    DE,HL       ; PUT PTR IN HL
        CALL  LD          ; COPY SEED TO (IX)
        LD    C,80H       ; SET EXPONENT
        CALL  NORML1      ; NORMALIZE
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;
; RANDOMIZE    USES REFRESH REGISTER TO MODIFY SEED
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
RANDMZ  LD   A,R          ; GET REFRESH REGISTER
        OR   A
        DAA               ; CONVERT TO DECIMAL
        LD   (SEED10),A
        RET
*H INT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; INTEGER FUNCTION -- FLOOR FUNCTION
;   INPUT,OUTPUT: IX POINTS TO FL PT NUMBER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INT     XOR  A
        CP   (IX+7)
        RET  Z            ; X WAS ZERO
        LD   A,8DH
        SUB  (IX+7)       ; # DIGITS RIGHT OF PT
        RET  C            ; X >= 10^13
        RET  Z            ; X >= 10^12
        CP   13
        JR   NC,INT40     ; X < 1
        LD   B,A
;
; SET POINTER TO 1ST DIGIT RIGHT OF PT
;
        LD   A,14
        SUB  B            ; 1+# DIGITS LEFT OF PT
        LD   E,A
        SRL  E            ; # BYTES TO SKIP
        LD   D,0
        PUSH IX           ; GET PTR
        POP  HL
        ADD  HL,DE
;
; CLEAR FIRST DIGIT IF A IS ODD
;
        AND  1
        JR   Z,INT10      ; DEC PT IS BETWEEN BYTES
        LD   A,(HL)       ; GET BYTE WITH DEC PT
        LD   C,A          ; SAVE RIGHT HALF
        AND  0F0H         ; CLEAR RIGHT HALF
        LD   (HL),A       ; REPLACE IT
        INC  HL
        LD   A,C          ; GET RIGHT HALF
        AND  0FH
;
; CLEAR REMAINING BYTES, OR-ING EACH INTO A FIRST
;
INT10   SRL  B            ; CALC # BYTES
        JR   Z,INT30      ; NONE TO CLEAR
INT20   OR   (HL)
        LD   (HL),0
        INC  HL
        DJNZ INT20
;
; POSSIBLE CORRECTION FOR NEGATIVE NUMBERS
```

```
;
INT30   BIT  7,(IX)
        RET  Z          ; POSITIVE
        AND  A
        RET  Z          ; INPUT WAS AN INTEGER
        LD   IY,C1
        CALL SUB        ; SUBTRACT 1 TO CORRECT
        RET
;
; X < 1, NONZERO:  OUTPUT 0 IF +, -1 IF -
;
INT40   BIT  7,(IX)
        JP   Z,ZERO     ; POSITIVE FRACTION
        CALL ONE
        SET  7,(IX)     ; NEGATIVE 1
        RET
*H SHORT FUNCTIONS
;
; BRK SEPARATES INTEGER AND FRACTION PARTS
;
;   INPUT:  IX  PTS TO X
;   OUTPUT: IX  PTS TO FRACTIONAL PART OF X (POSITIVE)
;           HL  CONTAINS 2'S COMPL INTEGER PART OF X,
;               LOW ORDER 16 BITS VALID IF X < 10^13
;           CY  SET IF INTEGER WON'T FIT IN 16 BITS
;           Z   SET IF EXACT INTEGER
;           A'  IS ORIGINAL SIGN
;
BRK     LD   A,(IX)     ; GET SIGN
        PUSH AF
        CALL ST04       ; COPY X
        CALL INT        ; TAKE INTEGER PART
        LD   IY,TEMP4
        CALL RSUB       ; CALC. FRACTION PART
        PUSH IX         ; SAVE PTR
        LD   IX,TEMP4
        CALL FIXA       ; CALC. BINARY INTEGER PART
        POP  IX
        EX   AF,AF'
        POP  AF         ; GET ORIGINAL SIGN
        EX   AF,AF'
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; ABSOLUTE VALUE FUNCTION
;   INPUT,OUTPUT: IX POINTS TO FL PT NUMBER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ABS     RES  7,(IX)     ; TURN OFF SIGN BIT
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; SIGN FUNCTION
;   INPUT,OUTPUT: IX POINTS TO FL PT NUMBER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SGN     XOR  A          ; CLEAR A
        CP   (IX+7)
        RET  Z          ; ZERO INPUT IS NOP
        LD   A,(IX)     ; GET SIGN
```

```
        AND  80H
        CALL ONE        ; OUTPUT 1
        OR   (IX)
        LD   (IX),A     ; SIGN OF INPUT
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; NEGATE FUNCTION
;   INPUT,OUTPUT: IX POINTS TO FL PT NUMBER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
NEG     LD   A,(IX)     ; GET SIGN BIT
        XOR  80H        ; COMPLEMENT IT
        LD   (IX),A     ; RESTORE
        RET
*H INTEGER ADD AND SUB
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; IADD, ISUB  INTEGER ADD AND SUBTRACT
;
;   INPUT:  IX  PTS TO X
;           IY  PTS TO Y
;   OUTPUT: IX  PTS TO SUM OR DIFFERENCE
;   ERRORS: OVERFLOW (POSITIVE OR NEGATIVE)
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IADD    CALL ASLOAD     ; LOAD ARGUMENTS
        ADC  HL,DE
        JP   PO,ASSTO   ; NO OVERFLOW
ASOV    JP   P,NINF     ; OVFLW SIGN POS
        JP   PINF
ISUB    CALL ASLOAD     ; LOAD ARGUMENTS
        SBC  HL,DE
        JP   PO,ASSTO   ; NO OVERFLOW
        JR   ASOV
;
; ASLOAD  LOADS ARGS TO HL, DE, RESETS CY
;
ASLOAD  LD   H,(IX+1)
        LD   L,(IX)
        LD   D,(IY+1)
        LD   E,(IY)
        AND  A
        RET
;
; ASSTO  STORES HL AS RESULT
;
ASSTO   LD   (IX),L
        LD   (IX+1),H
        RET
*H INTEGER MULTIPLY
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; IMUL  INTEGER MULTIPLY
;
;   INPUT:  IX  PTS TO X
;           IY  PTS TO Y
;   OUTPUT: IX  PTS TO PRODUCT
;   ERRORS: OVERFLOW (POS OR NEG)
;   REGISTERS:  DE IS MULTIPLICAND
```

```
;               AC IS MULTIPLIER AND ACCUM EXTENSION
;               HL IS ACCUM
;               B  IS LOOP COUNTER
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IMUL    CALL MDLOAD     ; LOAD ARGS
        LD   A,B        ; X IN AC
;
; 16 BIT BY 16 BIT UNSIGNED MULTIPLY
;
        LD   HL,0       ; CLEAR ACCUM
        LD   B,16       ; SET COUNTER
        RRA             ; INITIALIZE CARRY
        RR   C
IM10    JR   NC,IM20    ; MPLIER BIT IS ZERO
        ADD  HL,DE      ; ADD MULTIPLICAND
IM20    RR   H          ; ROT ACC-MPLICAND INTO CY
        RR   L
        RRA
        RR   C
        DJNZ IM10       ; COUNTER NOT 0
;
; TEST FOR > 16 BIT PRODUCT
;
        LD   B,A        ; PRODUCT IN BC
        LD   A,H
        OR   L
        JR   NZ,INF?    ; OVERFLOW
;
; TEST FOR >15 BITS, AND STORE RESULTS
;
MDST0   EX   AF,AF'     ; GET SIGN OF RESULT
        AND  80H
        JR   Z,MDS20    ; POS RESULT
;
; NEGATE BC
;
        LD   HL,0
        SBC  HL,BC
        JR   Z,MDS10    ; RESULT WAS ZERO
        JP   P,NINF     ; NEGATIVE OVERFLOW
MDS10   LD   (IX),L     ; STORE RESULT
        LD   (IX+1),H
        RET
;
; TEST FOR > 15 BITS
;
MDS20   BIT  7,B
        JR   NZ,PINF    ; POSITIVE OVERFLOW
        LD   (IX),C
        LD   (IX+1),B
        RET
;
; INF? CHECKS SIGN BIT IN A'
; PINF PRODUCES POS OVERFLOW
; NINF PRODUCES NEG OVERFLOW
;
IDV0    LD   HL,ERROR
        SET  4,(HL)     ; FLAG ERROR
INF?    EX   AF,AF'     ; GET SIGN BIT
```

```
        AND   80H
        JR    NZ,NINF     ; NEG SIGN
PINF    LD    (IX),0FFH   ; RESULT IS 7FFFH
        LD    (IX+1),7FH
        JR    FINF
NINF    LD    (IX),0      ; RESULT IS 8000H
        LD    (IX+1),80H
FINF    LD    HL,ERROR
        SET   6,(HL)      ; FLAG ERROR
        RET
*H INTEGER DIVIDE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; IDIV  INTEGER DIVIDE
;
;   INPUT:  IX  PTS TO X
;           IY  PTS TO Y
;   OUTPUT: IX  PTS TO X/Y
;   ERRORS: Y IS ZERO
;   ALGORITHM: NON-RESTORING DIVISION, 16 BY 15
;   REGISTERS:  AC IS DIVIDEND, ACCUM EXT
;               DE IS DIVISOR
;               HL IS ACCUM, REMAINDER AT END
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IDIV    CALL MDLOAD     ; LOAD ARGS
        LD   A,D
        OR   E
        JR   Z,IDV0     ; DIVISOR IS ZERO
        LD   A,B        ; X IN AC
;
; 16 BIT BY 15 BIT UNSIGNED DIVIDE
;
        LD   HL,0       ; CLEAR ACCUM
        LD   B,16       ; SET COUNTER
IDV10                   ; 0 <= HL < DE
        RL   C          ; ROT ACC-RESULT LEFT
        RLA
        ADC  HL,HL      ; NO CY POSSIBLE
        SBC  HL,DE      ; SUB DIVISOR
IDV20   CCF             ; CALC RESULT BIT
        JR   NC,IDV50   ; ACCUM WENT NEGATIVE
IDV30   DJNZ IDV10      ; COUNTER NOT 0
        JP   IDV60
IDV40                   ; -DE <= HL < 0
        RL   C          ; RO ACC-RESULT LEFT
        RLA
        ADC  HL,HL
        AND  A          ; TURN OFF CY
        ADC  HL,DE      ; ADD DIVISOR
        JR   C,IDV30    ; ACCUM WENT POSITIVE
        JR   Z,IDV20    ; ACCUM BECAME ZERO
IDV50   DJNZ IDV40      ; COUNTER NOT 0
IDV60   RL   C          ; SHIFT IN LAST RESULT BIT
        RLA
        LD   B,A        ; QUOTIENT IN BC
        JP   MDSTO      ; STORE RESULT
;
; MDLOAD  LOADS X INTO BC, Y INTO DE
;   TAKES ABS OF EACH, STORES XOR OF SIGNS
```

