```
DISPLAY SIG
*IN PROGRESS
0001   .OPT XRE,ERR,COU,LIS,MEM,GEN
0002 ;
0003 ;
0004 ;
0005 ;                  COPYRIGHT  MOS TECHNOLOGY INC.
0006 ;                    JANUARY, 1976   REV. 0
0007 ;
0008 ;
0009 ;
0010 ;
0011 ;
0012 ;
0013 ;
0014 ;       THE ALGORITHMS BELOW ARE BASED UPON THE SCIENTIFIC NOTATION
0015 ;       FOR REAL NUMBERS. IT IS ASUMED THAT THE NUMBERS CONSIST OF A
0016 ;       SIX BCD DIGIT MANTISSA, A TWO BCD DIGIT EXPONENT AND THE SIGNS
0017 ;       FOR THE MANTISSA, AS WELL AS THE EXPONENT. THE ARRANGEMENT
0018 ;       OF THE NUMERICAL INFORMATION THAT CONSTITUTES AN ARGUMENT FOR
0019 ;       THESE ROUTINES IS CONTAINED IN A CONTIGUOUS FIVE BYTE BLOCK:
0020 ;            1) THE FIRST BYTE CONTAINS THE SIGN INFORMATION, I.E.
0021 ;               THE SIGN OF THE MANTISSA (SM), AND THE SIGN OF THE
0022 ;               EXPONENT (SE). THE FORMAT OF THE SIGN DATA IS AS
0023 ;               FOLLOWS:-
0024 ;                   A) BIT 7 CONTAINS THE SIGN OF THE MANTISSA.
0025 ;                      (+=0 AND -=1)
0026 ;                   B) BIT 6 CONTAINS THE SIGN OF THE EXPONENT.
0027 ;                      (+=0 AND -=1).
0028 ;            2) THE NEXT THREE BYTES CONTAIN THE MANTISSA. THIS PER-
0029 ;               MITS SIX BCD DIGITS. THE MANTISSA IS ASSUMED TO BE
0030 ;               LEFT JUSTIFIED AND ANY ANSWER PRODUCED WILL ALSO BE
0031 ;               LEFT JUSTIFIED. THE DECIMAL POINT IS ASSUMED TO
0032 ;               FOLLOW THE MOST SIGNIFICANT DIGIT OF THE MANTISSA.
0033 ;            3) THE LAST BYTE CONTAINS THE EXPONENT OF TEN. THUS
0034 ;               THE EXPONENT MAY CONTAIN TWO BCD DIGITS, WHICH
0035 ;               VARY FROM 00 THRU 99.
0036 ;
0037 ;
0038 ;
0039 ;       NOTATION AND WORKING STORAGE NAMES
0040 ;
0041 ;       THE WORKING STORAGE (THE PAGE ZERO RAM) USED BY THESE
0042 ;       ROUTINES IS BROKEN DOWN INTO FIVE BYTE BLOCKS, A FEW TEMP-
0043 ;       ORARY STORAGE BYTES AND A FEW COUNTERS. THE FIVE BYTE BLOCKS
0044 ;       ARE NAMED RA,RB,RC,RD,RE,RF,RS AND RT. RA IS COMPOSED OF
0045 ;       (SA,MA,MA+1,MA+2,EA) WHERE
0046 ;            1) SA DENOTES THE SIGN BYTE OF RA.
0047 ;            2) MA DENOTES THE MOST SIGNIFICANT BYTE OF RA.
0048 ;            3) EA DENOTES THE EXPONENT OF TEN IN RA.
0049 ;            4) ( ,... ) DENOTES A CONTIGUOUS BYTE STRING.
0050 ;       NOTE THAT THE ASSEMBLER CONSIDERS 'RA' TO BE THE NAME OF
0051 ;       EXACTLY ONE BYTE. HOWEVER, WHEN USED IN THE COM ENTS
0052 ;       BELOW IT REFERS TO A FIVE BYTE BLOCK.
0053 ;       NOTE ALSO THAT THE SIGN OF THE MANTISSA OF RA IS OFTEN
0054 ;       DENOTED BY SMA, AND THE SIGN OF THE EXPONENT BY SEA. SIMILAR
0055 ;       REMARKS HOLD FOR RB THUR RT.
