ad 11/4/0

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DBKEB-A-D
PRODUCT NAME:	KE11F (PDP-11 FIS) EXERCISER
DATE CREATED:	1-AUG-72
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	KEN CHAPMAN

CONTENTS
----

1. ABSTRACT

This program exercises the KE11F floating point instructions (FADD, FSUB, FMUL, FDIV) with random number patterns. The answers are checked against results obtained using the corresponding FORTRAN software routines. About 200 passes should be run to establish credability.

2. REQUIREMENTS

2.1 Equipment

PDP-11 (KD11A) standard computer with KE11F option

2.2 Storage

The routines use memory locations 0 - 17500. The map at the end of the listings shows the absolute locations of the FORTRAN math routines which were assembled seperately and linked to the main program via LNKX11 on a DECsystem-10.

2.3 Preliminary programs

MAINDEC-11-DBKEA-A KE11F Instruction Tests.

3. LOADING PROCEDURE

Use standard procedure for ABS tapes.

4. STARTING PROCEDURE

4.1 Control switch settings

See 5.1.1 (all down for worst case testing)

4.2 Starting address

The program should always be started at 200.

4.3 Program and/or operator action

1) Load program into memory using ABS loader.
2) Load address 200.
3) Set switches (see 5.1.1) All down for worst case.

4) Press start.
5) The program will loop and bell will ring once every pass.

5. OPERATING PROCEDURE

5.1 Operational switch settings

```
SW<15> =  1 ......  HALT ON ERROR
SW<14> =  1 ......  SCOPE LOOP
SW<13> =  1 ......  INHIBIT PRINTOUT
SW<12> =  1 ......  INHIBIT TRACE TRAPPING
SW<11> =  1 ......  INHIBIT ITERATIONS OF SUBTEST
SW<10> =  1 ......  BELL ON ERROR
          Ø ......  BELL ON PASS COMPLETE
SW<Ø9> =  1 ......  LOOP ON ERROR
SW<Ø8> =  1 ......  LOOP ON TEST IN SW<6:Ø>
SW<Ø7> =  1 ......  INPUT DATA FROM THE TELETYPE
```

Caution! SW<8:Ø> are also used for ROM word match with KM11 maintenance card.

5.2 Subroutine Abstracts

5.2.1 TYPIN

If SW<7> is on a Ø, the program calculates a pseudo-random number to be used as input data. If SW<7> is on a 1, the program will ask for input data from the teletype at the beginning of each pass. The same data is used with all instructions (FADD, FSUB, FMUL, FDIV) for the entire pass. If SW<7> is put down after entering the data entry routine, that data is used as the starting numbers for the random number generator.

The input format is:

```
          Type input data:
          A1:  NNNNNN
          A2:  NNNNNN
          B1:  NNNNNN
          B2:  NNNNNN
```

Where:

```
          A1 = left word of first argument
          A2 = right word of first argument
          B1 = left word of second argument
          B2 = right word of second argument
```

I.e. A1,A2(+,-,*,/)B1,B2 = answer

NNNNNN = data typed by the operator

A1, A2, B1, and B2 must be 16 bit left justified octal numbers.

E.G.

```
      42 = 000042
  200000 = not accepted (17 bits)
    4812 = not accepted (8 is not octal)
```

They are assumed to be in floating point format, I.E. bit 15 of A1 and B1 are the sign bits, bits 7-14 of A1 and B1 are the exponents (excess 128 format) and the rest (bits 0-6 of A1 and B1 and all of A2 and B2) form the mantissa (normalized) less the hidden bit. For more information read the maintenance manual. A1, A2, B1, and B2 are put into RAND.A, RAND.B, RAND.C, and RAND.D respectively.

5.2.2 FORTAN

This routine make use of "polish mode" to link the FORTRAN MATH PACKAGE ROUTINES TO CALCULATE THE EXPECTED RESULT.

```
LOCATIONS  $ADD1, $ADD2  contain addition answer.
Locations  $SUB1, $SUB2  contain subtract answer.
Locations  $MUL1, $MUL2  contain multiply answer.
Locations  $DIV1, $DIV2  contain divide answer.
```

If a floating error occurrs (overflow, underflow, or divide by zero), these answers are meaningless. The locations $ADDPS, $SUBPS, $MULPS, or $DIVPS contains 340 and $ADDER, $SUBER, $MULER, or $DIVER, contain the conditions codes of the error.

5.2.3 SCOPE

This subroutine call is placed between each subtest in the test section. It records the starting address of each subtest as it is being entered in location "LAD$". If a scope loop is requested, the current subtest will be looped upon. SW<11> on a 1 inhibits iteration of subtests. The contents of LAD$ may be used to determine the last subtest successfully completed.

5.2.4 HLT

This routine prints out an error message (See 6.1). To inhibit typeouts, put SW<13> on a 1.

5.2.5 TRTRAP

If SW<12> is on a 0, the T-bit will be set on alternate passes. When the T-bit is set, the processor traps after each instruction. The first instruction executed upon trapping is an "RTT" which returns to the interrupted sequence of instructions. This sequence is continued until the end of the program is reached.

5.2.6 TRAPCATCHER

A ".+2" - "HALT" sequence is repeated from 0 - 776 to catch any unexpected traps. Thus any unexpected traps or interupts will HALT at the vector + 2.

5.2.7 FLOATING POINT TRAP (to 244)

All tests set the floating point trap vector (244) to point to the instruction following the floating point instruction. Thus, whether or not a trap occurs is only detected if the data or the stack pointer(s) are wrong.

6. ERRORS

6.1 Error printout

There are two formats for error typeout; one for normal numbers and one for floating errors (overflow, underflow and divide by zero).

6.1.1 The normal format (when no floating point error is indicated) is as follows:

```
AAAAAA  MMMMMM,MMMMMM S MMMMMM,MMMMMM
             PSW   SP       ANSWER
EXPECT:      NNN  NNN  NNNNNN,NNNNNN
GOT:         NNN  NNN  NNNNNN,NNNNNN
```

Where:
AAAAAA ==> PC of HLT instruction
MMMMMM ==> Input data (RAND.A, RAND.B, RAND.C, RAND.D)
S   ==> type of operation being tested (+,-,*, or /)

```
NNNNNN ==> results
           PSW =     processor status word
           SP =      stack pointer (not necessarily R6)
           ANSWER =  resulting answer off the stack
```

6.1.2 When a floating point error is indicated (overflow, underflow, or divide by zero) the format is as follows:

```
AAAAAA  MMMMMM,MMMMMM S MMMMMM,MMMMMM
       PSW   SP   ANS1   ANS2   ANS3   ANS4   ANS5   ANS6
EXPECT:  NNN  NNN  NNNNNN NNNNNN NNNNNN NNNNNN NNNNNN NNNNNN
GOT:     NNN  NNN  NNNNNN NNNNNN NNNNNN NNNNNN NNNNNN NNNNNN
```

```
Where:
AAAAAA ==> PC of HLT instruction
MMMMMM ==> input data (RAND.A, RAND.B, RAND.C, RAND.D)
S      ==> type of operation being tested (+,-,*, or /)
NNNNNN ==> results
           PSW =     processor status word
           SP =      stack pointer (not necessarily R6)
           ANS1 =    PC of interupted instruction (should be
                     FIS)
           ANS2 =    PSW at interupt time
           ANS3 =    input data (RAND.C)
           ANS4 =      "    "   (RAND.D)
           ANS5 =      "    "   (RAND.A)
           ANS6 =      "    "   (RAND.B)
```

To find the failing test, look at the listing above the address typed.

6.2 Error recovery

Restart at 200

6.3 Error count

An error count is kept in "ERRORS" (LOC 1002). It is cleared by restarting at 200.

7. RESTRICTIONS

None

8, MISCELLANEOUS

8.1 Execution time

A bell will ring within 5 seconds with all switches down, More than 200 passes should be run to insure a wide varieey of number patterns.

8.2 Stack Pointer

Stack is initally set to 604

8.3 Pass counter

A 32 bit (2 words) pass count is kept in "PCNT" (LOC 1004,1006). It is cleared by restarting at 200,

8.4 Power Fail

Each test can be power failed with no errors, To use, start the test as usual and power down then up at any time, The program should type "POWER" and continue to run from where power fail interupted with no other typeouts,

9, PROGRAM DESCRIPTION

This program tests all the FIS Instructions on the KE11F using all registers except 7 for the "stack pointer", The program has many subtests (the code between 2 SCOPE statements) which are run 256 times before continuing to the next, SW<11> on a 1 causes each subtest to be run only once, The address ICNT (LOC 1000) contains the iteration count in the left byte and the test number in the right byte, All the subtests should be run sequentially by starting at 200 not by starting at the beginning of the subtest, To loop on a particular subtest, put the test number (see listing) in SW<6:0> of the switch register and SW<8> on a 1, This test will be looped upon until SW<8> is put on a 0 or the right byte is changed, If the test is non-existant, the program will be run as usual,

The FORTRAN math routines, which are used to calculate the correct answers, were taken unmodified from the PDP-11 FORTRAN package and assembled as seperate modules, They were linked to the main programs via LNKX11 on a DECsystem-10 which produces a binary tape in the normal absolute format, Thus, the program loads and runs just like any other diagnostic program,

```
                                    .TITLE  MAINDEC-11-DBKEB-A      KE11F (PDP-11 FIS) EXERCISER,
        000000                      .ASECT
                                    .GLOBL $ADR,$SBR,$MLR,$DVR,$ERR,$ERRA

                            ;COPYRIGHT 1972, DIGITAL EQUIPMENT CORP., MAYNARD, MASS
                            ;PROGRAM BY KEN CHAPMAN
                            .REM;

                                    SWITCH          USE
                                    -------         ---
                                    7               TTY DATA INPUT
                                    8               LOOP ON TEST IN SW<6:0>
                                    9               LOOP ON ERROR
                                    10              0=BELL ON PASS COMPLETED
                                                    1=BELL ON ERROR
                                    11              INHIBIT ITERATIONS
                                    12              INHIBIT TRACE TRAP
                                    13              INHIBIT ERROR TYPEOUTS
                                    14              LOOP ON TEST
                                    15              HALT ON ERROR

                            ERROR MESSAGE FORMATS;
                            1,      WHEN NO FLOATING POINT ERROR IS INDICATED

                            AAAAAA    MMMMMM,MMMMMM S MMMMMM,MMMMMM
                                       PSW   SP      ANSWER
                            EXPECT:    NNN   NNN  NNNNNN,NNNNNN
                            GOT:       NNN   NNN  NNNNNN,NNNNNN

                            WHERE;
                            AAAAAA ==> PC OF HLT INSTRUCTION
                            MMMMMM ==> INPUT DATA (RAND,A, RAND,B, RAND,C, RAND,D)
                            S      ==> TYPE OF OPERATION BEING TESTED (+,-,*, OR /)
                            NNN    ==> RESULTS
                                    PSW =   PROCESSOR STATUS WORD
                                    SP =    STACK POINTER (NOT NECESSARILY R6)
                                    ANSWER= RESULTING ANSWER OFF THE STACK

                            2,      WHEN A FLOATING POINT ERROR IS INDICATED (OVERFLOW, UNDERFLOW,
                                            OR DIVIDE BY ZERO);
                            AAAAAA    MMMMMM,MMMMMM S MMMMMM,MMMMMM
                                       PSW   SP    ANS1     ANS2     ANS3     ANS4     ANS5     ANS6
                            EXPECT:    NNN   NNN  NNNNNN   NNNNNN   NNNNNN   NNNNNN   NNNNNN   NNNNNN
                            GOT:       NNN   NNN  NNNNNN   NNNNNN   NNNNNN   NNNNNN   NNNNNN   NNNNNN

                            WHERE;
                            AAAAAA, MMMMMM, S, NNN, PSW, AND SP ARE THE SAME AS ABOVE,
                                    ANS1 =  PC OF INTERUPTED INSTRUCTION (SHOULD BE FIS)
                                    ANS2 =  PSW AT INTERUPT TIME
                                    ANS3 =  INPUT DATA (RAND,C)
                                    ANS4 =    "    "   (RAND,D)
                                    ANS5 =    "    "   (RAND,A)
                                    ANS6 =    "    "   (RAND,B);
        104400                  SCOPE=  TRAP
        104000                  HLT=    EMT
        000004                  TYPE=   IOT
        177776                  PS=     177776
        177570                  SWR=    177570
        177570                  DISPLAY=SWR
        000007                  BELL=   7
        000000                  R0=     %0
        000001                  R1=     %1
        000002                  R2=     %2
        000003                  R3=     %3
        000004                  R4=     %4
        000005                  R5=     %5
        000005                  TTY=    %5
        000006                  SP=     %6
        000007                  PC=     %7
        100000                  SW15=   100000
        040000                  SW14=    40000
        020000                  SW13=    20000
        010000                  SW12=    10000
        004000                  SW11=     4000
        002000                  SW10=     2000
        001000                  SW09=     1000
        000400                  SW08=      400
        000001                  BIT0 =  000001
        000002                  BIT1 =  000002
        000004                  BIT2 =  000004
        000010                  BIT3 =  000010
        000020                  BIT4 =  000020
        000040                  BIT5 =  000040
        000100                  BIT6 =  000100
        000200                  BIT7 =  000200
        000400                  BIT8 =  000400
        001000                  BIT9 =  001000
        002000                  BIT10 = 002000
        004000                  BIT11 = 004000
        010000                  BIT12 = 010000
        020000                  BIT13 = 020000
        040000                  BIT14 = 040000
        100000                  BIT15 = 100000
        000000                  LEVEL0 = 000
        000040                  LEVEL1 = 040
        000100                  LEVEL2 = 100
        000140                  LEVEL3 = 140
        000200                  LEVEL4 = 200
        000240                  LEVEL5 = 240
        000300                  LEVEL6 = 300
        000340                  LEVEL7 = 340
```

```

                000000                  .=      0                       ;TRAP CATCHER FROM 0 - 776

                000200                  .=      200

        000200  000167  000604                  JMP     BEGIN           ;JUMP TO STARTING ADDRESS OF PROGRAM

                000204                  .=      204
        000204  000167  000736                  JMP     START           ;RESTART ADDRESS

                000600                  .=      600

                                        ;THE FOLLOWING LOCATIONS ARE USED FOR THE STACKS.  R6 IS INITIALLY SET
                                        ;       TO 604 (STACK0), AS ARE THE OTHER REGISTERS (R0 THRU R5) WHEN
                                        ;       THEY ARE TO BE USED AS THE FLOATING POINT STACK POINTER.
                                        ;       THE DATA IS PUT DIRECTLY ONTO THE STACK, NOT BY PUSHES.
                                        ;       IF NO ERROR OCCURES THE STACK POINTER (ANY REGISTER) IS POINTING
                                        ;       TO 610 (ANS1).  IF AN ERROR OCCURES, R6 IS POINTING TO 604,
                                        ;       SO THE TRAP PUTS THE RETURN ADDRESS AND PS IN 600 (STK1)
                                        ;       AND 602 (STK2) RESPECTIVELY.

        000600  000000                  STK1:   0
        000602  000000                  STK2:   0
        000604  000000                  STK3:   STACK0: 0
        000606  000000                  STK4:   STACK2: 0
        000610  000000                  STK5:   STACK4: ANS1:   0
        000612  000000                  STK6:   STACK6: ANS2:   0
        000614  000000                  $PSW:   0
        000616  000000                  $SP:    0

        000620  000000                  RAND.A: 0
        000622  000000                  RAND.B: 0
        000624  000000                  RAND.C: 0
        000626  000000                  RAND.D: 0

        000630  000000                  $ADDPS: 0
        000632  000000                  $ADD1:  0
        000634  000000                  $ADD2:  0
        000636  000000                  $ADDER: 0

        000640  000000                  $SUBPS: 0
        000642  000000                  $SUB1:  0
        000644  000000                  $SUB2:  0
        000646  000000                  $SUBER: 0

        000650  000000                  $MULPS: 0
        000652  000000                  $MUL1:  0
        000654  000000                  $MUL2:  0
        000656  000000                  $MULER: 0

        000660  000000                  $DIVPS: 0
        000662  000000                  $DIV1:  0
        000664  000000                  $DIV2:  0
        000666  000000                  $DIVER: 0
```

```
156
157  000670  000000                    SAVSTK: 0
158  000672  000000                    RNDFLG: 0                               ;FOR FLAGS TO KEEP TRACK OF ROUNDING
159
160
161  000674  105367  177726            RAND4$: DECB    RAND.D                  ;INSURE ALL ZEROES WORKS
162  000700  066767  177716  177712            ADD     RAND.B, RAND.A
163  000706  005567  177714                    ADC     RAND.D
164  000712  066767  177706  177702            ADD     RAND.C, RAND.B
165  000720  005567  177700                    ADC     RAND.C
166  000724  066767  177676  177672            ADD     RAND.D, RAND.C
167  000732  005567  177664                    ADC     RAND.B
168  000736  066767  177656  177662            ADD     RAND.A, RAND.D
169  000744  005567  177650                    ADC     RAND.A
170  000750  000207                            RTS     PC
171
172
173  000752  000006                    YESRT:  RTT                             ;TRACE TRAP SERVICE ROUTINE
174
175  000754  104000                    FISTRP: HLT                             ;ERRONIOUS FIS TRAP
176  000756  000002                            RTI
177
```

```
                001000                          .=      1000

        001000  000000                          ICNT:   0                       ;ITTERATION COUNT (HI BYTE); TEST # (LO BYTE)
        001002  000000                          ERRORS: 0                       ;ERROR COUNT LOCATION
        001004  000000  000000                  PCNT:   0,0                     ;PASS COUNT LOCATION

        001010  012706  000604                  BEGIN:  MOV     #STACK0,SP      ;SET UP STACK
        001014  012737  000752  000014                  MOV     #YESRT,@#14     ;SET UP TRACE TRAP
        001022  012700  000020                          MOV     #20,R0
        001026  012720  015256                          MOV     #.IOT,(R0)+     ;SET UP IOT VECTOR
        001032  012720  000340                          MOV     #340,(R0)+
        001036  012720  015536                          MOV     #PDOWNS,(R0)+   ;SET UP POWER FAIL VECTOR
        001042  012720  000340                          MOV     #340,(R0)+
        001046  012720  014020                          MOV     #HLTS,(R0)+     ;SET EMT VECTOR
        001052  012720  000340                          MOV     #340,(R0)+
        001056  012720  013644                          MOV     #SCOPES,(R0)+   ;SET TRAP VECTOR
        001062  012720  000340                          MOV     #340,   (R0)+
        001066  012737  000754  000244                  MOV     #FISTRP,@#244   ;SET UP FIS VECTOR
        001074  012737  000340  000246                  MOV     #340,@#246
        001102  012767  123456  177510                  MOV     #123456,RAND.A  ;PRIME THE RANDOM NUMBER GENERATOR
        001110  012767  107654  177504                  MOV     #107654,RAND.B
        001116  012767  070707  177500                  MOV     #070707,RAND.C
        001124  012767  125252  177474                  MOV     #125252,RAND.D
        001132  005067  177644                          CLR     ERRORS          ;CLEAR ERROR COUNTER
        001136  005067  177642                          CLR     PCNT            ;CLEAR PASS COUNTER
        001142  005067  177640                          CLR     PCNT+2
        001146  012706  000604                  START:  MOV     #STACK0,SP      ;SET UP STACK
        001152  012737  000140  177776                  MOV     #140,   @#PS    ;SET UP PROCESSOR STATUS
        001160  005067  177614                          CLR     ICNT
        001164  005067  012622                          CLR     LADS
        001170  005067  177476                          CLR     RNDFLG          ;CLEAR THE ROUNDING FLAGS
        001174  105737  177570                          TSTB    @#SWR           ;CHECK FOR TTY INPUT
        001200  100403                                  BMI     TYPIN
        001202  004767  177466                          JSR     PC,RAND4S
        001206  000464                                  BR      FORTAN          ;BRANCH TO ROUTINE TO CALCULATE ANSWERS

                                                ;THE FOLLOWING ROUTINE ACCEPTS DATA FROM THE TELETYPE.
                                                ;       THE FORMAT IS FIXED:    A1,A2 (+,-,*,/) B1,B2.
                                                ;       THE PROGRAM ASKES FOR ONE ARGUMENT AT A TIME, AND RE-ASKES
                                                ;       WHEN INVALID DATA IS ENTERED.

        001210  000004  001214                  TYPIN:  TYPE,   .+2
        001214  005015  054524  042520                  .ASCIZ  <15><12>"TYPE INPUT DATA:"<15><12>
        001222  044440  050116  052125
        001230  042040  052101  035101
        001236  005015     000
                001242                                  .EVEN
        001242  000004  001246                  1$:     TYPE,   .+2
        001246  030501  020072  000040                  .ASCIZ  "A1:  "
        001254  004567  011502                          JSR     R5,     READIN  ;ACCEPT FIRST ARGUEMENT FROM THE TTY
        001260  000620                                  RAND.A
        001262  103752                                  BCS     TYPIN
        001264  000004  001270                  2$:     TYPE,   .+2
```

```
001270  031101  020072  000040           .ASCIZ  "A2:  "
001276  004567  011460                   JSR     R5,     READIN  ;ACCEPT SECOND ARGUEMENT FROM THE TTY
001302  000622                           RAND,B
001304  103767                           BCS     2$
001306  001340                           BNE     TYPIN
001310  000004  001314           3$:     TYPE,   ,.+2
001314  030502  020072  000040           .ASCIZ  "B1:  "
001322  004567  011434                   JSR     R5,     READIN  ;ACCEPT THIRD ARGUEMENT FROM THE TTY
001326  000624                           RAND,C
001330  103767                           BCS     3$
001332  001326                           BNE     TYPIN
001334  000004  001340           4$:     TYPE,   ,.+2
001340  031102  020072  000040           .ASCIZ  "B2:  "
001346  004567  011410                   JSR     R5,     READIN  ;ACCEPT FOURTH ARGUEMENT FROM THE TTY
001352  000626                           RAND,D
001354  103767                           BCS     4$
001356  001314                           BNE     TYPIN

001360  005067  177244           FORTAN: CLR     $ADDPS          ;CLEAR ALL THE PS SAVE LOCATIONS
001364  005067  177250                   CLR     $SUBPS
001370  005067  177254                   CLR     $MULPS
001374  005067  177260                   CLR     $DIVPS

001400  004467  011460                   JSR     R4,     $POLSH  ;ENTER POLISH MODE
001404  013044                           $PUSH                   ;PUSH THE DATA ONTO THE STACK
001406  000000G                          $ADR                    ;FORTAN ADD ROUTINE
001410  013066                           $POPAD                  ;SAVE THE ADD ANSWERS
001412  013044                           $PUSH                   ;PUSH THE DATA ONTO THE STACK
001414  000000G                          $SBR                    ;FORTRAN SUBTRACT ROUTINE
001416  013144                           $POPSB                  ;SAVE THE SUBTRACT ANSWERS
001420  013044                           $PUSH                   ;PUSH THE DATA ONTO THE STACK
001422  000000G                          $MLR                    ;FORTRAN MULTIPLY ROUTINE
001424  013222                           $POPML                  ;SAVE THE MULTIPLY ANSWERS
001426  013044                           $PUSH                   ;PUSH THE DATA ONTO THE STACK
001430  000000G                          $DVR                    ;FORTRAN DIVIDE ROUTINE
001432  013300                           $POPDV                  ;SAVE THE DIVIDE ANSWERS
001434  013412                           $EXIT                   ;EXIT POLISH MODE

001436  104400                           SCOPE
```

```

                                        ;*******************************************************************
                                        ;TEST 1:        EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
                                        ;       RAND,A,RAND,B + RAND,C,RAND,D = ANS1,ANS2
                                        ;       STACK POINTER = RØ
                                        ;*******************************************************************

 ØØ144Ø  Ø127ØØ  ØØØ6Ø4          TST1:  MOV     #STACKØ,RØ      ;SET UP THE STACK POINTER
 ØØ1444  ØØ4767  Ø1213Ø                 JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

 ØØ145Ø  ØØØ24Ø                         NOP
 ØØ1452  075ØØØ                         FADD+   RØ              ;FLOATING ADD ON THE RØ STACK

 ØØ1454  Ø13767  177776  177132 1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
 ØØ1462  Ø1ØØ67  17713Ø                 MOV     RØ,     $SP     ;SAVE THE STACK POINTER
 ØØ1466  Ø26767  177136  17712Ø 6$:     CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
 ØØ1474  ØØ1Ø23                         BNE     4$              ;GO CHECK FOR ROUNDING ERROR

 ØØ1476  1Ø5767  177112                 TSTB    $PSW            ;CHECK FOR ERROR
 ØØ15Ø2  1ØØ464                         BMI     2$              ;BRANCH IF ERROR

 ØØ15Ø4  Ø12767  ØØØ61Ø  177156         MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
 ØØ1512  Ø26767  177152  177Ø76         CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
 ØØ152Ø  ØØ14Ø1                         BEQ     .+4             ;BRANCH IF OK
 ØØ1522  1Ø4ØØØ                         HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

 ØØ1524  Ø26767  1771Ø2  177Ø56         CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
 ØØ1532  ØØ1ØØ4                         BNE     4$
 ØØ1534  Ø26767  177Ø74  177Ø5Ø         CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
 ØØ1542  ØØ1515                         BEQ     3$
 ØØ1544  Ø32767  ØØØØØ2  17712Ø 4$:     BIT     #BIT1,  RNDFLG  ;CHECK THE ROUNDING FLAG
 ØØ1552  ØØ1Ø22                         BNE     5$
 ØØ1554  Ø52767  ØØØØØ2  17711Ø         BIS     #BIT1,  RNDFLG  ;SET ROUNDING FLAG
 ØØ1562  Ø62767  ØØØØØ1  177Ø44         ADD     #1,     $ADD2   ;INCREMENT FORTRAN ANSWER
 ØØ157Ø  ØØ5567  177Ø36                 ADC     $ADD1           ;ADD CARRY
 ØØ1574  1Ø2334                         BVC     6$              ;BRANCH IF NO OVERFLOW
 ØØ1576  ØØØ257                         CCC                     ;CLEAR ALL CONDITION CODES
 ØØ16ØØ  ØØØ262                         SEV                     ;SET V-BIT
 ØØ16Ø2  Ø13767  177776  177Ø26         MOV     @#PS,   $ADDER  ;SET [illegible] PSW FOR OVERFLOW
 ØØ161Ø  Ø12767  ØØØ34Ø  177Ø12         MOV     #34Ø,   $ADDPS  ;SET [illegible] TRAP PSW
 ØØ1616  ØØØ723                         BR      6$              ;TRY [illegible] AGAIN

 ØØ162Ø  132767  ØØØØØ2  177Ø45 5$:     BITB    #BIT1,  RNDFLG+1 ;CHECK "DEROUNDING" FLAG
 ØØ1626  ØØ1Ø1Ø                         BNE     7$              ;BRANCH IF SET
 ØØ163Ø  152767  ØØØØØ2  177Ø35         BISB    #BIT1,  RNDFLG+1 ;SET "DEROUNDING" FLAG
 ØØ1636  162767  ØØØØØ1  17677Ø         SUB     #1,     $ADD2   ;RESTORE ORIGINAL ANSWER
 ØØ1644  ØØ5667  176762                 SBC     $ADD1           ;SUBTRACT CARRY
 ØØ165Ø  1Ø4ØØØ                 7$:     HLT                     ;WRONG PSW OR ANSWER

 ØØ1652  ØØØ451                         BR      3$

 ØØ1654  Ø12767  ØØØ6Ø4  177ØØ6 2$:     MOV     #STACKØ,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
 ØØ1662  Ø26767  177ØØ2  176726         CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
 ØØ167Ø  ØØ14Ø1                         BEQ     .+4             ;BRANCH IF OK
```

```
001672  104000                             HLT                             ;STACK POINTER FOULED UP

001674  022767  001454  176676             CMP     #1$,     STK1           ;CHECK THE RTI ADDRESS ON THE STACK
001702  001401                             BEQ     .+4                     ;BRANCH IF OK
001704  104000                             HLT                             ;RTI ADDRESS NOT EQUAL TO #1$

001706  026767  176724  176666             CMP     $ADDER,  STK2           ;CHECK THE PSW ON THE STACK
001714  001401                             BEQ     .+4                     ;BRANCH IF OK
001716  104000                             HLT                             ;RTI PSW NOT EQUAL TO 200

001720  026767  176700  176656             CMP     RAND.C,  STK3           ;CHECK THE DATA ON THE STACK
001726  001401                             BEQ     .+4                     ;BRANCH IF OK
001730  104000                             HLT                             ;STK3 NOT EQUAL TO RAND.C

001732  026767  176670  176646             CMP     RAND.D,  STK4           ;CHECK THE DATA ON THE STACK
001740  001401                             BEQ     .+4                     ;BRANCH IF OK
001742  104000                             HLT                             ;STK4 NOT EQUAL TO RAND.D

001744  026767  176650  176636             CMP     RAND.A,  STK5           ;CHECK THE DATA ON THE STACK
001752  001401                             BEQ     .+4                     ;BRANCH IF OK
001754  104000                             HLT                             ;STK5 NOT EQUAL TO RAND.A

001756  026767  176640  176626             CMP     RAND.B,  STK6           ;CHECK THE DATA ON THE STACK
001764  001401                             BEQ     .+4                     ;BRANCH IF OK
001766  104000                             HLT                             ;STK6 NOT EQUAL TO RAND.B

001770  012716  001776                     MOV     #3$,     (SP)           ;RESET THE STACK
001774  000002                             RTI                             ;RESTORE THE STATUS (T-BIT)

001776  104400                     3$:     SCOPE