```
;   IN A'
;
MDLOAD  LD   B,(IX+1)    ; LOAD X
        LD   C,(IX)
        LD   A,B
        AND  80H
        JR   Z,MDL10     ; X IS POSITIVE
;
; NEGATE X
;
        LD   HL,0
        SBC  HL,BC
        LD   B,H
        LD   C,L
;
; LOAD Y
;
MDL10   LD   D,(IY+1)
        LD   E,(IY)
        XOR  D           ; FORM SIGN OF RESULT
        EX   AF,AF'      ; SAVE IT
        BIT  7,D
        RET  Z           ; Y IS POSITIVE
;
; NEGATE Y
;
        LD   HL,0
        AND  A
        SBC  HL,DE
        EX   DE,HL
        RET
*H IABS AND ISGN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; IABS, INEG
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IABS    BIT  7,(IX+1)
        RET  Z           ; X IS POSITIVE
INEG    LD   D,(IX+1)    ; LOAD X
        LD   E,(IX)
        LD   HL,0        ; NEGATE X
        AND  A
        SBC  HL,DE
        JP   PE,NINF     ; NEG OVERFLOW
        LD   (IX+1),H    ; STORE X
        LD   (IX),L
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; ISGN
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ISGN    LD   A,(IX+1)    ; GET SIGN
        BIT  7,A
        JR   Z,ISG10     ; X IS NONNEG
        LD   (IX),0FFH   ; STORE -1
        LD   (IX+1),0FFH
        RET
ISG10   OR   (IX)
```

```
        RET  Z          ; X IS ZERO
        LD   (IX+1),1   ; STORE +1
        LD   (IX),0
        RET
*H ICP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; ICP  INTEGER COMPARE
;
;   RETURNS CY SET IF X < Y
;   RETURNS Z  SET IF X = Y
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ICP     LD   B,(IX+1)   ; GET SIGN OF X
        LD   A,B
        AND  80H        ; TEST SIGN, CLEAR CY
        JR   NZ,ICP10   ; X IS NEG
        BIT  7,(IY+1)
        RET  NZ         ; Y IS NEG
        LD   A,B
        CP   (IY+1)     ; SIGNS ARE BOTH POS
        RET  NZ
        LD   A,(IX)
        CP   (IY)
        RET
ICP10   XOR  (IY+1)     ; TEST SIGN, CLEAR Z
        RLA             ; SIGN BIT INTO CY
        RET  C          ; SIGNS DIFFERENT
        LD   A,B
        CP   (IY+1)     ; BOTH SIGNS NEG
        RET  NZ
        LD   A,(IX)
        CP   (IY)
        RET
;
; REFERENCES
;
        EXTERNAL MULT,DIVI,ERROR,OVER,UNDER,SAVE
        EXTERNAL         SEED,SEED10,CRNDA,CRNDC0
        EXTERNAL         SRD,LD,MLDIG,MDTL
        EXTERNAL         ZER0,ONE,NORML1,SLD,SUB,C1,MA0
        EXTERNAL         ST04,TEMP4,RSUB,FIXA
        EXTERNAL         LMARK,LREST,LBACK,RC2,LIN
        GLOBAL  DTOB,BTOD
        GLOBAL RND,RANDMZ,INT,BRK,ABS,SGN,NEG
        GLOBAL IADD,ISUB,IMUL,IDIV,IABS,ISGN,ICP,INEG
*H POWER
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; X TO THE Y POWER
;
;   INPUT:  IX  PTS TO X
;           IY  PTS TO Y
;   OUTPUT: IX  PTS TO X^Y
;   ERRORS: NONE DIRECTLY
;   SPECIAL CASES:  IF Y IS 0, RETURN 1
;   ALGORITHM:  IF Y IS AN EXACT INTEGER -32768 <= Y < 32768
;               THEN USE A SEQUENCE OF SQUARING AND
;               MULTIPLYING BY X TO PRODUCE RESULT
;       OTHERWISE, RESULT IS EXP(Y*LOG(X))
```

```
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; DETERMINE IF Y IS AN INTEGER
;
POW     PUSH IY         ; SAVE PTR TO Y
        PUSH IX         ; SAVE PTR TO X
        PUSH IY         ; PT TO Y
        POP  IX
        CALL FIXA       ; CONVERT TO INTEGER
        POP  IX         ; RESTORE IX
        JR   Z,POW10    ; Y WAS AN INTEGER
;
; FOR NONINTEGER Y, TAKE EXP(Y*LOG(X))
;
        CALL LOG        ; LOG OF X
        POP  IY         ; RESTORE IY
        CALL MULT       ; MPY BY Y
        CALL EXP        ; EXP OF PRODUCT
        RET
;
; Y IS INTEGER, SET UP FOR LOOP, GET MAGNITUDE
;
POW10   XOR  A          ; CHECK FOR 0
        POP  IY         ; RESTORE IY
        CP   (IY+7)
        JP   Z,ONE      ; X TO 0 POWER
        PUSH HL         ; SAVE Y
        BIT  7,H
        JR   Z,POW15    ; Y IS POSITIVE
        EX   DE,HL      ; NEGATE Y
        LD   HL,0
        AND  A
        SBC  HL,DE
        EX   (SP),HL    ; SAVE NEGATED Y INSTEAD
        CALL REC        ; CALC 1/X
POW15   CALL STO4       ; SAVE COPY OF X
        POP  HL         ; GET Y
        LD   B,16       ; SET COUNTER
;
; LOOK FOR FIRST BIT
;
POW20   PUSH BC         ; SAVE COUNTER
        ADD  HL,HL      ; SHIFT Y LEFT 1
        JR   C,POW40    ; 1 BIT FOUND
        POP  BC         ; GET COUNTER
        DJNZ POW20      ; COUNTER NOT 0
        JP   ONE        ; THIS JUMP SHOULD NOT HAPPEN
;
; SQUARE FOR EACH ZERO BIT
; SQUARE AND MULTIPLY FOR EACH 1 BIT
;
POW30   PUSH BC         ; SAVE COUNTER
        ADD  HL,HL      ; SHIFT Y LEFT 1
        PUSH HL         ; SAVE Y
        PUSH AF         ; SAVE CY
        CALL STO5       ; SAVE COPY OF ACCUM
        LD   IY,TEMP5
        CALL MULT       ; SQUARE X
        POP  AF         ; GET CY
```

```
        JR   NC,POW35    ; WAS A 0 BIT
        CALL MULT4       ; MPY BY X
POW35   POP  HL          ; GET Y
POW40   POP  BC          ; GET COUNTER
        DJNZ POW30       ; COUNTER NOT 0
        RET
;
; IPOW  POWERS FOR INTEGER ARGUMENTS
;   USES REAL POW ROUTINE BY FLOATING ARGS
;   IN TEMP5, TEMP6, (UNUSED BY POW WHEN Y
;   IS INTEGER), CALLING POW, AND FIXING RESULT
;
IPOW    PUSH IX          ; SAVE ORIGINAL PTR
        LD   L,(IY)      ; GET Y
        LD   H,(IY+1)
        LD   IX,TEMP6
        CALL FLOAT1      ; FLOAT Y IN TEMP6
        POP  IX          ; GET ORIGINAL PTR
        PUSH IX
        LD   L,(IX)      ; GET X
        LD   H,(IX+1)
        LD   IX,TEMP5
        CALL FLOAT1      ; FLOAT X IN TEMP5
        LD   IY,TEMP6    ; PT TO Y
        CALL POW         ; CALL REAL ROUTINE
        CALL FIX         ; FIX RESULT
        POP  IX          ; GET ORIGINAL PTR
        LD   (IX),L      ; STORE RESULT
        LD   (IX+1),H
        RET
*H SQUARE ROOT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; SQUARE ROOT FUNCTION
;
;   INPUT:  IX PTS TO X
;   OUTPUT: IX PTS TO SQR(X)
;   ERRORS: X IS NEGATIVE, RETURNS SQR(ABS(X))
;   SPECIAL CASES:  X IS 0
;   ALGORITHM:  INPUT IS SCALED TO .1 <= X < 1
;       BY REMOVING THE EXPONENT.  HALF THE
;       EXPONENT IS ADDED TO THE RESULT EXPONENT
;       AND THE RESULT IS MULTIPLIED BY SQR(10)
;       IF THE ORIGINAL EXP WAS ODD.
;       AN INITIAL APPROXIMATION IS GIVEN BY
;       A 1ST ORDER POLYNOMIAL, AND IMPROVED TO
;       REQUIRED ACCURACY IN 4 ITERATIONS OF
;       NEWTON'S METHOD.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CHECK VALID ARGUMENTS
;
SQR     LD   A,(IX+7)
        OR   A
        JP   Z,ZERO      ; ZERO INPUT
        BIT  7,(IX)
        CALL NZ,SQERR    ; NEGATIVE INPUT
;
; SCALE BY STRIPPING EXPONENT
```

```
;
        PUSH AF          ; SAVE ORIGINAL EXP
        LD   (IX+7),80H  ; SET ZERO EXP
;
; CALC INITIAL APPROX
;
        CALL ST04        ; SAVE X
        PUSH IX
        LD   IX,TEMP4
        CALL HALF        ; CALC X/2 IN TEMP4
        POP  IX
        LD   IY,PSQ00
        CALL ADD         ; INIT APPROX.
;
; PERFORM 4 ITERATIONS
;
        LD   B,4
SQ10    PUSH BC          ; SAVE COUNTER
        CALL ST05        ; SAVE LAST APPROX IN TEMP5
        LD   HL,TEMP4
        CALL LD          ; GET X/2
        LD   IY,TEMP5
        CALL DIVI        ; DIVIDE BY LAST APPROX
        PUSH IX
        LD   IX,TEMP5
        CALL HALF        ; DIVIDE LAST APPROX BY 2
        POP  IX
        CALL ADD         ; CALC NEW APPROX
        POP  BC
        DJNZ SQ10        ; COUNTER NOT 0
;
; CALC EXPONENT, SCALE RESULT
;
        POP  AF          ; GET ORIGINAL EXP
        AND  A
        RRA              ; DIVIDE BY TWO
        RR   B           ; SAVE CY
        ADD  A,40H       ; CORRECT THE BIAS
        LD   (IX+7),A    ; STORE IT
        BIT  7,B         ; TEST ODD OR EVEN
        RET  Z           ; ORIGINAL EXP EVEN
        LD   IY,SQR10
        CALL MULT        ; MPY BY SQR(10)
        RET
;
; SQUARE ROOT OF NEGATIVE NUMBER
;
SQERR   LD   HL,ERROR
        SET  3,(HL)      ; FLAG ERROR
        RES  7,(IX)      ; MAKE POSITIVE
        RET
*H EXP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; EXPONENTIATION ROUTINE
;
;   INPUT:  IX  PTS TO X
;   OUTPUT: IX  PTS TO E↑X
;   ERRORS: OVER/UNDERFLOW IN HANDLING EXPONENT PART
;   ALGORITHM:  X:=X*2*LOG BASE 10 OF E
```

