```
000316' 001000  000257          07000           XRA     A               ;ZERO A
000317' 001000  000157          07020           MOV     L,A             ;SAVE ZERO IN L
000320' 001000  000220          07040           SUB     B               ;NEGATE LOWEST ORDER
000321' 001000  000107          07060           MOV     B,A             ;SAVE IT
000322' 001000  000175          07080           MOV     A,L             ;GET A ZERO
000323' 001000  000233          07100           SBB     E               ;NEGATE NEXT HIGHEST ORDER
000324' 001000  000137          07120           MOV     E,A             ;SAVE IT
000325' 001000  000175          07140           MOV     A,L             ;GET A ZERO
000326' 001000  000232          07160           SBB     D               ;NEGATE NEXT HIGHEST ORDER
000327' 001000  000127          07180           MOV     D,A             ;SAVE IT
000330' 001000  000175          07200           MOV     A,L             ;GET ZERO BACK
000331' 001000  000231          07220           SBB     C               ;NEGATE HIGHEST ORDER
000332' 001000  000117          07240           MOV     C,A             ;SAVE IT
000333' 001000  000311          07260           RET                     ;ALL DONE

                                07320           ;SHIFT C,D,E RIGHT
                                07340           ;A = SHIFT COUNT
                                07360           ;ALTERS A,B,C,D,E,L
                                07380           ;THE IDEA (EXCEPT IN 4K) IS TO SHIFT RIGHT 8 PLACES AS MANY TIMES AS
                                07400           ; POSSIBLE
000334' 001000  000006          07420   SHIFTR: MVI     B,0             ;ZERO OVERFLOW BYTE
000335' 000000  000000          07440   IFE     LENGTH,<
                                07460           INR     A>              ;ADD ONE TO SHIFT COUNT
                                07480   IFN     LENGTH,<
000336' 001000  000326          07500   SHFTR1: SUI     10              ;CAN WE SHIFT IT 8 RIGHT?
000337' 000000  000010
000340' 001000  000332          07520           JC      SHFTR2          ;NO, SHIFT IT ONE PLACE AT A TIME
000341' 000000  000353'
000342' 000000  000311'
                                07540           ;THIS LOOP SPEEDS THINGS UP BY SHIFTING 8 PLACES AT ONE TIME
000343' 001000  000103          07560           MOV     B,E             ;SHIFT NUMBER 1 BYTE RIGHT
000344' 001000  000132          07580           MOV     E,D
000345' 001000  000121          07600           MOV     D,C
000346' 001000  000016          07620           MVI     C,0             ;PUT 0 IN MO
000347' 000000  000000
000350' 001000  000303          07640           JMP     SHFTR1          ;TRY TO SHIFT 8 RIGHT AGAIN
000351' 000000  000336'
000352' 000000  000341'
000353' 001000  000306          07660   SHFTR2: ADI     11>             ;CORRECT SHIFT COUNT
000354' 000000  000011
000355' 001000  000157          07680           MOV     L,A             ;SAVE SHIFT COUNT
000356' 001000  000257          07700   SHFTR3: XRA     A               ;CLEAR CARRY
000357' 001000  000055          07720           DCR     L               ;ARE WE DONE SHIFTING?