0056 ;
0057 ;
0058 ;
```

```
0059          *=$0000
0060 ZERO=$00
0061 ONE=$01
0062 TWO=$02
0063 THREE=$03
0064 FOUR=$04
0065 FIVE=$05
0066 SIX=$06
0067 SEVEN=$07
0068 EIGHT=$08
0069 NINE=$09
0070 TEN=$10
0071 HTEN=$0A
0072 AADDR=$00
0073 BADDR=$05
0074 CADDR=$0A
0075 DADDR=$0F
0076 EADDR=$14
0077 FADDR=$19
0078 SADDR=$1E
0079 TADDR=$23
0080 NEGE=$40
0081 NEGM=$80
0082 NEGEM=$C0
0083 NTYN=$99
0084 NTYE=$98
0085 RA       *=*
0086 SA       *=*+1
0087 MA       *=*+3
0088 EA       *=*+1
0089 RB       *=*
0090 SB       *=*+1
0091 MB       *=*+3
0092 EB       *=*+1
0093 RC       *=*
0094 SC       *=*+1
0095 MC       *=*+3
0096 EC       *=*+1
0097 RD       *=*
0098 SD       *=*+1
0099 MD       *=*+3
0100 ED       *=*+1
0101 RE       *=*
0102 SE       *=*+1
0103 ME       *=*+3
0104 EE       *=*+1
0105 RF       *=*
0106 SF       *=*+1
0107 MF       *=*+3
0108 EF       *=*+1
0109 RS       *=*
0110 SS       *=*+1
0111 MS       *=*+3
0112 ES       *=*+1
0113 RT       *=*
0114 ST       *=*+1
0115 MT       *=*+3
0116 ET       *=*+1
0117 TEMP     *=*+1
0118 TEMP1    *=*+1
```

```
0119 INDEX    *=*+1
0120 JNDEX    *=*+1
0121 CNTA     *=*+1
0122 CNTB     *=*+1
0123 COUNT    *=*+1
0124 CNTC     *=*+1
0125 CNTD     *=*+1
0126 CNTE     *=*+1
0127          *=$8000
0128 ;
0129 ;
0130 ;
0131 ;         THIS ROUTINE PERFORMS THE ADDITION OF TWO NUMBERS IN SCIENTIF
0132 ;         NOTATION. THE NUMERICAL VALUES OF THE SUMMANDS ARE ARRANGED
0133 ;         AS FOLLOWS:-
0134 ;                    1) A BYTE WHICH CONTAINS THE SIGN INFORMATION
0135 ;                 FOR THE MANTISSA AND THE EXPONENT,
0136 ;                    2) THREE BYTES WHICH CONTAIN THE SIX BCD DIGITS
0137 ;                 OF THE MANTISSA,
0138 ;                    3) ONE BYTE WHICH CONTAINS THE TWO BCD DIGITS OF
0139 ;                 THE EXPONENT.
0140 ;          THE ADDITION (SUBTRACTION) IS PERFORMED BY COMPARING THE
0141 ;         ABSOLUTE VALUES OF THE SUMMANDS AND ALIGNING THE DECIMAL
0142 ;          POINTS BY RIGHT SHIFTING THE SMALLER SUMMAND. AFTER
0143 ;           ALIGNMENT A TEST IS PERFORMED TO DETERMINE WHETHER TO ADD
0144 ;           OR SUBTRACT. THIS TEST IS BASED ON THE SIGNS OF THE
0145 ;           MANTISSAS OF THE SUMMANDS.
0146 ;           AFTER THE ACTUAL ADDITION OR SUBTRACTION THE EXPONENT IS
0147 ;           CORRECTED (IF NECESSARY) AND THE ANSWER IS PUT INTO
0148 ;           SCIENTIFIC NOTATION.
0149 ;
0150 ;
0151 ;         THE NAMES OF THE SUMMANDS ARE RA AND RB. RA CONSISTS OF SA
0152 ;          FOLLOWED BY MA,MA+1,MA+2 AND EA. RB IS ARRANGED SIMILARLY.
0153 ;         THE SUM IS PLACED IN RC.
0154 ;
0155 ;
0156 ;
0157 ; COMPUTE EXCLUSIVE OR OF SIGN BITS AND STORE IN TEMP
0158 ;
0159 ADD      LDA SA
0160          EOR SB
0161          STA TEMP
0162           SED
0163 ;
0164 ; SET RD=0.
0165 ;
0166          LDX #DADDR
0167          JSR CLREG
0168 ;
0169 ; TEST (MA,...,MA+2) FOR ZERO.
0170 ;
0171          LDX #ONE
0172          JSR ZTEST
0173 ;
0174 ; IF (MA,...,MA+2)=0, SWAP RA AND RB.
0175 ;
0176          BCC ADD2
0177 ;
0178 ; TEST (MB,...,MB+2) FOR ZERO.
```

```
0179 ;
0180         LDX #SIX
0181         JSR ZTEST
0182 ;
0183 ; IF (MB,...,MB+2)=0, PREPARE FOR POINT ALIGNMEMT.
0184 ;
0185         BCC ADD3
0186 ;
0187 ; IF SEA = SEB, COMPARE (MA,...,MA+2) WITH (MB,...,MB+2).
0188 ;
0189         BIT TEMP
0190         BVC ADD6
0191 ;
0192 ; IF (MB,...,MB+2) IS NOT ZERO AND SEA IS NOT EQUAL TO SEB,
0193 ; TEST SEA.