```
;       SAVE:=INT(X)
;       X:=FRACTION PART(X) / 2
;       X:=10↑X, WHERE 0 <= X < .5
;       IF SAVE IS ODD THEN X:=X*SQR(10)
;       X:=X*10↑INT(SAVE/2)
;
;       THE APPROXIMATION TO 10↑X IS GIVEN BY :
;               (Q(X↑2) + X*P(X↑2)) / (Q(X↑2) - X*P(X↑2))
;       WHERE Q IS 3RD ORDER POLYNOMIAL, P IS 2ND ORDER.
;       REFERENCE IS "COMPUTER APPROXIMATIONS", HART, JOHN F.
;           ET AL, FUNCTION NUMBER 1444
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CONVERT E↑X TO 10↑X
;
EXP     LD   IY,LOGE2   ; MULT BY 2*LOG10(E)
        CALL MULT
;
; GET 8 BIT 2'S COMPL INTEGER PART IN A
;
        CALL BRK        ; BREAK INTEGER AND FRACTION PARTS
        JP   C,EXERR1   ; INTEGER PART > 16 BITS
        LD   A,H
        AND  1          ; SIGN BIT IN A0
        ADD  A,H        ; 0 IFF H WAS FF OR 00
        JP   NZ,EXERR2  ; INTEGER > 9 BITS
        PUSH HL         ; SAVE IT
;
; SCALE FRACTION PART TO 0 <= X < .5
;
        XOR  A
        CP   (IX+7)
        JR   Z,EX10     ; FRACTION IS ZERO
        CALL HALF
        JP   C,UNDER    ; UNDERFLOW
;
; EVALUATE RATIONAL APPROX. TO 10↑X
;
EX10    CALL XX2        ; STORE X, X↑2 IN TEMP4, 5
        LD   IY,PEX02
        LD   B,2        ; 2ND ORDER POLY
        CALL POLY       ; EVALUATE P(X↑2)
        CALL MULT4      ; CALC. X*P(X↑2)
        CALL ST04       ; SAVE X*P IN TEMP4
        LD   HL,TEMP5
        CALL LD         ; GET X↑2
        LD   IY,QEX02
        LD   B,2        ; 3RD ORDER POLY(1)
        CALL POLY1      ; EVALUATE Q(X↑2)
        CALL ST05       ; SAVE Q IN TEMP5
        LD   IY,TEMP4
        CALL SUB
        CALL ST06       ; SAVE Q-XP IN TEMP6
        LD   HL,TEMP5
        CALL LD         ; GET Q
        LD   IY,TEMP4
        CALL ADD        ; CALC. Q+XP
        LD   IY,TEMP6
        CALL DIVI       ; FINAL RESULT
```

```
;
; POSSIBLE SCALE BY SQR(10)
;
        POP   HL          ; GET 9 BIT INTEGER
        SRA   H           ; DIVIDE BY 2
        RR    L
        PUSH  HL          ; SAVE 8 BIT INTEGER
        LD    IY,SQR10
        CALL  C,MULT      ; INTEGER WAS ODD
;
; ADD INTEGER PART TO RESULT EXPONENT
;
        POP   BC          ; PUT INTEGER IN C
        LD    A,(IX+7)    ; GET EXP
        SUB   80H         ; REMOVE BIAS
        ADD   A,C         ; ADD INTEGER PART
        JP    PE,EXERR3   ; OVER/UNDERFLOW
        ADD   A,80H       ; ADD BIAS
        JP    Z,UNDER     ; UNDERFLOW
        LD    (IX+7),A    ; STORE EXP
        RET
;
; EXP ERRORS
;
EXERR1  EX    AF,AF'      ; GET ORIGINAL SIGN
        LD    H,A
EXERR2  BIT   7,H
        JP    NZ,UNDER    ; - SIGN
        JP    OVER
EXERR3  JP    P,UNDER     ; + SIGN AFTER OVERFLOW
        JP    OVER
*H LOG
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; LOG ROUTINE
;
;   INPUT:  IX  PTS TO X
;   OUTPUT: IX  PTS TO NATURAL LOG OF X
;   ERRORS: X IS NEGATIVE, RETURN LOG(ABS(X))
;           X IS ZERO, RETURN NEGATIVE OVERFLOW
;   SPECIAL CASES:  X IS 1, RETURN ZERO
;   ALGORITHM:  SAVE:=INT(X)
;       X:=SQR(10)*FRACTION-PART(X)
;       X:=LOG BASE E (X), WHERE  1/SQR(10) < X < SQR(10)
;       X:=X - LOG BASE E(SQR(10)) + SAVE*LOG BASE E (10)
;
;       LOG IS APPROXIMATED BY :
;               Z*P(Z^2) / Q(Z^2)  WHERE Z = (X-1) / (X+1)
;       AND P IS A 3RD ORDER POLYNOMIAL, AND Q IS 4TH ORDER.
;       REFERENCE FUNCTION IS 2686.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CHECK FOR VALID ARGUMENTS
;
LOG     BIT   7,(IX)
        CALL  NZ,LGERR1   ; NEGATIVE ARG
        LD    IY,C1
        CALL  CP
        JP    Z,ZERO      ; LOG(1)
```

```
        LD    A,(IX+7)    ; GET EXPONENT
        OR    A
        JP    Z,LGERR2    ; ZERO ARG
;
; TAKE LOG OF EXPONENT PART
;
        SUB   80H         ; REMOVE BIAS
        LD    L,A         ; PUT IN HL
        LD    H,0         ; SIGN EXTEND
        JR    NC,L010     ; + SIGN
        LD    H,0FFH
L010    PUSH HL           ; SAVE INTEGER
;
; SCALE TO 1/SQR(10) < X < SQR(10)
;
        LD    (IX+7),80H ; SCALE TO 0 < X < 1
        LD    IY,SQR10
        CALL MULT
;
; APPROXIMATION FOR LOGARITHM
;
        CALL ST04         ; SAVE X IN TEMP4
        PUSH IX           ; SAVE PTR
        LD    IX,TEMP4
        LD    IY,C1
        CALL ADD          ; X+1 IN TEMP4
        POP   IX
        LD    IY,C1
        CALL SUB          ; X-1 IN (IX)
        LD    IY,TEMP4
        CALL DIVI
        CALL XX2          ; STORE Z,Z↑2 IN TEMP4,5
        LD    IY,QL003
        LD    B,3
        CALL POLY1        ; EVALUATE Q(Z↑2)
        CALL ST06         ; Q IN TEMP6
        LD    HL,TEMP5
        CALL LD           ; LOAD Z↑2
        LD    IY,PL003
        LD    B,3
        CALL POLY         ; EVALUATE P(Z↑2)
        CALL MULT4        ; CALC. Z*P(Z↑2)
        LD    IY,TEMP6
        CALL DIVI         ; CALC ZP/Q
        LD    IY,LNSQ10
        CALL SUB          ; SUB LOGE(SQR(10))
        CALL ST04         ; SAVE IN TEMP4
;
; CONVERT 16 BIT EXPONENT PART TO FLOATING PT
;
        POP   HL          ; GET INTEGER
        CALL FLOAT1       ; FLOAT IT
        LD    IY,LN10
        CALL MULT         ; MPY BY LOG BASE E(10)
        LD    IY,TEMP4
        CALL ADD          ; FINAL RESULT
        RET
;
; ERRORS
;
```

```
LGERR1  LD    HL,ERROR
        SET   2,(HL)      ; FLAG LOG OF NEG
        RES   7,(HL)      ; TAKE ABS VALUE
        RET
LGERR2  SET   7,(IX)      ; SET NEGATIVE SIGN
        JP    OVER        ; LOG OF ZERO
*H TANGENT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; TANGENT ROUTINE
;
;   INPUT:  IX  PTS TO X (IN RADIANS)
;   OUTPUT: IX  PTS TO TAN(X)
;   ERRORS: NONE DIRECTLY
;   ALGORITHM:  X := X*4/PI
;       SAVE := INT(X) LAND 3  (SAVE LOW 2 BITS ONLY)
;       X := FRACTION-PART(X)
;       IF SAVE IS 0,2 THEN X := 1-X
;       X := TAN(X*PI/4),  WHERE 0 <= X <= PI/4
;       IF SAVE IS 0,3 THEN X := 1/X
;       IF SAVE IS 0,1 THEN X := -X
;
;       THE APPROXIMATION TO TAN(X*PI/4) IS GIVEN BY:
;               X*P(X↑2) / Q(X↑2)
;       WHERE P IS ORDER 3 POLYNOMIAL, Q IS ORDER 4
;       REFERENCE FUNCTION IS 4286
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; SCALE BY 4/PI
;
TAN     LD    IY,FOVRPI
        CALL  MULT
        CALL  BRK         ; GET INTEGER,FRACTION
        LD    A,L
        PUSH  AF          ; SAVE FLAG BITS
        AND   1           ; TEST FLAG
        JR    Z,TA10      ; FLAG WAS 0 OR 2
        LD    IY,C1
        CALL  RSUB        ; CALC. 1-FRACTION PART
;
; TAN APPROX
;
TA10    CALL  XX2         ; SAVE X,X↑2 IN TEMP4,5
        LD    IY,QTA03
        LD    B,3
        CALL  POLY1       ; EVALUATE Q(X↑2)
        CALL  ST06        ; SAVE Q
        LD    HL,TEMP5
        CALL  LD          ; GET X↑2
        LD    IY,PTA03
        LD    B,3
        CALL  POLY        ; EVALUATE P(X↑2)
        CALL  MULT4       ; CALC. X*P(X↑2)
        POP   AF          ; GET 2 BIT FLAG
        PUSH  AF
        AND   3
        JP    PE,TA20     ; FLAG WAS 0 OR 3
        CALL  ST04        ; SAVE X*P
        LD    HL,TEMP6
```

```
        CALL LD          ; GET Q
        LD   IY,TEMP4    ; SET FOR Q/XP
        JP   TA30
TA20    LD   IY,TEMP6    ; SET FOR XP/Q
TA30    CALL DIVI
;
; CALC. SIGN OF RESULT
;
        POP  AF          ; GET FLAG
        AND  2
        RET  Z           ; ON 0 OR 1
        SET  7,(IX)
        RET
*H COSINE,SINE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; COSINE ROUTINE
;   EFFECTIVELY ADDS PI/2 AND CALLS SINE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
COS     LD   IY,FOVRPI
        CALL MULT        ; SCALE BY 4/PI
        CALL BRK         ; GET INTEGER, FRACTION
        LD   A,L
        ADD  A,2         ; ADD EFFECTIVE PI/2
        JP   SI10
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; SINE ROUTINE
;
;   INPUT:  IX  PTS TO X (IN RADIANS)
;   OUTPUT: IX  PTS TO SIN(X)
;   ERRORS: NONE
;   ALGORITHM:  X := X*4/PI
;       SAVE := INT(X) LAND 7
;       X := FRACTION-PART(X)
;       IF SAVE IS 1,3,5,7 THEN X := 1-X
;       IF SAVE IS 0,3,4,7 THEN X := SIN(X*PI/4)
;       IF SAVE IS 1,2,5,6 THEN X := COS(X*PI/4)
;       IF SAVE IS 4,5,6,7 THEN X := -X
;
;       THE SINE APPROX FOR 0 <= X <= PI/4 IS:
;               X*P(X↑2) / Q(X↑2)
;       WHERE P IS A 3RD ORDER POLYNOMIAL, Q IS 3RD ORDER.
;       REFERENCE FUNCTION IS 3063.
;
;       THE COSINE APPROX IS:
;               P(X↑2) / Q(X↑2)
;       WHERE P IS A 3RD ORDER POLY, Q IS 3RD ORDER.
;       REFERENCE FUNCTION IS 3843.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SIN     LD   IY,FOVRPI
        CALL MULT        ; SCALE BY 4/PI
        CALL BRK         ; GET INTEGER AND FRACTION
        LD   A,L
SI10    PUSH AF          ; SAVE 3 BIT FLAG
        AND  1
        JR   Z,SI20      ; FLAG IS 0,2,4,6
        LD   IY,C1
```