000360' 001000  000310          07740           RZ                      ;RETURN IF WE ARE
                                07760   IFE     LENGTH,<
                                07780   SHRADD: CALL    SHFTRO>         ;SHIFT THE NUMBER RIGHT ONE
                                07800   IFN     LENGTH,<
000361' 001000  000171          07820           MOV     A,C             ;GET MO
000362' 001000  000037          07840   SHRADD: RAR                     ;ENTRY FROM FADD, SHIFT IT RIGHT
000363' 001000  000117          07860           MOV     C,A             ;SAVE IT
000364' 001000  000172          07880           MOV     A,D             ;SHIFT NEXT BYTE RIGHT
```



```
000365' 001000  000037          07900           RAR
000366' 001000  000127          07920           MOV     D,A
000367' 001000  000173          07940           MOV     A,E             ;SHIFT LOW ORDER RIGHT
000370' 001000  000037          07960           RAR
000371' 001000  000137          07980           MOV     E,A
000372' 001000  000170          08000           MOV     A,B             ;SHIFT OVERFLOW BYTE RIGHT
000373' 001000  000037          08020           RAR
000374' 001000  000107          08040           MOV     B,A>
000375' 001000  000303          08060           JMP     SHFTR3          ;SEE IF WE ARE DONE
000376' 000000  000356'
000377' 000000  000351'

                                08120   IFE     LENGTH,<
                                08140           ;SHIFT C,D,E,B RIGHT ONE
                                08160           ;THIS IS USED BY SHIFTR, FMULT, FADD
                                08180           ;ALTERS A,B,C,D,E
                                08200   SHFTRO: MOV     A,C             ;GET THE MO
                                08220   SHFROA: RAR                     ;SHIFT IT RIGHT, ENTRY FROM FMULT
                                08240           MOV     C,A
                                08260           MOV     A,D             ;SHIFT THE MO RIGHT
                                08280           RAR
                                08300           MOV     D,A
                                08320           MOV     A,E             ;SHIFT THE LO
                                08340           RAR
                                08360           MOV     E,A
                                08380           MOV     A,B             ;SHIFT THE EXTRA LO BYTE
                                08400           RAR
                                08420           MOV     B,A
                                08440           RET>                    ;ALL DONE
                                08460   PAGE
```

```
                                        08480   SUBTTL  NATURAL LOG FUNCTION
                                        08500   IFN     EXTFNC,<
                                        08520           ;CALCULATION IS BY:
                                        08540           ; LN(F*2**N)=(N+LOG2(F))*LN(2)
                                        08560           ;AN APPROXIMATION POLYNOMIAL IS USED TO CALCULATE LOG2(F)
                                        08600           ;CONSTANTS USED BY LOG
000400' 000000  000000                  08620   FONE:   000     ; 1
000401' 000000  000000                  08640           000
000402' 000000  000000                  08660           000
000403' 000000  000201                  08680           201
000404' 000000  000003                  08700   LOGCN2: 3       ;DEGREE+1
000405' 000000  000252                  08720           252     ; 0.598978650
000406' 000000  000126                  08740           126
000407' 000000  000031                  08760           031
000410' 000000  000200                  08780           200
000411' 000000  000361                  08800           361     ; 0.961470632
000412' 000000  000042                  08820           042
000413' 000000  000166                  08840           166
000414' 000000  000200                  08860           200
000415' 000000  000105                  08880           105     ; 2.88539129
000416' 000000  000252                  08900           252     ; NOTE: THE REFERENCE FOR THIS CONSTANT HAS 100 NOT 105
000417' 000000  000070                  08920           070     ; IN THE LOW ORDER BYTE.