```

```

                                            ;*****************************************************************
                                            ;TEST 2:         EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
                                            ;       RAND,A,RAND,B - RAND,C,RAND,D = ANS1,ANS2
                                            ;       STACK POINTER = R1
                                            ;*****************************************************************

ØØ2ØØØ  Ø127Ø1  ØØØ6Ø4           TST2:  MOV     #STACKØ,R1      ;SET UP THE STACK POINTER
ØØ2ØØ4  ØØ4767  Ø1157Ø                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

ØØ2Ø1Ø  ØØØ24Ø                          NOP
ØØ2Ø12  Ø75Ø11                          FSUB+   R1              ;FLOATING SUBTRACT ON THE R1 STACK

ØØ2Ø14  Ø13767  177776  176572   1$:    MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
ØØ2Ø22  Ø1Ø167  17657Ø                  MOV     R1,     $SP     ;SAVE THE STACK POINTER
ØØ2Ø26  Ø26767  1766Ø6  17656Ø   6$:    CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
ØØ2Ø34  ØØ1Ø23                          BNE     4$              ;GO CHECK FOR ROUNDING ERROR

ØØ2Ø36  1Ø5767  176552                  TSTB    $PSW            ;CHECK FOR ERROR
ØØ2Ø42  1ØØ464                          BMI     2$              ;BRANCH IF ERROR

ØØ2Ø44  Ø12767  ØØØ61Ø  176616          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
ØØ2Ø52  Ø26767  176612  176536          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
ØØ2Ø6Ø  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ2Ø62  1Ø4ØØØ                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

ØØ2Ø64  Ø26767  176552  176516          CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
ØØ2Ø72  ØØ1ØØ4                          BNE     4$
ØØ2Ø74  Ø26767  176544  17651Ø          CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
ØØ21Ø2  ØØ1515                          BEQ     3$
ØØ21Ø4  Ø32767  ØØØØØ4  17656Ø   4$:    BIT     #BIT2,  RNDFLG  ;CHECK THE ROUNDING FLAG
ØØ2112  ØØ1Ø22                          BNE     5$
ØØ2114  Ø52767  ØØØØØ4  17655Ø          BIS     #BIT2,  RNDFLG  ;SET ROUNDING FLAG
ØØ2122  Ø62767  ØØØØØ1  176514          ADD     #1,     $SUB2   ;INCREMENT FORTRAN ANSWER
ØØ213Ø  ØØ5567  1765Ø6                  ADC     $SUB1           ;ADD CARRY
ØØ2134  1Ø2334                          BVC     6$              ;BRANCH IF NO OVERFLOW
ØØ2136  ØØØ257                          CCC                     ;CLEAR ALL CONDITION CODES
ØØ214Ø  ØØØ262                          SEV                     ;SET V-BIT
ØØ2142  Ø13767  177776  176476          MOV     @#PS,   $SUBER  ;SET UP PSW FOR OVERFLOW
ØØ215Ø  Ø12767  ØØØ34Ø  176462          MOV     #34Ø,   $SUBPS  ;SET UP TRAP PSW
ØØ2156  ØØØ723                          BR      6$              ;TRY IT AGAIN

ØØ216Ø  132767  ØØØØØ4  1765Ø5   5$:    BITB    #BIT2,  RNDFLG+1 ;CHECK "DEROUNDING" FLAG
ØØ2166  ØØ1Ø1Ø                          BNE     7$              ;BRANCH IF SET
ØØ217Ø  152767  ØØØØØ4  176475          BISB    #BIT2,  RNDFLG+1 ;SET "DEROUNDING" FLAG
ØØ2176  162767  ØØØØØ1  17644Ø          SUB     #1,     $SUB2   ;RESTORE ORIGINAL ANSWER
ØØ22Ø4  ØØ5667  176432                  SBC     $SUB1           ;SUBTRACT CARRY
ØØ221Ø  1Ø4ØØØ                   7$:    HLT                     ;WRONG PSW OR ANSWER

ØØ2212  ØØØ451                          BR      3$

ØØ2214  Ø12767  ØØØ6Ø4  176446   2$:    MOV     #STACKØ,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
ØØ2222  Ø26767  176442  176366          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
ØØ223Ø  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
```

```
002232  104000                          HLT                          ;STACK POINTER FOULED UP

002234  022767  002014  176336          CMP     #1$,     STK1        ;CHECK THE RTI ADDRESS ON THE STACK
002242  001401                          BEQ     .+4                  ;BRANCH IF OK
002244  104000                          HLT                          ;RTI ADDRESS NOT EQUAL TO #1$

002246  026767  176374  176326          CMP     $SUBER, STK2         ;CHECK THE PSW ON THE STACK
002254  001401                          BEQ     .+4                  ;BRANCH IF OK
002256  104000                          HLT                          ;RTI PSW NOT EQUAL TO 200

002260  026767  176340  176316          CMP     RAND.C, STK3         ;CHECK THE DATA ON THE STACK
002266  001401                          BEQ     .+4                  ;BRANCH IF OK
002270  104000                          HLT                          ;STK3 NOT EQUAL TO RAND.C

002272  026767  176330  176306          CMP     RAND.D, STK4         ;CHECK THE DATA ON THE STACK
002300  001401                          BEQ     .+4                  ;BRANCH IF OK
002302  104000                          HLT                          ;STK4 NOT EQUAL TO RAND.D

002304  026767  176310  176276          CMP     RAND.A, STK5         ;CHECK THE DATA ON THE STACK
002312  001401                          BEQ     .+4                  ;BRANCH IF OK
002314  104000                          HLT                          ;STK5 NOT EQUAL TO RAND.A

002316  026767  176300  176266          CMP     RAND.B, STK6         ;CHECK THE DATA ON THE STACK
002324  001401                          BEQ     .+4                  ;BRANCH IF OK
002326  104000                          HLT                          ;STK6 NOT EQUAL TO RAND.B

002330  012716  002336                  MOV     #3$,     (SP)        ;RESET THE STACK
002334  000002                          RTI                          ;RESTORE THE STATUS (T-BIT)

002336  104400                  3$:     SCOPE

```

```
                                        ;***************************************************************
                                        ;TEST 3:        EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
                                        ;       RAND,A,RAND,B * RAND,C,RAND,D = ANS1,ANS2
                                        ;       STACK POINTER = R2
                                        ;***************************************************************

002340  012702  000604          TST3:   MOV     #STACK0,R2      ;SET UP THE STACK POINTER
002344  004767  011230                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

002350  000240                          NOP
002352  075022                          FMUL+   R2              ;FLOATING MULTIPLY ON THE R2 STACK

002354  013767  177776  176232  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
002362  010267  176230                  MOV     R2,     $SP     ;SAVE THE STACK POINTER
002366  026767  176256  176220  6$:     CMP     $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
002374  001023                          BNE     4$              ;GO CHECK FOR ROUNDING ERROR

002376  105767  176212                  TSTB    $PSW            ;CHECK FOR ERROR
002402  100464                          BMI     2$              ;BRANCH IF ERROR

002404  012767  000610  176256          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
002412  026767  176252  176176          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
002420  001401                          BEQ     .+4             ;BRANCH IF OK
002422  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

002424  026767  176222  176156          CMP     $MUL1,  ANS1    ;CHECK THE ANSWER
002432  001004                          BNE     4$
002434  026767  176214  176150          CMP     $MUL2,  ANS2    ;CHECK THE ANSWER
002442  001515                          BEQ     3$
002444  032767  000010  176220  4$:     BIT     #BIT3,  RNDFLG  ;CHECK THE ROUNDING FLAG
002452  001022                          BNE     5$
002454  052767  000010  176210          BIS     #BIT3,  RNDFLG  ;SET ROUNDING FLAG
002462  062767  000001  176164          ADD     #1,     $MUL2   ;INCREMENT FORTRAN ANSWER
002470  005567  176156                  ADC     $MUL1           ;ADD CARRY
002474  102334                          BVC     6$              ;BRANCH IF NO OVERFLOW
002476  000257                          CCC                     ;CLEAR ALL CONDITION CODES
002500  000262                          SEV                     ;SET V-BIT
002502  013767  177776  176146          MOV     @#PS,   $MULER  ;SET UP PSW FOR OVERFLOW
002510  012767  000340  176132          MOV     #340,   $MULPS  ;SET UP TRAP PSW
002516  000723                          BR      6$              ;TRY IT AGAIN

002520  132767  000010  176145  5$:     BITB    #BIT3,  RNDFLG+1 ;CHECK "DEROUNDING" FLAG
002526  001010                          BNE     7$              ;BRANCH IF SET
002530  152767  000010  176135          BISB    #BIT3,  RNDFLG+1 ;SET "DEROUNDING" FLAG
002536  162767  000001  176110          SUB     #1,     $MUL2   ;RESTORE ORIGINAL ANSWER
002544  005667  176102                  SBC     $MUL1           ;SUBTRACT CARRY
002550  104000                  7$:     HLT                     ;WRONG PSW OR ANSWER

002552  000451                          BR      3$

002554  012767  000604  176106  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
002562  026767  176102  176026          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
002570  001401                          BEQ     .+4             ;BRANCH IF OK
```

```
002572  104000                          HLT                             ;STACK POINTER FOULED UP

002574  022767  002354  175776          CMP     #1$,     STK1           ;CHECK THE RTI ADDRESS ON THE STACK
002602  001401                          BEQ     .+4                     ;BRANCH IF OK
002604  104000                          HLT                             ;RTI ADDRESS NOT EQUAL TO #1$

002606  026767  176044  175766          CMP     $MULER,  STK2           ;CHECK THE PSW ON THE STACK
002614  001401                          BEQ     .+4                     ;BRANCH IF OK
002616  104000                          HLT                             ;RTI PSW NOT EQUAL TO 200

002620  026767  176000  175756          CMP     RAND.C,  STK3           ;CHECK THE DATA ON THE STACK
002626  001401                          BEQ     .+4                     ;BRANCH IF OK
002630  104000                          HLT                             ;STK3 NOT EQUAL TO RAND.C

002632  026767  175770  175746          CMP     RAND.D,  STK4           ;CHECK THE DATA ON THE STACK
002640  001401                          BEQ     .+4                     ;BRANCH IF OK
002642  104000                          HLT                             ;STK4 NOT EQUAL TO RAND.D

002644  026767  175750  175736          CMP     RAND.A,  STK5           ;CHECK THE DATA ON THE STACK
002652  001401                          BEQ     .+4                     ;BRANCH IF OK
002654  104000                          HLT                             ;STK5 NOT EQUAL TO RAND.A

002656  026767  175740  175726          CMP     RAND.B,  STK6           ;CHECK THE DATA ON THE STACK
002664  001401                          BEQ     .+4                     ;BRANCH IF OK
002666  104000                          HLT                             ;STK6 NOT EQUAL TO RAND.B

002670  012716  002676                  MOV     #3$,     (SP)           ;RESET THE STACK
002674  000002                          RTI                             ;RESTORE THE STATUS (T-BIT)

002676  104400                  3$:     SCOPE

```

```


                                  ;*************************************************************************
                                  ;TEST 4:         EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
                                  ;                RAND,A,RAND,B / RAND,C,RAND,D = ANS1,ANS2
                                  ;                STACK POINTER = R3
                                  ;*************************************************************************

 ØØ27ØØ  Ø127Ø3  ØØØ6Ø4           TST4:   MOV     #STACKØ,R3      ;SET UP THE STACK POINTER
 ØØ27Ø4  ØØ4767  Ø1Ø67Ø                   JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

 ØØ271Ø  ØØØ24Ø                           NOP
 ØØ2712  Ø75Ø33                           FDIV+   R3              ;FLOATING DIVIDE ON THE R3 STACK

 ØØ2714  Ø13767  177776  175672   1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
 ØØ2722  Ø1Ø367  17567Ø                   MOV     R3,     $SP     ;SAVE THE STACK POINTER
 ØØ2726  Ø26767  175726  17566Ø   6$:     CMP     $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
 ØØ2734  ØØ1Ø23                           BNE     4$              ;GO CHECK FOR ROUNDING ERROR

 ØØ2736  1Ø5767  175652                   TSTB    $PSW            ;CHECK FOR ERROR
 ØØ2742  1ØØ464                           BMI     2$              ;BRANCH IF ERROR

 ØØ2744  Ø12767  ØØØ61Ø  175716           MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
 ØØ2752  Ø26767  175712  175636           CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
 ØØ276Ø  ØØ14Ø1                           BEQ     .+4             ;BRANCH IF OK
 ØØ2762  1Ø4ØØØ                           HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

 ØØ2764  Ø26767  175672  175616           CMP     $DIV1,  ANS1    ;CHECK THE ANSWER
 ØØ2772  ØØ1ØØ4                           BNE     4$
 ØØ2774  Ø26767  175664  17561Ø           CMP     $DIV2,  ANS2    ;CHECK THE ANSWER
 ØØ3ØØ2  ØØ1515                           BEQ     3$
 ØØ3ØØ4  Ø32767  ØØØØ2Ø  17566Ø   4$:     BIT     #BIT4,  RNDFLG  ;CHECK THE ROUNDING FLAG
 ØØ3Ø12  ØØ1Ø22                           BNE     5$
 ØØ3Ø14  Ø52767  ØØØØ2Ø  17565Ø           BIS     #BIT4,  RNDFLG  ;SET ROUNDING FLAG
 ØØ3Ø22  Ø62767  ØØØØØ1  175634           ADD     #1,     $DIV2   ;INCREMENT FORTRAN ANSWER
 ØØ3Ø3Ø  ØØ5567  175626                   ADC     $DIV1           ;ADD CARRY
 ØØ3Ø34  1Ø2334                           BVC     6$              ;BRANCH IF NO OVERFLOW
 ØØ3Ø36  ØØØ257                           CCC                     ;CLEAR ALL CONDITION CODES
 ØØ3Ø4Ø  ØØØ262                           SEV                     ;SET V-BIT
 ØØ3Ø42  Ø13767  177776  175616           MOV     @#PS,   $DIVER  ;SET UP PSW FOR OVERFLOW
 ØØ3Ø5Ø  Ø12767  ØØØ34Ø  1756Ø2           MOV     #34Ø,   $DIVPS  ;SET UP TRAP PSW
 ØØ3Ø56  ØØØ723                           BR      6$              ;TRY IT AGAIN

 ØØ3Ø6Ø  132767  ØØØØ2Ø  1756Ø5   5$:     BITB    #BIT4,  RNDFLG+1 ;CHECK "DEROUNDING" FLAG
 ØØ3Ø66  ØØ1Ø1Ø                           BNE     7$              ;BRANCH IF SET
 ØØ3Ø7Ø  152767  ØØØØ2Ø  175575           BISB    #BIT4,  RNDFLG+1 ;SET "DEROUNDING" FLAG
 ØØ3Ø76  162767  ØØØØØ1  17556Ø           SUB     #1,     $DIV2   ;RESTORE ORIGINAL ANSWER
 ØØ31Ø4  ØØ5667  175552                   SBC     $DIV1           ;SUBTRACT CARRY
 ØØ311Ø  1Ø4ØØØ                   7$:     HLT                     ;WRONG PSW OR ANSWER

 ØØ3112  ØØØ451                           BR      3$

 ØØ3114  Ø12767  ØØØ6Ø4  175546   2$:     MOV     #STACKØ,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
 ØØ3122  Ø26767  175542  175466           CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
 ØØ313Ø  ØØ14Ø1                           BEQ     .+4             ;BRANCH IF OK
```

```
003132  104000                           HLT                            ;STACK POINTER FOULED UP

003134  022767  002714  175436           CMP     #1$,     STK1          ;CHECK THE RTI ADDRESS ON THE STACK
003142  001401                           BEQ     .+4                    ;BRANCH IF OK
003144  104000                           HLT                            ;RTI ADDRESS NOT EQUAL TO #1$

003146  026767  175514  175426           CMP     $DIVER,  STK2          ;CHECK THE PSW ON THE STACK
003154  001401                           BEQ     .+4                    ;BRANCH IF OK
003156  104000                           HLT                            ;RTI PSW NOT EQUAL TO 200

003160  026767  175440  175416           CMP     RAND.C,  STK3          ;CHECK THE DATA ON THE STACK
003166  001401                           BEQ     .+4                    ;BRANCH IF OK
003170  104000                           HLT                            ;STK3 NOT EQUAL TO RAND.C

003172  026767  175430  175406           CMP     RAND.D,  STK4          ;CHECK THE DATA ON THE STACK
003200  001401                           BEQ     .+4                    ;BRANCH IF OK
003202  104000                           HLT                            ;STK4 NOT EQUAL TO RAND.D

003204  026767  175410  175376           CMP     RAND.A,  STK5          ;CHECK THE DATA ON THE STACK
003212  001401                           BEQ     .+4                    ;BRANCH IF OK
003214  104000                           HLT                            ;STK5 NOT EQUAL TO RAND.A

003216  026767  175400  175366           CMP     RAND.B,  STK6          ;CHECK THE DATA ON THE STACK
003224  001401                           BEQ     .+4                    ;BRANCH IF OK
003226  104000                           HLT                            ;STK6 NOT EQUAL TO RAND.B

003230  012716  003236                   MOV     #3$,     (SP)          ;RESET THE STACK
003234  000002                           RTI                            ;RESTORE THE STATUS (T-BIT)

003236  104400                   3$:     SCOPE

```

```

                                     ;*********************************************************************
                                     ;TEST 5:        EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
                                     ;        RAND,A,RAND,B + RAND,C,RAND,D = ANS1,ANS2
                                     ;        STACK POINTER = R4
                                     ;*********************************************************************

003240  012704  000604               TST5:   MOV     #STACK0,R4         ;SET UP THE STACK POINTER
003244  004767  010330                       JSR     PC,     PUSHR      ;PUT THE DATA ON THE STACK

003250  000240                               NOP
003252  075004                               FADD+   R4                 ;FLOATING ADD ON THE R4 STACK

003254  013767  177776  175332  1$:          MOV     @#PS,   $PSW       ;SAVE PROCESSOR STATUS
003262  010467  175330                       MOV     R4,     $SP        ;SAVE THE STACK POINTER
003266  026767  175336  175320               CMP     $ADDPS, $PSW       ;CHECK THE PROCESSOR STATUS
003274  001401                               BEQ     .+4                ;BRANCH IF OK
003276  104000                               HLT                        ;PSW NOT EQUAL TO $ADDPS

003300  105767  175310                       TSTB    $PSW               ;CHECK FOR ERROR
003304  100423                               BMI     2$                 ;BRANCH IF ERROR

003306  012767  000610  175354               MOV     #STACK4,SAVSTK     ;SAVE PROPER STACK ADDRESS FOR TYPING
003314  026767  175350  175274               CMP     SAVSTK, $SP        ;CHECK THE STACK POINTER
003322  001401                               BEQ     .+4                ;BRANCH IF OK
003324  104000                               HLT                        ;STACK POINTER NOT EQUAL TO #STACK4

003326  026767  175300  175254               CMP     $ADD1,  ANS1       ;CHECK THE ANSWER
003334  001401                               BEQ     .+4                ;BRANCH IF OK
003336  104000                               HLT                        ;LEFT HALF OF ANSWER WRONG

003340  026767  175270  175244               CMP     $ADD2,  ANS2       ;CHECK THE ANSWER
003346  001401                               BEQ     .+4                ;BRANCH IF OK
003350  104000                               HLT                        ;RIGHT HALF OF ANSWER WRONG

003352  000451                               BR      3$

003354  012767  000604  175306  2$:          MOV     #STACK0,SAVSTK     ;SAVE STACK ADDRESS FOR TYPING
003362  026767  175302  175226               CMP     SAVSTK, $SP        ;CHECK THE STACK POINTER
003370  001401                               BEQ     .+4                ;BRANCH IF OK
003372  104000                               HLT                        ;STACK POINTER FOULED UP

003374  022767  003254  175176               CMP     #1$,    STK1       ;CHECK THE RTI ADDRESS ON THE STACK
003402  001401                               BEQ     .+4                ;BRANCH IF OK
003404  104000                               HLT                        ;RTI ADDRESS NOT EQUAL TO #1$

003406  026767  175224  175166               CMP     $ADDER, STK2       ;CHECK THE PSW ON THE STACK
003414  001401                               BEQ     .+4                ;BRANCH IF OK
003416  104000                               HLT                        ;RTI PSW NOT EQUAL TO 200

003420  026767  175200  175156               CMP     RAND,C, STK3       ;CHECK THE DATA ON THE STACK
003426  001401                               BEQ     .+4                ;BRANCH IF OK
003430  104000                               HLT                        ;STK3 NOT EQUAL TO RAND,C

```

```
003432  026767  175170  175146              CMP     RAND,D, STK4     ;CHECK THE DATA ON THE STACK
003440  001401                              BEQ     .+4              ;BRANCH IF OK
003442  104000                              HLT                      ;STK4 NOT EQUAL TO RAND,D

003444  026767  175150  175136              CMP     RAND,A, STK5     ;CHECK THE DATA ON THE STACK
003452  001401                              BEQ     .+4              ;BRANCH IF OK
003454  104000                              HLT                      ;STK5 NOT EQUAL TO RAND,A

003456  026767  175140  175126              CMP     RAND,B, STK6     ;CHECK THE DATA ON THE STACK
003464  001401                              BEQ     .+4              ;BRANCH IF OK
003466  104000                              HLT                      ;STK6 NOT EQUAL TO RAND,B

003470  012716  003476                      MOV     #3$,     (SP)    ;RESET THE STACK
003474  000002                              RTI                      ;RESTORE THE STATUS (T-BIT)

003476  104400                      3$:     SCOPE



                                    ;*********************************************************************
                                    ;TEST 6:         EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
                                    ;       RAND,A,RAND,B - RAND,C,RAND,D = ANS1,ANS2
                                    ;       STACK POINTER = R5
                                    ;*********************************************************************

003500  012705  000604              TST6:   MOV     #STACK0,R5       ;SET UP THE STACK POINTER
003504  004767  010070                      JSR     PC,      PUSHR   ;PUT THE DATA ON THE STACK

003510  000240                              NOP
003512  075015                              FSUB    R5               ;FLOATING SUBTRACT ON THE R5 STACK

003514  013767  177776  175072      1$:     MOV     @#PS,    $PSW    ;SAVE PROCESSOR STATUS
003522  010567  175070                      MOV     R5,      $SP     ;SAVE THE STACK POINTER
003526  026767  175106  175060              CMP     $SUBPS, $PSW     ;CHECK THE PROCESSOR STATUS
003534  001401                              BEQ     .+4              ;BRANCH IF OK
003536  104000                              HLT                      ;PSW NOT EQUAL TO $SUBPS

003540  105767  175050                      TSTB    $PSW             ;CHECK FOR ERROR
003544  100423                              BMI     2$               ;BRANCH IF ERROR

003546  012767  000610  175114              MOV     #STACK4,SAVSTK   ;SAVE PROPER STACK ADDRESS FOR TYPING
003554  026767  175110  175034              CMP     SAVSTK, $SP      ;CHECK THE STACK POINTER
003562  001401                              BEQ     .+4              ;BRANCH IF OK
003564  104000                              HLT                      ;STACK POINTER NOT EQUAL TO #STACK4

003566  026767  175050  175014              CMP     $SUB1,   ANS1    ;CHECK THE ANSWER
003574  001401                              BEQ     .+4              ;BRANCH IF OK
003576  104000                              HLT                      ;LEFT HALF OF ANSWER WRONG

003600  026767  175040  175004              CMP     $SUB2,   ANS2    ;CHECK THE ANSWER
003606  001401                              BEQ     .+4              ;BRANCH IF OK
003610  104000                              HLT                      ;RIGHT HALF OF ANSWER WRONG

003612  000451                              BR      3$