```
        CALL RSUB        ; CALC. 1-X
SI20    POP  AF          ; GET FLAG
        PUSH AF
        AND  3
        JP   PO,SI30     ; 1,2,5,6
;
; SIN APPROX
;
        CALL XX2         ; STORE X,X^2 IN TEMP4,5
        LD   IY,QSI02
        LD   B,2
        CALL POLY1       ; EVALUATE Q(X^2)
        CALL ST06        ; SAVE Q
        LD   HL,TEMP5
        CALL LD          ; GET X^2
        LD   IY,PSI03
        LD   B,3
        CALL POLY        ; EVALUATE P(X^2)
        CALL MULT4       ; CALC. X*P(X^2)
        LD   IY,TEMP6
        CALL DIVI        ; CALC. XP/Q
        JP   SI40
;
; COS APPROX
;
SI30    CALL XX2         ; STORE X,X^2 IN TEMP4,5
        LD   IY,QCO02
        LD   B,2
        CALL POLY1       ; EVALUATE Q(X^2)
        CALL ST06
        LD   HL,TEMP5
        CALL LD          ; GET X^2
        LD   IY,PCO03
        LD   B,3
        CALL POLY        ; EVALUATE P(X^2)
        LD   IY,TEMP6
        CALL DIVI        ; CALC. P/Q
;
; CALC SIGN OF RESULT
;
SI40    POP  AF          ; GET FLAG
        AND  4
        RET  Z           ; 0,1,2,3
        SET  7,(IX)
        RET
*H ARCTANGENT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; ARCTANGENT ROUTINE
;
;   INPUT:  IX  PTS TO X
;   OUTPUT: IX  PTS TO ATN(X)  (IN RADIANS)
;   ERRORS:  NONE
;   ALGORITHM:   SAVE := SGN(X)
;       FLAG := 0
;       IF TAN(3*PI/8) <= X THEN X:=1/X, FLAG:=1
;       IF TAN(PI/8) <= X THEN X:=(X-1)/(X+1), FLAG:=2
;       X := ATN(X)   , WHERE 0 <= X <= TAN(PI/8)
;       IF FLAG = 1 THEN X := X + PI/4
;       IF FLAG = 2 THEN X := PI/2 - X
```

```
;       X := X*SAVE
;
;       APPROXIMATION FOR ATN(X) IS:
;               X*P(X^2) / Q(X^2)
;       WHERE P IS 3RD ORDER POLYNOMIAL, Q IS 4TH ORDER.
;       REFERENCE FUNCTION IS 5075
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; TAKE ABS VALUE, DETERMINE RANGE
;
ATN     LD   A,(IX)     ; GET SIGN
        AND  80H
        PUSH AF         ; SAVE SIGN
        RES  7,(IX)     ; MAKE POSITIVE
        LD   IY,TAPI8
        CALL CP         ; COMPARE TO 1ST LIMIT
        LD   B,0
        JR   C,AT20     ; < TAN(PI/8)
        LD   IY,TA3PI8
        CALL CP         ; COMPARE TO 2ND LIMIT
        JR   C,AT10     ; < TAN(3*PI/8)
;
; TAN(3*PI/8) <= X
;   TAKE 1/X NOW, SUB FROM PI/2 AT END
;
        CALL REC        ; RECIPROCAL
        LD   B,2        ; SET FLAG TO 2
        JP   AT20
;
; TAN (PI/8) <= X < TAN(3*PI/8)
;   CALC (X-1)/(X+1) NOW, ADD PI/4 AT END
;
AT10    CALL ST04       ; SAVE X IN TEMP4
        PUSH IX
        LD   IX,TEMP4
        LD   IY,C1
        CALL ADD        ; X+1 IN TEMP4
        POP  IX
        LD   IY,C1
        CALL SUB        ; X-1 IN ACCUM
        LD   IY,TEMP4
        CALL DIVI
        LD   B,1        ; SET FLAG TO 1
;
; APPROX FOR ARCTAN
;
AT20    PUSH BC         ; SAVE FLAG
        CALL XX2        ; SAVE X,X^2
        LD   IY,QAT03
        LD   B,3
        CALL POLY1      ; CALC. Q(X^2)
        CALL ST06       ; SAVE Q
        LD   HL,TEMP5
        CALL LD         ; GET X^2
        LD   IY,PAT03
        LD   B,3
        CALL POLY       ; CALC P(X^2)
        CALL MULT4      ; X*P
        LD   IY,TEMP6
```

```
        CALL DIVI       ; X*P/Q
        POP  BC         ; GET FLAG
        BIT  0,B
        JR   Z,AT30     ; FLAG IS 0 OR 2
; SECOND INTERVAL (FLAG IS 1)
        LD   IY,PI4
        CALL ADD        ; ADD PI/4
        JP   AT40
AT30    BIT  1,B
        JR   Z,AT40     ; FLAG IS 0
; THIRD INTERVAL (FLAG IS 2)
        LD   IY,PI2
        CALL RSUB       ; SUB FROM PI/2
;
; GIVE RESULT SIGN OF ARGUMENT
;
AT40    POP  AF         ; GET SIGN
        RET  Z          ; SIGN WAS POSITIVE
        SET  7,(IX)     ; MAKE NEGATIVE
        RET
*H POLY
;
; POLYNOMIAL EVALUATOR
;
;   INPUT:  IX  PTS TO ACCUM
;           IY  PTS TO HIGHEST COEFFICIENT
;           TEMP5 CONTAINS X
;           B   CONTAINS ORDER (ORDER-1 FOR POLY1)
;   OUTPUT: IX  PTS TO RESULT
;           (POLY1 ASSUMES FIRST COEFF. IS 1)
;
POLY    PUSH BC         ; SAVE COUNTER
        PUSH IY         ; SAVE COEFF. PTR
        JP   POLY10
POLY1   PUSH BC         ; SAVE COUNTER
        PUSH IY         ; SAVE PTR
        CALL ADD
        LD   IY,TEMP5   ; PT TO X
POLY10  CALL MULT
        POP  IY
        POP  BC
        LD   DE,-8      ; ADJUST PTR
        ADD  IY,DE
        DJNZ POLY1      ; COUNTER NOT 0
        CALL ADD        ; FINAL ADD
        RET
;
; POLYNOMIAL UTILITIES
;
XX2     CALL ST04       ; STORE X IN TEMP4
        CALL MULT4      ; CALC. X^2
        CALL ST05
        RET
MULT4   LD   IY,TEMP4
        CALL MULT
        RET
*H CONSTANTS
SQR10   DEFW 1603H      ; 3.1622 77660 168
        DEFW 7722H
        DEFW 0166H
```

```
          DEFW 8168H
LNSQ10    DEFW 1501H       ; 1.1512 92546 497
          DEFW 9212H
          DEFW 6454H
          DEFW 8197H
LN10      DEFW 3002H       ; 2.3025 85092 994
          DEFW 8525H
          DEFW 2909H
          DEFW 8194H
LOGE2     DEFW 6808H       ; .86858 89638 065
          DEFW 8958H
          DEFW 8063H
          DEFW 8065H
FOVRPI    DEFW 2701H       ; 1.2732 39544 735
          DEFW 3932H
          DEFW 4754H
          DEFW 8135H
TAPI8     DEFW 1404H       ; .41421 35623 731
          DEFW 3521H
          DEFW 3762H
          DEFW 8031H
TA3PI8    DEFW 4102H       ; 2.4142 13562 373
          DEFW 1342H
          DEFW 2356H
          DEFW 8173H
PI4       DEFW 8507H       ; .78539 81633 974
          DEFW 8139H
          DEFW 3963H
          DEFW 8074H
PI2       DEFW 5701H       ; 1.5707 96326 795
          DEFW 9607H
          DEFW 6732H
          DEFW 8195H
PSQ00     DEFW 7101H       ; .17157 28752 538
          DEFW 2857H
          DEFW 2575H
          DEFW 8038H
PEX00     DEFW 1705H       ; 5178.0 91991 516
          DEFW 9180H
          DEFW 1599H
          DEFW 8416H
PEX01     DEFW 3108H       ; 831.40 67212 937
          DEFW 6740H
          DEFW 2921H
          DEFW 8337H
PEX02     DEFW 8301H       ; 18.312 36015 928
          DEFW 3612H
          DEFW 5901H
          DEFW 8228H
QEX00     DEFW 4904H       ; 4497.6 33557 406
          DEFW 3376H
          DEFW 7455H
          DEFW 8406H
QEX01     DEFW 7002H       ; 2709.3 16940 852
          DEFW 1693H
          DEFW 0894H
          DEFW 8452H
QEX02     DEFW 5901H       ; 159.37 41523 603
          DEFW 4137H
          DEFW 3652H
```

```
          DEFW 8303H
; QEX03 IS 1
PL000     DEFW 6502H      ; 265.52 24908 516
          DEFW 2452H
          DEFW 8590H
          DEFW 8316H
PL001     DEFW 2984H      ; -429.48 34828 658
          DEFW 3448H
          DEFW 8682H
          DEFW 8358H
PL002     DEFW 9701H      ; 197.64 46297 035
          DEFW 4664H
          DEFW 7029H
          DEFW 8335H
PL003     DEFW 2782H      ; -22.764 76157 115
          DEFW 7664H
          DEFW 7115H
          DEFW 8215H
QL000     DEFW 3201H      ; 132.76 12454 259
          DEFW 1276H
          DEFW 4245H
          DEFW 8359H
QL001     DEFW 5882H      ; -258.99 54899 200
          DEFW 5499H
          DEFW 9289H
          DEFW 8300H
QL002     DEFW 5801H      ; 158.60 18962 727
          DEFW 1860H
          DEFW 2796H
          DEFW 8327H
QL003     DEFW 1483H      ; -31.416 48448 282
          DEFW 4816H
          DEFW 8244H
          DEFW 8282H
; QL004 IS 1
PTA00     DEFW 0801H      ; 10888 600.43 728
          DEFW 6088H
          DEFW 3704H
          DEFW 8828H
PTA01     DEFW 9588H      ; -89588 8.4400 677
          DEFW 8488H
          DEFW 0640H
          DEFW 8677H
PTA02     DEFW 4101H      ; 14189. 85425 276
          DEFW 8589H
          DEFW 5242H
          DEFW 8576H
PTA03     DEFW 5684H      ; -45649 31943 867
          DEFW 3149H
          DEFW 3894H
          DEFW 8267H
QTA00     DEFW 3801H      ; 13863 796.66 357
          DEFW 7963H
          DEFW 6366H
          DEFW 8857H
QTA01     DEFW 9983H      ; -39913 09.518 035
          DEFW 0913H
          DEFW 8051H
          DEFW 8735H
QTA02     DEFW 3501H      ; 13538 2.7128 051
```