000420' 000000  000202                  08940           202
                                        08960   LOG:    FSIGN                   ;CHECK FOR A NEGATIVE OR ZERO ARGUMENT
000422' 001000  000352                  09000           JPE     FCERR           ;FAC .LE. 0, BLOW HIM OUT OF THE WATER
000423' 000000  000000*
000424' 000000  000376'
                                        09020                                   ;FSIGN ONLY RETURNS 0,1 OR 377 IN A
                                        09040                                   ;THE PARITY WILL BE EVEN IF A HAS 0 OR 377
000425' 001000  000041                  09060           LXI     H,FAC           ;GET POINTER TO EXPONENT
000426' 000000  000235*
000427' 000000  000423'
000430' 001000  000176                  09080           MOV     A,M             ;GET EXPONENT IN A
000431' 001000  000001                  09100           MOVRI   200,065,004,363 ;GET SQR(,5)
000432' 000000  000065
000433' 000000  000200
000434' 001000  000021
000435' 000000  000363
000436' 000000  000004
000437' 001000  000220                  09120           SUB     B               ;REMOVE EXCESS 200
000440' 001000  000365                  09140           PUSH    PSW             ;SAVE EXPONENT FOR LATER
000441' 001000  000160                  09160           MOV     M,B             ;SET EXP TO 200, RESULT IS NUM IN (,5,1)
000442' 001000  000325                  09180           PUSHR                   ;SAVE SQR(,5)
000443' 001000  000305
000444' 001000  000315                  09200           CALL    FADD            ;CALCULATE (F-SQR(,5))/(F+SQR(,5))
000445' 000000  000025'
000446' 000000  000426'
000447' 001000  000301                  09220           POPR                    ;GET SQR(,5) BACK
000450' 001000  000321
000451' 001000  000004                  09240           INR     B               ;GET SQR(2)
000452' 001000  000315                  09260           CALL    FDIV            ;WHERE F=NUMBER LEFT IN FAC
```



```
000453' 000000  000655'
000454' 000000  000445'
000455' 001000  000041                  09280           LXI     H,FONE          ;THE CALCULATION IS EQUIVALENT TO THE ABOVE,
000456' 000000  000400'
000457' 000000  000453'
000460' 001000  000315                  09300           CALL    FSUBS           ; BUT DONE IN A DIFFERENT ORDER
000461' 000000  000011'
000462' 000000  000456'
000463' 001000  000041                  09320           LXI     H,LOGCN2        ;EVALUATE APPROXIMATION POLYNOMIAL
000464' 000000  000404'
000465' 000000  000461'
000466' 001000  000315                  09340           CALL    POLYX
000467' 000000  002615'
000470' 000000  000464'
000471' 001000  000001                  09360           MOVRI   200,200,000,000 ;GET -1/2
000472' 000000  000200
000473' 000000  000200
000474' 001000  000021
000475' 000000  000000
000476' 000000  000000
000477' 001000  000315                  09380           CALL    FADD            ;ADD IN LAST CONSTANT
000500' 000000  000025'
000501' 000000  000467'
000502' 001000  000361                  09400           POP     PSW             ;RETRIEVE ORIGINAL EXPONENT
000503' 001000  000315                  09420           CALL    FINLOG          ;ADD IT TO ORIGINAL NUMBER
000504' 000000  001731'
000505' 000000  000500'
000506' 001000  000001                  09440   MULLN2: MOVRI   200,261,162,030>;GET LN(2)
000507' 000000  000061
000510' 000000  000200
000511' 001000  000021
000512' 000000  000030
000513' 000000  000162
                                        09460   ;       JMP     FMULT           ;MULTIPLY BY LN(2)
                                        09480   PAGE
```

```
                                                09500   SUBTTL  FLOATING MULTIPLICATION AND DIVISION
                                                09520           ;MULTIPLICATION                 FAC:=ARG*FAC
                                                09540           ;ALTERS A,B,C,D,E,H,L
                                                09560   IFE     EXTFNC,<
                                                09580   FMULTS: CALL    MOVRM>                  ;ENTRY WITH POINTER TO ARG IN (HL)
                                                09600   IFN     LENGTH-2,<