```

```
ØØ3614  Ø12767  ØØØ6Ø4  175Ø46  2$:     MOV     #STACKØ,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
ØØ3622  Ø26767  175Ø42  174766          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
ØØ363Ø  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ3632  1Ø4ØØØ                          HLT                     ;STACK POINTER FOULED UP

ØØ3634  Ø22767  ØØ3514  174736          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
ØØ3642  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ3644  1Ø4ØØØ                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

ØØ3646  Ø26767  174774  174726          CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
ØØ3654  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ3656  1Ø4ØØØ                          HLT                     ;RTI PSW NOT EQUAL TO 2ØØ

ØØ366Ø  Ø26767  17474Ø  174716          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
ØØ3666  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ367Ø  1Ø4ØØØ                          HLT                     ;STK3 NOT EQUAL TO RAND.C

ØØ3672  Ø26767  17473Ø  1747Ø6          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
ØØ37ØØ  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ37Ø2  1Ø4ØØØ                          HLT                     ;STK4 NOT EQUAL TO RAND.D

ØØ37Ø4  Ø26767  17471Ø  174676          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
ØØ3712  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ3714  1Ø4ØØØ                          HLT                     ;STK5 NOT EQUAL TO RAND.A

ØØ3716  Ø26767  1747ØØ  174666          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
ØØ3724  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ3726  1Ø4ØØØ                          HLT                     ;STK6 NOT EQUAL TO RAND.B

ØØ373Ø  Ø12716  ØØ3736                          MOV     #3$,    (SP)    ;RESET THE STACK
ØØ3734  ØØØØØ2                          RTI                     ;RESTORE THE STATUS (T=BIT)

ØØ3736  1Ø44ØØ                  3$:     SCOPE



                                ;*****************************************************************
                                ;TEST 7:        EXERCISE FMUL (PDP=11 FLOATING MULTIPLY INSTRUCTION)
                                ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
                                ;       STACK POINTER = SP
                                ;*****************************************************************

ØØ374Ø  Ø127Ø6  ØØØ6Ø4          TST7:   MOV     #STACKØ,SP      ;SET UP THE STACK POINTER
ØØ3744  ØØ4767  ØØ763Ø                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

ØØ375Ø  ØØØ24Ø                          NOP
ØØ3752  Ø75Ø26                          FMUL+   SP              ;FLOATING MULTIPLY ON THE SP STACK

ØØ3754  Ø13767  177776  174632  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
ØØ3762  Ø1Ø667  17463Ø                  MOV     SP,     $SP     ;SAVE THE STACK POINTER
ØØ3766  Ø26767  174656  17462Ø          CMP     $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
ØØ3774  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ3776  1Ø4ØØØ                          HLT                     ;PSW NOT EQUAL TO $MULPS

ØØ4ØØØ  1Ø5767  17461Ø                  TSTB    $PSW            ;CHECK FOR ERROR
```

```
 004004  100424                              BMI     2$               ;BRANCH IF ERROR

 004006  012767  000610  174654              MOV     #STACK4,SAVSTK   ;SAVE PROPER STACK ADDRESS FOR TYPING
 004014  026767  174650  174574              CMP     SAVSTK, $SP      ;CHECK THE STACK POINTER
 004022  001401                              BEQ     .+4              ;BRANCH IF OK
 004024  104000                              HLT                      ;STACK POINTER NOT EQUAL TO #STACK4

 004026  026767  174620  174554              CMP     $MUL1,  ANS1     ;CHECK THE ANSWER
 004034  001401                              BEQ     .+4              ;BRANCH IF OK
 004036  104000                              HLT                      ;LEFT HALF OF ANSWER WRONG

 004040  026767  174610  174544              CMP     $MUL2,  ANS2     ;CHECK THE ANSWER
 004046  001401                              BEQ     .+4              ;BRANCH IF OK
 004050  104000                              HLT                      ;RIGHT HALF OF ANSWER WRONG

 004052  024646                              CMP     -(SP),  -(SP)    ;RESTORE THE STACK
 004054  000451                              BR      3$

 004056  012767  000600  174604  2$:         MOV     #STK1,  SAVSTK   ;SAVE PROPER STACK ADDRESS FOR TYPING
 004064  026767  174600  174524              CMP     SAVSTK, $SP      ;CHECK THE STACK POINTER
 004072  001401                              BEQ     .+4              ;BRANCH IF OK
 004074  104000                              HLT                      ;STACK POINTER FOULED UP

 004076  022767  003754  174474              CMP     #1$,    STK1     ;CHECK THE RTI ADDRESS ON THE STACK
 004104  001401                              BEQ     .+4              ;BRANCH IF OK
 004106  104000                              HLT                      ;RTI ADDRESS NOT EQUAL TO #1$

 004110  026767  174542  174464              CMP     $MULER, STK2     ;CHECK THE PSW ON THE STACK
 004116  001401                              BEQ     .+4              ;BRANCH IF OK
 004120  104000                              HLT                      ;RTI PSW NOT EQUAL TO 200

 004122  026767  174476  174454              CMP     RAND.C, STK3     ;CHECK THE DATA ON THE STACK
 004130  001401                              BEQ     .+4              ;BRANCH IF OK
 004132  104000                              HLT                      ;STK3 NOT EQUAL TO RAND.C

 004134  026767  174466  174444              CMP     RAND.D, STK4     ;CHECK THE DATA ON THE STACK
 004142  001401                              BEQ     .+4              ;BRANCH IF OK
 004144  104000                              HLT                      ;STK4 NOT EQUAL TO RAND.D

 004146  026767  174446  174434              CMP     RAND.A, STK5     ;CHECK THE DATA ON THE STACK
 004154  001401                              BEQ     .+4              ;BRANCH IF OK
 004156  104000                              HLT                      ;STK5 NOT EQUAL TO RAND.A

 004160  026767  174436  174424              CMP     RAND.B, STK6     ;CHECK THE DATA ON THE STACK
 004166  001401                              BEQ     .+4              ;BRANCH IF OK
 004170  104000                              HLT                      ;STK6 NOT EQUAL TO RAND.B

 004172  012716  004200                      MOV     #3$,    (SP)     ;RESET THE STACK
 004176  000002                              RTI                      ;RESTORE THE STATUS (T-BIT)

 004200  104400                  3$:         SCOPE

```

```

                                        ;*******************************************************************
                                        ;TEST 1Ø:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
                                        ;         RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
                                        ;         STACK POINTER = RØ
                                        ;*******************************************************************

Ø042Ø2  Ø127ØØ  ØØØ6Ø4          TST1Ø:  MOV     #STACKØ,RØ      ;SET UP THE STACK POINTER
Ø042Ø6  ØØ4767  ØØ7366                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

ØØ4212  ØØØ24Ø                          NOP
ØØ4214  Ø75Ø3Ø                          FDIV+   RØ              ;FLOATING DIVIDE ON THE RØ STACK

ØØ4216  Ø13767  177776  17437Ø  1$:     MOV     @#PS,   SPSW    ;SAVE PROCESSOR STATUS
ØØ4224  Ø1ØØ67  174366                  MOV     RØ,     SSP     ;SAVE THE STACK POINTER
ØØ423Ø  Ø26767  174424  174356          CMP     SDIVPS, SPSW    ;CHECK THE PROCESSOR STATUS
ØØ4236  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ424Ø  1Ø4ØØØ                          HLT                     ;PSW NOT EQUAL TO SDIVPS

ØØ4242  1Ø5767  174346                  TSTB    SPSW            ;CHECK FOR ERROR
ØØ4246  1ØØ423                          BMI     2$              ;BRANCH IF ERROR

ØØ425Ø  Ø12767  ØØØ61Ø  174412          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
ØØ4256  Ø26767  1744Ø6  174332          CMP     SAVSTK, SSP     ;CHECK THE STACK POINTER
ØØ4264  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4266  1Ø4ØØØ                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

ØØ427Ø  Ø26767  174366  174312          CMP     SDIV1,  ANS1    ;CHECK THE ANSWER
ØØ4276  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ43ØØ  1Ø4ØØØ                          HLT                     ;LEFT HALF OF ANSWER WRONG

ØØ43Ø2  Ø26767  174356  1743Ø2          CMP     SDIV2,  ANS2    ;CHECK THE ANSWER
ØØ431Ø  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4312  1Ø4ØØØ                          HLT                     ;RIGHT HALF OF ANSWER WRONG

ØØ4314  ØØØ451                          BR      3$

ØØ4316  Ø12767  ØØØ6Ø4  174344  2$:     MOV     #STACKØ,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
ØØ4324  Ø26767  17434Ø  174264          CMP     SAVSTK, SSP     ;CHECK THE STACK POINTER
ØØ4332  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4334  1Ø4ØØØ                          HLT                     ;STACK POINTER FOULED UP

ØØ4336  Ø22767  ØØ4216  174234          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
ØØ4344  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4346  1Ø4ØØØ                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

ØØ435Ø  Ø26767  174312  174224          CMP     SDIVER, STK2    ;CHECK THE PSW ON THE STACK
ØØ4356  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ436Ø  1Ø4ØØØ                          HLT                     ;RTI PSW NOT EQUAL TO 2ØØ

ØØ4362  Ø26767  174236  174214          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
ØØ437Ø  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4372  1Ø4ØØØ                          HLT                     ;STK3 NOT EQUAL TO RAND.C

```

```
004374  026767  174226  174204              CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
004402  001401                              BEQ     .+4             ;BRANCH IF OK
004404  104000                              HLT                     ;STK4 NOT EQUAL TO RAND.D

004406  026767  174206  174174              CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
004414  001401                              BEQ     .+4             ;BRANCH IF OK
004416  104000                              HLT                     ;STK5 NOT EQUAL TO RAND.A

004420  026767  174176  174164              CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
004426  001401                              BEQ     .+4             ;BRANCH IF OK
004430  104000                              HLT                     ;STK6 NOT EQUAL TO RAND.B

004432  012716  004440                      MOV     #3$,    (SP)    ;RESET THE STACK
004436  000002                              RTI                     ;RESTORE THE STATUS (T-BIT)

004440  104400                      3$:     SCOPE



                                    ;*****************************************************************
                                    ;TEST 11:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
                                    ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
                                    ;       STACK POINTER = R1
                                    ;*****************************************************************

004442  012701  000604              TST11:  MOV     #STACK0,R1      ;SET UP THE STACK POINTER
004446  004767  007126                      JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

004452  000240                              NOP
004454  075001                              FADD+   R1              ;FLOATING ADD ON THE R1 STACK

004456  013767  177776  174130      1$:     MOV     @#PS,   SPSW    ;SAVE PROCESSOR STATUS
004464  010167  174126                      MOV     R1,     SSP     ;SAVE THE STACK POINTER
004470  026767  174134  174116              CMP     SADDPS, SPSW    ;CHECK THE PROCESSOR STATUS
004476  001401                              BEQ     .+4             ;BRANCH IF OK
004500  104000                              HLT                     ;PSW NOT EQUAL TO SADDPS

004502  105767  174106                      TSTB    SPSW            ;CHECK FOR ERROR
004506  100423                              BMI     2$              ;BRANCH IF ERROR

004510  012767  000610  174152              MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
004516  026767  174146  174072              CMP     SAVSTK, SSP     ;CHECK THE STACK POINTER
004524  001401                              BEQ     .+4             ;BRANCH IF OK
004526  104000                              HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

004530  026767  174076  174052              CMP     SADD1,  ANS1    ;CHECK THE ANSWER
004536  001401                              BEQ     .+4             ;BRANCH IF OK
004540  104000                              HLT                     ;LEFT HALF OF ANSWER WRONG

004542  026767  174066  174042              CMP     SADD2,  ANS2    ;CHECK THE ANSWER
004550  001401                              BEQ     .+4             ;BRANCH IF OK
004552  104000                              HLT                     ;RIGHT HALF OF ANSWER WRONG

004554  000451                              BR      3$

```

```
ØØ4556  Ø12767  ØØØ6Ø4  1741Ø4  2$:     MOV     #STACKØ,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
ØØ4564  Ø26767  1741ØØ  174Ø24          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
ØØ4572  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4574  1Ø4ØØØ                          HLT                     ;STACK POINTER FOULED UP

ØØ4576  Ø22767  ØØ4456  173774          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
ØØ46Ø4  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ46Ø6  1Ø4ØØØ                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

ØØ461Ø  Ø26767  174Ø22  173764          CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
ØØ4616  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ462Ø  1Ø4ØØØ                          HLT                     ;RTI PSW NOT EQUAL TO 2ØØ

ØØ4622  Ø26767  173776  173754          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
ØØ463Ø  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4632  1Ø4ØØØ                          HLT                     ;STK3 NOT EQUAL TO RAND.C

ØØ4634  Ø26767  173766  173744          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
ØØ4642  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4644  1Ø4ØØØ                          HLT                     ;STK4 NOT EQUAL TO RAND.D

ØØ4646  Ø26767  173746  173734          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
ØØ4654  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ4656  1Ø4ØØØ                          HLT                     ;STK5 NOT EQUAL TO RAND.A

ØØ466Ø  Ø26767  173736  173724          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
ØØ4666  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ467Ø  1Ø4ØØØ                          HLT                     ;STK6 NOT EQUAL TO RAND.B

ØØ4672  Ø12716  ØØ47ØØ                  MOV     #3$,    (SP)    ;RESET THE STACK
ØØ4676  ØØØØØ2                          RTI                     ;RESTORE THE STATUS (T-BIT)

ØØ47ØØ  1Ø44ØØ                  3$:     SCOPE


                                ;*************************************************************************
                                ;TEST 12:       EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
                                ;       RAND.A,RAND.B - RAND.C,RAND.D = ANS1,ANS2
                                ;       STACK POINTER = R2
                                ;*************************************************************************

ØØ47Ø2  Ø127Ø2  ØØØ6Ø4          TST12:  MOV     #STACKØ,R2      ;SET UP THE STACK POINTER
ØØ47Ø6  ØØ4767  ØØ6666                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

ØØ4712  ØØØ24Ø                          NOP
ØØ4714  Ø75Ø12                          FSUB+   R2              ;FLOATING SUBTRACT ON THE R2 STACK

ØØ4716  Ø13767  177776  17367Ø  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
ØØ4724  Ø1Ø267  173666                  MOV     R2,     $SP     ;SAVE THE STACK POINTER
ØØ473Ø  Ø26767  1737Ø4  173656          CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
ØØ4736  ØØ14Ø1                          BEQ     .+4             ;BRANCH IF OK
ØØ474Ø  1Ø4ØØØ                          HLT                     ;PSW NOT EQUAL TO $SUBPS

ØØ4742  1Ø5767  173646                  TSTB    $PSW            ;CHECK FOR ERROR
```

```
004746  100423                              BMI     2$                   ;BRANCH IF ERROR

004750  012767  000610  173712              MOV     #STACK4,SAVSTK       ;SAVE PROPER STACK ADDRESS FOR TYPING
004756  026767  173706  173632              CMP     SAVSTK, $SP          ;CHECK THE STACK POINTER
004764  001401                              BEQ     .+4                  ;BRANCH IF OK
004766  104000                              HLT                          ;STACK POINTER NOT EQUAL TO #STACK4

004770  026767  173646  173612              CMP     $SUB1,  ANS1         ;CHECK THE ANSWER
004776  001401                              BEQ     .+4                  ;BRANCH IF OK
005000  104000                              HLT                          ;LEFT HALF OF ANSWER WRONG

005002  026767  173636  173602              CMP     $SUB2,  ANS2         ;CHECK THE ANSWER
005010  001401                              BEQ     .+4                  ;BRANCH IF OK
005012  104000                              HLT                          ;RIGHT HALF OF ANSWER WRONG

005014  000451                              BR      3$

005016  012767  000604  173644      2$:     MOV     #STACK0,SAVSTK       ;SAVE STACK ADDRESS FOR TYPING
005024  026767  173640  173564              CMP     SAVSTK, $SP          ;CHECK THE STACK POINTER
005032  001401                              BEQ     .+4                  ;BRANCH IF OK
005034  104000                              HLT                          ;STACK POINTER FOULED UP

005036  022767  004716  173534              CMP     #1$,    STK1         ;CHECK THE RTI ADDRESS ON THE STACK
005044  001401                              BEQ     .+4                  ;BRANCH IF OK
005046  104000                              HLT                          ;RTI ADDRESS NOT EQUAL TO #1$

005050  026767  173572  173524              CMP     $SUBER, STK2         ;CHECK THE PSW ON THE STACK
005056  001401                              BEQ     .+4                  ;BRANCH IF OK
005060  104000                              HLT                          ;RTI PSW NOT EQUAL TO 200

005062  026767  173536  173514              CMP     RAND.C, STK3         ;CHECK THE DATA ON THE STACK
005070  001401                              BEQ     .+4                  ;BRANCH IF OK
005072  104000                              HLT                          ;STK3 NOT EQUAL TO RAND.C

005074  026767  173526  173504              CMP     RAND.D, STK4         ;CHECK THE DATA ON THE STACK
005102  001401                              BEQ     .+4                  ;BRANCH IF OK
005104  104000                              HLT                          ;STK4 NOT EQUAL TO RAND.D

005106  026767  173506  173474              CMP     RAND.A, STK5         ;CHECK THE DATA ON THE STACK
005114  001401                              BEQ     .+4                  ;BRANCH IF OK
005116  104000                              HLT                          ;STK5 NOT EQUAL TO RAND.A

005120  026767  173476  173464              CMP     RAND.B, STK6         ;CHECK THE DATA ON THE STACK
005126  001401                              BEQ     .+4                  ;BRANCH IF OK
005130  104000                              HLT                          ;STK6 NOT EQUAL TO RAND.B

005132  012716  005140                      MOV     #3$,    (SP)         ;RESET THE STACK
005136  000002                              RTI                          ;RESTORE THE STATUS (T-BIT)

005140  104400                      3$:     SCOPE

```

```

                                          ;*******************************************************************
                                          ;TEST 13:        EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
                                          ;         RAND,A,RAND,B * RAND,C,RAND,D = ANS1,ANS2
                                          ;         STACK POINTER = R3
                                          ;*******************************************************************

 005142  012703  000604           TST13:  MOV     #STACK0,R3       ;SET UP THE STACK POINTER
 005146  004767  006426                   JSR     PC,     PUSHR    ;PUT THE DATA ON THE STACK

 005152  000240                           NOP
 005154  075023                           FMUL+   R3               ;FLOATING MULTIPLY ON THE R3 STACK

 005156  013767  177776  173430   1$:     MOV     @#PS,   $PSW     ;SAVE PROCESSOR STATUS
 005164  010367  173426                   MOV     R3,     $SP      ;SAVE THE STACK POINTER
 005170  026767  173454  173416           CMP     $MULPS, $PSW     ;CHECK THE PROCESSOR STATUS
 005176  001401                           BEQ     .+4              ;BRANCH IF OK
 005200  104000                           HLT                      ;PSW NOT EQUAL TO $MULPS

 005202  105767  173406                   TSTB    $PSW             ;CHECK FOR ERROR
 005206  100423                           BMI     2$               ;BRANCH IF ERROR

 005210  012767  000610  173452           MOV     #STACK4,SAVSTK   ;SAVE PROPER STACK ADDRESS FOR TYPING
 005216  026767  173446  173372           CMP     SAVSTK, $SP      ;CHECK THE STACK POINTER
 005224  001401                           BEQ     .+4              ;BRANCH IF OK
 005226  104000                           HLT                      ;STACK POINTER NOT EQUAL TO #STACK4

 005230  026767  173416  173352           CMP     $MUL1,  ANS1     ;CHECK THE ANSWER
 005236  001401                           BEQ     .+4              ;BRANCH IF OK
 005240  104000                           HLT                      ;LEFT HALF OF ANSWER WRONG

 005242  026767  173406  173342           CMP     $MUL2,  ANS2     ;CHECK THE ANSWER
 005250  001401                           BEQ     .+4              ;BRANCH IF OK
 005252  104000                           HLT                      ;RIGHT HALF OF ANSWER WRONG

 005254  000451                           BR      3$

 005256  012767  000604  173404   2$:     MOV     #STACK0,SAVSTK   ;SAVE STACK ADDRESS FOR TYPING
 005264  026767  173400  173324           CMP     SAVSTK, $SP      ;CHECK THE STACK POINTER
 005272  001401                           BEQ     .+4              ;BRANCH IF OK
 005274  104000                           HLT                      ;STACK POINTER FOULED UP

 005276  022767  005156  173274           CMP     #1$,    STK1     ;CHECK THE RTI ADDRESS ON THE STACK
 005304  001401                           BEQ     .+4              ;BRANCH IF OK
 005306  104000                           HLT                      ;RTI ADDRESS NOT EQUAL TO #1$

 005310  026767  173342  173264           CMP     $MULER, STK2     ;CHECK THE PSW ON THE STACK
 005316  001401                           BEQ     .+4              ;BRANCH IF OK
 005320  104000                           HLT                      ;RTI PSW NOT EQUAL TO 200

 005322  026767  173276  173254           CMP     RAND,C, STK3     ;CHECK THE DATA ON THE STACK
 005330  001401                           BEQ     .+4              ;BRANCH IF OK
 005332  104000                           HLT                      ;STK3 NOT EQUAL TO RAND,C

```

```
ØØ5334  Ø26767  173266  173244           CMP     RAND,D, STK4     ;CHECK THE DATA ON THE STACK
ØØ5342  ØØ14Ø1                           BEQ     .+4              ;BRANCH IF OK
ØØ5344  1Ø4ØØØ                           HLT                      ;STK4 NOT EQUAL TO RAND,D

ØØ5346  Ø26767  173246  173234           CMP     RAND,A, STK5     ;CHECK THE DATA ON THE STACK
ØØ5354  ØØ14Ø1                           BEQ     .+4              ;BRANCH IF OK
ØØ5356  1Ø4ØØØ                           HLT                      ;STK5 NOT EQUAL TO RAND,A

ØØ536Ø  Ø26767  173236  173224           CMP     RAND,B, STK6     ;CHECK THE DATA ON THE STACK
ØØ5366  ØØ14Ø1                           BEQ     .+4              ;BRANCH IF OK
ØØ537Ø  1Ø4ØØØ                           HLT                      ;STK6 NOT EQUAL TO RAND,B

ØØ5372  Ø12716  ØØ54ØØ                   MOV     #3$,     (SP)    ;RESET THE STACK
ØØ5376  ØØØØØ2                           RTI                      ;RESTORE THE STATUS (T-BIT)

ØØ54ØØ  1Ø44ØØ                   3$:     SCOPE


                                 ;*************************************************************************
                                 ;TEST 14:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
                                 ;       RAND,A,RAND,B / RAND,C,RAND,D = ANS1,ANS2
                                 ;       STACK POINTER = R4
                                 ;*************************************************************************

ØØ54Ø2  Ø127Ø4  ØØØ6Ø4           TST14:  MOV     #STACKØ,R4       ;SET UP THE STACK POINTER
ØØ54Ø6  ØØ4767  ØØ6166                   JSR     PC,      PUSHR   ;PUT THE DATA ON THE STACK

ØØ5412  ØØØ24Ø                           NOP
ØØ5414  Ø75Ø34                           FDIV+   R4               ;FLOATING DIVIDE ON THE R4 STACK

ØØ5416  Ø13767  177776  17317Ø   1$:     MOV     @#PS,    $PSW    ;SAVE PROCESSOR STATUS
ØØ5424  Ø1Ø467  173166                   MOV     R4,      $SP     ;SAVE THE STACK POINTER
ØØ543Ø  Ø26767  173224  173156           CMP     $DIVPS,  $PSW    ;CHECK THE PROCESSOR STATUS
ØØ5436  ØØ14Ø1                           BEQ     .+4              ;BRANCH IF OK
ØØ544Ø  1Ø4ØØØ                           HLT                      ;PSW NOT EQUAL TO $DIVPS

ØØ5442  1Ø5767  173146                   TSTB    $PSW             ;CHECK FOR ERROR
ØØ5446  1ØØ423                           BMI     2$               ;BRANCH IF ERROR

ØØ545Ø  Ø12767  ØØØ61Ø  173212           MOV     #STACK4,SAVSTK   ;SAVE PROPER STACK ADDRESS FOR TYPING
ØØ5456  Ø26767  1732Ø6  173132           CMP     SAVSTK,  $SP     ;CHECK THE STACK POINTER
ØØ5464  ØØ14Ø1                           BEQ     .+4              ;BRANCH IF OK
ØØ5466  1Ø4ØØØ                           HLT                      ;STACK POINTER NOT EQUAL TO #STACK4

ØØ547Ø  Ø26767  173166  173112           CMP     $DIV1,   ANS1    ;CHECK THE ANSWER
ØØ5476  ØØ14Ø1                           BEQ     .+4              ;BRANCH IF OK
ØØ55ØØ  1Ø4ØØØ                           HLT                      ;LEFT HALF OF ANSWER WRONG

ØØ55Ø2  Ø26767  173156  1731Ø2           CMP     $DIV2,   ANS2    ;CHECK THE ANSWER
ØØ551Ø  ØØ14Ø1                           BEQ     .+4              ;BRANCH IF OK
ØØ5512  1Ø4ØØØ                           HLT                      ;RIGHT HALF OF ANSWER WRONG

ØØ5514  ØØØ451                           BR      3$