```
          DEFW 2738H
          DEFW 8012H
          DEFW 8651H
QTA03     DEFW 0181H        ; -1014.6 56190 253
          DEFW 5646H
          DEFW 0219H
          DEFW 8453H
; QTA04 IS 1
PSI00     DEFW 0602H        ; 20664 34.333 700
          DEFW 3464H
          DEFW 3733H
          DEFW 8700H
PSI01     DEFW 8181H        ; -18160 3.9879 741
          DEFW 3960H
          DEFW 9787H
          DEFW 8641H
PSI02     DEFW 5903H        ; 3599.9 30694 964
          DEFW 3099H
          DEFW 4969H
          DEFW 8464H
PSI03     DEFW 0182H        ; -20.107 48329 459
          DEFW 4807H
          DEFW 9432H
          DEFW 8259H
QSI00     DEFW 6302H        ; 26310 65.910 265
          DEFW 6510H
          DEFW 0291H
          DEFW 8765H
QSI01     DEFW 9203H        ; 39270. 24277 465
          DEFW 2470H
          DEFW 7427H
          DEFW 8565H
QSI02     DEFW 7802H        ; 278.11 91948 108
          DEFW 9111H
          DEFW 8194H
          DEFW 8308H
; QSI03 IS 1
PCO00     DEFW 2901H        ; 12905 39.465 904
          DEFW 3905H
          DEFW 5946H
          DEFW 8704H
PCO01     DEFW 7483H        ; -37456 7.0391 572
          DEFW 7056H
          DEFW 1539H
          DEFW 8672H
PCO02     DEFW 3401H        ; 13432. 30098 654
          DEFW 3032H
          DEFW 8609H
          DEFW 8554H
PCO03     DEFW 1281H        ; -112.31 45082 334
          DEFW 4531H
          DEFW 2308H
          DEFW 8334H
QCO00     DEFW 2901H        ; 12905 39.465 904
          DEFW 3905H
          DEFW 5946H
          DEFW 8704H
QCO01     DEFW 3402H        ; 23467. 77310 725
          DEFW 7767H
          DEFW 0731H
```

```
          DEFW 8525H
QC002     DEFW 0902H        ; 209.69 51819 673
          DEFW 5169H
          DEFW 9681H
          DEFW 8373H
; QC003 IS 1
PAT00     DEFW 1602H        ; 216.06 23078 972
          DEFW 2306H
          DEFW 8907H
          DEFW 8372H
PAT01     DEFW 2203H        ; 322.66 20700 133
          DEFW 2066H
          DEFW 0170H
          DEFW 8333H
PAT02     DEFW 3201H        ; 132.70 23981 640
          DEFW 2370H
          DEFW 1698H
          DEFW 8340H
PAT03     DEFW 2801H        ; 12.888 38303 416
          DEFW 3888H
          DEFW 3430H
          DEFW 8216H
QAT00     DEFW 1602H        ; 216.06 23078 972
          DEFW 2306H
          DEFW 8907H
          DEFW 8372H
QAT01     DEFW 9403H        ; 394.68 28393 123
          DEFW 2868H
          DEFW 3139H
          DEFW 8323H
QAT02     DEFW 2102H        ; 221.05 08830 284
          DEFW 0805H
          DEFW 0283H
          DEFW 8384H
QAT03     DEFW 8503H        ; 38.501 48650 835
          DEFW 4801H
          DEFW 0865H
          DEFW 8235H
; QAT04 IS 1
;
; REFERENCES
;
          GLOBAL            POW,IPOW,SQR,EXP,LOG,TAN,COS,SIN,ATN
          EXTERNAL          ERROR,OVER,UNDER,ZERO,ONE,C1
          EXTERNAL          ADD,SUB,RSUB,MULT,DIVI,REC
          EXTERNAL          HALF
          EXTERNAL          ST04,ST05,ST06,TEMP4,TEMP5,TEMP6
          EXTERNAL          LD,CP,FIX,FIXA,FLOAT,FLOAT1,BRK
          EXTERNAL          SRB,NORMLZ

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; MISCELLANEOUS ROUTINES REQUIRED AS EXTERNALS FOR MATH PACKAGE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
LMARK     LD   (POINT),HL ; STORE PTR FOR DTOB
          RET
LREST     RET               ; ENTRY FOR UNSUCCESSFUL DTOB (NO DIGITS FOUND)
LBACK     RET               ; DUMMY ROUTINE (BACK UP PTR FOR BASIC)
RC2       RET               ; NORMAL RETURN, IF SOME KIND OF NUMBER FOUND
```

```
LIN     PUSH HL          ; SAVE HL
        LD   HL,(POINT)  ; GET PTR
LIN10   LD   A,(HL)      ; GET NEXT CHAR
        INC  HL          ; ADVANCE PTR
        CP   ' '
        JR   Z,LIN10     ; IGNORE BLANKS
        LD   (POINT),HL  ; SAVE PTR
        POP  HL          ; RESTORE HL
        RET
SCP     RET              ; STRING COMPARE
POINT   DEFS 2           ; TEMP PTR STORAGE
ERROR   DEFB 0           ; ERROR FLAGS FOR MATH PACKAGE
ILLNUM  EQU  7           ; ILLEGAL NUMBER (DTOB)
OVRFLW  EQU  6           ; OVERFLOW
UNDFLW  EQU  5           ; UNDERFLOW
DVDZRO  EQU  4           ; DIVISION BY ZERO
SQRNGN  EQU  3           ; SQUARE ROOT OF NEGATIVE NUMBER
LOGNGN  EQU  2           ; LOG OF NEGATIVE NUMBER
;
; REFERENCES
;
        GLOBAL           ERROR,SCP,LMARK,LREST,LBACK,RC2,LIN
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; CALCULATOR PROGRAM USING BCD MATH PACKAGE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INIT    LD   (SSP),SP    ; SAVE STACK PTR
        CALL INOPN       ; OPEN TTY INPUT
        CALL OUTOPN      ; OPEN TTY OUTPUT
        CALL RANDMZ
INIT10  LD   HL,CLRSCR
        CALL OUTSTR      ; CLEAR SCREEN
        LD   HL,HEADNG
        CALL OUTSTR      ; PRINT HEADING
        CALL MENU        ; PRINT MENU
        CALL CLRALL      ; CLEAR REGISTERS
        XOR  A
        LD   (ERROR),A   ; CLEAR ERROR FLAG
        CALL DPYALL      ; DISPLAY REGISTERS
        CALL DPYMEM      ; DISPLAY MEMORY
;
; MAIN LOOP
;
LOOP    LD   HL,CLRLIN
        CALL OUTSTR      ; CLEAR INPUT LINE
        LD   A,(ERROR)
        AND  A           ; CHECK FOR ERRORS
        LD   HL,ERRMSG
        CALL NZ,OUTSTR   ; PRINT ERROR MESSAGE
        CALL INCHR       ; GET A CHAR
        LD   HL,ERROR
        LD   (HL),0      ; CLEAR ERRORS
        CP   '.'
        JR   Z,LOOP05    ; ENTER NUMBER
        CP   '0'
        JR   C,LOOP10    ; NOT A DIGIT
        CP   '9'+1
        JR   NC,LOOP10   ; NOT A DIGIT
LOOP05  CALL NUM         ; A DIGIT
LOOP10  CP   'K'
        JP   Z,INIT10    ; RESTART
        LD   HL,LOOZR    ; SET UP FOR ZERO ARG OPS
        LD   IX,REGT
        PUSH HL          ; SET UP RET ADDR
        CP   '#'
        JP   Z,RND
        CP   'Z'
        JP   Z,ZERO
        CP   'P'
        JP   Z,PI
        CP   '='
        JP   Z,DUP       ; DUPLICATE
        CP   'R'
        JP   Z,MEMR
        LD   IX,REG4     ; SET UP FOR UNARY OPS
        LD   HL,LOOUN
        EX   (SP),HL     ; SET NEW RET ADDR
        CP   'Q'
        JP   Z,SQR       ; SQUARE ROOT
        CP   'E'
        JP   Z,EXP
```

```
          CP   'L'
          JP   Z,LOG
          CP   'T'
          JP   Z,TAN
          CP   'C'
          JP   Z,COS
          CP   'S'
          JP   Z,SIN
          CP   'A'
          JP   Z,ATN
          CP   'V'
          JP   Z,REC
          CP   'F'
          JP   Z,BRK
          CP   'I'
          JP   Z,INT
          CP   'B'
          JP   Z,ABS
          CP   'N'
          JP   Z,NEG
          CP   'X'
          JP   Z,XCH
          CP   'W'
          JP   Z,MEMW
          LD   IX,REG3     ; SET UP FOR BINARY OPS
          LD   IY,REG4
          LD   HL,LOOBIN
          EX   (SP),HL     ; NEW RET ADDR
          CP   '+'
          JP   Z,ADD       ; ADDITION
          CP   '-'
          JP   Z,SUB
          CP   '*'
          JP   Z,MULT
          CP   '/'
          JP   Z,DIVI
          CP   '^'
          JP   Z,POW
          CP   'D'
          RET  Z           ; POP COMMAND
          LD   HL,LOOP     ; SET UP FOR OTHERS
          EX   (SP),HL     ; NEW RET ADDR
          CP   0DH         ; CR
          RET  Z           ; CR IS ENTER
          LD   A,0FFH
          LD   (ERROR),A   ; FLAG COMMAND ERROR
          RET              ; ALL UNDEFINED CHARS
LOOZR     CALL PUSH
          JP   LOOP
LOOBIN    LD   HL,REG4-1   ; SRC
          LD   DE,REGT-1
          LD   BC,NUMSIZ*3
          LDDR             ; COPY REG1,2,3 INTO 2,3,4
          CALL DPYALL      ; REDISPLAY REGS
          JP   LOOP
;
LOOUN     LD   HL,DREG4
          LD   IX,REG4
          CALL DPYR        ; REDISPLAY REG4
          CALL DPYROU      ; DISPLAY ROUNDED RESULT
```