0194 ;
0195 ADD1    BIT SA
0196 ;
0197 ; IF SEA=0, SKIP NEXT INSTRUCTION
0198 ;
0199         BVC ADD3
0200 ;
0201 ; IF SEA = 1, SWAP RA AND RB.
0202 ;
0203 ADD2    JSR ASB
0204 ;
0205 ;
0206 ;
0207 ; PREPARE FOR POINT ALIGNMENT
0208 ; IF SEA IS EQUAL TO SEB, TEST SEA AT ADD9.
0209 ; IF SEA IS NOT EQUAL TO SEB, ADD EA TO EB. IF THE SUM IS
0210 ; LESS THAN 99 STORE SUM IN COUNT.
0211 ; IF THE SUM IS GREATER THAN 99, SET RB = 0.
0212 ;
0213 ;
0214 ;
0215 ADD3    SED
0216         BIT TEMP
0217         BVC ADD9
0218         LDA EA
0219         CLC
0220         ADC EB
0221         BCS ADD5
0222 ;
0223 ; COMPARE COUNT WITH 6, IF GREATER THAN OR EQUAL TO 6
0224 ; SET RB = 0.
0225 ;
0226 ADD4    STA COUNT
0227         CMP #SIX
0228         BEQ ADD5
0229         BCS ADD5
0230         JMP ADD41
0231 ;
0232 ; COMPARE RA TO RB AND SWAP, IF NECESSARY, SO THAT THE ABSOLUTE
0233 ; VALUE OF RA IS GREATER THAN OR EQUAL TO THAT OF RB.
0234 ;
0235 ADD6    JSR COMPAB
0236 ;
0237 ; COMPARE ABS(RA) TO ABS(RB)
0238 ; LOAD TEST VALUE CNTA
```

```
0239 ; IF CNTA = 0 , GO TO ADD8
0240 ; OTHERWISE SWAP RA WITH RB.
0241 ;
0242          LDA CNTA
0243          BEQ ADD8
0244 ADD7     JSR ASB
0245 ADD8     LDA EA
0246          CMP EB
0247          BEQ ADD3
0248          BCS ADD1
0249          JMP ADD7
0250 ;
0251 ; COMPUTE COUNT VALUE FROM EA,EB,SEA
0252 ;
0253 ADD9     SEC
0254          BIT SA
0255          BVS ADD10
0256          LDA EA
0257          SBC EB
0258          JMP ADD4
0259 ADD10    LDA EB
0260          SBC EA
0261          JMP ADD4
0262 ;
0263 ;
0264 ;
0265 ; USEING COUNT VALUE COMPUTE BY TABLE LOOKUP THE NUMBER OF RIGHT SHIFT
0266 ; NECESSARY TO ALIGN THE MANTISSAS OF THE SUMMANDS. MOVE AND
0267 ; SHIFT THE MANTISSA OF B TO THE MANTISSA OF D. ADTAB1 CONTAINS
0268 ; THE NUMBER OF BYTES TO BE MOVED AND ADTAB2 CONTAINS THE OFFSET
0269 ; FROM RA OF THE FIRST BYTE OF THE MANTISSA OF RD WHICH WILL RE-
0270 ; CEIVE DATA FROM THE MANTISSA OF B.
0271 ;
0272 ;
0273 ;
0274 ADD41    LDY COUNT
0275          LDA ADTAB1,Y
0276          STA CNTB
0277          LDX ADTAB2,Y
0278          LDY #BADDR+1
0279          JSR MVC
0280          LDA COUNT
0281          LSR A
0282          BCC ADD5
0283 ;
0284 ; LEFT SHIFT (MD,...,ED) ONE DIGIT.
0285 ;
0286          LDX #THREE
0287 ADD42    ASL MD+3
0288          ROL MD+2
0289          ROL MD+1
0290          ROL MD
0291          DEX
0292          BPL ADD42
0293 ;
0294 ; MOVE RA TO RC.
0295 ;
0296 ADD5     LDY #ADDR+1
0297          LDX #CADR+1
0298          LDA #THREE
```

```
0299          JSR MVMG
0300  ;
0301  ; DETERMINE WHETHER TO ADD OR SUBTRACT
0302  ;
0303          BIT TEMP
0304          BMI ADD13
0305  ;
0306  ; PERFORM THE ADDITION (MC,...,MC+2) = (MC,...,MC+2)+(MD,...,MD+2).
0307  ;
0308          LDX #TWO
0309          CLC
0310  ADD11   LDA MC,X
0311          ADC MD,X
0312          STA MC,X
0313          DEX
0314          BPL ADD11
0315  ;
0316  ;
0317  ;
0318  ; IF THIS ADDITION CAUSES AN OVERFLOW RIGHT SHIFT (SC,...,MC+2).