```

```
005516  012767  000604  173144  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
005524  026767  173140  173064          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
005532  001401                          BEQ     .+4             ;BRANCH IF OK
005534  104000                          HLT                     ;STACK POINTER FOULED UP

005536  022767  005416  173034          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
005544  001401                          BEQ     .+4             ;BRANCH IF OK
005546  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

005550  026767  173112  173024          CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
005556  001401                          BEQ     .+4             ;BRANCH IF OK
005560  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200

005562  026767  173036  173014          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
005570  001401                          BEQ     .+4             ;BRANCH IF OK
005572  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C

005574  026767  173026  173004          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
005602  001401                          BEQ     .+4             ;BRANCH IF OK
005604  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D

005606  026767  173006  172774          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
005614  001401                          BEQ     .+4             ;BRANCH IF OK
005616  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A

005620  026767  172776  172764          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
005626  001401                          BEQ     .+4             ;BRANCH IF OK
005630  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B

005632  012716  005640                  MOV     #3$,    (SP)    ;RESET THE STACK
005636  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)

005640  104400                  3$:     SCOPE


                                ;*******************************************************************
                                ;TEST 15:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
                                ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
                                ;       STACK POINTER = R5
                                ;*******************************************************************

005642  012705  000604          TST15:  MOV     #STACK0,R5      ;SET UP THE STACK POINTER
005646  004767  005726                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

005652  000240                          NOP
005654  075005                          FADD+   R5              ;FLOATING ADD ON THE R5 STACK

005656  013767  177776  172730  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
005664  010567  172726                  MOV     R5,     $SP     ;SAVE THE STACK POINTER
005670  026767  172734  172716          CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
005676  001401                          BEQ     .+4             ;BRANCH IF OK
005700  104000                          HLT                     ;PSW NOT EQUAL TO $ADDPS

005702  105767  172706                  TSTB    $PSW            ;CHECK FOR ERROR
```

```
005706  100423                              BMI     2$                  ;BRANCH IF ERROR

005710  012767  000610  172752              MOV     #STACK4,SAVSTK      ;SAVE PROPER STACK ADDRESS FOR TYPING
005716  026767  172746  172672              CMP     SAVSTK, $SP         ;CHECK THE STACK POINTER
005724  001401                              BEQ     .+4                 ;BRANCH IF OK
005726  104000                              HLT                         ;STACK POINTER NOT EQUAL TO #STACK4

005730  026767  172676  172652              CMP     $ADD1,  ANS1        ;CHECK THE ANSWER
005736  001401                              BEQ     .+4                 ;BRANCH IF OK
005740  104000                              HLT                         ;LEFT HALF OF ANSWER WRONG

005742  026767  172666  172642              CMP     $ADD2,  ANS2        ;CHECK THE ANSWER
005750  001401                              BEQ     .+4                 ;BRANCH IF OK
005752  104000                              HLT                         ;RIGHT HALF OF ANSWER WRONG

005754  000451                              BR      3$

005756  012767  000604  172704      2$:     MOV     #STACK0,SAVSTK      ;SAVE STACK ADDRESS FOR TYPING
005764  026767  172700  172624              CMP     SAVSTK, $SP         ;CHECK THE STACK POINTER
005772  001401                              BEQ     .+4                 ;BRANCH IF OK
005774  104000                              HLT                         ;STACK POINTER FOULED UP

005776  022767  005656  172574              CMP     #1$,    STK1        ;CHECK THE RTI ADDRESS ON THE STACK
006004  001401                              BEQ     .+4                 ;BRANCH IF OK
006006  104000                              HLT                         ;RTI ADDRESS NOT EQUAL TO #1$

006010  026767  172622  172564              CMP     $ADDER, STK2        ;CHECK THE PSW ON THE STACK
006016  001401                              BEQ     .+4                 ;BRANCH IF OK
006020  104000                              HLT                         ;RTI PSW NOT EQUAL TO 200

006022  026767  172576  172554              CMP     RAND.C, STK3        ;CHECK THE DATA ON THE STACK
006030  001401                              BEQ     .+4                 ;BRANCH IF OK
006032  104000                              HLT                         ;STK3 NOT EQUAL TO RAND.C

006034  026767  172566  172544              CMP     RAND.D, STK4        ;CHECK THE DATA ON THE STACK
006042  001401                              BEQ     .+4                 ;BRANCH IF OK
006044  104000                              HLT                         ;STK4 NOT EQUAL TO RAND.D

006046  026767  172546  172534              CMP     RAND.A, STK5        ;CHECK THE DATA ON THE STACK
006054  001401                              BEQ     .+4                 ;BRANCH IF OK
006056  104000                              HLT                         ;STK5 NOT EQUAL TO RAND.A

006060  026767  172536  172524              CMP     RAND.B, STK6        ;CHECK THE DATA ON THE STACK
006066  001401                              BEQ     .+4                 ;BRANCH IF OK
006070  104000                              HLT                         ;STK6 NOT EQUAL TO RAND.B

006072  012716  006100                      MOV     #3$,    (SP)        ;RESET THE STACK
006076  000002                              RTI                         ;RESTORE THE STATUS (T-BIT)

006100  104400                      3$:     SCOPE

```

```

                                        ;*************************************************************************
                                        ;TEST 16:       EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
                                        ;       RAND,A,RAND,B - RAND,C,RAND,D = ANS1,ANS2
                                        ;       STACK POINTER = SP
                                        ;*************************************************************************

 ØØ61Ø2  Ø127Ø6  ØØØ6Ø4                 TST16:  MOV     #STACKØ,SP      ;SET UP THE STACK POINTER
 ØØ61Ø6  ØØ4767  ØØ5466                         JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

 ØØ6112  ØØØ24Ø                                 NOP
 ØØ6114  Ø75Ø16                                 FSUB+   SP              ;FLOATING SUBTRACT ON THE SP STACK

 ØØ6116  Ø13767  177776  17247Ø  1$:            MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
 ØØ6124  Ø1Ø667  172466                         MOV     SP,     $SP     ;SAVE THE STACK POINTER
 ØØ613Ø  Ø26767  1725Ø4  172456                 CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
 ØØ6136  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
 ØØ614Ø  1Ø4ØØØ                                 HLT                     ;PSW NOT EQUAL TO $SUBPS

 ØØ6142  1Ø5767  172446                         TSTB    $PSW            ;CHECK FOR ERROR
 ØØ6146  1ØØ424                                 BMI     2$              ;BRANCH IF ERROR

 ØØ615Ø  Ø12767  ØØØ61Ø  172512                 MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
 ØØ6156  Ø26767  1725Ø6  172432                 CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
 ØØ6164  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
 ØØ6166  1Ø4ØØØ                                 HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

 ØØ617Ø  Ø26767  172446  172412                 CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
 ØØ6176  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
 ØØ62ØØ  1Ø4ØØØ                                 HLT                     ;LEFT HALF OF ANSWER WRONG

 ØØ62Ø2  Ø26767  172436  1724Ø2                 CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
 ØØ621Ø  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
 ØØ6212  1Ø4ØØØ                                 HLT                     ;RIGHT HALF OF ANSWER WRONG

 ØØ6214  Ø24646                                 CMP     -(SP),  -(SP)   ;RESTORE THE STACK
 ØØ6216  ØØØ451                                 BR      3$

 ØØ622Ø  Ø12767  ØØØ6ØØ  172442  2$:            MOV     #STK1,  SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
 ØØ6226  Ø26767  172436  172362                 CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
 ØØ6234  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
 ØØ6236  1Ø4ØØØ                                 HLT                     ;STACK POINTER FOULED UP

 ØØ624Ø  Ø22767  ØØ6116  172332                 CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
 ØØ6246  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
 ØØ625Ø  1Ø4ØØØ                                 HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

 ØØ6252  Ø26767  17237Ø  172322                 CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
 ØØ626Ø  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
 ØØ6262  1Ø4ØØØ                                 HLT                     ;RTI PSW NOT EQUAL TO 2ØØ

 ØØ6264  Ø26767  172334  172312                 CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
 ØØ6272  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
 ØØ6274  1Ø4ØØØ                                 HLT                     ;STK3 NOT EQUAL TO RAND.C

 006276  026767  172324  172302            CMP     RAND.D, STK4     ;CHECK THE DATA ON THE STACK
 006304  001401                            BEQ     .+4              ;BRANCH IF OK
 006306  104000                            HLT                      ;STK4 NOT EQUAL TO RAND.D

 006310  026767  172304  172272            CMP     RAND.A, STK5     ;CHECK THE DATA ON THE STACK
 006316  001401                            BEQ     .+4              ;BRANCH IF OK
 006320  104000                            HLT                      ;STK5 NOT EQUAL TO RAND.A

 006322  026767  172274  172262            CMP     RAND.B, STK6     ;CHECK THE DATA ON THE STACK
 006330  001401                            BEQ     .+4              ;BRANCH IF OK
 006332  104000                            HLT                      ;STK6 NOT EQUAL TO RAND.B

 006334  012716  006342                    MOV     #3$,    (SP)     ;RESET THE STACK
 006340  000002                            RTI                      ;RESTORE THE STATUS (T-BIT)

 006342  104400                    3$:     SCOPE



                                   ;*************************************************************
                                   ;TEST 17:        EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
                                   ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
                                   ;       STACK POINTER = R0
                                   ;*************************************************************

 006344  012700  000604            TST17:  MOV     #STACK0,R0       ;SET UP THE STACK POINTER
 006350  004767  005224                    JSR     PC,     PUSHR    ;PUT THE DATA ON THE STACK

 006354  000240                            NOP
 006356  075020                            FMUL+   R0               ;FLOATING MULTIPLY ON THE R0 STACK

 006360  013767  177776  172226    1$:     MOV     @#PS,   $PSW     ;SAVE PROCESSOR STATUS
 006366  010067  172224                    MOV     R0,     $SP      ;SAVE THE STACK POINTER
 006372  026767  172252  172214            CMP     $MULPS, $PSW     ;CHECK THE PROCESSOR STATUS
 006400  001401                            BEQ     .+4              ;BRANCH IF OK
 006402  104000                            HLT                      ;PSW NOT EQUAL TO $MULPS

 006404  105767  172204                    TSTB    $PSW             ;CHECK FOR ERROR
 006410  100423                            BMI     2$               ;BRANCH IF ERROR

 006412  012767  000610  172250            MOV     #STACK4,SAVSTK   ;SAVE PROPER STACK ADDRESS FOR TYPING
 006420  026767  172244  172170            CMP     SAVSTK, $SP      ;CHECK THE STACK POINTER
 006426  001401                            BEQ     .+4              ;BRANCH IF OK
 006430  104000                            HLT                      ;STACK POINTER NOT EQUAL TO #STACK4

 006432  026767  172214  172150            CMP     $MUL1,  ANS1     ;CHECK THE ANSWER
 006440  001401                            BEQ     .+4              ;BRANCH IF OK
 006442  104000                            HLT                      ;LEFT HALF OF ANSWER WRONG

 006444  026767  172204  172140            CMP     $MUL2,  ANS2     ;CHECK THE ANSWER
 006452  001401                            BEQ     .+4              ;BRANCH IF OK
 006454  104000                            HLT                      ;RIGHT HALF OF ANSWER WRONG

 006456  000451                            BR      3$
```

```

006460  012767  000604  172202  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
006466  026767  172176  172122          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
006474  001401                          BEQ     .+4             ;BRANCH IF OK
006476  104000                          HLT                     ;STACK POINTER FOULED UP

006500  022767  006360  172072          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
006506  001401                          BEQ     .+4             ;BRANCH IF OK
006510  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

006512  026767  172140  172062          CMP     $MULER, STK2    ;CHECK THE PSW ON THE STACK
006520  001401                          BEQ     .+4             ;BRANCH IF OK
006522  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200

006524  026767  172074  172052          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
006532  001401                          BEQ     .+4             ;BRANCH IF OK
006534  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C

006536  026767  172064  172042          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
006544  001401                          BEQ     .+4             ;BRANCH IF OK
006546  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D

006550  026767  172044  172032          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
006556  001401                          BEQ     .+4             ;BRANCH IF OK
006560  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A

006562  026767  172034  172022          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
006570  001401                          BEQ     .+4             ;BRANCH IF OK
006572  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B

006574  012716  006602                  MOV     #3$,    (SP)    ;RESET THE STACK
006600  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)

006602  104400                  3$:     SCOPE


                                ;*************************************************************************
                                ;TEST 20:        EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
                                ;       RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
                                ;       STACK POINTER = R1
                                ;*************************************************************************

006604  012701  000604          TST20:  MOV     #STACK0,R1      ;SET UP THE STACK POINTER
006610  004767  004764                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

006614  000240                          NOP
006616  075031                          FDIV    R1              ;FLOATING DIVIDE ON THE R1 STACK

006620  013767  177776  171766  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
006626  010167  171764                  MOV     R1,     $SP     ;SAVE THE STACK POINTER
006632  026767  172022  171754          CMP     $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
006640  001401                          BEQ     .+4             ;BRANCH IF OK
006642  104000                          HLT                     ;PSW NOT EQUAL TO $DIVPS

```

```
006644  105767  171744                  TSTB    $PSW              ;CHECK FOR ERROR
006650  100423                          BMI     2$                ;BRANCH IF ERROR

006652  012767  000610  172010          MOV     #STACK4,SAVSTK    ;SAVE PROPER STACK ADDRESS FOR TYPING
006660  026767  172004  171730          CMP     SAVSTK, $SP       ;CHECK THE STACK POINTER
006666  001401                          BEQ     .+4               ;BRANCH IF OK
006670  104000                          HLT                       ;STACK POINTER NOT EQUAL TO #STACK4

006672  026767  171764  171710          CMP     $DIV1,  ANS1      ;CHECK THE ANSWER
006700  001401                          BEQ     .+4               ;BRANCH IF OK
006702  104000                          HLT                       ;LEFT HALF OF ANSWER WRONG

006704  026767  171754  171700          CMP     $DIV2,  ANS2      ;CHECK THE ANSWER
006712  001401                          BEQ     .+4               ;BRANCH IF OK
006714  104000                          HLT                       ;RIGHT HALF OF ANSWER WRONG

006716  000451                          BR      3$

006720  012767  000604  171742  2$:     MOV     #STACK0,SAVSTK    ;SAVE STACK ADDRESS FOR TYPING
006726  026767  171736  171662          CMP     SAVSTK, $SP       ;CHECK THE STACK POINTER
006734  001401                          BEQ     .+4               ;BRANCH IF OK
006736  104000                          HLT                       ;STACK POINTER FOULED UP

006740  022767  006620  171632          CMP     #1$,    STK1      ;CHECK THE RTI ADDRESS ON THE STACK
006746  001401                          BEQ     .+4               ;BRANCH IF OK
006750  104000                          HLT                       ;RTI ADDRESS NOT EQUAL TO #1$

006752  026767  171710  171622          CMP     $DIVER, STK2      ;CHECK THE PSW ON THE STACK
006760  001401                          BEQ     .+4               ;BRANCH IF OK
006762  104000                          HLT                       ;RTI PSW NOT EQUAL TO 200

006764  026767  171634  171612          CMP     RAND.C, STK3      ;CHECK THE DATA ON THE STACK
006772  001401                          BEQ     .+4               ;BRANCH IF OK
006774  104000                          HLT                       ;STK3 NOT EQUAL TO RAND.C

006776  026767  171624  171602          CMP     RAND.D, STK4      ;CHECK THE DATA ON THE STACK
007004  001401                          BEQ     .+4               ;BRANCH IF OK
007006  104000                          HLT                       ;STK4 NOT EQUAL TO RAND.D

007010  026767  171604  171572          CMP     RAND.A, STK5      ;CHECK THE DATA ON THE STACK
007016  001401                          BEQ     .+4               ;BRANCH IF OK
007020  104000                          HLT                       ;STK5 NOT EQUAL TO RAND.A

007022  026767  171574  171562          CMP     RAND.B, STK6      ;CHECK THE DATA ON THE STACK
007030  001401                          BEQ     .+4               ;BRANCH IF OK
007032  104000                          HLT                       ;STK6 NOT EQUAL TO RAND.B

007034  012716  007042                  MOV     #3$,    (SP)      ;RESET THE STACK
007040  000002                          RTI                       ;RESTORE THE STATUS (T-BIT)

007042  104400                  3$:     SCOPE

```

```

                                       ;*******************************************************************
                                       ;TEST 21:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
                                       ;        RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
                                       ;        STACK POINTER = R2
                                       ;*******************************************************************

ØØ7Ø44  Ø127Ø2  ØØØ6Ø4                 TST21:  MOV     #STACKØ,R2      ;SET UP THE STACK POINTER
ØØ7Ø5Ø  ØØ4767  ØØ4524                         JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

ØØ7Ø54  ØØØ24Ø                                 NOP
ØØ7Ø56  Ø75ØØ2                                 FADD+   R2              ;FLOATING ADD ON THE R2 STACK

ØØ7Ø6Ø  Ø13767  177776  171526  1$:    MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
ØØ7Ø66  Ø10267  171524                         MOV     R2,     $SP     ;SAVE THE STACK POINTER
ØØ7Ø72  Ø26767  171532  171514                 CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
ØØ71ØØ  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
ØØ71Ø2  1Ø4ØØØ                                 HLT                     ;PSW NOT EQUAL TO $ADDPS

ØØ71Ø4  1Ø5767  1715Ø4                         TSTB    $PSW            ;CHECK FOR ERROR
ØØ711Ø  1ØØ423                                 BMI     2$              ;BRANCH IF ERROR

ØØ7112  Ø12767  ØØØ61Ø  17155Ø                 MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
ØØ712Ø  Ø26767  171544  17147Ø                 CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
ØØ7126  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
ØØ713Ø  1Ø4ØØØ                                 HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

ØØ7132  Ø26767  171474  17145Ø                 CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
ØØ714Ø  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
ØØ7142  1Ø4ØØØ                                 HLT                     ;LEFT HALF OF ANSWER WRONG

ØØ7144  Ø26767  171464  17144Ø                 CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
ØØ7152  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
ØØ7154  1Ø4ØØØ                                 HLT                     ;RIGHT HALF OF ANSWER WRONG

ØØ7156  ØØØ451                                 BR      3$

ØØ716Ø  Ø12767  ØØØ6Ø4  1715Ø2  2$:    MOV     #STACKØ,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
ØØ7166  Ø26767  171476  171422                 CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
ØØ7174  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
ØØ7176  1Ø4ØØØ                                 HLT                     ;STACK POINTER FOULED UP

ØØ72ØØ  Ø22767  ØØ7Ø6Ø  171372                 CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
ØØ72Ø6  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
ØØ721Ø  1Ø4ØØØ                                 HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

ØØ7212  Ø26767  17142Ø  171362                 CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
ØØ722Ø  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
ØØ7222  1Ø4ØØØ                                 HLT                     ;RTI PSW NOT EQUAL TO 2ØØ

ØØ7224  Ø26767  171374  171352                 CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
ØØ7232  ØØ14Ø1                                 BEQ     .+4             ;BRANCH IF OK
ØØ7234  1Ø4ØØØ                                 HLT                     ;STK3 NOT EQUAL TO RAND.C

```

```
007236  026767  171364  171342          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
007244  001401                          BEQ     .+4             ;BRANCH IF OK
007246  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D

007250  026767  171344  171332          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
007256  001401                          BEQ     .+4             ;BRANCH IF OK
007260  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A

007262  026767  171334  171322          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
007270  001401                          BEQ     .+4             ;BRANCH IF OK
007272  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B

007274  012716  007302                  MOV     #3$,    (SP)    ;RESET THE STACK
007300  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)

007302  104400                  3$:     SCOPE



                                ;************************************************************************
                                ;TEST 22:       EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
                                ;       RAND.A,RAND.B - RAND.C,RAND.D = ANS1,ANS2
                                ;       STACK POINTER = R3
                                ;************************************************************************

007304  012703  000604          TST22:  MOV     #STACK0,R3      ;SET UP THE STACK POINTER
007310  004767  004264                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

007314  000240                          NOP
007316  075013                          FSUB+   R3              ;FLOATING SUBTRACT ON THE R3 STACK

007320  013767  177776  171266  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
007326  010367  171264                  MOV     R3,     $SP     ;SAVE THE STACK POINTER
007332  026767  171302  171254          CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
007340  001401                          BEQ     .+4             ;BRANCH IF OK
007342  104000                          HLT                     ;PSW NOT EQUAL TO $SUBPS

007344  105767  171244                  TSTB    $PSW            ;CHECK FOR ERROR
007350  100423                          BMI     2$              ;BRANCH IF ERROR

007352  012767  000610  171310          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
007360  026767  171304  171230          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
007366  001401                          BEQ     .+4             ;BRANCH IF OK
007370  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

007372  026767  171244  171210          CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
007400  001401                          BEQ     .+4             ;BRANCH IF OK
007402  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG

007404  026767  171234  171200          CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
007412  001401                          BEQ     .+4             ;BRANCH IF OK
007414  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG

007416  000451                          BR      3$

```

```
 007420  012767  000604  171242  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
 007426  026767  171236  171162          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
 007434  001401                          BEQ     .+4             ;BRANCH IF OK
 007436  104000                          HLT                     ;STACK POINTER FOULED UP

 007440  022767  007320  171132          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
 007446  001401                          BEQ     .+4             ;BRANCH IF OK
 007450  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

 007452  026767  171170  171122          CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
 007460  001401                          BEQ     .+4             ;BRANCH IF OK
 007462  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200

 007464  026767  171134  171112          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
 007472  001401                          BEQ     .+4             ;BRANCH IF OK
 007474  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C

 007476  026767  171124  171102          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
 007504  001401                          BEQ     .+4             ;BRANCH IF OK
 007506  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D

 007510  026767  171104  171072          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
 007516  001401                          BEQ     .+4             ;BRANCH IF OK
 007520  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A

 007522  026767  171074  171062          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
 007530  001401                          BEQ     .+4             ;BRANCH IF OK
 007532  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B

 007534  012716  007542                  MOV     #3$,    (SP)    ;RESET THE STACK
 007540  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)

 007542  104400                  3$:     SCOPE



                                 ;*************************************************************************
                                 ;TEST 23:        EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
                                 ;        RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
                                 ;        STACK POINTER = R4
                                 ;*************************************************************************

 007544  012704  000604          TST23:  MOV     #STACK0,R4      ;SET UP THE STACK POINTER
 007550  004767  004024                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

 007554  000240                          NOP
 007556  075024                          FMUL+   R4              ;FLOATING MULTIPLY ON THE R4 STACK

 007560  013767  177776  171026  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
 007566  010467  171024                  MOV     R4,     $SP     ;SAVE THE STACK POINTER
 007572  026767  171052  171014          CMP     $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
 007600  001401                          BEQ     .+4             ;BRANCH IF OK
 007602  104000                          HLT                     ;PSW NOT EQUAL TO $MULPS

 007604  105767  171004                  TSTB    $PSW            ;CHECK FOR ERROR
```

```
007610  100423                             BMI     2$               ;BRANCH IF ERROR

007612  012767  000610  171050             MOV     #STACK4,SAVSTK   ;SAVE PROPER STACK ADDRESS FOR TYPING
007620  026767  171044  170770             CMP     SAVSTK, $SP      ;CHECK THE STACK POINTER
007626  001401                             BEQ     .+4              ;BRANCH IF OK
007630  104000                             HLT                      ;STACK POINTER NOT EQUAL TO #STACK4

007632  026767  171014  170750             CMP     $MUL1,   ANS1    ;CHECK THE ANSWER
007640  001401                             BEQ     .+4              ;BRANCH IF OK
007642  104000                             HLT                      ;LEFT HALF OF ANSWER WRONG

007644  026767  171004  170740             CMP     $MUL2,   ANS2    ;CHECK THE ANSWER
007652  001401                             BEQ     .+4              ;BRANCH IF OK
007654  104000                             HLT                      ;RIGHT HALF OF ANSWER WRONG

007656  000451                             BR      3$

007660  012767  000604  171002  2$:        MOV     #STACK0,SAVSTK   ;SAVE STACK ADDRESS FOR TYPING
007666  026767  170776  170722             CMP     SAVSTK, $SP      ;CHECK THE STACK POINTER
007674  001401                             BEQ     .+4              ;BRANCH IF OK
007676  104000                             HLT                      ;STACK POINTER FOULED UP

007700  022767  007560  170672             CMP     #1$,     STK1    ;CHECK THE RTI ADDRESS ON THE STACK
007706  001401                             BEQ     .+4              ;BRANCH IF OK
007710  104000                             HLT                      ;RTI ADDRESS NOT EQUAL TO #1$

007712  026767  170740  170662             CMP     $MULER, STK2     ;CHECK THE PSW ON THE STACK
007720  001401                             BEQ     .+4              ;BRANCH IF OK
007722  104000                             HLT                      ;RTI PSW NOT EQUAL TO 200

007724  026767  170674  170652             CMP     RAND.C, STK3     ;CHECK THE DATA ON THE STACK
007732  001401                             BEQ     .+4              ;BRANCH IF OK
007734  104000                             HLT                      ;STK3 NOT EQUAL TO RAND.C

007736  026767  170664  170642             CMP     RAND.D, STK4     ;CHECK THE DATA ON THE STACK
007744  001401                             BEQ     .+4              ;BRANCH IF OK
007746  104000                             HLT                      ;STK4 NOT EQUAL TO RAND.D

007750  026767  170644  170632             CMP     RAND.A, STK5     ;CHECK THE DATA ON THE STACK
007756  001401                             BEQ     .+4              ;BRANCH IF OK
007760  104000                             HLT                      ;STK5 NOT EQUAL TO RAND.A

007762  026767  170634  170622             CMP     RAND.B, STK6     ;CHECK THE DATA ON THE STACK
007770  001401                             BEQ     .+4              ;BRANCH IF OK
007772  104000                             HLT                      ;STK6 NOT EQUAL TO RAND.B

007774  012716  010002                     MOV     #3$,     (SP)    ;RESET THE STACK
010000  000002                             RTI                      ;RESTORE THE STATUS (T-BIT)

010002  104400                     3$:     SCOPE

```

```

                                  ;*****************************************************************
                                  ;TEST 24:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
                                  ;        RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
                                  ;        STACK POINTER = R5
                                  ;*****************************************************************

 010004  012705  000604           TST24:  MOV     #STACK0,R5         ;SET UP THE STACK POINTER
 010010  004767  003564                   JSR     PC,     PUSHR      ;PUT THE DATA ON THE STACK

 010014  000240                           NOP
 010016  075035                           FDIV+   R5                 ;FLOATING DIVIDE ON THE R5 STACK

 010020  013767  177776  170566   1$:     MOV     @#PS,   $PSW       ;SAVE PROCESSOR STATUS
 010026  010567  170564                   MOV     R5,     $SP        ;SAVE THE STACK POINTER
 010032  026767  170622  170554           CMP     $DIVPS, $PSW       ;CHECK THE PROCESSOR STATUS
 010040  001401                           BEQ     .+4                ;BRANCH IF OK
 010042  104000                           HLT                        ;PSW NOT EQUAL TO $DIVPS

 010044  105767  170544                   TSTB    $PSW               ;CHECK FOR ERROR
 010050  100423                           BMI     2$                 ;BRANCH IF ERROR

 010052  012767  000610  170610           MOV     #STACK4,SAVSTK     ;SAVE PROPER STACK ADDRESS FOR TYPING
 010060  026767  170604  170530           CMP     SAVSTK, $SP        ;CHECK THE STACK POINTER
 010066  001401                           BEQ     .+4                ;BRANCH IF OK
 010070  104000                           HLT                        ;STACK POINTER NOT EQUAL TO #STACK4

 010072  026767  170564  170510           CMP     $DIV1,  ANS1       ;CHECK THE ANSWER
 010100  001401                           BEQ     .+4                ;BRANCH IF OK
 010102  104000                           HLT                        ;LEFT HALF OF ANSWER WRONG

 010104  026767  170554  170500           CMP     $DIV2,  ANS2       ;CHECK THE ANSWER
 010112  001401                           BEQ     .+4                ;BRANCH IF OK
 010114  104000                           HLT                        ;RIGHT HALF OF ANSWER WRONG

 010116  000451                           BR      3$

 010120  012767  000604  170542   2$:     MOV     #STACK0,SAVSTK     ;SAVE STACK ADDRESS FOR TYPING
 010126  026767  170536  170462           CMP     SAVSTK, $SP        ;CHECK THE STACK POINTER
 010134  001401                           BEQ     .+4                ;BRANCH IF OK
 010136  104000                           HLT                        ;STACK POINTER FOULED UP

 010140  022767  010020  170432           CMP     #1$,    STK1       ;CHECK THE RTI ADDRESS ON THE STACK
 010146  001401                           BEQ     .+4                ;BRANCH IF OK
 010150  104000                           HLT                        ;RTI ADDRESS NOT EQUAL TO #1$

 010152  026767  170510  170422           CMP     $DIVER, STK2       ;CHECK THE PSW ON THE STACK
 010160  001401                           BEQ     .+4                ;BRANCH IF OK
 010162  104000                           HLT                        ;RTI PSW NOT EQUAL TO 200

 010164  026767  170434  170412           CMP     RAND.C, STK3       ;CHECK THE DATA ON THE STACK
 010172  001401                           BEQ     .+4                ;BRANCH IF OK
 010174  104000                           HLT                        ;STK3 NOT EQUAL TO RAND.C