```
        JP   LOOP
*H SPECIAL OPERATIONS
;
; INPUT A NUMBER AND PUSH IT
;   DIGITS, . , E, N, ARE ACCEPTED
;   D ABORTS ENTER MODE
;   A DIGIT HAS ALREADY BEEN TYPED
;
;
; INITIALIZE BUFFER, FLAG, COUNT, PTR
;
NUM     LD   (INBUF+1),A; SAVE FIRST DIGIT
        LD   A,'+'
        LD   (INBUF),A  ; INIT SIGN
        LD   HL,INBUF+2
        LD   (INBPTR),HL; INIT PTR
        LD   B,INBLEN-2
NUM10   LD   (HL),' '   ; INITIALIZE BUFFER
        INC  HL
        DJNZ NUM10
        LD   A,0
        LD   (NFLG),A   ; CLEAR EXP FLAG
        LD   A,ENTLEN+2
        LD   (ENTERL),A ; INIT COUNT
;
; CONVERT AND DISPLAY INBUF--REGT
; SEND ERROR MESSAGES
;
NUM20   LD   HL,ENTERL
        CALL OUTSTR     ; DISPLAY ENTER LINE
        CALL INCHR      ; GET A CHAR
        LD   B,A        ; SAVE CHAR
        LD   A,(NFLG)
        AND  A          ; TEST FLAG
        LD   A,B        ; RESTORE CHAR
        JR   NZ,NUM35   ; IN EXPONENT PART
;
; PROCESS MANTISSA
;
        CP   'D'
        JR   C,NUM25    ; NOT A DIGIT
        CP   '9'+1
        JR   C,NUM60    ; A DIGIT
NUM25   CP   '.'
        JR   Z,NUM60    ; A DECIMAL PT
        CP   'E'
        JR   Z,NUM65    ; SET UP FOR EXPONENT
        CP   'N'
        JR   Z,NUM55    ; NEGATE
;
; END OF NUMBER INPUT, PUSH RESULT
;
NUM30   CP   'D'
        JR   NZ,NUM32   ; NOT DELETE
        POP  HL         ; KILL RET ADDR
        JP   LOOP       ; IGNORE NUMBER
NUM32   PUSH AF         ; SAVE THIS CHAR
        LD   HL,INBUF
        LD   IX,REGT
        CALL DTOB       ; CONVERT NUMBER IN REGT
```

```
        CALL PUSH       ; PUSH REGT ONTO STACK
        LD   HL,CLRLIN
        CALL OUTSTR     ; CLEAR NUMBER DISPLAY
        POP  AF
        RET
;
; PROCESS EXPONENT PART
;
NUM35   CP   '0'
        JR   C,NUM37    ; NOT A DIGIT
        CP   '9'+1
        JR   C,NUM60    ; A DIGIT
NUM37   CP   'N'
        JR   NZ,NUM30   ; NOT NEGATE
;
; NEGATE EXPONENT
;
NUM40   LD   HL,(ESGPTR); GET PTR TO SIGN
        LD   A,(HL)     ; GET SIGN
        CP   '+'
        JR   Z,NUM50    ; POSITIVE
        LD   (HL),'+'
        JP   NUM20
NUM50   LD   (HL),'-'
        JP   NUM20
;
; NEGATE MANTISSA
;
NUM55   LD   A,(INBUF)  ; GET SIGN
        CP   '+'
        JR   Z,NUM57    ; POSITIVE
        LD   A,'+'
        LD   (INBUF),A
        JP   NUM20
NUM57   LD   A,'-'
        LD   (INBUF),A
        JP   NUM20
;
; STORE CHAR IN INBUF
;
NUM60   CALL NUM70      ; STORE CHAR
        JP   NUM20
;
; SET EFLAG, ADJUST COUNT, PTR
;
NUM65   CALL NUM70      ; STORE E IN INBUF
        LD   A,0FFH
        LD   (NFLG),A   ; TURN ON FLAG
        LD   (ESGPTR),HL; SAVE PTR TO SIGN
        LD   A,'+'
        CALL NUM70      ; DEFAULT EXP SIGN
        JP   NUM20
;
; ROUTINE TO STORE INTO INBUF
;   RETURNS PTR VALUE IN HL
;
NUM70   LD   B,A        ; SAVE CHAR
        LD   HL,(INBPTR); GET PTR
        LD   A,(ENTERL) ; GET COUNT
        CP   ENTLEN+INBLEN
```

```
        RET  Z          ; COUNT MAX, IGNORE
        INC  A          ; INC IT
        LD   (ENTERL),A
        LD   (HL),B     ; STORE CHAR
        INC  HL
        LD   (INBPTR),HL
        RET
;
; DUP  DUPLICATE TOP OF STACK
;
DUP     LD   HL,REG4
        LD   DE,REGT
        LD   BC,NUMSIZ
        LDIR            ; COPY REG4 TO REGT
        RET
;
; EXCHANGE REGISTERS
;
XCH     LD   DE,REGT+7  ; DEST
        LD   HL,REGT-1  ; SRC
        LD   BC,NUMSIZ*2
        LDDR            ; COPY REG3,4 TO REG4,T
        LD   HL,REGT
        LD   DE,REG3
        LD   BC,NUMSIZ
        LDIR            ; COPY REGT TO REG3
        LD   HL,DREG3
        LD   IX,REG3
        CALL DPYR       ; DISPLAY NEW REG3
        RET
;
; EXIT TO RIO
;
RIO     LD   SP,(SSP)   ; RESTORE SP
        RET             ; RETURN TO RIO
;
; PI ROUTINE RETURNS PI
;
PI      PUSH IX         ; COPY IX TO DE
        POP  DE
        LD   HL,CPI
        LD   BC,NUMSIZ
        LDIR
        RET
;
; MEMORY READ
;
MEMR    CALL GETDIG     ; GET MEM ADDR
        JR   C,MEMR99   ; ERROR
        PUSH IX         ; COPY REG PTR TO HL
        POP  HL
        LD   DE,REGT    ; PT TO TEMP REG
        LD   BC,NUMSIZ
        LDIR            ; COPY VALUE
        RET
MEMR99  POP  HL         ; THROW AWAY RET ADDR
        JP   LOOP
;
; MEMORY WRITE
;
```

```
MEMW    CALL GETDIG     ; GET MEM ADDR
        RET  C          ; ERROR
        PUSH HL         ; SAVE DISPLAY PTR
        PUSH IX         ; COPY REG PTR TO DE
        POP  DE
        LD   HL,REG4    ; SRC IS REG4
        LD   BC,NUMSIZ
        LDIR            ; COPY REG4 TO MEM
        POP  HL         ; GET DPY PTR
        CALL DPYR       ; REDISPLAY MEM REG
        RET
*H SPECIAL FORMATTING
DPYROU  LD   HL,REG4    ; SRC
        LD   DE,REGT
        LD   BC,NUMSIZ
        LDIR            ; COPY REG4 TO REGT
        LD   IX,REGT
        LD   IY,ROUCON  ; ROUNDING CONSTANT
        LD   A,(IX+NUMSIZ-1)
        LD   (IY+NUMSIZ-1),A
        LD   A,(IX)     ; GET SIGN
        AND  80H
        LD   (IY),A
        CALL ADD        ; ROUND TO 10 PLACES
;
; CLEAR DISPLAY AREA
;
        LD   HL,DROU+ADRSIZ+1
        LD   B,18
DPYR05  LD   (HL),' '
        INC  HL
        DJNZ DPYR05
        LD   HL,DREGT
        PUSH HL         ; SAVE HL
        CALL BTOD       ; CONVERT NUMBER
        EX   AF,AF'     ; SAVE EXPONENT IN A'
;
; SET UP HL PTING TO INPUT, DE TO OUTPUT STRINGS
;
        POP  HL         ; GET PTR TO ASCII STRING
        LD   DE,DROU+ADRSIZ+1; OUTPUT PTR
;
; CHECK FOR ZERO INPUT
;
        XOR  A          ; CHECK FOR ZERO
        CP   (IX+7)
        JR   Z,STR90    ; INPUT IS ZERO
;
; CALCULATE # SIGNIFICANT DIGITS
;
        PUSH HL         ; SAVE INPUT PTR
        LD   BC,10      ; MAX COUNT
        ADD  HL,BC      ; PT TO LAST DIGIT
        LD   A,'0'      ; ASCII ZERO
STR10   CP   (HL)       ; COMPARE DIGIT
        JR   NZ,STR20   ; DIGIT IS SIGNIFICANT
        DEC  HL         ; DECR PTR
        DEC  C          ; DECR COUNTER
        JR   NZ,STR10   ; COUNTER NOT ZERO
        POP  HL         ; RESTORE PTR
```

```
        JR   STR90      ; THIS JR SHOULDN'T HAPPEN
STR20   POP  HL         ; RESTORE PTR
;
; SET SIGN
;
        LD   A,(HL)
        LD   (DE),A
        INC  HL
        INC  DE
;
; CHECK IF X >= 1
;
        EX   AF,AF'     ; GET EXPONENT
        LD   B,A        ; SAVE IN B
        AND  A
        JR   Z,STR40    ; EXPONENT IS ZERO
        JP   M,STR60    ; EXPONENT IS < 0
        CP   11
        JP   NC,STR80   ; TOO BIG
;
; PRINT DIGITS TO LEFT OF DECIMAL PT
;
STR30   LD   A,(HL)     ; GET CHAR
        LD   (DE),A     ; STORE IN STRING
        INC  HL         ; ADVANCE PTRS
        INC  DE
        DEC  C          ; DECR # SIGNIF DIGITS
        DJNZ STR30      ; EXP COUNTER NOT ZERO
;
; CHECK IF ANY SIGNIFICANT DIGITS LEFT
;
        LD   A,C        ; GET # DIGITS
        AND  A
        JP   M,STR100   ; NONE LEFT
        JR   Z,STR100   ; NONE LEFT
;
; PRINT DECIMAL PT
;
STR40   LD   A,'.'
        LD   (DE),A
        INC  DE
;
; PRINT SIGNIFICANT DIGITS TO RIGHT OF PT
;
STR50   LDIR            ; (B IS ZERO, C IS COUNTER)
        JR   STR100
;
; CHECK IF NUMBER WILL FIT IN 10 PLACE FRACTION
;
STR60   LD   A,C        ; GET # SIG DIGITS
        SUB  B          ; ADD # LEADING ZEROS
        CP   11
        JR   NC,STR80   ; WON'T FIT, USE E NOTATION
;
; OUTPUT DECIMAL PT AND LEADING ZEROS
;
        LD   A,'.'
        LD   (DE),A
        INC  DE
        LD   A,'0'      ; ASCII 0
```