0000514' 001000  000041                         09620           XWD     1000,041        ;"LXI   H" AROUND NEXT 2 BYTES
0000515' 001000  000301                         09640   FMULTT: POPR>                   ;ENTRY IF ARGUMENT IS ON THE STACK
0000516' 001000  000321
0000517' 001000  000357                         09660   FMULT:  FSIGN                   ;CHECK IF FAC IS ZERO
0000520' 001000  000310                         09680           RZ                      ;IF IT IS, RESULT IS ZERO
0000521' 001000  000056                         09700           MVI     L,0             ;ADD THE TWO EXPONENTS, L IS A FLAG
0000522' 000000  000000
0000523' 001000  000315                         09720           CALL    MULDIV          ;FIX UP THE EXPONENTS
0000524' 000000  001035'
0000525' 000000  000504'
                                                09740           ;SAVE THE NUMBER IN THE REGISTERS SO WE CAN ADD IT FAST
0000526' 001000  000171                         09760           MOV     A,C             ;GET MO
0000527' 001000  000062                         09780           STA     FMULTA+1        ;STORE MO OF REGISTERS
0000530' 000000  000606'
0000531' 000000  000524'
0000532' 001000  000353                         09800           XCHG                    ;STORE THE TWO LO'S OF THE REGISTERS
0000533' 001000  000042                         09820           SHLD    FMULTB+1
0000534' 000000  000601'
0000535' 000000  000530'
0000536' 001000  000001                         09840           LXI     B,$CODE         ;ZERO THE PRODUCT REGISTERS
0000537' 000000  000000*
0000540' 000000  000534'
0000541' 001000  000120                         09860           MOV     D,B
0000542' 001000  000130                         09880           MOV     E,B
0000543' 001000  000041                         09900           LXI     H,NORMAL        ;PUT ADDRESS OF NORMAL, WERE WE FINISH UP,
0000544' 000000  000146'
0000545' 000000  000537'
0000546' 001000  000345                         09920           PUSH    H               ; ON THE STACK
0000547' 001000  000041                         09940           LXI     H,FMULT2        ;PUT FMULT2 ON THE STACK TWICE, SO AFTER
0000550' 000000  000557'
0000551' 000000  000544'
0000552' 001000  000345                         09960           PUSH    H               ; WE MULTIPLY BY THE LO BYTE, WE WILL
0000553' 001000  000345                         09980           PUSH    H               ; MULTIPLY BY THE MO AND HO
0000554' 001000  000041                         10000           LXI     H,FACLO         ;GET ADDRESS OF LO OF FAC
0000555' 000000  000075*
0000556' 000000  000550'
0000557' 001000  000176                         10020   FMULT2: MOV     A,M             ;GET BYTE TO MULTIPLY BY
0000560' 001000  000043                         10040           INX     H               ;MOVE POINTER TO NEXT BYTE
                                                10060   IFN     LENGTH,<
                                                10080           ORA     A
0000561' 001000  000267                         10100           JZ      FMULT3>         ;ARE WE MULTIPLYING BY ZERO?
0000562' 001000  000312
0000563' 000000  000632'
0000564' 000000  000555'
0000565' 001000  000345                         10120           PUSH    H               ;SAVE POINTER
                                                10140   IFE     LENGTH,<
                                                10160           MVI     L,10>           ;SET UP A COUNT
                                                10180   IFN     LENGTH,<
```



```
0000566' 001000  000353                         10200           XCHG                    ;GET LO'S IN (HL)
0000567' 001000  000036                         10220           MVI     E,10>           ;SET UP A COUNT
0000570' 000000  000010
                                                10240
                                                10260   ;THE PRODUCT WILL BE FORMED IN C,D,E,B, THIS WILL BE IN C,M,L,B PART OF THE
                                                10280   ;TIME IN ORDER TO USE THE "DAD" INSTRUCTION,  AT FMULT2, WE GET THE NEXT
                                                10300   ;BYTE OF THE MANTISSA IN THE FAC TO MULTIPLY BY,  ((HL) POINTS TO IT)