```

```
010176  026767  170424  170402            CMP     RAND.D, STK4      ;CHECK THE DATA ON THE STACK
010204  001401                            BEQ     .+4               ;BRANCH IF OK
010206  104000                            HLT                       ;STK4 NOT EQUAL TO RAND.D

010210  026767  170404  170372            CMP     RAND.A, STK5      ;CHECK THE DATA ON THE STACK
010216  001401                            BEQ     .+4               ;BRANCH IF OK
010220  104000                            HLT                       ;STK5 NOT EQUAL TO RAND.A

010222  026767  170374  170362            CMP     RAND.B, STK6      ;CHECK THE DATA ON THE STACK
010230  001401                            BEQ     .+4               ;BRANCH IF OK
010232  104000                            HLT                       ;STK6 NOT EQUAL TO RAND.B

010234  012716  010242                    MOV     #3$,    (SP)      ;RESET THE STACK
010240  000002                            RTI                       ;RESTORE THE STATUS (T-BIT)

010242  104400                    3$:     SCOPE


                                  ;*******************************************************************
                                  ;TEST 25:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
                                  ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
                                  ;       STACK POINTER = SP
                                  ;*******************************************************************

010244  012706  000604            TST25:  MOV     #STACK0,SP        ;SET UP THE STACK POINTER
010250  004767  003324                    JSR     PC,     PUSHR     ;PUT THE DATA ON THE STACK

010254  000240                            NOP
010256  075006                            FADD    SP                ;FLOATING ADD ON THE SP STACK

010260  013767  177776  170326    1$:     MOV     @#PS,   $PSW      ;SAVE PROCESSOR STATUS
010266  010667  170324                    MOV     SP,     $SP       ;SAVE THE STACK POINTER
010272  026767  170332  170314            CMP     $ADDPS, $PSW      ;CHECK THE PROCESSOR STATUS
010300  001401                            BEQ     .+4               ;BRANCH IF OK
010302  104000                            HLT                       ;PSW NOT EQUAL TO $ADDPS

010304  105767  170304                    TSTB    $PSW              ;CHECK FOR ERROR
010310  100424                            BMI     2$                ;BRANCH IF ERROR

010312  012767  000610  170350            MOV     #STACK4,SAVSTK    ;SAVE PROPER STACK ADDRESS FOR TYPING
010320  026767  170344  170270            CMP     SAVSTK, $SP       ;CHECK THE STACK POINTER
010326  001401                            BEQ     .+4               ;BRANCH IF OK
010330  104000                            HLT                       ;STACK POINTER NOT EQUAL TO #STACK4

010332  026767  170274  170250            CMP     $ADD1,  ANS1      ;CHECK THE ANSWER
010340  001401                            BEQ     .+4               ;BRANCH IF OK
010342  104000                            HLT                       ;LEFT HALF OF ANSWER WRONG

010344  026767  170264  170240            CMP     $ADD2,  ANS2      ;CHECK THE ANSWER
010352  001401                            BEQ     .+4               ;BRANCH IF OK
010354  104000                            HLT                       ;RIGHT HALF OF ANSWER WRONG

010356  024646                            CMP     -(SP),  -(SP)     ;RESTORE THE STACK
010360  000451                            BR      3$
```

```

010362  012767  000600  170300  2$:     MOV     #STK1,  SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
010370  026767  170274  170220          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
010376  001401                          BEQ     .+4             ;BRANCH IF OK
010400  104000                          HLT                     ;STACK POINTER FOULED UP

010402  022767  010260  170170          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
010410  001401                          BEQ     .+4             ;BRANCH IF OK
010412  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

010414  026767  170216  170160          CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
010422  001401                          BEQ     .+4             ;BRANCH IF OK
010424  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200

010426  026767  170172  170150          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
010434  001401                          BEQ     .+4             ;BRANCH IF OK
010436  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C

010440  026767  170162  170140          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
010446  001401                          BEQ     .+4             ;BRANCH IF OK
010450  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D

010452  026767  170142  170130          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
010460  001401                          BEQ     .+4             ;BRANCH IF OK
010462  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A

010464  026767  170132  170120          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
010472  001401                          BEQ     .+4             ;BRANCH IF OK
010474  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B

010476  012716  010504                  MOV     #3$,    (SP)    ;RESET THE STACK
010502  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)

010504  104400                  3$:     SCOPE



                                ;*********************************************************************
                                ;TEST 26:       EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
                                ;               RAND.A,RAND.B - RAND.C,RAND.D = ANS1,ANS2
                                ;               STACK POINTER = R0
                                ;*********************************************************************

010506  012700  000604          TST26:  MOV     #STACK0,R0      ;SET UP THE STACK POINTER
010512  004767  003062                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

010516  000240                          NOP
010520  075010                          FSUB+   R0              ;FLOATING SUBTRACT ON THE R0 STACK

010522  013767  177776  170064  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
010530  010067  170062                  MOV     R0,     $SP     ;SAVE THE STACK POINTER
010534  026767  170100  170052          CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
010542  001401                          BEQ     .+4             ;BRANCH IF OK
010544  104000                          HLT                     ;PSW NOT EQUAL TO $SUBPS

```

```
010546  105767  170042                  TSTB    $PSW            ;CHECK FOR ERROR
010552  100423                          BMI     2$              ;BRANCH IF ERROR

010554  012767  000610  170106          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
010562  026767  170102  170026          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
010570  001401                          BEQ     .+4             ;BRANCH IF OK
010572  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

010574  026767  170042  170006          CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
010602  001401                          BEQ     .+4             ;BRANCH IF OK
010604  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG

010606  026767  170032  167776          CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
010614  001401                          BEQ     .+4             ;BRANCH IF OK
010616  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG

010620  000451                          BR      3$

010622  012767  000604  170040  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
010630  026767  170034  167760          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
010636  001401                          BEQ     .+4             ;BRANCH IF OK
010640  104000                          HLT                     ;STACK POINTER FOULED UP

010642  022767  010522  167730          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
010650  001401                          BEQ     .+4             ;BRANCH IF OK
010652  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

010654  026767  167766  167720          CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
010662  001401                          BEQ     .+4             ;BRANCH IF OK
010664  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200

010666  026767  167732  167710          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
010674  001401                          BEQ     .+4             ;BRANCH IF OK
010676  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C

010700  026767  167722  167700          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
010706  001401                          BEQ     .+4             ;BRANCH IF OK
010710  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D

010712  026767  167702  167670          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
010720  001401                          BEQ     .+4             ;BRANCH IF OK
010722  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A

010724  026767  167672  167660          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
010732  001401                          BEQ     .+4             ;BRANCH IF OK
010734  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B

010736  012716  010744                  MOV     #3$,    (SP)    ;RESET THE STACK
010742  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)

010744  104400                  3$:     SCOPE

```

```

                                   ;*************************************************************************
                                   ;TEST 27:        EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
                                   ;        RAND,A,RAND,B * RAND,C,RAND,D = ANS1,ANS2
                                   ;        STACK POINTER = R1
                                   ;*************************************************************************

 010746  012701  000604            TST27:  MOV     #STACK0,R1          ;SET UP THE STACK POINTER
 010752  004767  002622                    JSR     PC,     PUSHR       ;PUT THE DATA ON THE STACK

 010756  000240                            NOP
 010760  075021                            FMUL+   R1                  ;FLOATING MULTIPLY ON THE R1 STACK

 010762  013767  177776  167624    1$:     MOV     @#PS,   $PSW        ;SAVE PROCESSOR STATUS
 010770  010167  167622                    MOV     R1,     $SP         ;SAVE THE STACK POINTER
 010774  026767  167650  167612            CMP     $MULPS, $PSW        ;CHECK THE PROCESSOR STATUS
 011002  001401                            BEQ     .+4                 ;BRANCH IF OK
 011004  104000                            HLT                         ;PSW NOT EQUAL TO $MULPS

 011006  105767  167602                    TSTB    $PSW                ;CHECK FOR ERROR
 011012  100423                            BMI     2$                  ;BRANCH IF ERROR

 011014  012767  000610  167646            MOV     #STACK4,SAVSTK      ;SAVE PROPER STACK ADDRESS FOR TYPING
 011022  026767  167642  167566            CMP     SAVSTK, $SP         ;CHECK THE STACK POINTER
 011030  001401                            BEQ     .+4                 ;BRANCH IF OK
 011032  104000                            HLT                         ;STACK POINTER NOT EQUAL TO #STACK4

 011034  026767  167612  167546            CMP     $MUL1,  ANS1        ;CHECK THE ANSWER
 011042  001401                            BEQ     .+4                 ;BRANCH IF OK
 011044  104000                            HLT                         ;LEFT HALF OF ANSWER WRONG

 011046  026767  167602  167536            CMP     $MUL2,  ANS2        ;CHECK THE ANSWER
 011054  001401                            BEQ     .+4                 ;BRANCH IF OK
 011056  104000                            HLT                         ;RIGHT HALF OF ANSWER WRONG

 011060  000451                            BR      3$

 011062  012767  000604  167600    2$:     MOV     #STACK0,SAVSTK      ;SAVE STACK ADDRESS FOR TYPING
 011070  026767  167574  167520            CMP     SAVSTK, $SP         ;CHECK THE STACK POINTER
 011076  001401                            BEQ     .+4                 ;BRANCH IF OK
 011100  104000                            HLT                         ;STACK POINTER FOULED UP

 011102  022767  010762  167470            CMP     #1$,    STK1        ;CHECK THE RTI ADDRESS ON THE STACK
 011110  001401                            BEQ     .+4                 ;BRANCH IF OK
 011112  104000                            HLT                         ;RTI ADDRESS NOT EQUAL TO #1$

 011114  026767  167536  167460            CMP     $MULER, STK2        ;CHECK THE PSW ON THE STACK
 011122  001401                            BEQ     .+4                 ;BRANCH IF OK
 011124  104000                            HLT                         ;RTI PSW NOT EQUAL TO 200

 011126  026767  167472  167450            CMP     RAND.C, STK3        ;CHECK THE DATA ON THE STACK
 011134  001401                            BEQ     .+4                 ;BRANCH IF OK
 011136  104000                            HLT                         ;STK3 NOT EQUAL TO RAND.C

 Ø1114Ø  026767  167462  16744Ø          CMP     RAND.D, STK4      ;CHECK THE DATA ON THE STACK
 Ø11146  ØØ14Ø1                          BEQ     .+4               ;BRANCH IF OK
 Ø1115Ø  1Ø4ØØØ                          HLT                       ;STK4 NOT EQUAL TO RAND.D

 Ø11152  Ø26767  167442  16743Ø          CMP     RAND.A, STK5      ;CHECK THE DATA ON THE STACK
 Ø1116Ø  ØØ14Ø1                          BEQ     .+4               ;BRANCH IF OK
 Ø11162  1Ø4ØØØ                          HLT                       ;STK5 NOT EQUAL TO RAND.A

 Ø11164  Ø26767  167432  16742Ø          CMP     RAND.B, STK6      ;CHECK THE DATA ON THE STACK
 Ø11172  ØØ14Ø1                          BEQ     .+4               ;BRANCH IF OK
 Ø11174  1Ø4ØØØ                          HLT                       ;STK6 NOT EQUAL TO RAND.B

 Ø11176  Ø12716  Ø112Ø4                  MOV     #3$,    (SP)      ;RESET THE STACK
 Ø112Ø2  ØØØØØ2                          RTI                       ;RESTORE THE STATUS (T-BIT)

 Ø112Ø4  1Ø44ØØ                  3$:     SCOPE



                                 ;*************************************************************************
                                 ;TEST 3Ø:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
                                 ;       RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
                                 ;       STACK POINTER = R2
                                 ;*************************************************************************

 Ø112Ø6  Ø127Ø2  ØØØ6Ø4          TST3Ø:  MOV     #STACKØ,R2        ;SET UP THE STACK POINTER
 Ø11212  ØØ4767  ØØ2362                  JSR     PC,     PUSHR     ;PUT THE DATA ON THE STACK

 Ø11216  ØØØ24Ø                          NOP
 Ø1122Ø  Ø75Ø32                          FDIV+   R2                ;FLOATING DIVIDE ON THE R2 STACK

 Ø11222  Ø13767  177776  167364  1$:     MOV     @#PS,   $PSW      ;SAVE PROCESSOR STATUS
 Ø1123Ø  Ø1Ø267  167362                  MOV     R2,     $SP       ;SAVE THE STACK POINTER
 Ø11234  Ø26767  16742Ø  167352          CMP     $DIVPS, $PSW      ;CHECK THE PROCESSOR STATUS
 Ø11242  ØØ14Ø1                          BEQ     .+4               ;BRANCH IF OK
 Ø11244  1Ø4ØØØ                          HLT                       ;PSW NOT EQUAL TO $DIVPS

 Ø11246  1Ø5767  167342                  TSTB    $PSW              ;CHECK FOR ERROR
 Ø11252  1ØØ423                          BMI     2$                ;BRANCH IF ERROR

 Ø11254  Ø12767  ØØØ61Ø  1674Ø6          MOV     #STACK4,SAVSTK    ;SAVE PROPER STACK ADDRESS FOR TYPING
 Ø11262  Ø26767  1674Ø2  167326          CMP     SAVSTK, $SP       ;CHECK THE STACK POINTER
 Ø1127Ø  ØØ14Ø1                          BEQ     .+4               ;BRANCH IF OK
 Ø11272  1Ø4ØØØ                          HLT                       ;STACK POINTER NOT EQUAL TO #STACK4

 Ø11274  Ø26767  167362  1673Ø6          CMP     $DIV1,  ANS1      ;CHECK THE ANSWER
 Ø113Ø2  ØØ14Ø1                          BEQ     .+4               ;BRANCH IF OK
 Ø113Ø4  1Ø4ØØØ                          HLT                       ;LEFT HALF OF ANSWER WRONG

 Ø113Ø6  Ø26767  167352  167276          CMP     $DIV2,  ANS2      ;CHECK THE ANSWER
 Ø11314  ØØ14Ø1                          BEQ     .+4               ;BRANCH IF OK
 Ø11316  1Ø4ØØØ                          HLT                       ;RIGHT HALF OF ANSWER WRONG

 Ø1132Ø  ØØØ451                          BR      3$

```

```
011322  012767  000604  167340  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
011330  026767  167334  167260          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
011336  001401                          BEQ     .+4             ;BRANCH IF OK
011340  104000                          HLT                     ;STACK POINTER FOULED UP

011342  022767  011222  167230          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
011350  001401                          BEQ     .+4             ;BRANCH IF OK
011352  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

011354  026767  167306  167220          CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
011362  001401                          BEQ     .+4             ;BRANCH IF OK
011364  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200

011366  026767  167232  167210          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
011374  001401                          BEQ     .+4             ;BRANCH IF OK
011376  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C

011400  026767  167222  167200          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
011406  001401                          BEQ     .+4             ;BRANCH IF OK
011410  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D

011412  026767  167202  167170          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
011420  001401                          BEQ     .+4             ;BRANCH IF OK
011422  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A

011424  026767  167172  167160          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
011432  001401                          BEQ     .+4             ;BRANCH IF OK
011434  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B

011436  012716  011444                  MOV     #3$,    (SP)    ;RESET THE STACK
011442  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)

011444  104400                  3$:     SCOPE



                                ;*************************************************************************
                                ;TEST 31:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
                                ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
                                ;       STACK POINTER = R3
                                ;*************************************************************************

011446  012703  000604          TST31:  MOV     #STACK0,R3      ;SET UP THE STACK POINTER
011452  004767  002122                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

011456  000240                          NOP
011460  075003                          FADD+   R3              ;FLOATING ADD ON THE R3 STACK

011462  013767  177776  167124  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
011470  010367  167122                  MOV     R3,     $SP     ;SAVE THE STACK POINTER
011474  026767  167130  167112          CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
011502  001401                          BEQ     .+4             ;BRANCH IF OK
011504  104000                          HLT                     ;PSW NOT EQUAL TO $ADDPS

011506  105767  167102                  TSTB    $PSW            ;CHECK FOR ERROR
```

```
Ø11512  1ØØ423                               BMI     2$                  ;BRANCH IF ERROR

Ø11514  Ø12767  ØØØ61Ø  167146               MOV     #STACK4,SAVSTK      ;SAVE PROPER STACK ADDRESS FOR TYPING
Ø11522  Ø26767  167142  167Ø66               CMP     SAVSTK, $SP         ;CHECK THE STACK POINTER
Ø1153Ø  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø11532  1Ø4ØØØ                               HLT                         ;STACK POINTER NOT EQUAL TO #STACK4

Ø11534  Ø26767  167Ø72  167Ø46               CMP     $ADD1,   ANS1       ;CHECK THE ANSWER
Ø11542  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø11544  1Ø4ØØØ                               HLT                         ;LEFT HALF OF ANSWER WRONG

Ø11546  Ø26767  167Ø62  167Ø36               CMP     $ADD2,   ANS2       ;CHECK THE ANSWER
Ø11554  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø11556  1Ø4ØØØ                               HLT                         ;RIGHT HALF OF ANSWER WRONG

Ø1156Ø  ØØØ451                               BR      3$

Ø11562  Ø12767  ØØØ6Ø4  1671ØØ      2$:      MOV     #STACKØ,SAVSTK      ;SAVE STACK ADDRESS FOR TYPING
Ø1157Ø  Ø26767  167Ø74  167Ø2Ø               CMP     SAVSTK, $SP         ;CHECK THE STACK POINTER
Ø11576  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø116ØØ  1Ø4ØØØ                               HLT                         ;STACK POINTER FOULED UP

Ø116Ø2  Ø22767  Ø11462  16677Ø               CMP     #1$,     STK1       ;CHECK THE RTI ADDRESS ON THE STACK
Ø1161Ø  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø11612  1Ø4ØØØ                               HLT                         ;RTI ADDRESS NOT EQUAL TO #1$

Ø11614  Ø26767  167Ø16  16676Ø               CMP     $ADDER, STK2        ;CHECK THE PSW ON THE STACK
Ø11622  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø11624  1Ø4ØØØ                               HLT                         ;RTI PSW NOT EQUAL TO 2ØØ

Ø11626  Ø26767  166772  16675Ø               CMP     RAND.C, STK3        ;CHECK THE DATA ON THE STACK
Ø11634  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø11636  1Ø4ØØØ                               HLT                         ;STK3 NOT EQUAL TO RAND.C

Ø1164Ø  Ø26767  166762  16674Ø               CMP     RAND.D, STK4        ;CHECK THE DATA ON THE STACK
Ø11646  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø1165Ø  1Ø4ØØØ                               HLT                         ;STK4 NOT EQUAL TO RAND.D

Ø11652  Ø26767  166742  16673Ø               CMP     RAND.A, STK5        ;CHECK THE DATA ON THE STACK
Ø1166Ø  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø11662  1Ø4ØØØ                               HLT                         ;STK5 NOT EQUAL TO RAND.A

Ø11664  Ø26767  166732  16672Ø               CMP     RAND.B, STK6        ;CHECK THE DATA ON THE STACK
Ø11672  ØØ14Ø1                               BEQ     .+4                 ;BRANCH IF OK
Ø11674  1Ø4ØØØ                               HLT                         ;STK6 NOT EQUAL TO RAND.B

Ø11676  Ø12716  Ø117Ø4                       MOV     #3$,     (SP)       ;RESET THE STACK
Ø117Ø2  ØØØØØ2                               RTI                         ;RESTORE THE STATUS (T-BIT)

Ø117Ø4  1Ø44ØØ                      3$:      SCOPE
```

```

                                     ;*******************************************************************
                                     ;TEST 32:        EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
                                     ;        RAND,A,RAND,B - RAND,C,RAND,D = ANS1,ANS2
                                     ;        STACK POINTER = R4
                                     ;*******************************************************************

011706  012704  000604               TST32:  MOV     #STACK0,R4      ;SET UP THE STACK POINTER
011712  004767  001662                       JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

011716  000240                               NOP
011720  075014                               FSUB+   R4              ;FLOATING SUBTRACT ON THE R4 STACK

011722  013767  177776  166664       1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
011730  010467  166662                       MOV     R4,     $SP     ;SAVE THE STACK POINTER
011734  026767  166700  166652               CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
011742  001401                               BEQ     .+4             ;BRANCH IF OK
011744  104000                               HLT                     ;PSW NOT EQUAL TO $SUBPS

011746  105767  166642                       TSTB    $PSW            ;CHECK FOR ERROR
011752  100423                               BMI     2$              ;BRANCH IF ERROR

011754  012767  000610  166706               MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
011762  026767  166702  166626               CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
011770  001401                               BEQ     .+4             ;BRANCH IF OK
011772  104000                               HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

011774  026767  166642  166606               CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
012002  001401                               BEQ     .+4             ;BRANCH IF OK
012004  104000                               HLT                     ;LEFT HALF OF ANSWER WRONG

012006  026767  166632  166576               CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
012014  001401                               BEQ     .+4             ;BRANCH IF OK
012016  104000                               HLT                     ;RIGHT HALF OF ANSWER WRONG

012020  000451                               BR      3$

012022  012767  000604  166640       2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
012030  026767  166634  166560               CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
012036  001401                               BEQ     .+4             ;BRANCH IF OK
012040  104000                               HLT                     ;STACK POINTER FOULED UP

012042  022767  011722  166530               CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
012050  001401                               BEQ     .+4             ;BRANCH IF OK
012052  104000                               HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

012054  026767  166566  166520               CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
012062  001401                               BEQ     .+4             ;BRANCH IF OK
012064  104000                               HLT                     ;RTI PSW NOT EQUAL TO 200

012066  026767  166532  166510               CMP     RAND,C, STK3    ;CHECK THE DATA ON THE STACK
012074  001401                               BEQ     .+4             ;BRANCH IF OK
012076  104000                               HLT                     ;STK3 NOT EQUAL TO RAND,C

```

```
012100  026767  166522  166500           CMP     RAND.D, STK4      ;CHECK THE DATA ON THE STACK
012106  001401                           BEQ     .+4               ;BRANCH IF OK
012110  104000                           HLT                       ;STK4 NOT EQUAL TO RAND.D

012112  026767  166502  166470           CMP     RAND.A, STK5      ;CHECK THE DATA ON THE STACK
012120  001401                           BEQ     .+4               ;BRANCH IF OK
012122  104000                           HLT                       ;STK5 NOT EQUAL TO RAND.A

012124  026767  166472  166460           CMP     RAND.B, STK6      ;CHECK THE DATA ON THE STACK
012132  001401                           BEQ     .+4               ;BRANCH IF OK
012134  104000                           HLT                       ;STK6 NOT EQUAL TO RAND.B

012136  012716  012144                   MOV     #3$,    (SP)      ;RESET THE STACK
012142  000002                           RTI                       ;RESTORE THE STATUS (T-BIT)

012144  104400                   3$:     SCOPE



                                 ;*******************************************************************
                                 ;TEST 33:        EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
                                 ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
                                 ;       STACK POINTER = R5
                                 ;*******************************************************************

012146  012705  000604           TST33:  MOV     #STACK0,R5        ;SET UP THE STACK POINTER
012152  004767  001422                   JSR     PC,     PUSHR     ;PUT THE DATA ON THE STACK

012156  000240                           NOP
012160  075025                           FMUL+   R5                ;FLOATING MULTIPLY ON THE R5 STACK

012162  013767  177776  166424   1$:     MOV     @#PS,   $PSW      ;SAVE PROCESSOR STATUS
012170  010567  166422                   MOV     R5,     $SP       ;SAVE THE STACK POINTER
012174  026767  166450  166412           CMP     $MULPS, $PSW      ;CHECK THE PROCESSOR STATUS
012202  001401                           BEQ     .+4               ;BRANCH IF OK
012204  104000                           HLT                       ;PSW NOT EQUAL TO $MULPS

012206  105767  166402                   TSTB    $PSW              ;CHECK FOR ERROR
012212  100423                           BMI     2$                ;BRANCH IF ERROR

012214  012767  000610  166446           MOV     #STACK4,SAVSTK    ;SAVE PROPER STACK ADDRESS FOR TYPING
012222  026767  166442  166366           CMP     SAVSTK, $SP       ;CHECK THE STACK POINTER
012230  001401                           BEQ     .+4               ;BRANCH IF OK
012232  104000                           HLT                       ;STACK POINTER NOT EQUAL TO #STACK4

012234  026767  166412  166346           CMP     $MUL1,  ANS1      ;CHECK THE ANSWER
012242  001401                           BEQ     .+4               ;BRANCH IF OK
012244  104000                           HLT                       ;LEFT HALF OF ANSWER WRONG

012246  026767  166402  166336           CMP     $MUL2,  ANS2      ;CHECK THE ANSWER
012254  001401                           BEQ     .+4               ;BRANCH IF OK
012256  104000                           HLT                       ;RIGHT HALF OF ANSWER WRONG

012260  000451                           BR      3$