```
STR70   LD   (DE),A      ; STORE A 0
        INC  DE          ; ADV PTR
        INC  B           ; INC COUNTER
        JR   NZ,STR70    ; COUNTER NOT 0
        JR   STR50       ; GO PRINT SIG DIGITS
;
; USE E NOTATION FOR NUMBER
;
STR80   LD   BC,10       ; #DIGITS
        LDI              ; TRANSFER LEADING DIGIT
        LD   A,'.'
        LD   (DE),A      ; DECIMAL PT
        INC  DE
        LDIR             ; REST OF DIGITS
        INC  HL          ; SKIP 3 DIGITS
        INC  HL
        INC  HL
        INC  DE          ; SKIP A SPACE
        LD   BC,5        ; CHARS LEFT IN EXP
        LDIR
        JR   STR100
;
; OUTPUT A ZERO
;
STR90   LD   A,'0'
        INC  DE
        LD   (DE),A
STR100  LD   HL,DROU
        CALL OUTSTR      ; DISPLAY IT
        RET
*H UTILITIES
;
; DISPLAY A REGISTER
;   HL PTS TO DISPLAY BUFFER
;   IX PTS TO REGISTER VALUE
;
DPYR    PUSH HL          ; SAVE OUTPUT PTR
        PUSH IX          ; COPY IX TO HL
        POP  HL
        LD   DE,REGT     ; DEST IS TEMP REG
        LD   BC,NUMSIZ
        LDIR             ; COPY REGISTER TO TEMP
        LD   IX,REGT     ; PT IX TO IT
        LD   HL,DREGT    ; PT TO TEMP DPY REG
        CALL BTOD        ; CONVERT TO ASCII
        POP  HL          ; GET PTR
        PUSH HL
        LD   BC,ADRSIZ+3
        ADD  HL,BC       ; PT TO SIGN BYTE
        EX   DE,HL
        LD   HL,DREGT    ; PT TO BTOD OUTPUT
        LD   BC,NUMDIG+1
        LDI              ; COPY SIGN
        LDI              ; COPY FIRST DIGIT
        LD   A,'.'
        LD   (DE),A      ; DECIMAL PT
        INC  DE
        LDIR             ; COPY REST OF DIGITS
        LD   A,' '
        LD   (DE),A      ; BLANK
```

```
        INC   DE
        INC   HL
        LD    BC,NUMEXP
        LDIR              ; EXPONENT
        POP   HL
        CALL  OUTSTR      ; PRINT RESULT
        RET
;
; PUSH, DPYALL
;   VALUE IN TEMP REG IS PUSHED ONTO STACK
;   ALL REGISTERS ARE REDISPLAYED
;
PUSH    LD    HL,REG2     ; SOURCE
        LD    DE,REG1     ; DEST
        LD    BC,NUMSIZ*4
        LDIR              ; MOVE ALL VALUES UP 1
DPYALL  LD    HL,DREG1
        LD    IX,REG1
        CALL  DPYR
        LD    HL,DREG2
        LD    IX,REG2
        CALL  DPYR
        LD    HL,DREG3
        LD    IX,REG3
        CALL  DPYR
        LD    HL,DREG4
        LD    IX,REG4
        CALL  DPYR
        CALL  DPYROU      ; DISPLAY ROUNDED RESULT
        RET
;
; CLRALL   CLEAR ALL REGISTERS
;
CLRALL  LD    HL,REG1
        LD    B,NUMSIZ*15
CLRA10  LD    (HL),0
        INC   HL
        DJNZ  CLRA10
        RET
;
; DISPLAY MEMORIES
;
DPYMEM  LD    HL,DMEM0    ; FIRST MEM REG
        LD    IX,MEM0
        LD    B,10        ; COUNT
DPYM10  PUSH  BC          ; SAVE COUNT
        PUSH  HL          ; SAVE REG PTR
        PUSH  IX          ; SAVE DPY PTR
        CALL  DPYR
        POP   IX
        LD    BC,NUMSIZ   ; DISPLAY SIZE
        ADD   IX,BC       ; PT TO NEXT DISPLAY
        POP   HL
        LD    BC,DSIZE+1
        ADD   HL,BC       ; PT TO NEXT REG
        POP   BC
        DJNZ  DPYM10      ; COUNT NOT 0
        RET
;
; DISPLAY MENU
```

```
;
MENU    LD   HL,MENU1
        CALL OUTSTR
        LD   HL,MENU2
        CALL OUTSTR
        LD   HL,MENU3
        CALL OUTSTR
        LD   HL,MENU4
        CALL OUTSTR
        LD   HL,MENU5
        CALL OUTSTR
        LD   HL,MENU6
        CALL OUTSTR
        RET
;
; GETDIG  GETS ADDRESSES FOR MEMORY OPERATIONS
;    INPUTS A CHAR, MULTIPLIES BY NUMSIZ AND DSIZE,
;    ADDS TO REG AND DPY PTRS
;    RETURNS CY IF NOT SUCCESSFUL
;
GETDIG  CALL INCHR      ; GET A CHAR
        CP   '0'
        JR   C,GETD99   ; NOT A DIGIT
        CP   '9'+1
        JR   NC,GETD99  ; NOT A DIGIT
        AND  0FH        ; CONVERT TO BINARY
        LD   C,A        ; SAVE IN C
GETD10  XOR  A          ; CLEAR A
        LD   B,NUMSIZ
GETD20  ADD  A,C        ; MULTIPLY BY NUMSIZ
        DJNZ GETD20
        LD   IX,MEMO    ; PT TO FIRST REG
        LD   E,A        ; PUT RESULT IN DE
        LD   D,0
        ADD  IX,DE      ; PT TO ADDRESSED MEM
        EX   DE,HL      ; PUT DISPL IN HL
        LD   E,C        ; PUT ORIGINAL VAL IN DE
        LD   D,0
        LD   B,DSIZE+1-NUMSIZ
GETD30  ADD  HL,DE      ; MULTIPLY BY DSIZ
        DJNZ GETD30
        LD   DE,DMEMO   ; PT TO FIRST DPY
        ADD  HL,DE      ; PT TO ADDRESSED MEM DPY
        AND  A          ; CY OFF
        RET
GETD99  CP   'D'
        SCF             ; TURN ON CY FLAG
        RET  Z          ; DELETE COMMAND
        LD   A,0FFH
        LD   (ERROR),A  ; FLAG ERROR
        RET
*H I/O UTILITIES
;
; INPUT FILE OPEN
;
INOPN   LD   IY,INVECT  ; PT TO VECTOR
        LD   (IY+1),4   ; OPEN
        LD   (IY+4),0   ; LEN
        CALL SYSTEM
        RET
```

```
OUTOPN  LD    IY,OTVECT   ; PT TO VECTOR
        LD    (IY+1),4    ; OPEN
        LD    (IY+4),0    ; LEN
        CALL  SYSTEM
        RET
;
; INCHR   INPUT 1 CHAR
;   RETURNED IN A
;
INCHR   LD    IY,INVECT
        LD    (IY+1),0CH  ; READ LINE
        LD    (IY+4),1    ; DATA LENGTH
        LD    HL,INCBUF
        LD    (IY+2),L    ; DATA ADDR
        LD    (IY+3),H
        CALL  SYSTEM
        LD    (IY+1),40H  ; READ STATUS
        LD    (IY+4),1    ; DATA LENGTH
        LD    HL,CSTAT    ; GET CSAT ADDR
        LD    (IY+2),L
        LD    (IY+3),H
        CALL  SYSTEM
        LD    A,(CSTAT)   ; GET FLAGS
        BIT   5,A
        JP    Z,RIO       ; ESC PENDING
        LD    A,(INCBUF)  ; GET CHAR
        BIT   6,A
        RET   Z           ; NOT UPPER CASE
        RES   5,A         ; MAKE UPPER CASE
        RET
;
; OUTSTR  OUTPUT A STRING
;   HL PTS TO COUNT FOLLOWED BY STRING
;
OUTSTR  LD    A,(HL)      ; GET COUNT
        LD    IY,OTVECT
        LD    (IY+1),10H  ; WRITE ASCII
        LD    (IY+4),A
        INC   HL
        LD    (IY+2),L
        LD    (IY+3),H
        CALL  SYSTEM
        RET
;
; VECTORS
;
INVECT  DEFB  1
        DEFB  4
        DEFW  0
        DEFW  0
        DEFW  0
        DEFW  0
        DEFB  0
        DEFW  0
OTVECT  DEFB  2
        DEFB  4
        DEFW  0
        DEFW  0
        DEFW  0
        DEFW  0
```

```
        DEFB 0
        DEFW 0
*H CONSTANTS, WORK AREAS
;
; EQUATES
;
ASCESC  EQU  1BH         ; ESCAPE CHAR
SYSTEM  EQU  1403H
NUMSIZ  EQU  8           ; SIZE OF DATA
NUMDIG  EQU  13          ; NUMBER OF SIGNIF. DIGITS
NUMEXP  EQU  4           ; NUMBER OF EXPON DIGITS + 1
;
; CONSTANTS
;
CPI     DEFW 1403H       ; 3.141592653590
        DEFW 9215H
        DEFW 3565H
        DEFW 8190H
ROUCON  DEFW 0
        DEFW 0
        DEFW 500H
        DEFW 0
;
; MESSAGES, STRINGS
;
HEADNG  DEFB 27+ADRSIZ
        ADDR 1,20
        DEFM 'Z & O   C A L C U L A T O R'
;
; MENU LINES
;
MENU1   DEFB 60+ADRSIZ
        ADDR 4,8
        DEFM '+ Y+X       '
        DEFM 'Q SQR(X)    '
        DEFM 'T TAN(X)    '
        DEFM 'V 1/X       '
        DEFM 'X EXCH X,Y  '
MENU2   DEFB 60+ADRSIZ
        ADDR 5,8
        DEFM '- Y-X       '
        DEFM 'E EXP(X)    '
        DEFM 'S SIN(X)    '
        DEFM 'N CHG SGN   '
        DEFM '= DUP X     '
MENU3   DEFB 60+ADRSIZ
        ADDR 6,8
        DEFM '* Y*X       '
        DEFM 'L LOG(X)    '
        DEFM 'C COS(X)    '
        DEFM 'B ABS(X)    '
        DEFM 'D DEL/POP   '
MENU4   DEFB 60+ADRSIZ
        ADDR 7,8
        DEFM '/ Y/X       '
        DEFM '# RANDOM    '
        DEFM 'A ARCTAN(X) '
        DEFM 'F FRACTION  '
        DEFM 'R<N> READ   '
MENU5   DEFB 60+ADRSIZ
```

```
          ADDR 8,8
          DEFM '↑ Y↑X       '
          DEFM 'Z ZERO      '
          DEFM 'P PI        '
          DEFM 'I INTEGER   '
          DEFM 'W<N> WRITE  '
MENU6     DEFB 60+ADRSIZ
          ADDR 9,8
          DEFM '            '
          DEFM '            '
          DEFM '            '
          DEFM '<ESC> EXIT  '
          DEFM 'K CLEAR ALL '
ERRMSG    DEFB 5+ADRSIZ
          ADDR 22,1
          DEFM 'ERROR'
NOP       DEFB 1
          DEFB ' '
CLRLIN    DEFB 39+2*ADRSIZ
          ADDR 22,1         ; ROW 20, COL 1
          DEFM '                    '
          DEFM '                   '
          ADDR 22,1         ; ROW 20, COL 1
PRMPTA    DEFB 2+ADRSIZ
          ADDR 22,1         ; ROW 20, COL 1
          DEFB '>'
FRSTDG    DEFS 1
PROMPT    DEFB 1
          DEFB '>'
CLRSCR    DEFB CLRSIZ
          CLEAR
;
; DISPLAY BUFFERS
;
DSIZE     EQU  22+ADRSIZ
DREG1     DEFB DSIZE
          ADDR 13,8
          DEFM '  '
          DEFS 20
DREG2     DEFB DSIZE
          ADDR 14,8
          DEFM '  '
          DEFS 20
DREG3     DEFB DSIZE
          ADDR 15,8
          DEFM 'Y '
          DEFS 20
DREG4     DEFB DSIZE
          ADDR 16,8
          DEFM 'X '
          DEFS 20
DREGT     DEFS NUMDIG+NUMEXP+2
DMEM0     DEFB DSIZE
          ADDR 13,40
          DEFM '0 '
          DEFS 20
DMEM1     DEFB DSIZE
          ADDR 14,40
          DEFM '1 '
          DEFS 20
```