                                                10320   ;(THE FMULT2 SUBROUTINE PRESERVES (HL))  IN 8K, IF THE BYTE IS ZERO, WE JUST
                                                10340   ;SHIFT THE PRODUCT 8 RIGHT,  THIS BYTE IS THEN SHIFTED RIGHT AND SAVED IN D
                                                10360   ;(M IN 4K),  THE CARRY DETERMINES IF WE SHOULD ADD IN THE SECOND FACTOR
                                                10380   ;IF WE DO, WE ADD IT TO C,M,L,  B IS ONLY USED TO DETERMINE WHICH WAY WE
                                                10400   ;ROUND,  WE THEN SHIFT C,M,L,B (C,D,E,B) IN 4K RIGHT ONE TO GET READY FOR THE
                                                10420   ;NEXT TIME THROUGH THE LOOP,  NOTE THAT THE CARRY IS SHIFTED INTO THE MSB OF
                                                10440   ;C,  E HAS A COUNT (L IN 4K) TO DETERMINE WHEN WE HAVE LOOKED AT ALL THE BITS
                                                10460   ;OF D (H IN 4K),
0000571' 001000  000037                         10480   FMULT4: RAR                     ;ROTATE BYTE RIGHT
                                                10500   IFE     LENGTH,<
                                                10520           MOV     M,A>            ;SAVE THE COUNT
                                                10540   IFN     LENGTH,<
0000572' 001000  000127                         10560           MOV     D,A>            ;SAVE IT
0000573' 001000  000171                         10580           MOV     A,C             ;GET MO
0000574' 001000  000322                         10600           JNC     FMULT5          ;DON'T ADD IN NUMBER IF BIT WAS ZERO
0000575' 000000  000627'
0000576' 000000  000563'
                                                10620   IFE     LENGTH,<
                                                10640           XCHG>                   ;PUT THE LO'S IN (HL)
0000577' 001000  000325                         10660           PUSH    D               ;SAVE COUNTERS
0000600' 001000  000021                         10680   FMULTB: LXI     D,$CODE         ;GET LO'S OF NUMBER TO ADD, THIS IS SET ABOVE
0000601' 000000  000537*
0000602' 000000  000575'
0000603' 001000  000031                         10700           DAD     D               ;ADD THEM IN
0000604' 001000  000321                         10720           POP     D               ;GET COUNTERS BACK
0000605' 001000  000316                         10740   FMULTA: ACI     0               ;ADD IN MO, THIS IS SET UP ABOVE
0000606' 000000  000000
                                                10760   IFE     LENGTH,<
                                                10780           XCHG                    ;PUT THE LO'S BACK IN (DE)
                                                10800   FMULTS: CALL    SHFROA          ;SHIFT THE RESULT RIGHT ONE
                                                10820           DCR     L               ;ARE WE DONE?
                                                10840           MOV     A,M>            ;GET NUMBER WE ARE MULTIPLYING BY
                                                10860   IFN     LENGTH,<
0000607' 001000  000037                         10880   FMULTS: RAR                     ;ROTATE RESULT RIGHT ONE
0000610' 001000  000117                         10900           MOV     C,A
0000611' 001000  000174                         10920           MOV     A,M             ;ROTATE NEXT BYTE
0000612' 001000  000037                         10940           RAR
0000613' 001000  000147                         10960           MOV     M,A
0000614' 001000  000175                         10980           MOV     A,L             ;ROTATE NEXT LOWER ORDER
0000615' 001000  000037                         11000           RAR
0000616' 001000  000157                         11020           MOV     L,A
0000617' 001000  000170                         11040           MOV     A,B             ;ROTATE LO
0000620' 001000  000037                         11060           RAR
0000621' 001000  000107                         11080           MOV     B,A
0000622' 001000  000035                         11100           DCR     E               ;ARE WE DONE?