```

```
012262  012767  000604  166400  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
012270  026767  166374  166320          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
012276  001401                          BEQ     .+4             ;BRANCH IF OK
012300  104000                          HLT                     ;STACK POINTER FOULED UP

012302  022767  012162  166270          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
012310  001401                          BEQ     .+4             ;BRANCH IF OK
012312  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

012314  026767  166336  166260          CMP     $MULER, STK2    ;CHECK THE PSW ON THE STACK
012322  001401                          BEQ     .+4             ;BRANCH IF OK
012324  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200

012326  026767  166272  166250          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
012334  001401                          BEQ     .+4             ;BRANCH IF OK
012336  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C

012340  026767  166262  166240          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
012346  001401                          BEQ     .+4             ;BRANCH IF OK
012350  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D

012352  026767  166242  166230          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
012360  001401                          BEQ     .+4             ;BRANCH IF OK
012362  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A

012364  026767  166232  166220          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
012372  001401                          BEQ     .+4             ;BRANCH IF OK
012374  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B

012376  012716  012404                  MOV     #3$,    (SP)    ;RESET THE STACK
012402  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)

012404  104400                  3$:     SCOPE



                                ;*******************************************************************
                                ;TEST 34:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
                                ;       RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
                                ;       STACK POINTER = SP
                                ;*******************************************************************

012406  012706  000604          TST34:  MOV     #STACK0,SP      ;SET UP THE STACK POINTER
012412  004767  001162                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

012416  000240                          NOP
012420  075036                          FDIV+   SP              ;FLOATING DIVIDE ON THE SP STACK

012422  013767  177776  166164  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
012430  010667  166162                  MOV     SP,     $SP     ;SAVE THE STACK POINTER
012434  026767  166220  166152          CMP     $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
012442  001401                          BEQ     .+4             ;BRANCH IF OK
012444  104000                          HLT                     ;PSW NOT EQUAL TO $DIVPS

012446  105767  166142                  TSTB    $PSW            ;CHECK FOR ERROR
```

```
012452  100424                              BMI     2$              ;BRANCH IF ERROR

012454  012767  000610  166206              MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
012462  026767  166202  166126              CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
012470  001401                              BEQ     .+4             ;BRANCH IF OK
012472  104000                              HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

012474  026767  166162  166106              CMP     $DIV1,  ANS1    ;CHECK THE ANSWER
012502  001401                              BEQ     .+4             ;BRANCH IF OK
012504  104000                              HLT                     ;LEFT HALF OF ANSWER WRONG

012506  026767  166152  166076              CMP     $DIV2,  ANS2    ;CHECK THE ANSWER
012514  001401                              BEQ     .+4             ;BRANCH IF OK
012516  104000                              HLT                     ;RIGHT HALF OF ANSWER WRONG

012520  024646                              CMP     -(SP),  -(SP)   ;RESTORE THE STACK
012522  000451                              BR      3$

012524  012767  000600  166136      2$:     MOV     #STK1,  SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
012532  026767  166132  166056              CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
012540  001401                              BEQ     .+4             ;BRANCH IF OK
012542  104000                              HLT                     ;STACK POINTER FOULED UP

012544  022767  012422  166026              CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
012552  001401                              BEQ     .+4             ;BRANCH IF OK
012554  104000                              HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

012556  026767  166104  166016              CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
012564  001401                              BEQ     .+4             ;BRANCH IF OK
012566  104000                              HLT                     ;RTI PSW NOT EQUAL TO 200

012570  026767  166030  166006              CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
012576  001401                              BEQ     .+4             ;BRANCH IF OK
012600  104000                              HLT                     ;STK3 NOT EQUAL TO RAND.C

012602  026767  166020  165776              CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
012610  001401                              BEQ     .+4             ;BRANCH IF OK
012612  104000                              HLT                     ;STK4 NOT EQUAL TO RAND.D

012614  026767  166000  165766              CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
012622  001401                              BEQ     .+4             ;BRANCH IF OK
012624  104000                              HLT                     ;STK5 NOT EQUAL TO RAND.A

012626  026767  165770  165756              CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
012634  001401                              BEQ     .+4             ;BRANCH IF OK
012636  104000                              HLT                     ;STK6 NOT EQUAL TO RAND.B

012640  012716  012646                      MOV     #3$,    (SP)    ;RESET THE STACK
012644  000002                              RTI                     ;RESTORE THE STATUS (T-BIT)

012646  104400                      3$:     SCOPE