0319  ; INSERT A ONE IN MSD OF MC
0320  ; AND CORRECT SIGN AND EXPONENT OF ANSWER
0321  ;
0322  ;
0323  ;
0324          BCC ADD120
0325          LDA #ONE
0326          STA RC
0327          JSR RSC
0328          LDA EA
0329          SEC
0330          BIT SA
0331          BVC ADD110
0332          SBC #ONE
0333          STA EC
0334          BNE ADD12
0335          LDA #$BF
0336          AND SA
0337          STA SC
0338  ADD12   RTS
0339  ADD120  LDA SA
0340          STA SC
0341          RTS
0342  ;
0343  ; CORRECT EXPONENT
0344  ;
0345  ADD110  ADC #ZERO
0346          STA EC
0347          BCC ADD120
0348  ;
0349  ; IF THE EXPONENT OVERFLOWS SET RC = FF FFFFFF FF.
0350  ;
0351          JSR INFIN
0352          RTS
0353  ;
0354  ; SUBTRACT (MD,...,MD+2) FROM (MC,...,MC+2) AND ADJUST SIGN AND
0355  ; EXPONENT OF ANSWER.
0356  ;
0357  ADD13   LDX #TWO
0358          SEC
```

```
0359 ADD14   LDA MC,X
0360         SBC MD,X
0361         STA MC,X
0362         DEX
0363         BPL ADD14
0364 ;
0365 ; MOVE EXPONENT AND SIGN OF ANSWER TO RC.
0366 ;
0367         LDA EA
0368         STA EC
0369         LDA SA
0370         STA SC
0371 ;
0372 ; TEST THE MANTISSA OF RC FOR ZERO.
0373 ;
0374         LDX #$0B
0375         JSR ZTEST
0376         BCC ADD18
0377 ;
0378 ; IF THE MANTISSA OF RC IS NOT ZERO , TEST THE MSD OF MC.
0379 ;
0380 ADD15   LDA MC
0381         AND #$F0
0382         BNE ADD19
0383 ;
0384 ; IF THE MSD OF MC IS ZERO, LEFT SHIFT (MC,...,MC+2) BY ONE DIGIT.
0385 ;
0386         LDX #THREE
0387 ADD16   ASL MC+2
0388         ROL MC+1
0389         ROL MC
0390         DEX
0391         BPL ADD16
0392 ;
0393 ; TEST THE SIGN OF THE EXPONENT OF RC
0394 ;
0395 ADD17   BIT SC
0396         SEC
0397         LDA EC
0398 ;
0399 ; IF THE SIGN OF THE EXPONENT OF RC IS POSITIVE , DECREMENT EC.
0400 ;
0401         BVC ADD20
0402 ;
0403 ; IF THE SIGN OF EC IS NEGATIVE, INCREMENT EC.
0404 ;
0405         ADC #ZERO
0406         STA EC
0407 ;
0408 ; IF EC IS LESS THAN OR EQUAL TO 99, TEST MC AT ADD15
0409 ; OTHERWISE, SET RC=0.
0410 ;
0411         BCC ADD15
0412 ADD18   JSR ZEROA
0413 ADD19   RTS
0414 ;
0415 ; DECREMENT EC.
0416 ;
0417 ADD20   SBC #ONE
0418         STA EC
```

```
0419 ;
0420 ; IF EC-1=99, CORRECT SEC,EC, AND GO TO MC TEST AT ADD15
0421 ;
0422         BCS ADD15
0423 ;
0424 ; CORRECT EC.
0425 ;
0426         LDA #ONE
0427         STA EC
0428 ;
0429 ; CORRECT SIGN OF EC.
0430 ;
0431         LDA #$40
0432         ORA SC
0433         STA SC
0434         JMP ADD15
0435 ;
0436 ;
0437 ;
0438 ;       THIS ROUTINE COMPUTES THE PRODUCT OF TWO BCD NUMBERS IN
0439 ;       SCIENTIFIC NOTATION. THE PRODUCT IS FORMED BY REPEATED
0440 ;       ADDITION AND APPROPRIATE MANIPULATION OF THE SIGNS AND
0441 ;       EXPONENTS OF THE ARGUMENTS. THE RESULT IS LEFT JUST-
0442 ;       IFIED AND IN SCIENTIFIC NOTATION. THE NAMES OF THE FACTORS
0443 ;       ARE RA AND RB. THE PRODUCT IS PLACED IN RC.
0444 ;
0445 ;
0446 ;
0447 MLTPLY SED
0448         LDX #CADDR
0449         JSR CLREG
0450 ; MOVE RB TO RD.
0451 ;
0452         LDY #BADDR
0453         LDX #DADDR
0454         JSR MVRG
0455 ;
0456 ; CLEAR EB AND SB, SET SWITCH VALUE TEMP1=0.
0457 ;
0458         LDA #ZERO
0459         STA EB
0460         STA TEMP1
0461         STA SB
0462 ;
0463 ; TEST (MA,...,MA+2) FOR ZERO.
0464 ;
0465         LDX #ONE
0466         JSR ZTEST
0467 ;
0468 ; IF (MA,...,MA+2) = 0, SET RC=0.