0000623' 001000  000172                         11120           MOV     A,D>            ;GET NUMBER WE ARE MULTIPLYING BY
```

```
000624' 001000  000302          11140           JNZ     FMULT4          ;MULTIPLY AGAIN IF WE ARE NOT DONE
000625' 000000  000571'
000626' 000000  000601'
                                11160   IFN     LENGTH,<
000627' 001000  000353          11180           XCHG>                   ;GET LO'S IN (HL)
000630' 001000  000341          11200   POPMRT: POP     H               ;GET POINTER TO NUMBER TO MULTIPLY BY
000631' 001000  000311          11220           RET                     ;ALL DONE
                                11240   IFN     LENGTH,<
000632' 001000  000103          11260   FMULT3: MOV     B,E             ;MULTIPLY BY ZERO: SHIFT EVERYTHING 8 RIGHT
000633' 001000  000132          11280           MOV     E,D
000634' 001000  000121          11300           MOV     D,C
000635' 001000  000117          11320           MOV     C,A             ;SHIFT IN 8 ZEROS ON THE LEFT
000636' 001000  000311          11340           RET>                    ;ALL DONE

                                11400           ;DIVIDE FAC BY 10
                                11420           ;ALTERS A,B,C,D,E,H,L
000637' 001000  000315          11440   DIV10:  CALL    PUSHF           ;SAVE NUMBER
000640' 000000  001205'
000641' 000000  000625'
                                11460   IFN     LENGTH-2,<
000642' 001000  000001          11480           MOVRI   204,040,000,000 ;LOAD CONSTANT '10' INTO REGISTERS
000643' 000000  000040
000644' 000000  000204
000645' 001000  000021
000646' 000000  000000
000647' 000000  000000
000650' 001000  000315          11500           CALL    MOVFR>          ;MOVE THE CONSTANT TO THE FAC
000651' 000000  001225'
000652' 000000  000640'
                                11520   IFE     LENGTH-2,<
                                11540           LXI     H,FTEN          ;GET POINTER TO THE CONSTANT '10'
                                11560           CALL    MOVFM>          ;MOVE TEN INTO THE FAC
000653' 001000  000301          11580   FDIVT:  POPR                    ;GET NUMBER BACK IN REGISTERS
000654' 001000  000321
                                11600                                   ;FALL INTO DIVIDE AND WE ARE DONE

                                11660           ;DIVISION       FAC:=ARG/FAC
                                11680           ;ALTERS A,B,C,D,E,H,L
000655' 001000  000357          11700   FDIV:   FSIGN                   ;CHECK FOR DIVISION BY ZERO
000656' 001000  000312          11720           JZ      DV0ERR          ;HE IS TRYING TO GET AWAY WITH IT
000657' 000000  000000*
000660' 000000  000651'
000661' 001000  000056          11740           MVI     L,377           ;SUBTRACT THE TWO EXPONENTS, L IS A FLAG
000662' 000000  000377
000663' 001000  000315          11760           CALL    MULDIV          ;FIX UP THE EXPONENTS AND THINGS
000664' 000000  001035'
000665' 000000  000657'
000666' 001000  000064          11780           INR     M               ;ADD 2 TO EXPONENT TO CORRECT SCALING
000667' 001000  000064          11800           INR     M
                                11820           ;HERE WE SAVE THE FAC IN MEMORY SO WE CAN SUBTRACT IT FROM THE NUMBER
                                11840           ;IN THE REGISTERS QUICKLY.
```



```
000670' 001000  000053          11860           DCX     H               ;POINT TO MO
000671' 001000  000176          11880           MOV     A,M             ;GET MO
000672' 001000  000062          11900           STA     FDIVA+1         ;SAVE IT
000673' 000000  000734'
000674' 000000  000664'
000675' 001000  000053          11920           DCX     H               ;SAVE MIDDLE ORDER
000676' 001000  000176          11940           MOV     A,M
000677' 001000  000062          11960           STA     FDIVB+1         ;PUT IT WHERE NOTHING WILL HURT IT