```

```

012650  062767  000001  166130              ADD     #1,     PCNT+2  ;COUNT PASSES
012656  005567  166122                      ADC     PCNT

012662     001                      DONE:
012662  032737  002000  177570              BIT     #SW10,@#SWR     ;RING THE BELL?
012670  001002                              BNE     1$              ;NO!
012672  000004  000007                      TYPE    ,BELL
012676  005046                      1$:     CLR     -(6)            ;CLEAR TRACE TRAP
012700  032737  010000  177570              BIT     #SW12,@#SWR     ;RUN WITH TRT?
012706  001010                              BNE     2$
012710  005167  000044                      COM     ,TBIT
012714  100005                              BPL     2$
012716  052716  000020                      BIS     #20,(6)         ;SET TRACE TRAP
012722  012746  012754                      MOV     #3$,-(6)        ;JUMP TO START OF TEST
012726  000002                              RTI
012730  012746  012736              2$:     MOV     #4$,-(6)        ;JUMP TO START OF TEST
012734  000002                              RTI                     ;RETURN
012736  013700  000042              4$:     MOV     @#42,R0         ;GET MONITOR ADDRESS
012742  001404                              BEQ     3$              ;IF NONE
012744  004710                              JSR     7,(0)           ;GO TO MONITOR
012746  000240                              NOP
012750  000240                              NOP
012752  000240                              NOP
012754  000137  001146              3$:     JMP     @#START         ;RETURN

012760  000000                      .TBIT:  0



                                    ;SUBROUTINE TO READ TTY INPUT AND SAVE OCTAL NUMBER

012762  004767  002124              READIN: JSR     PC,READS
012766  012702  015212                      MOV     #INPUT,R2
012772  012501                              MOV     (R5)+,R1
012774  005011                              CLR     (R1)
012776  112203                      1$:     MOVB    (R2)+,R3        ;STORE DATA
013000  001420                              BEQ     4$              ;BRANCH IF DONE
013002  162703  000060                      SUB     #60,R3
013006  000241                              CLC
013010  032703  177770                      BIT     #177770,R3
013014  001010                              BNE     2$
013016  006311                              ASL     (R1)
013020  103407                              BCS     3$
013022  006311                              ASL     (R1)
013024  103405                              BCS     3$
013026  006311                              ASL     (R1)
013030  103403                              BCS     3$
013032  050311                              BIS     R3,(R1)
013034  000760                              BR      1$
013036  000261                      2$:     SEC                     ;SET C-BIT IF NOT
013040  000244                      3$:     CLZ
013042  000205                      4$:     RTS     R5
```

```

Ø13Ø44  Ø16746  165552          $PUSH:  MOV     RAND,B,-(SP)
Ø13Ø5Ø  Ø16746  165544                  MOV     RAND,A,-(SP)
Ø13Ø54  Ø16746  165546                  MOV     RAND,D,-(SP)
Ø13Ø6Ø  Ø16746  16554Ø                  MOV     RAND,C,-(SP)
Ø13Ø64  ØØØ134                  $POLSH: JMP     @(R4)+

Ø13Ø66  ØØ5767  165536          $POPAD: TST     $ADDPS          ;CHECK FOR ERROR
Ø13Ø72  ØØ1145                          BNE     $SKIP           ;BRANCH IF PS SET
Ø13Ø74  Ø32716  Ø776ØØ                  BIT     #776ØØ, (SP)    ;CHECK FOR  ZERO
Ø131ØØ  ØØ1Ø1Ø                          BNE     1$              ;BRANCH IF NOT
Ø131Ø2  Ø13767  177776  16552Ø          MOV     @#PS,   $ADDPS  ;Z-BIT IN PSW
Ø1311Ø  ØØ5Ø67  165516                  CLR     $ADD1           ;ZERO ANSWER
Ø13114  ØØ5Ø67  165514                  CLR     $ADD2
Ø1312Ø  ØØØ532                          BR      $SKIP

Ø13122  ØØ5716                  1$:     TST     (SP)            ;GET N-BIT, CLEAR C-BIT, V-BIT
Ø13124  Ø13767  177776  165476          MOV     @#PS,   $ADDPS  ;SET THE PSW SAVE
Ø13132  Ø12667  165474                  MOV     (SP)+,  $ADD1
Ø13136  Ø12667  165472                  MOV     (SP)+,  $ADD2
Ø13142  ØØØ134                          JMP     @(R4)+

Ø13144  ØØ5767  16547Ø          $POPSB: TST     $SUBPS          ;CHECK FOR ERROR
Ø1315Ø  ØØ1116                          BNE     $SKIP           ;BRANCH IF PS SET
Ø13152  Ø32716  Ø776ØØ                  BIT     #776ØØ, (SP)    ;CHECK FOR  ZERO
Ø13156  ØØ1Ø1Ø                          BNE     1$              ;BRANCH IF NOT
Ø1316Ø  Ø13767  177776  165452          MOV     @#PS,   $SUBPS  ;Z-BIT IN PSW
Ø13166  ØØ5Ø67  16545Ø                  CLR     $SUB1           ;ZERO ANSWER
Ø13172  ØØ5Ø67  165446                  CLR     $SUB2
Ø13176  ØØØ5Ø3                          BR      $SKIP

Ø132ØØ  ØØ5716                  1$:     TST     (SP)            ;GET N-BIT, CLEAR C-BIT, V-BIT
Ø132Ø2  Ø13767  177776  16543Ø          MOV     @#PS,   $SUBPS  ;SET THE PSW SAVE
Ø1321Ø  Ø12667  165426                  MOV     (SP)+,  $SUB1
Ø13214  Ø12667  165424                  MOV     (SP)+,  $SUB2
Ø1322Ø  ØØØ134                          JMP     @(R4)+

Ø13222  ØØ5767  165422          $POPML: TST     $MULPS          ;CHECK FOR ERROR
Ø13226  ØØ1Ø67                          BNE     $SKIP           ;BRANCH IF PS SET
Ø1323Ø  Ø32716  Ø776ØØ                  BIT     #776ØØ, (SP)    ;CHECK FOR  ZERO
Ø13234  ØØ1Ø1Ø                          BNE     1$              ;BRANCH IF NOT
Ø13236  Ø13767  177776  1654Ø4          MOV     @#PS,   $MULPS  ;Z-BIT IN PSW
Ø13244  ØØ5Ø67  1654Ø2                  CLR     $MUL1           ;ZERO ANSWER
Ø1325Ø  ØØ5Ø67  1654ØØ                  CLR     $MUL2
Ø13254  ØØØ454                          BR      $SKIP

Ø13256  ØØ5716                  1$:     TST     (SP)            ;GET N-BIT, CLEAR C-BIT, V-BIT
Ø1326Ø  Ø13767  177776  165362          MOV     @#PS,   $MULPS  ;SET THE PSW SAVE
Ø13266  Ø12667  16536Ø                  MOV     (SP)+,  $MUL1
Ø13272  Ø12667  165356                  MOV     (SP)+,  $MUL2
Ø13276  ØØØ134                          JMP     @(R4)+
013300  032767  077600  165316  $POPDV: BIT     #77600,RAND,C   ;CHECK FOR DIVIDED BY ZERO
013306  001010                          BNE     1$
013310  000277                          SCC                     ;SET ALL CONDITION CODES
013312  000244                          CLZ                     ;CLEAR THE Z-BIT
013314  013767  177776  165344          MOV     @#PS,   $DIVER  ;SET UP DIVIDE BY ZERO CC'S
013322  012767  000340  165330          MOV     #340,   $DIVPS  ;SET UP PSW
013330  005767  165324          1$:     TST     $DIVPS          ;CHECK FOR ERROR
013334  001024                          BNE     $SKIP           ;BRANCH IF PS SET
013336  032716  077600                  BIT     #77600, (SP)    ;CHECK FOR  ZERO
013342  001010                          BNE     2$              ;BRANCH IF NOT
013344  013767  177776  165306          MOV     @#PS,   $DIVPS  ;Z-BIT IN PSW
013352  005067  165304                  CLR     $DIV1           ;ZERO ANSWER
013356  005067  165302                  CLR     $DIV2
013362  000411                          BR      $SKIP

013364  005716                  2$:     TST     (SP)            ;GET N-BIT, CLEAR C-BIT, V-BIT
013366  013767  177776  165264          MOV     @#PS,   $DIVPS  ;SET THE PSW SAVE
013374  012667  165262                  MOV     (SP)+,  $DIV1
013400  012667  165260                  MOV     (SP)+,  $DIV2
013404  000134                          JMP     @(R4)+

013406  022626                  $SKIP:  CMP     (SP)+,  (SP)+   ;POP GARBAGE OFF THE STACK
013410  000134                          JMP     @(R4)+

013412  000204                  $EXIT:  RTS     R4              ;EXIT POLISH MODE

013414  016500  000002          $ERR:   MOV     2(5),   R0      ;PUT CODE INTO R0
013420  022700  004003          $ERRA:  CMP     #4003,  R0      ;CHECK FOR DIVIDE BY ZERO
013424  001464                          BEQ     8$              ;SKIP OUT

013426  122700  000003                  CMPB    #3,     R0      ;CHECK FOR OVERFLOW
013432  001006                          BNE     2$              ;BRANCH IF NOT
013434  000257                          CCC                     ;CLEAR ALL CONDITION CODES
013436  000262                          SEV                     ;SET THE V-BIT
013440  013767  177776  165146          MOV     @#PS,   $PSW    ;SET UP PSW FOR OVERFLOW
013446  000405                          BR      3$

013450  000257                  2$:     CCC                     ;CLEAR ALL CONDITION CODES
013452  000272                          SNV                     ;SET N-BIT AND V-BIT
013454  013767  177776  165132          MOV     @#PS,   $PSW    ;SET UP PSW FOR UNDERFLOW
013462  105000                  3$:     CLRB    R0              ;CLEAR LOW BYTE
013464  000300                          SWAB    R0              ;HIGH BYTE INTO LOW
013466  162700  000002                  SUB     #2,     R0      ;CHECK FOR ADD/SUB
013472  001021                          BNE     5$              ;BRANCH IF NOT
013474  005767  165130                  TST     $ADDPS          ;CHECK FOR ADD
013500  001007                          BNE     4$              ;BRANCH IF NOT
013502  016767  165106  165126          MOV     $PSW,   $ADDER  ;SET UP ADD ERROR PSW
013510  012767  000340  165112          MOV     #340,   $ADDPS  ;SET UP ADD PSW
013516  000427                          BR      8$

013520  016767  165070  165120  4$:     MOV     $PSW,   $SUBER  ;SET UP SUBTRACT ERROR PSW
013526  012767  000340  165104          MOV     #340,   $SUBPS  ;SET UP SUBTRACT PSW
013534  000420                          BR      8$
```

```

013536  162700  000004          5$:     SUB     #4,     R0      ;CHECK FOR MUL
013542  003407                          BLE     6$              ;BRANCH IF NOT
013544  016767  165044  165104          MOV     $PSW,   $MULER  ;SET UP MULTIPLY ERROR PSW
013552  012767  000340  165070          MOV     #340,   $MULPS  ;SET UP MULTIPLY PSW
013560  000406                          BR      8$

013562  016767  165026  165076  6$:     MOV     $PSW,   $DIVER  ;SET UP DIVIDE ERROR PSW
013570  012767  000340  165062  7$:     MOV     #340,   $DIVPS  ;SET UP DIVIDE PSW
013576  000205                  8$:     RTS     R5              ;RETURN TO FORTRAN

                                ;SUBROUTINE TO PUSH DATA ONTO STACK

013600  016767  165016  165004  PUSHR:  MOV     RAND.B, STACK6  ;PUT DATA ON THE STACK
013606  016767  165006  164774          MOV     RAND.A, STACK4
013614  016767  165006  164764          MOV     RAND.D, STACK2
013622  016767  164776  164754          MOV     RAND.C, STACK0
013630  011637  000244                  MOV     (SP),   @#244   ;SET UP TRAP VECTOR
013634  062737  000004  000244          ADD     #4,     @#244
013642  000207                          RTS     PC

013644  032737  000400  177570  SCOPE$: BIT     #SW08,@#SWR     ;KILL LOUB OR LOOP ON SPEC. TEST
013652  001412                          BEQ     1$
013654  013767  177570  000134          MOV     @#SWR,  SCOTMP  ;SAVE SWR
013662  042767  177600  000126          BIC     #177600,SCOTMP  ;CLR ALL BUT TEST NO.
013670  126767  000122  165102          CMPB    SCOTMP,ICNT     ;ON RIGHT TEST?   -SW6-0-
013676  001434                          BEQ     OVER$
013700  032737  040000  177570  1$:     BIT     #SW14,@#SWR     ;LOOP ON TEST
013706  001026                          BNE     KIT$
013710  032737  004000  177570          BIT     #SW11,@#SWR     ;KILL ITERATIONS
013716  001012                          BNE     SVLAD$
013720  105767  165055                  TSTB    ICNT+1
013724  001404                          BEQ     2$              ;BRANCH IF FIRST
013726  126767  000062  165045          CMPB    TIMES,ICNT+1    ;DONE?
013734  001013                          BNE     KIT$            ;BRANCH IF NOT
013736  112767  000001  165035  2$:     MOVB    #1,ICNT+1       ;FIRST ITERATION
013744  105267  165030          SVLAD$: INCB    ICNT            ;COUNT TEST NUMBERS
013750  011667  000036                  MOV     (6),LAD$        ;SAVE LOOP ADDRESS
013754  016737  165020  177570          MOV     ICNT,@#DISPLAY  ;DISPLAY TEST NO. AND ITERATION COUNT
013762  000002                          RTI                     ;RETURN

013764  105267  165011          KIT$:   INCB    ICNT+1
013770  016737  165004  177570  OVER$:  MOV     ICNT,@#DISPLAY  ;SET UP DISPLAY
013776  005767  000010                  TST     LAD$            ;FIRST ONE?
014002  001760                          BEQ     SVLAD$
014004  016716  000002                  MOV     LAD$,(6)        ;FUDGE RETURN ADDRESS
014010  000002                          RTI                     ;FIXES PS

014012  000000                  LAD$:   0                       ;LOOP ADDRESS
014014  000377                  TIMES:  377                     ;RUN 377 TIMES
014016  000000                  SCOTMP: 0
```

```
Ø14Ø2Ø  Ø32737  ØØ2ØØØ  17757Ø  HLTS:   BIT     #SW1Ø,@#SWR        ;BELL ON ERROR?
Ø14Ø26  ØØ14Ø2                          BEQ     1$                 ;NO - SKIP
Ø14Ø3Ø  ØØØØØ4  ØØØØØ7                  TYPE    ,BELL              ;RING BELL
Ø14Ø34  ØØ5267  164742          1$:     INC     ERRORS             ;COUNT THE NUMBER OF ERRORS
Ø14Ø4Ø  Ø32737  Ø2ØØØØ  17757Ø          BIT     #SW13,@#SWR        ;SKIP TYPEOUT IF SET
Ø14Ø46  ØØ1Ø17                          BNE     2$                 ;SKIP TYPEOUTS
Ø14Ø5Ø  ØØØØØ4  Ø15362                  TYPE    ,RETURN
Ø14Ø54  Ø11667  ØØØØ6Ø                  MOV     (6),HLTADS         ;PUT ADDRESS OF INSTRUCTION ON STACK
Ø14Ø6Ø  162767  ØØØØØ2  ØØØØ52          SUB     #2,HLTADS
Ø14Ø66  Ø167Ø5  ØØØØ46                  MOV     HLTADS,TTY         ;TYPE HLTADS IN OCTAL
Ø14Ø72  ØØ4767  ØØ13ØØ                  JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
Ø14Ø76  ØØØØØ4  Ø1537Ø                  TYPE    ,SPACE+3
Ø141Ø2  ØØ4767  ØØØØ34                  JSR     PC,     ERRORS     ;GO TO USER ERROR ROUTINE
Ø141Ø6  ØØ5737  17757Ø          2$:     TST     @#SWR              ;HALT ON ERROR
Ø14112  1ØØØØ1                          BPL     .+4                ;SKIP IF CONTINUE
Ø14114  ØØØØØØ                          HALT                       ;HALT ON ERROR!
Ø14116  Ø32737  ØØ1ØØØ  17757Ø          BIT     #SWØ9,@#SWR        ;CHECK FOR INHIBIT LOOP ON ERROR
Ø14124  ØØ1ØØ1                          BNE     .+4                ;SKIP IF LOOP ON ERROR
Ø14126  ØØØØØ2                          RTI
Ø1413Ø  1Ø5Ø67  164645                  CLRB    ICNT+1
Ø14134  ØØØ167  177624                  JMP     KITS               ;LOOP ON TEST UNTIL NO ERRORS

Ø1414Ø  ØØØØØØ                  HLTADS: Ø

Ø14142  Ø1ØØ46                  ERRORS: MOV     RØ,     -(SP)      ;SAVE RØ
Ø14144  Ø1Ø146                          MOV     R1,     -(SP)      ;SAVE R1
Ø14146  ØØØØØ4  Ø1537Ø                  TYPE,   SPACE+3
Ø14152  Ø167Ø5  164442                  MOV     RAND,A,TTY         ;TYPE RAND,A IN OCTAL
Ø14156  ØØ4767  ØØ1214                  JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
Ø14162  ØØØØØ4  Ø147Ø2                  TYPE,   COMMA
Ø14166  Ø167Ø5  16443Ø                  MOV     RAND,B,TTY         ;TYPE RAND,B IN OCTAL
Ø14172  ØØ4767  ØØ12ØØ                  JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
Ø14176  Ø137ØØ  ØØØ244                  MOV     @#244,  RØ         ;GET PC+2 OF INSTRUCTION
Ø142Ø2  Ø14ØØ1                          MOV     -(RØ),  R1         ;GET THE INSTRUCTION
Ø142Ø4  Ø427Ø1  177747                  BIC     #177747,R1         ;MASK ALL BUT TYPE (+,-,*,/)
Ø1421Ø  ØØ62Ø1                          ASR     R1                 ;DIV BY 2
Ø14212  Ø12767  Ø14662  ØØØØØ6          MOV     #SIGNS, 1$         ;SET TO TOP OF SIGN TABLE
Ø1422Ø  Ø6Ø167  ØØØØØ2                  ADD     R1,     1$         ;ADD OFFSET
Ø14224  ØØØØØ4                          TYPE
Ø14226  Ø14662                  1$:     SIGNS                      ;TYPE THE RIGHT SIGN
Ø1423Ø  Ø167Ø5  16437Ø                  MOV     RAND,C,TTY         ;TYPE RAND,C IN OCTAL
Ø14234  ØØ4767  ØØ1136                  JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
Ø1424Ø  ØØØØØ4  Ø147Ø2                  TYPE,   COMMA
Ø14244  Ø167Ø5  164356                  MOV     RAND,D,TTY         ;TYPE RAND,D IN OCTAL
Ø1425Ø  ØØ4767  ØØ1122                  JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
Ø14254  ØØ63Ø1                          ASL     R1                 ;RESET TABLE POINTER
Ø14256  Ø627Ø1  ØØØ63Ø                  ADD     #$ADDPS,R1
Ø14262  1Ø5767  164326                  TSTB    $PSW               ;CHECK FOR ERROR CONDITIONS
Ø14266  1ØØ46Ø                          BMI     3$                 ;BRANCH IF ERROR
Ø1427Ø  ØØØØØ4  Ø147Ø4                  TYPE,   HEAD1
Ø14274  ØØØØØ4  Ø15Ø62                  TYPE,   EXPECT
Ø143ØØ  Ø121Ø5                          MOV     (R1)+,TTY          ;TYPE (R1)+ IN OCTAL
Ø143Ø2  ØØ4767  ØØ1Ø7Ø                  JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
Ø143Ø6  ØØØØØ4  Ø1537Ø                  TYPE,   SPACE+3
```

```
014312  012705  000610              MOV     #STACK4,TTY        ;TYPE #STACK4 IN OCTAL
014316  004767  001054              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014322  000004  015370              TYPE,   SPACE+3
014326  012105                      MOV     (R1)+,TTY          ;TYPE (R1)+ IN OCTAL
014330  004767  001042              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014334  000004  014702              TYPE,   COMMA
014340  011105                      MOV     (R1),TTY           ;TYPE (R1) IN OCTAL
014342  004767  001030              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014346  000004  015076              TYPE,   GOT
014352  016705  164236              MOV     $PSW,TTY           ;TYPE $PSW IN OCTAL
014356  004767  001014              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014362  000004  015370              TYPE,   SPACE+3
014366  016705  164224              MOV     $SP,TTY            ;TYPE $SP IN OCTAL
014372  004767  001000              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014376  000004  015370              TYPE,   SPACE+3
014402  016705  164202              MOV     ANS1,TTY           ;TYPE ANS1 IN OCTAL
014406  004767  000764              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014412  000004  014702              TYPE,   COMMA
014416  016705  164170              MOV     ANS2,TTY           ;TYPE ANS2 IN OCTAL
014422  004767  000750              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014426  000510                      BR      7$

014430  000004  014751      3$:     TYPE,   HEAD2
014434  000004  015062              TYPE,   EXPECT
014440  012105                      MOV     (R1)+,TTY          ;TYPE (R1)+ IN OCTAL
014442  004767  000730              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014446  000004  015370              TYPE,   SPACE+3
014452  016705  164212              MOV     SAVSTK,TTY         ;TYPE SAVSTK IN OCTAL
014456  004767  000714              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014462  000004  015370              TYPE,   SPACE+3
014466  005720                      TST     (R0)+              ;UPDATE R0 TO RIGHT ADDRESS
014470  010005                      MOV     R0,TTY             ;TYPE R0 IN OCTAL
014472  004767  000700              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014476  000004  015370              TYPE,   SPACE+3
014502  022121                      CMP     (R1)+,  (R1)+      ;ADD 4 TO R1
014504  011105                      MOV     (R1),TTY           ;TYPE (R1) IN OCTAL
014506  004767  000664              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014512  000004  015370              TYPE,   SPACE+3
014516  016705  164102              MOV     RAND.C,TTY         ;TYPE RAND.C IN OCTAL
014522  004767  000650              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014526  000004  015370              TYPE,   SPACE+3
014532  016705  164070              MOV     RAND.D,TTY         ;TYPE RAND.D IN OCTAL
014536  004767  000634              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014542  000004  015370              TYPE,   SPACE+3
014546  016705  164046              MOV     RAND.A,TTY         ;TYPE RAND.A IN OCTAL
014552  004767  000620              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014556  000004  015370              TYPE,   SPACE+3
014562  016705  164034              MOV     RAND.B,TTY         ;TYPE RAND.B IN OCTAL
014566  004767  000604              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014572  000004  015076              TYPE,   GOT
014576  016705  164012              MOV     $PSW,TTY           ;TYPE $PSW IN OCTAL
014602  004767  000570              JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
014606  000004  015370              TYPE,   SPACE+3
014612  016705  164000              MOV     $SP,TTY            ;TYPE $SP IN OCTAL
```

```
Ø14616  ØØ4767  ØØØ554                    JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
Ø14622  Ø127Ø1  ØØØ6ØØ                    MOV     #STK1,  R1         ;SET UP TABLE POINTER
Ø14626  Ø127ØØ  ØØØØØ6                    MOV     #6,     RØ
Ø14632  ØØØØØ4  Ø1537Ø          6$:       TYPE,   SPACE+3
Ø14636  Ø12125                            MOV     (R1)+,TTY          ;TYPE (R1)+ IN OCTAL
Ø1464Ø  ØØ4767  ØØØ532                    JSR     %7,PRINTR          ;TYPE LEADING ZERO'S
Ø14644  ØØ53ØØ                            DEC     RØ                 ;COUNT DOWN
Ø14646  ØØ1371                            BNE     6$                 ;KEEP TYPING
Ø1465Ø  ØØØØØ4  Ø15362          7$:       TYPE,   RETURN
Ø14654  Ø126Ø1                            MOV     (SP)+,  R1         ;RESTORE R1
Ø14656  Ø126ØØ                            MOV     (SP)+,  RØ         ;RESTORE RØ
Ø1466Ø  ØØØ2Ø7                            RTS     PC

Ø14662  Ø2544Ø  ØØØØ4Ø          SIGNS:    .ASCIZ  " + "
Ø14666  Ø2644Ø  ØØØØ4Ø                    .ASCIZ  " - "
Ø14672  Ø2504Ø  ØØØØ4Ø                    .ASCIZ  " * "
Ø14676  Ø2744Ø  ØØØØ4Ø                    .ASCIZ  " / "

Ø147Ø2  ØØØØ54                  COMMA:    .ASCIZ  ","

Ø147Ø4  ØØ5Ø15  Ø2ØØ4Ø  Ø2ØØ4Ø  HEAD1:    .ASCIZ  <15><12>"          PSW      SP        ANSWER"
Ø14712  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø
Ø1472Ø  Ø5ØØ4Ø  053523  Ø2ØØ4Ø
Ø14726  Ø2ØØ4Ø  Ø5144Ø  Ø2Ø12Ø
Ø14734  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø
Ø14742  Ø471Ø1  053523  Ø511Ø5
Ø1475Ø     ØØØ
Ø14751     Ø15  Ø2ØØ12  Ø2ØØ4Ø  HEAD2:    .ASCIZ  <15><12>"          PSW      SP      STK1      STK2      STK3      STK4      STK5
Ø14756  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø
Ø14764  Ø2ØØ4Ø  Ø5152Ø  Ø2Ø127
Ø14772  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø5Ø123
Ø15ØØØ  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø5144Ø
Ø15ØØ6  Ø45524  Ø2ØØ61  Ø2ØØ4Ø
Ø15Ø14  Ø5144Ø  Ø45524  Ø2ØØ62
Ø15Ø22  Ø2ØØ4Ø  Ø5144Ø  Ø45524
Ø15Ø3Ø  Ø2ØØ63  Ø2ØØ4Ø  Ø5144Ø
Ø15Ø36  Ø45524  Ø2ØØ64  Ø2ØØ4Ø
Ø15Ø44  Ø5144Ø  Ø45524  Ø2ØØ65
Ø15Ø52  Ø2ØØ4Ø  Ø5144Ø  Ø45524
Ø15Ø6Ø  ØØØØ66
Ø15Ø62  ØØ5Ø15  Ø541Ø5  Ø4252Ø  EXPECT:   .ASCIZ  <15><12>"EXPECT:  "
Ø15Ø7Ø  Ø521Ø3  Ø2ØØ72  ØØØØ4Ø
Ø15Ø76  ØØ5Ø15  Ø475Ø7  Ø35124  GOT:      .ASCIZ  <15><12>"GOT:     "
Ø151Ø4  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØ4Ø
                                          .EVEN
```

```

015112  010346                          READS:  MOV     R3,-(6)         ;SAVE R3
015114  012703  015212                  1$:     MOV     #INPUT,R3       ;GET ADDRESS
015120  022703  015252                  2$:     CMP     #.QUES, R3      ;CHECK FOR BUFFER OVERFLOW
015124  001412                                  BEQ     4$              ;ABORT
015126  105737  177560                          TSTB    @#177560        ;WAIT FOR
015132  100375                                  BPL     .-4             ;A CHARACTER
015134  113713  177562                          MOVB    @#177562,(3)    ;GET CHARACTER
015140  142713  000200                          BICB    #200,(3)        ;GET RID OF JUNK
015144  122713  000177                          CMPB    #177,(3)        ;IS IT A RUBOUT
015150  001003                                  BNE     3$              ;SKIP IF NOT
015152  000004  015252                  4$:     TYPE    ,.QUES          ;TYPE A '?'
015156  000756                                  BR      1$              ;ZAP THE BUFFER AND LOOP
015160  111367  000210                  3$:     MOVB    (3),.TYPE       ;SET UP FOR TYPING
015164  000004  015374                          TYPE    ,.TYPE          ;ECHO IT
015170  122723  000015                          CMPB    #15,(3)+        ;CHECK FOR RETURN
015174  001351                                  BNE     2$              ;LOOP IF NOT RETURN
015176  105063  177777                          CLRB    -1(3)           ;ZAP RETURN (THE 15)
015202  000004  000012                          TYPE    ,12             ;TYPE A LINE FEED
015206  012603                                  MOV     (6)+,R3         ;RESTORE R3
015210  000207                                  RTS     PC              ;RETURN

015212  000020                          INPUT:  .BLKW   20
015252  006477  000012                  .QUES:  .ASCIZ  "?"<15><12>

015256  010546                          .IOT:   MOV     TTY,-(6)        ;SAVE TTY
015260  017605  000002                          MOV     @2(6),TTY       ;GET ADDRESS TO BE TYPED
015264  032705  177400                          BIT     #177400,TTY     ;IS IT A TYPEM?
015270  001004                                  BNE     1$              ;NO
015272  010567  000076                          MOV     TTY,.TYPE       ;GET THE CHARACTER
015276  012705  015374                          MOV     #.TYPE,TTY      ;FUDGE THE ADDRESS
015302  105715                          1$:     TSTB    (TTY)           ;TERMINATOR?
015304  001406                                  BEQ     2$              ;GET OUT IF SO
015306  112537  177566                          MOVB    (TTY)+,@#177566 ;LOAD AND TYPE THE CHARACTER
015312  105737  177564                          TSTB    @#177564        ;IS THE PRINTER READY
015316  100375                                  BPL     .-4             ;WAIT UNTIL IT IS
015320  000770                                  BR      1$              ;GET THE NEXT CHARACTER
015322  017646  000002                  2$:     MOV     @2(6),-(6)      ;GET ADDRESS TO BE TYPED
015326  062766  000002  000004                  ADD     #2,4(6)         ;ADD 2 TO THE ADDRESS
015334  022666  000002                          CMP     (6)+,2(6)       ;IS IT .+2?
015340  001006                                  BNE     3$              ;NO
015342  062705  000002                          ADD     #2,TTY          ;ADD 2 TO THE ADDRESS
015346  042705  000001                          BIC     #1,TTY          ;BACK UP TO AN EVEN BYTE
015352  010566  000002                          MOV     TTY,2(6)        ;RESTORE ADDRESS
015356  012605                          3$:     MOV     (6)+,TTY        ;RESTORE TTY
015360  000002                                  RTI                     ;RETURN

015362  005015     000                  RETURN: .ASCIZ  <15><12>        ;RETURN AND LINEFEED
015365     015  020012  020040          SPACE:  .ASCIZ  <15><12>"   "   ;RETURN AND 3 SPACES
015372     000
        015374                          .EVEN
015374  000000                          .TYPE:  0                       ;CHARACTER TYPE LOCATION
```

```

015376  112767  000001  000130  PRINTR: MOVB    #1,.PR          ;SET ZERO FILL SWITCH
015404  000402                          BR      .+6             ;SKIP
015406  005067  000122          PRINTS: CLR     .PR             ;SUPRESS LEADING ZERO'S
015412  112767  177772  000115          MOVB    #-6,.PR+1       ;SET COUNT
015420  010446                          MOV     R4,-(6)         ;SAVE R4
015422  012704  015524                  MOV     #.PRBUF,R4      ;SET POINTER TO FIRST ASCII CHAR.
015426  105014                          CLRB    (4)             ;CLEAR FIRST BYTE
015430  000405                          BR      .PRF            ;ROTATE FIRST BIT
015432  105014                  .PRL:   CLRB    (4)             ;CLEAR BYTE OF CHARACTER
015434  006105                          ROL     TTY             ;ROTATE BIT INTO C
015436  106114                          ROLB    (4)             ;PACK IT
015440  006105                          ROL     TTY             ;ROTATE BIT INTO C
015442  106114                          ROLB    (4)             ;PACK IT
015444  006105                  .PRF:   ROL     TTY             ;ROTATE BIT INTO C
015446  106114                          ROLB    (4)             ;PACK IT
015450  105714                          TSTB    (4)             ;IS IT ZERO?
015452  001402                          BEQ     .+6             ;SKIP INC
015454  105267  000054                  INCB    .PR             ;SET FILL SWITCH
015460  105767  000050                  TSTB    .PR             ;CHECK FILL SWITCH
015464  001402                          BEQ     .+6             ;SKIP BITSET
015466  152724  000060                  BISB    #'0,(4)+        ;MAKE INTO ASCII CHAR
015472  105267  000037                  INCB    .PR+1           ;INC COUNT
015476  001355                          BNE     .PRL            ;REPEAT
015500  022704  015524                  CMP     #.PRBUF,R4      ;EMPTY BUFFER?
015504  001002                          BNE     .+6             ;SKIP IF NOT
015506  112724  000060                  MOVB    #'0,(4)+        ;LOAD 1 ZERO
015512  105014                          CLRB    (4)             ;NULL TERMINATOR
015514  000004  015524                  TYPE    ,.PRBUF         ;TYPE IT
015520  012604                          MOV     (6)+,R4         ;RESTORE R4
015522  000207                          RTS     PC              ;RETURN

015524  000004                  .PRBUF: .BLKW   4               ;OUTPUT BUFFER
015534  000000                  .PR:    0                       ;COUNT AND SWITCH
```

```

015536  012777  015652  000120  PDOWNS: MOV     #ILLUP,@PUVECS  ;SET FOR FAST UP
015544  012777  000340  000114          MOV     #340,@PUVECS+2  ;PRIO17
015552  010046                          MOV     R0,-(6)         ;PUSH R0 ON STACK
015554  010146                          MOV     R1,-(6)         ;PUSH R1 ON STACK
015556  010246                          MOV     R2,-(6)         ;PUSH R2 ON STACK
015560  010346                          MOV     R3,-(6)         ;PUSH R3 ON STACK
015562  010446                          MOV     R4,-(6)         ;PUSH R4 ON STACK
015564  010546                          MOV     R5,-(6)         ;PUSH R5 ON STACK
015566  010667  000064                  MOV     SP,.SAVR6       ;SAVE SP
015572  012777  015602  000064          MOV     #PUPS,@PUVECS   ;SET UP VECTOR
015600  000000                          HALT

015602  016706  000050          PUPS:   MOV     .SAVR6,SP       ;GET SP
015606  005001                          CLR     R1              ;WAIT LOOP FOR THE TTY
015610  005201                  1$:     INC     R1              ;WAIT FOR THE INC
015612  001376                          BNE     1$              ;OF  WORD
015614  012605                          MOV     (6)+,R5         ;POP STACK INTO R5
015616  012604                          MOV     (6)+,R4         ;POP STACK INTO R4
015620  012603                          MOV     (6)+,R3         ;POP STACK INTO R3
015622  012602                          MOV     (6)+,R2         ;POP STACK INTO R2
015624  012601                          MOV     (6)+,R1         ;POP STACK INTO R1
015626  012600                          MOV     (6)+,R0         ;POP STACK INTO R0
015630  012777  015536  000022          MOV     #PDOWNS,@PDVECS ;SET UP THE POWER DOWN VECTOR
015636  012777  000340  000016          MOV     #340,@PDVECS+2  ;PRIO17
015644  000004  015670                  TYPE    ,POWERS
015650  000002                          RTI

015652  000000                  ILLUP:  HALT                    ;THE POWER UP SEQUENCE WAS STARTED
015654  000776                          BR      .-2             ; BEFORE THE POWER DOWN WAS COMPLETE

015656  000000                  .SAVR6: 0                       ;PUT THE SP HERE
015660  000024  000026          PDVECS: 24,26                   ;POWER DOWN VECTOR
015664  000024  000026          PUVECS: 24,26                   ;POWER UP VECTOR
015670  005015  047520  042527  POWERS: .ASCIZ  <15><12>"POWER"
015676  000122
                                        .EVEN

        000001                  .END
```

```
ANS1      ØØØ61Ø         ANS2      ØØØ612         BEGIN     ØØ1Ø1Ø         BELL    = ØØØØØ7
BITØ    = ØØØØØ1         BIT1    = ØØØØØ2         BIT1Ø   = ØØ2ØØØ       BIT11   = ØØ4ØØØ
BIT12   = Ø1ØØØØ         BIT13   = Ø2ØØØØ         BIT14   = Ø4ØØØØ       BIT15   = 1ØØØØØ
BIT2    = ØØØØØ4         BIT3    = ØØØØ1Ø         BIT4    = ØØØØ2Ø       BIT5    = ØØØØ4Ø
BIT6    = ØØØ1ØØ         BIT7    = ØØØ2ØØ         BIT8    = ØØØ4ØØ       BIT9    = ØØ1ØØØ
CCC     = ØØØ257         COMMA     Ø147Ø2         DISPLA  = 17757Ø       DONE      Ø12662
ERRORS    ØØ1ØØ2         ERROR$    Ø14142         EXPECT    Ø15Ø62       FADD    = Ø75ØØØ
FDIV    = Ø75Ø3Ø         FISTRP    ØØØ754         FMUL    = Ø75Ø2Ø       FORTAN    ØØ136Ø
FSUB    = Ø75Ø1Ø         GOT       Ø15Ø76         HEAD1     Ø147Ø4       HEAD2     Ø14751
HLT     = 1Ø4ØØØ         HLTAD$    Ø1414Ø         HLT$      Ø14Ø2Ø       ICNT      ØØ1ØØØ
ILLUP     Ø15652         INPUT     Ø15212         KIT$      Ø13764       LAD$      Ø14Ø12
LEVELØ  = ØØØØØØ         LEVEL1  = ØØØØ4Ø         LEVEL2  = ØØØ1ØØ       LEVEL3  = ØØØ14Ø
LEVEL4  = ØØØ2ØØ         LEVEL5  = ØØØ24Ø         LEVEL6  = ØØØ3ØØ       LEVEL7  = ØØØ34Ø
N       = ØØØØ35         OVER$     Ø1377Ø         PC      =%ØØØØØ7       PCNT      ØØ1ØØ4
PDOWN$    Ø15536         PDVEC$    Ø1566Ø         POWER$    Ø1567Ø       PRINTR    Ø15376
PRINTS    Ø154Ø6         PS      = 177776         PUP$      Ø156Ø2       PUSHR     Ø136ØØ
PUVEC$    Ø15664         RAND.A    ØØØ62Ø         RAND.B    ØØØ622       RAND.C    ØØØ624
RAND.D    ØØØ626         RAND4$    ØØØ674         READIN    Ø12762       READ$     Ø15112
RETURN    Ø15362         RNDFLG    ØØØ672         RØ      =%ØØØØØØ       R1      =%ØØØØØ1
R2      =%ØØØØØ2         R3      =%ØØØØØ3         R4      =%ØØØØØ4       R5      =%ØØØØØ5
SAVSTK    ØØØ67Ø         SCC     = ØØØ277         SCOPE   = 1Ø44ØØ       SCOPE$    Ø13644
SCOTMP    Ø14Ø16         SIGN$     Ø14662         SNV     = ØØØ272       SP      =%ØØØØØ6
SPACE     Ø15365         STACKØ    ØØØ6Ø4         STACK2    ØØØ6Ø6       STACK4    ØØØ61Ø
STACK6    ØØØ612         START     ØØ1146         STK1      ØØØ6ØØ       STK2      ØØØ6Ø2
STK3      ØØØ6Ø4         STK4      ØØØ6Ø6         STK5      ØØØ61Ø       STK6      ØØØ612
SVLAD$    Ø13744         SWR     = 17757Ø         SWØ8    = ØØØ4ØØ       SWØ9    = ØØ1ØØØ
SW1Ø    = ØØ2ØØØ         SW11    = ØØ4ØØØ         SW12    = Ø1ØØØØ       SW13    = Ø2ØØØØ
SW14    = Ø4ØØØØ         SW15    = 1ØØØØØ         TIMES     Ø14Ø14       TST1      ØØ144Ø