0469 ;
0470         BCC MLT1
0471 ;
0472 ; IF (MA,...,MA+2) IS NOT ZERO, TEST (MB,...,MB+2) FOR ZERO.
0473 ;
0474         LDX #SIX
0475         JSR ZTEST
0476 ;
0477 ; IF (MB,...,MB+2) IS NOT ZERO, BEGIN FORMATION OF PRODUCT.
0478 ;
```

```
0479          BCS MLT3
0480 MLT1     JSR ZEROA
0481          LDA #ZERO
0482          STA SC
0483          RTS
0484 MLT2     LDA SD
0485          STA SC
0486          LDA ED
0487          STA EC
0488          RTS
0489 ;
0490 ; ACTUAL MULTIPLICATION STARTS HERE. INDEX REGISTER Y ACTS AS LOOP COU
0491 ; ER AND A SUBMULTIPLIER SELECTOR.
0492 ;
0493 MLT3     LDY #SIX
0494 MLT4     TYA
0495 ;
0496 ; DIVIDE THE CONTENTS OF Y BY TWO AND TEST FOR PARITY (ODD OR EVEN).
0497 ; THIS ALLOWS ONE TO SELECT THE HIGH ORDER OR LOW ORDER DIGIT IN
0498 ; A BYTE OF THE MULTIPLIER.
0499 ;
0500          LSR A
0501          TAX
0502          TYA
0503          AND #ONE
0504 ;
0505 ; IF PARITY IS ODD, USE LOW ORDER DIGIT AS SUBMULTIPLIER.
0506 ;
0507          BNE MLT7
0508 ;
0509 ; IF THE PARITY IS EVEN, USE THE HIGH ORDER DIGIT AS SUBMULTIPLIER.
0510 ;
0511          LDA MA,X
0512          LSR A
0513          LSR A
0514          LSR A
0515          LSR A
0516 ;
0517 ; DECREMENT ACC AND CHECK FOR BORROW.
0518 ;
*END
```

```
DISPLAY SIGMAM
*IN PROGRESS
0001 ;
0002 ;
0003 ;     NOTE: ADD 518 TO THE LINE NUMBERS IN THIS PORTION OF THE LISTING
0004 ;
0005 ;
0006 ;
0007 MLT5    SEC
0008         SBC #ONE
0009 ;
0010 ; IF BORROW OCCURS, CHECK FOR END OF MULTIPLY.
0011 ;
0012         BCC MLT8
0013 ;
0014 ; IF NO BORROW, STORE DECREMENTED VALUE (IN ACC)IN COUNT.
0015 ;
0016         STA COUNT
0017 ;
0018 ; ADD (SB,MB,MB+1,MB+2) TO (SC,MC,MC+1,MC+2).
0019 ;
0020         CLC
0021         LDX #THREE
0022 MLT6    LDA RB,X
0023         ADC RC,X
0024         STA RC,X
0025         DEX
0026         BPL MLT6
0027 ;
0028 ; LOAD COUNT INTO ACC AND GO DECREMENT SUBMULTIPLIER.
0029 ;
0030         LDA COUNT
0031         JMP MLT5
0032 MLT7    LDA MA,X
0033         AND #$0F
0034         JMP MLT5
0035 ;
0036 ; TEST FOR END OF MULTIPLY.
0037 ;
0038 MLT8    DFY
0039 ;
0040 ; IF END, GO TO MLT9, OTHERWISE RIGHT SHIFT RC AND SELECT NEW
0041 ; SUBMULTIPLIER FROM MA,X.
0042 ;
0043         BMI MLT9
0044         STY TEMP
0045         JSR RSC
0046         LDY TEMP
0047         JMP MLT4
0048 ;
0049 ;
0050 ;
0051 ;
0052 ; THIS IS A PORTION OF THE DIVIDE EXIT ROUTINE WHICH IS SHARED IN
0053 ; PART BY THE MULTIPLY ROUTINE, THE PREFIX OF 'MD' USED IN STATEMENT
0054 ; NAMES INDICATES SOME OF THE SHARED CODE.
0055 ;
0056 ;
0057 ;
0058 DIV6    LDA #ONE
```

```
0059          STA TEMP1
0060          LDY #DADDR
0061          LDX #CADDR
0062          LDA CNTC
0063          EOR #$40
0064          STA SB
0065          JMP MD12
0066 ;
0067 ; SUBTRACT EB FROM EA.
0068 ;
0069 MD10     SEC
0070          SBC EB
0071 ;
0072 ; IF EB IS SMALLER THAN EA STORE ACC IN ED. OTHERWISE SUBTRACT
0073 ; EA FROM EB AND STORE RESULT IN ED.
0074 ;
0075          BCS MD2
0076          SEC
0077          LDA EB
0078          SBC EA
0079          STA ED
0080          LDA SB
0081          JMP MD3
0082 MD11     LDA SA
0083          JMP MD3
0084 ;
0085 ; RESTORE RB, BY MOVING RD TO RB.