000700' 000000  000730'
000701' 000000  000673'
000702' 001000  000053          11980           DCX     H               ;SAVE LO
000703' 001000  000176          12000           MOV     A,M
000704' 001000  000062          12020           STA     FDIVC+1
000705' 000000  000724'
000706' 000000  000700'

                                12060   ;THE NUMERATOR WILL BE KEPT IN B,H,L.  THE QUOTIENT WILL BE FORMED IN C,D,E,
                                12080   ;TO GET A BIT OF THE QUOTIENT, WE FIRST SAVE B,H,L ON THE STACK, THEN
                                12100   ;SUBTRACT THE DENOMINATOR THAT WE SAVED IN MEMORY.  THE CARRY INDICATES
                                12120   ;WHETHER OR NOT B,H,L WAS BIGGER THAN THE DENOMINATOR.  IF B,H,L WAS BIGGER,
                                12140   ;THE NEXT BIT OF THE QUOTIENT IS A ONE.  TO GET THE OLD B,H,L OFF THE STACK,
                                12160   ;WE POP THEM INTO THE PSW.  IF THE DENOMINATOR WAS BIGGER, THE NEXT BIT OF
                                12180   ;THE QUOTIENT IS ZERO, AND WE GET THE OLD B,H,L BACK BY POPPING IT OFF THE
                                12200   ;STACK.  WE HAVE TO KEEP AN EXTRA BIT OF THE QUOTIENT IN FDIVG+1 IN CASE THE
                                12220   ;DENOMINATOR WAS BIGGER.  THEN B,H,L WILL GET SHIFTED LEFT.  IF THE MSB  OF
                                12240   ;B WAS ONE, IT HAS TO BE STORED SOMEWHERE, SO WE STORE IT IN FDIVG+1.  THEN
                                12260   ;THE NEXT TIME THROUGH THE LOOP B,H,L WILL LOOK BIGGER BECAUSE IT HAS AN
                                12280   ;EXTRA MO BIT IN FDIVG+1, WE ARE DONE DIVIDING WHEN THE MSB OF C IS A ONE.
                                12300   ;THIS OCCURS WHEN WE HAVE CALCULATED 24 BITS OF THE QUOTIENT.  WHEN WE JUMP
                                12320   ;TO ROUND, THE 25TH BIT OF THE QUOTIENT DETERMINES WHETHER WE ROUND OR NOT.
                                12340   ;IT IS IN THE MSB OF A.  IF INITIALLY THE DENOMINATOR IS BIGGER THAN THE
                                12360   ;NUMERATOR, THE FIRST BIT OF THE QUOTIENT WILL BE ZERO.  THIS MEANS WE
                                12380   ;WILL GO THROUGH THE DIVIDE LOOP 26 TIMES, SINCE IT STOPS ON THE 25TH BIT
                                12400   ;AFTER THE FIRST NON-ZERO BIT OF THE EXPONENT.  SO, THIS QUOTIENT WILL LOOK
                                12420   ;SHIFTED LEFT ONE FROM THE QUOTIENT OF TWO NUMBERS IN WHICH THE NUMERATOR IS
                                12440   ;BIGGER.  THIS CAN ONLY OCCUR ON THE FIRST TIME THROUGH THE LOOP, SO C,D,E
                                12460   ;ARE ALL ZERO.  SO, IF WE FINISH THE LOOP AND C,D,E ARE ALL ZERO, THEN WE
                                12480   ;MUST DECREMENT THE EXPONENT TO CORRECT FOR THIS.
000707' 001000  000101          12500           MOV     B,C             ;GET NUMBER IN B,H,L
000710' 001000  000353          12520           XCHG
000711' 001000  000257          12540           XRA     A               ;ZERO C,D,E AND HIGHEST ORDER
000712' 001000  000117          12560           MOV     C,A
000713' 001000  000127          12580           MOV     D,A
000714' 001000  000137          12600           MOV     E,A
000715' 001000  000062          12620           STA     FDIVG+1
000716' 000000  000737'
000717' 000000  000705'
000720' 001000  000345          12640   FDIV1:  PUSH    H               ;SAVE LO'S OF NUMBER
000721' 001000  000305          12660           PUSH    B               ;SAVE MO OF NUMBER
000722' 001000  000175          12680           MOV     A,L             ;SUBTRACT NUMBER THAT WAS IN FAC
000723' 001000  000326          12700   FDIVC:  SUI     0               ;SUBTRACT LO
000724' 000000  000000
000725' 001000  000157          12720           MOV     L,A             ;SAVE IT
```