```
DMEM2   DEFB DSIZE
        ADDR 15,40
        DEFM '2 '
        DEFS 20
DMEM3   DEFB DSIZE
        ADDR 16,40
        DEFM '3 '
        DEFS 20
DMEM4   DEFB DSIZE
        ADDR 17,40
        DEFM '4 '
        DEFS 20
DMEM5   DEFB DSIZE
        ADDR 18,40
        DEFM '5 '
        DEFS 20
DMEM6   DEFB DSIZE
        ADDR 19,40
        DEFM '6 '
        DEFS 20
DMEM7   DEFB DSIZE
        ADDR 20,40
        DEFM '7 '
        DEFS 20
DMEM8   DEFB DSIZE
        ADDR 21,40
        DEFM '8 '
        DEFS 20
DMEM9   DEFB DSIZE
        ADDR 22,40
        DEFM '9 '
        DEFS 20
DROU    DEFB 18+ADRSIZ
        ADDR 19,10
        DEFS 18
;
; REGISTERS
;
REG1    DEFS NUMSIZ
REG2    DEFS NUMSIZ
REG3    DEFS NUMSIZ
REG4    DEFS NUMSIZ
REGT    DEFS NUMSIZ
MEM0    DEFS NUMSIZ
MEM1    DEFS NUMSIZ
MEM2    DEFS NUMSIZ
MEM3    DEFS NUMSIZ
MEM4    DEFS NUMSIZ
MEM5    DEFS NUMSIZ
MEM6    DEFS NUMSIZ
MEM7    DEFS NUMSIZ
MEM8    DEFS NUMSIZ
MEM9    DEFS NUMSIZ
;
; BUFFERS
;
ENTLEN  EQU  7+ADRSIZ   ; SIZE OF ENTER MSG
INBLEN  EQU  30         ; SIZE OF INBUF
ENTERL  DEFS 1          ; MSG LENGTH
        ADDR 22,1       ; ENTER MESSAGE
```

```
        DEFM 'ENTER: '
INBUF   DEFS INBLEN
        DEFS 1          ; EXP SIGN MAY GO HERE
NFLG    DEFS 1          ; NUMBER FLAG
INBPTR  DEFS 2          ; PTR IN INBUF
ESGPTR  DEFS 2          ; PTR TO EXP SIGN
INCBUF  DEFS 1          ; INPUT CHAR
CSTAT   DEFS 1          ; CONSOLE STATUS
SSP     DEFS 2          ; SAVED STACK PTR
;
; REFERENCES
;
        EXTERNAL        DTOB,BTOD
        EXTERNAL        ADD,SUB,MULT,DIVI,POW
        EXTERNAL        INT,ABS,SGN,BRK,RND
        EXTERNAL        REC,NEG,RANDMZ,ZERO
        EXTERNAL        SQR,EXP,LOG,TAN,COS,SIN,ATN
        EXTERNAL        ERROR
ASM #1 CAL (M NOL O=CAL.OBJ)
LINK $=4400 CAL DMATHM DMATH0 DMATH1 DMATH2 (NOM)
```

```
*H MULTIPLICATION
;
; 8 BIT BY 8 BIT UNSIGNED MULTIPLY  306 CYCLES AVG.
;
;   MULTIPLICAND IN E
;   MULTIPLIER IN   C
;   PRODUCT   IN   AE
;
MP88    XOR  A          ; CLEAR ACCUM
        LD   B,8        ; SET COUNTER
        RR   E          ; 1ST MPLIER BIT IN CY
MP88A   JR   NC,$+3     ; BIT WAS A 0
        ADD  A,C        ; ADD MULTIPLICAND
        RRA             ; ROTATE ACC-MULTIPLIER INTO CY
        RR   E
        DJNZ MP88A      ; COUNTER NOT 0
        RET
*EJECT
;
; 16 BIT BY 16 BIT UNSIGNED MULTIPLY  930 CYCLES AVG.
;
;   MULTIPLICAND IN DE
;   MULTIPLIER  IN  AC
;   PRODUCT  IN   HLAC
;
MP16    LD   HL,0       ; CLEAR ACCUM
        LD   B,16       ; SET COUNTER
        RRA             ; 1ST MPLIER BIT IN CY
        RR   C
MP161   JR   NC,MP162   ; MPLIER BIT IS 0
        ADD  HL,DE      ; ADD MULTIPLICAND
MP162   RR   H          ; ROT ACC-MPLIER INTO CY
        RR   L
        RRA
        RR   C
        DJNZ MP161      ; COUNTER NOT 0
        RET
*EJECT
;
; 24 BIT BY 24 BIT UNSIGNED MULTIPLY   2022 CYCLES AVG.
;
;   MULTIPLICAND IN CDE
;   MULTIPLIER IN H'L'D'
;   PRODUCT IN AHLH'L'D'
;
MP24    XOR  A          ; CLEAR ACCUM
        LD   H,A
        LD   L,A
        LD   B,24       ; SET COUNTER
        EXX             ; ALT BANK
        RR   H          ; ROT 1ST MPLIER BIT INTO CY
        RR   L
        RR   D
        EXX             ; MAIN BANK
MP241   JR   NC,MP242   ; MPLIER BIT WAS 0
        ADD  HL,DE      ; ADD MPLICAND
        ADC  A,C
MP242   RRA             ; ROT ACC-MPLIER INTO CY
        RR   H
        RR   L
```

```
        EXX             ; ALT BANK
        RR   H
        RR   L
        RR   D
        EXX             ; MAIN BANK
        DJNZ MP241      ; COUNTER NOT 0
        RET
*EJECT
;
; 32 BIT BY 32 BIT UNSIGNED MULTIPLY   3619 CYCLES AVG.
;
;   MULTIPLICAND IN D'E'DE
;   MULTIPLIER  IN  B'C'CA
;   PRODUCT IN H'L'HLB'C'CA
;
MP32    LD   HL,0       ; CLEAR ACCUM
        EXX             ; ALT BANK
        LD   HL,0
        EXX             ; MAIN BANK
        LD   B,33       ; SET COUNTER
        JP   MP323
MP321   JR   NC,MP322   ; MPLIER BIT IS 0
        ADD  HL,DE      ; ADD MULTIPLICAND
        EXX             ; ALT BANK
        ADC  HL,DE
        EXX             ; MAIN BANK
MP322   EXX             ; ALT BANK
        RR   H          ; ROTATE ACCUM INTO CY
        RR   L
        EXX             ; MAIN BANK
        RR   H
        RR   L
MP323   EXX             ; ALT BANK
        RR   B          ; ROTATE MPLIER INTO CY
        RR   C
        EXX             ; MAIN BANK
        RR   C
        RRA
        DJNZ MP321      ; COUNTER NOT 0
        RET
*H DIVISION
;
; 8 BIT BY 8 BIT UNSIGNED DIVIDE   346 CYCLES AVG.
; USES RESTORING DIVISION, PRODUCING COMPLEMENTED QUOTIENT
;   E IS DIVIDEND
;   C IS DIVISOR
;   A IS QUOTIENT
;   B IS REMAINDER
;
DV88    XOR  A          ; CLEAR ACCUM
        LD   B,8        ; LOOP COUNTER
DV8A    RL   E          ; ROTATE CY INTO ACC-DIVIDEND
        RLA             ; CY WILL BE OFF
        SUB  C          ; TRIAL SUB DIVISOR
        JR   NC,$+3     ; SUBTRACT OK
        ADD  A,C        ; RESTORE ACCUM, SET CY
        DJNZ DV8A
        LD   B,A        ; PUT REMAINDER IN B
        LD   A,E        ; GET QUOTIENT
        RLA             ; SHIFT IN LAST RESULT BIT
```

```
        CPL             ; COMPLEMENT BITS
        RET
*EJECT
;
; 16 BIT BY 8 BIT UNSIGNED DIVIDE  473 CYCLES AVG.
;   RESTORING DIVISION, FAKING A 9 BIT ACCUM
;
;   DIVIDEND IN AE
;   DIVISOR IN  C
;   QUOTIENT IN E
;   REMAINDER IN A
;   CY ON FOR OVERFLOW   (A >= C ON INPUT)
;
DV168   CP   C          ; CHECK FOR OVERFLOW
        CCF
        RET  C          ; A >= C
        LD   B,8        ; LOOP COUNTER
        JP   DV163
DV161   SUB  C          ; SUB DIVISOR
DV162   SCF             ; FORCE RESULT TO 1
        DEC  B
        JR   Z,DV165    ; COUNTER IS 0
DV163   RL   E          ; ROTATE CY INTO ACC-DIVIDEND
        RLA
        JR   C,DV161    ; NEED 9 BIT ACCUM
        SUB  C          ; TRY SUB DIVISOR
        JR   NC,DV162   ; SUB OK, OUTPUT A 1
        ADD  A,C        ; RESTORE ACCUM
        AND  A          ; MAKE CY 0
        DJNZ DV163      ; COUNTER NOT 0
DV165   RL   E          ; GET LAST RESULT BIT, 0 CY
        RET
*EJECT
;
; 16 BIT BY 15 BIT UNSIGNED DIVIDE   1164 CYCLES AVG.
;   NON-RESTORING DIVISION ALGORITHM
;
;   DIVIDEND IN AC
;   DIVISOR IN  DE  (15 BITS)
;   QUOTIENT IN AC
;   REMAINDER IN HL
;
DV15    LD   HL,0       ; CLEAR ACCUM
        LD   B,16       ; SET COUNTER
DV151                   ; 0 <= HL < DE
        RL   C          ; ROTATE ACC-RESULT LEFT
        RLA
        ADC  HL,HL      ; NO CY POSSIBLE
        SBC  HL,DE      ; SUBTRACT DIVISOR
DV152   CCF             ; CALC RESULT BIT
        JR   NC,DV155   ; ACCUM WENT NEGATIVE
DV153   DJNZ DV151      ; COUNTER NOT 0
        RL   C          ; SHIFT IN LAST RESULT BIT
        RLA
        RET
DV154                   ; -DE <= HL < 0
        RL   C          ; ROTATE ACC-RESULT LEFT
        RLA
        ADC  HL,HL
        AND  A          ; TURN OFF CY
```