0086 ;
0087 MLT9     LDY #DADDR
0088          LDX #BADDR
0089 MD12     JSR MVRG
0090 ;
0091 ; COMPUTE THE EXCLUSIVE OR OF SA, SB AND STORE RESULT IN TEMP.
0092 ; TEST TEMP AND LOAD ACC WITH EXPONENT OF RA.
0093 ;
0094          LDA SB
0095          EOR SA
0096          STA TEMP
0097          BIT TEMP
0098          LDA EA
0099 ;
0100 ; IF THE EXPONENTS HAVE DIFFERENT SIGNS GO TO MD10 TO SUBTRACT
0101 ; EB FROM EA, OR EA FROM EB DEPENDING ON WHICH IS LARGER.
0102 ;
0103          BVS MD10
0104 ;
0105 ; IF THE EXPONENTS HAVE THE SAME SIGN ADD EA TO EB AND CHECK FOR
0106 ; OVERFLOW OF EXPONENT.
0107 ;
0108 MD1      CLC
0109          ADC EB
0110          BCC MD2
0111          BNE MD59
0112          LDA TEMP1
0113          BEQ MDOV2
0114          BIT SA
0115          BVS MD7
0116          LDA CNTA
0117          BEQ MD61
0118          LDA #ZERO
```

```
0119          STA CNTA
0120 MDOV1    LDA #NTYN
0121          JMP MD2
0122 MDOV2    BIT SA
0123          BVC MD61
0124          LDA SC
0125          BEQ MDOV1
0126          JSR RSC
0127          JMP MDOV1
0128 ;
0129 ; STORE ACC IN ED AND CHECK ACC FOR ZERO.
0130 ;
0131 MD2      STA ED
0132          BNE MD11
0133 ;
0134 ; IF ED=0, CHANGE SIGN OF EXPONENT OF ANSWER AND STORE IN SD
0135 ;
0136          LDA SA
0137          AND #$BF
0138 MD3      STA SD
0139 ;
0140 ; TEST TO SEE IF THE SIGNS OF THE MANTISSAS ARE UNLIKE.
0141 ;
0142 MD4      LDA TEMP
0143 ;
0144 ; IF SMA AND SMB ARE UNLIKE , SET SMD=1(-).
0145 ;
0146          BMI MD8
0147 ;
0148 ; IF THE SIGNS ARE THE SAME SET SMD=0(+).
0149 ;
0150          LDA #$7F
0151          AND SD
0152 MD5      STA SD
0153 ;
0154 ; CHECK MULTIPLY/DIVIDE INDICATOR. TEMP1=1, IF DIVIDE AND
0155 ; TEMP1=0, IF MULTIPLY.
0156 ;
0157          LDA TEMP1
0158          BNE DIVEXT
0159 ;
0160 ; CHECK SC FOR ZERO.
0161 ;
0162          LDA RC
0163          BEQ MD51
0164 ;
0165 ; IF SC IS NONZERO RIGHT SHIFT (SC,MC,MC+1,MC+2).
0166 ;
0167          JSR RSC
0168 ;
0169 ; CORRECT EXPONENT OF ANSWER IN ED.
0170 ;
0171          LDA ED
0172          BIT SD
0173          BVS MD9
0174          CLC
0175          ADC #ONE
0176          BEQ MD6
0177          STA ED
0178 MD51     JMP MLT2
```

```
0179 ;
0180 ; IF AN OVERFLOW CONDITION HAS OCCURED, SET RC=0 OR RC=FFFFFFFFFF.
0181 ;
0182 MD59    LDA SA
0183         STA SD
0184 MD6     BIT SD
0185         BVS MD7
0186 MD61    JSR INFIN
0187         RTS
0188 MD7     JSR ZEROA
0189         RTS
0190 ;
0191 ; CORRECT SIGN OF THE MANTISSA OF THE ANSWER IN SD.
0192 ;
0193 MD8     LDA #$80
0194         ORA SD
0195         JMP MD5
0196 DIVEXT  LDA CNTC
0197         STA SB
0198         LDA CNTA
0199         BEQ MD51
0200 DVEXT0  BIT SD
0201         LDA ED
0202         SEC
0203         BVC DVEXT2
0204         ADC #ZERO
0205         BEQ MD6
0206 DVEXT1  STA ED
0207         JMP MLT2
0208 DVEXT2  BEQ DVEXT3
0209         SBC #ONE
0210         JMP DVEXT1
0211 DVEXT3  LDA SD
0212         ORA #$40
0213         STA SD
0214         JMP DVEXT0
0215 ;
0216 ; CORRECT EXPONENT OF ANSWER IN ED
0217 ;
0218 MD9     SEC
0219         SBC #ONE
0220         BEQ MD2
0221         STA ED
0222         JMP MLT2
0223 ;
0224 ;
0225 ;
0226 ;          THIS ROUTINE COMPUTES THE QUOTIENT OF TWO NUMBERS IN
0227 ;          SCIENTIFIC NOTATION. THE ARGUMENTS ARE ASSUMED TO BE IN
0228 ;          BCD AND FORMATTED AS SPECIFIED ABOVE. THE QUOTIENT IS
0229 ;          FORMED BY SUBTRACTING (SB,...,MB+2) FROM
0230 ;          (SC,...,MC+2) AND KEEPING TRACK OF HOW MANY TIMES THIS
0231 ;          SUBTRACTION CAN BE PERFORMED WITHOUT CAUSING A BORROW.
0232 ;          THE ARGUMENTS ARE RA AND RB. THE QUOTIENT IS RC. THE
0233 ;          QUOTIENT IS LEFT JUSTIFIED AND IN SCIENTIFIC NOTATION.
0234 ;
0235 ;
0236 ;
0237 DIVIDE  SED
0238 ;
```

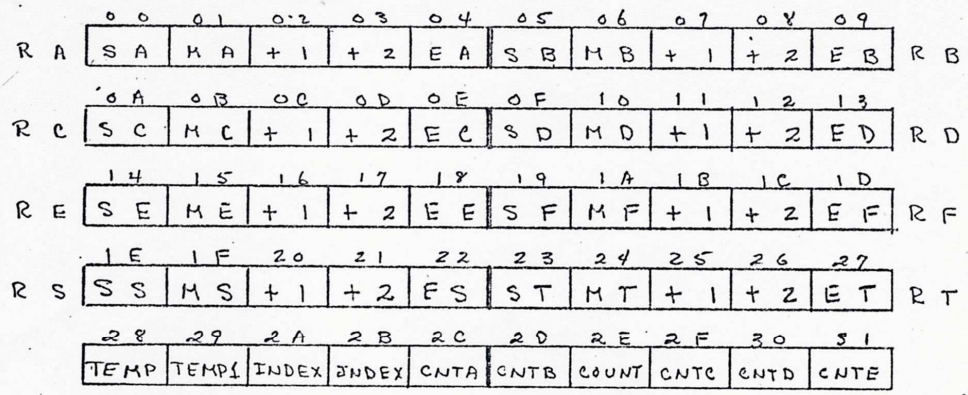
```
0239 ; TEST (MA,...,MA+2) FOR ZERO.
0240 ;
0241         LDX #ONE
0242         JSR ZTEST
0243 ;
0244 ; IF (MA,...,MA+2)=0, SET RC=0.
0245 ;
0246         BCC MD7
0247 ;
0248 ; TEST (MB,...,MB+2) FOR ZERO.
0249 ;
0250         LDX #SIX
0251         JSR ZTEST
0252 ;
0253 ; IF (MB,...,MB+2)=0,SET RC=FF FFFFFF FF.
0254 ;
0255         BCC MD61
0256 ;
0257 ; MOVE RA TO RC.
0258 ;
0259         LDY #AADDR
0260         LDX #CADDR
0261         JSR MVRG
0262 ;
0263 ; STORE SB IN CNTC.
0264 ;
0265         LDA SB
0266         STA CNTC
0267 ;
0268 ; CLEAR SC,SB,AND EC
0269 ;
0270         LDA #ZERO
0271         STA SC
0272         STA SB
0273         STA EC
0274 ;
0275 ; SET RD =0.
0276 ;
0277         LDX #DADDR
0278         JSR CLREG
0279 ;
0280 ; COMPARE (MA,...,MA+2) WITH (MB,...,MB+2) AND STORE THE RESULT IN
0281 ; CNTA.
0282 ;
0283         JSR COMPAB
0284         CLC
0285 ;
0286 ; DIVISION STARTS HERE.
0287 ; COUNT THE NUMBER OF TIMES (SB,...,MB+2) CAN BE SUBTRACTED FROM
0288 ; (SC,...,MC+2).
0289 ;
0290 DIV0    LDA MD+2
0291         ADC #ZERO
0292         STA MD+2
0293 ;
0294 ; SUBTRACT (SB,...,MB+2) FROM (SC,...,MC+2).
0295 ;
0296 DIV1    SEC
0297         LDX #THREE
0298 DIV2    LDA RC,X
```

```
0299          SBC RB,X
0300          STA RC,X
0301          DEX
0302          BPL DIV2
0303 ;
0304 ; IF NO BORROW, INCREMENT MD+2.
0305 ;
0306          BCS DIV0
0307 ;
0308 ; IF BORROW, CORRECT (SC,...MC+2).
0309 ;
0310          LDX #THREE
0311 DIV3     LDA RC,X
0312          ADC RB,X
0313          STA RC,X
0314          DEX
0315          BPL DIV3
0316 ;
0317 ; CHECK TO SEE IF SIX DIGITS OF QUOTIENT HAVE BEEN GENERATED.
0318 ;
0319          LDA MD
0320          AND #$F0
0321 ;
0322 ; IF THE ANSWER TO THE PRECEEDING QUESTION IS YES, PREPARE THE
0323 ; SIGNS AND EXPONENT OF THE ANSWER.
0324 ;
0325          BNE DIV51
0326 ;
0327 ; IF THE ANSWER IS NO, LEFT SHIFT (SC,...,MD+2).
0328 ;
0329          LDY #THREE
0330 DIV4     LDX #EIGHT
0331          CLC
0332 DIV5     ROL RC,X
0333          DEX
0334          BPL DIV5
0335          DEY
0336          BPL DIV4
0337          JMP DIV1
0338 DIV51    JMP DIV6
0339 ;
0340 ;
0341 ;
0342 ; THIS SUBROUTINE COMPARES (MA,...,MA+2) WITH (MB,...,MB+2) AND
0343 ; SETS CNTA=1, IF (MA,...,MA+2)IS LESS THAN
0344 ; (MB,...,MB+2) AND CNTA=0, IF OTHERWISE.
0345 ;
0346 ;
0347 ;
0348 COMPAB LDA #ZERO
0349          STA CNTA
0350          SEC
0351          LDX #TWO
0352 COM1     LDA MA,X
0353          SBC MB,X
0354          DEX
0355          BPL COM1
0356          BCC COM2
0357          RTS
0358 COM2     INC CNTA
```

```
0359          RTS
0360 ;
0361 ;
0362 ;
0363 ; THIS SUBROUTINE EXCHANGES RA WITH RB.
0364 ;
0365 ;
0366 ;
0367 ASB     LDX #FIVE
0368 ASB1    DEX
0369          BMI ASB2
0370          LDA RA,X
0371          LDY RB,X
0372          STA RB,X
0373          STY RA,X
0374          JMP ASB1
0375 ASB2    RTS
0376 ;
0377 ;
0378 ;
0379 ; THIS ROUTINE TESTS (SA+X,...,MA+1+X) FOR ZERO. THE CARRY FLAG IS SET
0380 ; IF ANY BYTE IS NONZERO. OTHERWISE CARRY IS RESET.
0381 ;
0382 ;
0383 ;
0384 ZTEST   CLC
0385          LDA RA,X
0386          ADC MA,X
0387          ADC MA+1,X
0388          BCS ASB2
0389          BEQ ASB2
0390          SEC
0391          RTS
0392 ;
0393 ;
0394 ;
0395 ; THIS ROUTINE SETS A FIVE BYTE BLOCK EQUAL TO ZERO, STARTING AT
0396 ; ADDRESS RA+X.
0397 ;
0398 ;
0399 ;
0400 CLREG   LDY #FOUR
0401 CLRG    LDA #ZERO
0402 CLR1    STA RA,X
0403          INX
0404          DEY
0405          BPL CLR1
0406          RTS
0407 ;
0408 ;
0409 ;
0410 ; THIS ROUTINE MOVES THE FIVE BYTE BLOCK STARTING AT RA+Y
0411 ; TO RA+X.
0412 ;
0413 ;
0414 ;
0415 MVRG    LDA #FOUR
0416 MVNG    STA CNTB
0417 MVC     LDA RA,Y
0418          STA RA,X
```

```
0419          INX
0420          INY
0421          DEC CNTB
0422          BPL MVC
0423          RTS
0424 ;
0425 ;
0426 ;
0427 ; THIS ROUTINE SETS RC=0.
0428 ;
0429 ;
0430 ;
0431 ZEROA     LDA #ZERO
0432 ZEROZ     LDX #FOUR
0433 ZEROY     STA RC,X
0434          DEX
0435          BPL ZEROY
0436          RTS
0437 ;
0438 ;
0439 ;
0440 ; THIS ROUTINE SETS RC=FF FFFFFF FF.
0441 ;
0442 ;
0443 ;
0444 INFIN     LDA #$FF
0445          JMP ZEROZ
0446 ;
0447 ;
0448 ;
0449 ; THIS ROUTINE RIGHT SHIFTS (SC,...,SC+3) BY ONE DIGIT.
0450 ;
0451 ;
0452 ;
0453 RSC       LDX #THREE
0454 RSC1      ASL RC+3
0455          ROL RC+2
0456          ROL RC+1
0457          ROL RC
0458          DEX
0459          BPL RSC1
0460          LDA RC+2
0461          STA RC+3
0462          LDA RC+1
0463          STA RC+2
0464          LDA RC
0465          STA RC+1
0466          LDA #ZERO
0467          STA RC
0468          RTS
0469 ADTAB1   .BYTE $03,$03,$02,$02,$01,$01
0470 ADTAB2   .BYTE $10,$11,$11,$12,$12,$13
0471          .END
*END

*GO
```



NAME	OFFSET FROM RA
AADDR	$00
BADDR	$05
CADDR	$0A
DADDR	$0F
EADDR	$14
FADDR	$19
SADDR	$1E
TADDR	$23

Jan., 1976 REV. 0