TST1Ø     ØØ42Ø2         TST11     ØØ4442         TST12     ØØ47Ø2       TST13     ØØ5142
TST14     ØØ54Ø2         TST15     ØØ5642         TST16     ØØ61Ø2       TST17     ØØ6344
TST2      ØØ2ØØØ         TST2Ø     ØØ66Ø4         TST21     ØØ7Ø44       TST22     ØØ73Ø4
TST23     ØØ7544         TST24     Ø1ØØØ4         TST25     Ø1Ø244       TST26     Ø1Ø5Ø6
TST27     Ø1Ø746         TST3      ØØ234Ø         TST3Ø     Ø112Ø6       TST31     Ø11446
TST32     Ø117Ø6         TST33     Ø12146         TST34     Ø124Ø6       TST4      ØØ27ØØ
TST5      ØØ324Ø         TST6      ØØ35ØØ         TST7      ØØ374Ø       TTY     =%ØØØØØ5
TYPE    = ØØØØØ4         TYPIN     ØØ121Ø         YESRT     ØØØ752       $ADDER    ØØØ636
$ADDPS    ØØØ63Ø         $ADD1     ØØØ632         $ADD2     ØØØ634       $ADR    = ****** G
$DIVER    ØØØ666         $DIVPS    ØØØ66Ø         $DIV1     ØØØ662       $DIV2     ØØØ664
$DVR    = ****** G       $ERR      Ø13414 G       $ERRA     Ø1342Ø G     $EXIT     Ø13412
$MLR    = ****** G       $MULER    ØØØ656         $MULPS    ØØØ65Ø       $MUL1     ØØØ652
$MUL2     ØØØ654         $POLSH    Ø13Ø64         $POPAD    Ø13Ø66       $POPDV    Ø133ØØ
$POPML    Ø13222         $POPSB    Ø13144         $PSW      ØØØ614       $PUSH     Ø13Ø44
$SBR    = ****** G       $SKIP     Ø134Ø6         $SP       ØØØ616       $SUBER    ØØØ646
$SUBPS    ØØØ64Ø         $SUB1     ØØØ642         $SUB2     ØØØ644       .BIT    = 177777
.IOT      Ø15256         .PR       Ø15534         .PRBUF    Ø15524       .PRF      Ø15444
.PRL      Ø15432         .QUES     Ø15252         .SAVR6    Ø15656       .TBIT     Ø1276Ø
.TYPE     Ø15374         .       = Ø157ØØ

          ØØØØØØ


 ERRORS DETECTED:  Ø
```

```
                                         .TITLE  $ADR
            ØØØØØØ'                      .CSECT
                                         .GLOBL  $ADR,$SBR,$ERR
                                     ;   $ADR  ---- THE REAL ADD ROUTINE
                                     ;   COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                                     ;   REPLACE THE TWO ITEMS ON TOP OF THE STACK
                                     ;   WITH THEIR SUM.
                                     ;   $SBR  ---- THE REAL SUBTRACT ROUTINE
                                     ;   SUBTRACT THE TOP STACK ITEM FROM THE SECOND ITEM
                                     ;   REPLACE THEM BOTH WITH THE DIFFERENCE.
            ØØØØØØ                       RØ=%Ø
            ØØØØØ1                       R1=%1
            ØØØØØ2                       R2=%2
            ØØØØØ3                       R3=%3
            ØØØØØ4                       R4=%4
            ØØØØØ5                       R5=%5
            ØØØØØ6                       SP=%6
            ØØØØØ7                       PC=%7
            ØØØØØØ                       SIGNS=Ø
            ØØØØØ4                       A1=4
            ØØØØØ6                       B1=6
            ØØØØ1Ø                       A2=8.
            ØØØØ12                       B2=1Ø.
            177302                       AC=177302
            177304                       MQ=177304
            177312                       NOR=177312
            177316                       ASH=177316
            ØØØØØØ                       FØ=%Ø
 ØØØØØØ'    Ø62716  1ØØØØØ         $SBR: ADD     #1ØØØØØ,@SP      ;CHANGE THE SIGN OF TOP ITEM
               ØØ1                       .IFDF   FPU
                                   $ADR: .WORD   17ØØØ1  ;;SETF
                                         .WORD   172426  ;;LDF   (SP)+,FØ         ;GET OPERAND
                                         .WORD   172Ø26  ;;ADDF  (SP)+,FØ         ;ADD
                                         .WORD   174Ø46  ;;STF___FØ,-(SP)         ;SUM TO STACK
                                         JMP     @(R4)+
               ØØØ                       .ENDC
               ØØ1                       .IFNDF  FPU
 ØØØØØ4'    Ø1Ø446                 $ADR: MOV     R4,-(SP)
 ØØØØØ6'    ØØ5Ø46                       CLR     -(SP)   ;CLEAR SIGNS
 ØØØØ1Ø'    ØØ5ØØ2                       CLR     R2      ;CLEAR EXPONENTS
 ØØØØ12'    ØØ5ØØ3                       CLR     R3
 ØØØØ14'    ØØ6366  ØØØØØ6               ASL     B1(SP)  ;SHIFT B1
 ØØØØ2Ø'    ØØ6166  ØØØØØ4               ROL     A1(SP)  ;SHIFT A1
 ØØØØ24'    1566Ø3  ØØØØØ5               BISB    A1+1(SP),R3     ;GET E1
 ØØØØ3Ø'    ØØ1574                       BEQ     OUT     ;JUMP IF ZERO
 ØØØØ32'    1Ø6116                       ROLB    @SP     ;GET S1
 ØØØØ34'    ØØ6366  ØØØØ12               ASL     B2(SP)  ;SHIFT B2
 ØØØØ4Ø'    ØØ6166  ØØØØ1Ø               ROL     A2(SP)  ;SHIFT A2
 ØØØØ44'    1566Ø2  ØØØØ11               BISB    A2+1(SP),R2     ;GET E2
 ØØØØ5Ø'    ØØ1Ø14                       BNE     A2NZ    ;JUMP IF NOT Ø
 ØØØØ52'    1Ø6Ø16                       RORB    @SP     ;RECONSTRUCT A1,B1
 ØØØØ54'    ØØ6Ø66  ØØØØØ4               ROR     A1(SP)
 ØØØØ6Ø'    ØØ6Ø66  ØØØØØ6               ROR     B1(SP)
 ØØØØ64'    Ø16666  ØØØØØ4  ØØØØ1Ø       MOV     A1(SP),A2(SP)    ;FIRST ARG TO TOP OF STACK
```

```
000072' 016666  000006  000012                  MOV     B1(SP),B2(SP)
000100' 000550                                  BR      OUT     ;DONE
000102' 106166  000001                  A2NZ:   ROLB    SIGNS+1(SP)     ;GET S2
000106' 112766  000001  000011                  MOVB    #1,A2+1(SP)     ;INSERT NORMAL BIT
000114' 112766  000001  000005                  MOVB    #1,A1+1(SP)     ;INSERT NORMAL BIT
000122' 160302                                  SUB     R3,R2   ;R2=E2-E1, R3=E1
000124' 003005                                  BGT     EXPA    ;JUMP IF E2>=E1
000126' 016600  000010                          MOV     A2(SP),R0       ;R0=A2
000132' 016601  000012                          MOV     B2(SP),R1       ;R1=B2
000136' 000415                                  BR      SCHK    ;CHECK SIGNS
000140' 060203                          EXPA:   ADD     R2,R3   ;R2=E2-E1,R3=E2,E2>E1
000142' 016600  000004                          MOV     A1(SP),R0       ;R0=A1
000146' 016601  000006                          MOV     B1(SP),R1       ;R1=B1
000152' 016666  000010  000004                  MOV     A2(SP),A1(SP)
000160' 016666  000012  000006                  MOV     B2(SP),B1(SP)
000166' 000316                                  SWAB    @SP     ;EXCHANGE SIGNS
000170' 005402                                  NEG     R2      ;E1-E2
000172' 126616  000001                  SCHK:   CMPB    SIGNS+1(SP),@SP ;SEE IF SIGNS ARE THE SAME
000176' 001403                                  BEQ     ECHK    ;YES, CHECK EXPONENTS
000200' 005401                                  NEG     R1      ;NEGATE FRACTION
000202' 005500                                  ADC     R0
000204' 005400                                  NEG     R0
000206' 005702                          ECHK:   TST     R2
000210' 001450                                  BEQ     SHFTD   ;JUMP IF E1=E2
000212' 022702  177747                  SHFT:   CMP     #-25.,R2        ;IS THERE ANY POINT IN SHIFTING?
000216' 003405                                  BLE     SHFTR   ;YES
000220' 016600  000004                          MOV     A1(SP),R0       ;NO, ANSWER IS OPERAND
000224' 016601  000006                          MOV     B1(SP),R1       ;WITH THE LARGER EXPONENT
000230' 000456                                  BR      NORMD
                   002                          .IFDF   EAE
                                        SHFTR:  MOV     R1,@#MQ ;MOVE FRACTION TO AC,MQ
                                                MOV     R0,@#AC
                                                MOV     R2,@#ASH        ;SHIFT RIGHT TO EQUALIZE EXPONENTS
                                                MOV     @#MQ,R1 ;RECOVER SHIFTED FRACTION
                                                MOV     @#AC,R0
                   001                          .ENDC
                   002                          .IFDF   MULDIV
                                        SHFTR:  .WORD   073002  ;;ASHC  R2,R0
                   001                          .ENDC
                   002                          .IFNDF  EAE&MULDIV
000232' 022702  177770                  SHFTR:  CMP     #-8.,R2 ;CHECK # OF BITS TO SHIFT
000236' 003431                                  BLE     SHFTR0  ;JUMP IF NOT MORE THAN 1/2 WORD
000240' 005004                                  CLR     R4      ;SET UP EXTENSION BITS
000242' 005700                                  TST     R0      ;BASED ON HIGH ORDER FRACTION
000244' 100001                                  BPL     NCOMP   ;JUMP IF +
000246' 005104                                  COM     R4      ;- OTHERWISE
000250' 022702  177760                  NCOMP:  CMP     #-16.,R2
000254' 002405                                  BLT     SHFTRL  ;JUMP IF LESS THAN ONE WORD TO SHIFT
000256' 010001                                  MOV     R0,R1   ;SHIFT RIGHT A WHOLE WORD
000260' 010400                                  MOV     R4,R0   ;USE EXTENSION BITS
000262' 062702  000020                          ADD     #16.,R2 ;ACCOUNT FOR SHIFT
000266' 001421                                  BEQ     SHFTD
000270' 022702  177770                  SHFTRL: CMP     #-8.,R2
000274' 003412                                  BLE     SHFTR0  ;JUMP IF NOT MORE THAN 1/2 WORD
```

```
ØØØ276' Ø627Ø2  ØØØØ2Ø              ADD     #16.,R2 ;SHIFT LEFT 16-X
ØØØ3Ø2' ØØ63Ø1              SHFTL:  ASL     R1
ØØØ3Ø4' ØØ61ØØ                      ROL     RØ
ØØØ3Ø6' ØØ61Ø4                      ROL     R4
ØØØ31Ø' ØØ53Ø2                      DEC     R2      ;COUNT LOOP
ØØØ312' ØØ3373                      BGT     SHFTL
ØØØ314' Ø1ØØØ1                      MOV     RØ,R1   ;PUT RESULT IN RØ, R1
ØØØ316' Ø1Ø4ØØ                      MOV     R4,RØ
ØØØ32Ø' ØØØ4Ø4                      BR      SHFTD
ØØØ322' ØØ62ØØ              SHFTRØ: ASR     RØ      ;SHIFT A MIN AND B MIN
ØØØ324' ØØ6ØØ1                      ROR     R1
ØØØ326' ØØ52Ø2                      INC     R2      ;REDUCE EXPONENT DIFFERENCE
ØØØ33Ø' ØØ2774                      BLT     SHFTRØ
            ØØ1                     .ENDC
ØØØ332' Ø666ØØ  ØØØØØ4      SHFTD:  ADD     A1(SP),RØ       ;A1+A2
ØØØ336' Ø666Ø1  ØØØØØ6              ADD     B1(SP),R1       ;B1+B2
ØØØ342' ØØ55ØØ                      ADC     RØ
ØØØ344' 126616  ØØØØØ1              CMPB    SIGNS+1(SP),@SP
ØØØ35Ø' ØØ1Ø34                      BNE     SUB     ;GO CLEAN UP SUBTRACT
ØØØ352' Ø3ØØ27  ØØ1ØØØ              BIT     RØ,#1ØØØ
ØØØ356' ØØ14Ø3                      BEQ     NORMD   ;JUMP IF NO NORMAL BIT OVERFLOW
ØØØ36Ø' ØØ62ØØ                      ASR     RØ
ØØØ362' ØØ6ØØ1                      ROR     R1
ØØØ364' ØØ52Ø3                      INC     R3      ;INCREASE EXPONENT
ØØØ366' ØØØ3Ø3              NORMD:  SWAB    R3      ;MOVE EXPONENT LEFT
ØØØ37Ø' ØØ1Ø2Ø                      BNE     OVER    ;JUMP IF OVERFLOW
ØØØ372' 15ØØØ3                      BISB    RØ,R3
ØØØ374' ØØ6Ø16                      ROR     @SP     ;INSERT SIGN
ØØØ376' ØØ6ØØ3                      ROR     R3
ØØØ4ØØ' ØØ6ØØ1                      ROR     R1
ØØØ4Ø2' ØØ55Ø1                      ADC     R1      ;ROUND SUM
ØØØ4Ø4' ØØ55Ø3                      ADC     R3
ØØØ4Ø6' 1Ø2411                      BVS     OVER    ;JUMP IF OVERFLOW ON ROUND
ØØØ41Ø' 1Ø341Ø                      BCS     OVER
ØØØ412' Ø1Ø366  ØØØØ1Ø      STORE:  MOV     R3,A2(SP)       ;STORE EXPONENT AND SIGN
ØØØ416' Ø1Ø166  ØØØØ12              MOV     R1,B2(SP)       ;INSERT LOW ORDER FRACTION
ØØØ422' ØØ5726              OUT:    TST     (SP)+   ;POP SIGNS
ØØØ424' Ø126Ø4                      MOV     (SP)+,R4
ØØØ426' Ø22626                      CMP     (SP)+,(SP)+     ;POP FIRST ARGUMENT
ØØØ43Ø' ØØØ134                      JMP     @(R4)+  ;DONE, RETURN
                            ;
ØØØ432' ØØ4567  ØØØØØØG     OVER:   JSR     R5,$ERR ;ERROR 3,2
ØØØ436' ØØØ771                      BR      OUT
ØØØ44Ø'    ØØ3                      .BYTE   3
ØØØ441'    ØØ2                      .BYTE   2
                            ;
ØØØ442' ØØ57ØØ              SUB:    TST     RØ      ;CHECK HIGH ORDER RESULT FRACTION
ØØØ444' ØØ3ØØ5                      BGT     BIT9    ;IF POSITIVE SIGN IS OK
ØØØ446' ØØ1413                      BEQ     ZTEST   ;CHECK FOR ZERO RESULT
ØØØ45Ø' ØØ54ØØ                      NEG     RØ      ;GET ABSOLUTE VALUE
ØØØ452' ØØ54Ø1                      NEG     R1
ØØØ454' ØØ56ØØ                      SBC     RØ
ØØØ456' ØØØ316                      SWAB    @SP     ;EXCHANGE SIGNS
ØØØ46Ø'                     BIT9:
```

```
               ØØ2                            .IFDF   EAE
                                              BIT     RØ,#7ØØ
                                              BNE     BIT9A   ;JUMP IF NOT MORE THAN 2 TO SHIFT
                                              MOV     R1,@#MQ ;RESULT FRACTION TO AC,MQ
                                              MOV     RØ,@#AC
                                              CLR     @#NOR   ;NORMALIZE
                                              SUB     @#NOR,R3        ;ADJUST EXPONENT
                                              MOV     #-6,@#ASH       ;SHIFT TO CORRECT POSITION
                                              ADD     #6,R3   ;COMPENSATE EXPONENT
                                              BLE     UNDERF  ;JUMP IF UNDERFLOW
                                              MOV     @#AC,RØ
                                              MOV     @#MQ,R1 ;GET FRACTION BACK
                                              BR      NORMD
               ØØ1                            .ENDC
ØØØ46Ø' Ø3ØØ27  ØØØ4ØØ               BIT9A:   BIT     RØ,#4ØØ
ØØØ464' ØØ1Ø14                                BNE     UTEST   ;JUMP IF NORMAL BIT FOUND
ØØØ466' ØØ53Ø3                                DEC     R3      ;DECREASE EXPONENT
ØØØ47Ø' ØØ63Ø1                                ASL     R1      ;DOUBLE FRACTION
ØØØ472' ØØ61ØØ                                ROL     RØ
ØØØ474' ØØØ771                                BR      BIT9A   ;TRY AGAIN
ØØØ476' ØØ57Ø1                       ZTEST:   TST     R1      ;CHECK LOW ORDER PART
               ØØ2                            .IFDF   EAE
                                              BNE     BIT9
                                              BR      ZERO
               ØØ1                            .ENDC
               ØØ2                            .IFNDF  EAE
ØØØ5ØØ' ØØ1415                                BEQ     ZERO
ØØØ5Ø2' ØØØ3Ø1                                SWAB    R1      ;SAVE NORMALIZE SOME TIME
ØØØ5Ø4' 15Ø1ØØ                                BISB    R1,RØ   ;MOVE BITS LEFT
ØØØ5Ø6' 1Ø5ØØ1                                CLRB    R1
ØØØ51Ø' 1627Ø3  ØØØØ1Ø                        SUB     #8.,R3  ;TELL EXPONENT ABOUT IT
ØØØ514' ØØØ761                                BR      BIT9
               ØØ1                            .ENDC
ØØØ516' ØØ57Ø3                       UTEST:   TST     R3      ;CHECK FOR UNDERFLOW
ØØØ52Ø' ØØ3322                                BGT     NORMD   ;JUMP IF NONE
ØØØ522' ØØ4567  ØØØØØØG              UNDERF:  JSR     R5,$ERR ;ERROR 5,2
ØØØ526' ØØØ4Ø1                                BR      UNDER
ØØØ53Ø'    ØØ5                                .BYTE   5
ØØØ531'    ØØ2                                .BYTE   2
ØØØ532' ØØ5ØØ1                       UNDER:   CLR     R1      ;UNDERFLOW, TREAT AS Ø
ØØØ534' ØØ5ØØ3                       ZERO:    CLR     R3      ;CLEAR EXPONENT
ØØØ536' ØØØ725                                BR      STORE
               ØØØ                            .ENDC
            ØØØØØ1                            .END
```

```
$ADR    MACY11,620  22-AUG-72  11:41  PAGE 5
ADR04.PAL        SYMBOL TABLE

AC      = 177302         ASH     = 177316         A1      = 000004         A2      = 000010
A2NZ      000102R        BIT9      000460R        BIT9A     000460R        B1      = 000006
B2      = 000012         ECHK      000206R        EXPA      000140R        F0      =%000000
MQ      = 177304         NCOMP     000250R        NOR     = 177312         NORMD     000366R
OUT       000422R        OVER      000432R        PC      =%000007         R0      =%000000
R1      =%000001         R2      =%000002         R3      =%000003         R4      =%000004
R5      =%000005         SCHK      000172R        SHFT      000212R        SHFTD     000332R
SHFTL     000302R        SHFTR     000232R        SHFTRL    000270R        SHFTR0    000322R
SIGNS   = 000000         SP      =%000006         STORE     000412R        SUB       000442R
UNDER     000532R        UNDERF    000522R        UTEST     000516R        ZERO      000534R
ZTEST     000476R        $ADR      000004RG       $ERR    = ****** G       $SBR      000000RG
.       = 000540R

          000540

 ERRORS DETECTED:  0
```

```
                                          .TITLE  $MLR05
           000000'                        .CSECT
                                          .GLOBL  $MLR,$ERRA
                                  ;       $MLR    THE REAL MULTIPLY ROUTINE
                                  ;
                                  ;
                                  ;       $MLR    V005A
                                  ;
                                  ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                                  ;
                                  ;       CALLED IN POLISH MODE.
                                  ;       REPLACES THE TOP TWO REALS ON THE STACK
                                  ;       WITH THEIR PRODUCT.
           000000                         R0=%0
           000001                         R1=%1
           000002                         R2=%2
           000003                         R3=%3
           000004                         R4=%4
           000005                         R5=%5
           000006                         SP=%6
           000007                         PC=%7
           177304                         MQ=177304
           177311                         SR=177311
           177314                         LSH=177314
           000000                         F0=%0
           000010                 A=8.
           000014                 B=12.
           000010                 RESLT=8.
           000002                 SIGN=2
              001                         .IFDF   FPU
                                  $MLR:   .WORD   170001  ;;SETF
                                          .WORD   172426  ;;LDF    (SP)+,F0         ;GET MULTIPLICAND
                                          .WORD   171026  ;;MULF   (SP)+,F0         ;MULTIPLY
                                          .WORD   174046  ;;STF    F0,-(SP)         ;PRODUCT TO STACK
                                          JMP     @(R4)+
              000                         .ENDC
              001                         .IFNDF  FPU
 000000' 010446                   $MLR:   MOV     R4,-(SP)
 000002' 010546                           MOV     R5,-(SP)
              002                         .IFNDF  EAE&MULDIV
 000004' 016602  000004                   MOV     A+0-4(SP),R2
 000010' 006302                           ASL     R2      ;SHIFT MULTIPLICAND
 000012' 006146                           ROL     -(SP)   ;KEEP SIGN
 000014' 005046                           CLR     -(SP)   ;CLEAR EXPONENT
 000016' 000302                           SWAB    R2
 000020' 110216                           MOVB    R2,@SP  ;KEEP MULTIPLICAND EXPONENT
 000022' 001507                           BEQ     ZERO1   ;JUMP IF ANSWER IS ZERO
 000024' 000261                           SEC             ;INSERT NORMAL BIT
 000026' 006002                           ROR     R2
 000030' 105002                           CLRB    R2
 000032' 156602  000013                   BISB    A+3(SP),R2
 000036' 005003                           CLR     R3
 000040' 156603  000012                   BISB    A+2(SP),R3
 000044' 000303                           SWAB    R3
```

```
000046' 006366  000014                   ASL     B(SP)   ;SHIFT HIGH MULTIPLIER
000052' 005566  000002                   ADC     SIGN(SP)        ;GET PRODUCT SIGN
000056' 105766  000015                   TSTB    B+1(SP)
000062' 001467                           BEQ     ZERO1   ;JUMP IF ZERO
000064' 006066  000014                   ROR     B(SP)   ;SIGN IS NOW ZERO
000070' 005000                           CLR     R0      ;CLEAR PRODUCT
000072' 005001                           CLR     R1
000074' 016604  000016                   MOV     B+2(SP),R4      ;GET LOW ORDER MULTIPLIER
000100' 001406                           BEQ     B2Z
000102' 012705  000017           B2NZ:   MOV     #15.,R5
000106' 004767  000220                   JSR     PC,MULT0
000112' 004767  000160                   JSR     PC,MULT ;DO LAST LOW BIT FULL PRECISION
000116' 016604  000014           B2Z:    MOV     B(SP),R4        ;GET HIGH ORDER BITS
000122' 012705  000007                   MOV     #7,R5   ;THERE ARE ONLY SEVEN OF THEM
000126' 004767  000144                   JSR     PC,MULT
000132' 004767  000144                   JSR     PC,MULT1        ;GO DO THE NORMAL BIT
000136' 062604                           ADD     (SP)+,R4        ;ADD EXPONENTS
                001                      .ENDC
                                 ;       EAE CODE
                002                      .IFDF   EAE
                                 ;       (A1+A2*2**-16)*(B1+B2*2**-16)
                                         MOV     #MQ,R4  ;POINT TO MQ
                                         MOV     #100000,R5      ;GET LEADING BIT
                                         MOV     B+2-4(SP),@R4   ;LOW ORDER B TO MQ
                                         MOV     B+0-4(SP),-(R4) ;HIGH TO AC
                                         BEQ     ZERO    ;JUMP IF 0
                                         INC     @#LSH   ;GET SIGN
                                         RORB    @#SR
                                         ROL     -(SP)   ;SAVE IT
                                         MOV     (R4)+,-(SP)     ;SAVE EXPONENT
                                         CLRB    @SP     ;RIGHT JUSTIFY IT
                                         SWAB    @SP
                                         MOV     #7,@#LSH        ;MOVE FRACTION LEFT
                                         MOV     @R4,-(SP)       ;SAVE B2
                                         BIS     R5,-(R4)        ;INSERT NORMAL BIT
                                         MOV     (R4)+,-(SP)     ;SAVE B1
                                         MOV     A+2+4(SP),@R4   ;LOW ORDER A TO MQ
                                         MOV     A+0+4(SP),-(R4) ;HIGH TO AC
                                         BEQ     ZERO2   ;JUMP IF 0
                                         INC     @#LSH   ;GET SIGN
                                         RORB    @#SR
                                         ADC     6(SP)   ;GET RESULT SIGN
                                         MOV     @R4,R3  ;GET EXPONENT
                                         CLRB    R3
                                         SWAB    R3
                                         ADD     R3,4(SP)        ;GET SUM OF EXPONENTS
                                         MOV     #7,@#LSH        ;LEFT JUSTIFY FRACTION
                                         MOV     (R4)+,R2        ;SAVE A1
                                         BIS     R5,R2   ;INSERT NORMAL BIT
                                         CLR     R0      ;CLEAR PRODUCT
                                         CLR     R1
                                         MOV     (R4)+,R3        ;SAVE A2
                                         BNE     A2NZ
                                         TST     -(R4)   ;POINT TO MQ
```

```
                                      BR      A2Z     ;SHORT CUT
                              A2NZ:   MOV     @SP,@R4 ;GET B1*A2
                                      CMP     -(R4),-(R4)     ;POINT TO AC
                                      ADD     R3,@R4  ;A2, 2'S COMP CORRECTION
                                      TST     R3
                                      BPL     A2P
                                      ADD     @SP,@R4 ;B1, CORRECTION
                              A2P:    MOV     (R4)+,R1        ;HIGH PRODUCT TO R1
                              A2Z:    MOV     2(SP),(R4)+     ;B2 TO MQ
                                      BNE     B2NZ
                                      TST     -(R4)   ;POINT TO MQ
                                      BR      B2Z     ;SHORT CUT
                              B2NZ:   MOV     R2,@R4  ;GET B2*A1
                                      CMP     -(R4),-(R4)     ;POINT TO AC
                                      ADD     2(SP),@R4       ;B2, CORRECTION
                                      TST     2(SP)
                                      BPL     B2P     ;JUMP IF B2 +
                                      ADD     R2,@R4  ;A1, CORRECTION
                              B2P:    ADD     (R4)+,R1        ;HIGH PRODUCT TO R1
                                      ADC     RØ
                              B2Z:    MOV     R2,(R4)+        ;A1 TO MQ
                                      ADD     R2,RØ
                                      MOV     @SP,@R4 ;GET A1*B1
                                      ADD     (SP)+,RØ
                                      ADD     -(R4),R1
                                      ADC     RØ
                                      ADD     -(R4),RØ        ;AC+RØ
                                      TST     (SP)+   ;POP B2
                                      MOV     (SP)+,R4        ;GET SUM OF EXPONENTS
              ØØ1                     .ENDC
                              ;       MUL/DIV CODE
              ØØ2                     .IFDF   MULDIV
                              ;       (A1+A2*2**-16)*(B1+B2*2**-16)
                                      MOV     B+2-4(SP),R5    ;LOW ORDER B
                                      MOV     B+Ø-4(SP),R4    ;HIGH ORDER
                                      BEQ     ZERO
                                      .WORD   Ø73427,1        ;;      ASHC    #1,R4   ;GET SIGN BIT
                                      ROL     -(SP)   ;SAVE IT
                                      MOV     R4,-(SP)        ;SAVE EXPONENT
                                      CLRB    @SP
                                      SWAB    @SP     ;RIGHT JUSTIFY
                                      .WORD   Ø73427,7        ;;      ASHC    #7,R4   ;LEFT JUSTIFY FRACTION
                                      MOV     R5,-(SP)        ;SAVE B2
                                      BIS     #1ØØØØØ,R4      ;INSERT NORMAL BIT
                                      MOV     R4,-(SP)        ;SAVE B1
                                      MOV     A+2+4(SP),R3    ;GET A2
                                      MOV     A+Ø+4(SP),R2    ;GET A1
                                      BEQ     ZERO2   ;JUMP IF RESULT TO BE Ø
                                      .WORD   Ø73227,1        ;;      ASHC    #1,R2   ;GET SIGN
                                      ADC     6(SP)   ;GET RESULT SIGN
                                      MOV     R2,RØ   ;GET EXPONENT
                                      CLRB    RØ
                                      SWAB    RØ
                                      ADD     RØ,4(SP)        ;GET SUM OF EXPONENTS
```

```
                                        .WORD   Ø73227,7        ;;      ASHC    #7,R2   ;GET A1
                                        BIS     #1ØØØØØ,R2      ;INSERT NORMAL BIT
                                        CLR     RØ      ;CLEAR ACCUMULATOR
                                        CLR     R1
                                        TST     R3      ;CHECK A2
                                        BEQ     A2Z     ;JUMP IF Ø
                                        .WORD   Ø7Ø4Ø3  ;;      MUL     R3,R4   ;GET A2*B1
                                        ADD     R3,R4
                                        TST     R3
                                        BPL     A2P     ;JUMP IF A2 +
                                        ADD     @SP,R4  ;B1 CORRECTION
                                A2P:    MOV     R4,R1   ;A2*B1*2**-16
                                A2Z:    MOV     2(SP),R4        ;B2 TO MULTIPLIER
                                        BEQ     B2Z     ;JUMP IF Ø
                                        .WORD   Ø7Ø4Ø2  ;;      MUL     R2,R4   ;GET A1*B2
                                        ADD     2(SP),R4
                                        TST     2(SP)
                                        BPL     B2P     ;JUMP IF B2 +
                                        ADD     R2,R4   ;A1 CORRECTION
                                B2P:    ADD     R4,R1   ;A1*B2*2**-16
                                        ADC     RØ
                                B2Z:    MOV     R2,R4   ;A1 TO MULTIPLIER
                                        ADD     R2,RØ
                                        .WORD   Ø7Ø416  ;;      MUL     @SP,R4  ;GET A1*B1
                                        ADD     (SP)+,RØ
                                        ADD     R5,R1   ;LOW ORDER A1*B1
                                        ADC     RØ
                                        ADD     R4,RØ   ;HIGH ORDER A1*B1
                                        TST     (SP)+   ;POP B2
                                        MOV     (SP)+,R4        ;GET SUM OF EXPONENTS
                  ØØ1                   .ENDC
ØØØ14Ø' ØØ61Ø1                          ROL     R1      ;SHIFT OUT NORMAL BIT
ØØØ142' ØØ61ØØ                          ROL     RØ
ØØØ144' 1Ø34Ø3                          BCS     NORM    ;JUMP IF IT WAS FOUND
ØØØ146' ØØ61Ø1                          ROL     R1
ØØØ15Ø' ØØ61ØØ                          ROL     RØ      ;MUST HAVE GOT IT NOW
ØØØ152' ØØ53Ø4                          DEC     R4      ;ADJUST EXPONENT
ØØØ154' 1627Ø4  ØØØ2ØØ          NORM:   SUB     #2ØØ,R4 ;TAKE OUT ONE OF THE EXCESS 128'S
ØØØ16Ø' ØØ3436                          BLE     UNDER   ;JUMP IF UNDERFLOW
ØØØ162' Ø227Ø4  ØØØ377                  CMP     #377,R4
ØØØ166' ØØ2427                          BLT     OVER    ;JUMP IF OVERFLOW
ØØØ17Ø' 1Ø5ØØ1                          CLRB    R1
ØØØ172' 15ØØØ1                          BISB    RØ,R1
ØØØ174' ØØØ3Ø1                          SWAB    R1
ØØØ176' 1Ø5ØØØ                          CLRB    RØ
ØØØ2ØØ' 15Ø4ØØ                          BISB    R4,RØ
ØØØ2Ø2' ØØØ3ØØ                          SWAB    RØ
ØØØ2Ø4' ØØ6Ø26                          ROR     (SP)+   ;GET PRODUCT SIGN
ØØØ2Ø6' ØØ6ØØØ                          ROR     RØ      ;INSERT IT IN RESULT
ØØØ21Ø' ØØ6ØØ1                          ROR     R1
ØØØ212' ØØ55Ø1                          ADC     R1
ØØØ214' ØØ55ØØ                          ADC     RØ
ØØØ216' 1Ø3414                          BCS     OVER1   ;JUMP IF OVERFLOW ON ROUND
ØØØ22Ø' 1Ø2413                          BVS     OVER1
```

```
000222' 010066  000010      OUT:    MOV     R0,RESLT(SP)    ;PUT OUT ANSWER
000226' 010166  000012              MOV     R1,RESLT+2(SP)
000232' 012605                      MOV     (SP)+,R5
000234' 012604                      MOV     (SP)+,R4
000236' 022626                      CMP     (SP)+,(SP)+     ;FLUSH TOP ARGUMENT
000240' 000134                      JMP     @(R4)+  ;RETURN
            002                     .IFDF   EAE!MULDIV
                            ZERO2:  CMP     (SP)+,(SP)+     ;POP B1,B2
            001                     .ENDC
000242' 022626              ZERO1:  CMP     (SP)+,(SP)+     ;POP SIGN AND EXPONENT
000244' 000411                      BR      ZERO
000246' 005726              OVER:   TST     (SP)+   ;FLUSH SIGN
000250' 012700  006003      OVER1:  MOV     #6003,R0        ;ERROR 3,12
000254' 000403                      BR      ECALL
000256' 012700  003405      UNDER:  MOV     #3405,R0        ;ERROR 5,7
000262' 005726                      TST     (SP)+   ;FLUSH SIGN
000264' 004567  000000G     ECALL:  JSR     R5,$ERRA        ;CALL ERROR
000270' 005000              ZERO:   CLR     R0      ;CLEAR RESULT
000272' 005001                      CLR     R1
000274' 000752                      BR      OUT
            002                     .IFNDF  EAE&MULDIV
000276' 006204              MULT:   ASR     R4      ;TEST NEXT MULTIPLIER BIT
000300' 103004                      BCC     X0      ;JUMP IF IT IS 0
000302' 060301              MULT1:  ADD     R3,R1
000304' 005500                      ADC     R0
000306' 103406                      BCS     COVER
000310' 060200                      ADD     R2,R0
000312' 006000              X0:     ROR     R0      ;NOW SHIFT PRODUCT
000314' 006001                      ROR     R1
000316' 005305                      DEC     R5      ;COUNT LOOP
000320' 003366                      BGT     MULT    ;AGAIN PLEASE
000322' 000207                      RTS     PC      ;RETURN TO CALLER
000324' 060200              COVER:  ADD     R2,R0   ;FIRST ADD OVERFLOWED R0
000326' 000261                      SEC             ;SHOW THIS OVERFLOW TO SHIFT
000330' 000770                      BR      X0
000332' 006204              MULT0:  ASR     R4      ;REDUCED PRECISION MULTIPLY
000334' 103001                      BCC     X00
000336' 060200                      ADD     R2,R0   ;USE ONLY HIGH ORDER MULTIPLICAND
000340' 006000              X00:    ROR     R0
000342' 006001                      ROR     R1
000344' 005305                      DEC     R5
000346' 003371                      BGT     MULT0
000350' 000207                      RTS     PC
            001                     .ENDC
            000                     .ENDC
        000001                      .END
```

```
MLRØ5.PAL       SYMBOL TABLE

A       = ØØØØ1Ø        B       = ØØØØ14        B2NZ      ØØØ1Ø2R        B2Z       ØØØ116R
COVER     ØØØ324R       ECALL     ØØØ264R       FØ      =%ØØØØØØ         LSH     = 177314
MQ      = 1773Ø4        MULT      ØØØ276R       MULTØ     ØØØ332R        MULT1     ØØØ3Ø2R
NORM      ØØØ154R       OUT       ØØØ222R       OVER      ØØØ246R        OVER1     ØØØ25ØR
PC      =%ØØØØØ7        RESLT   = ØØØØ1Ø        RØ      =%ØØØØØØ         R1      =%ØØØØØ1
R2      =%ØØØØØ2        R3      =%ØØØØØ3        R4      =%ØØØØØ4         R5      =%ØØØØØ5
SIGN    = ØØØØØ2        SP      =%ØØØØØ6        SR      = 177311         UNDER     ØØØ256R
XØ        ØØØ312R       XØØ       ØØØ34ØR       ZERO      ØØØ27ØR        ZERO1     ØØØ242R
$ERRA   = ****** G      $MLR      ØØØØØØRG      .       = ØØØ352R

          ØØØ352

 ERRORS DETECTED:  Ø
```

```
                                     .TITLE  $DVRØ5
            ØØØØØØ'                  .CSECT
                                     .GLOBL  $DVR,$ERRA
                             ;       $DVR --- THE REAL DIVIDE ROUTINE
                             ;
                             ;
                             ;       $DVR    VØØ5A
                             ;
                             ;       COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                             ;
                             ;       CALLED IN THE POLISH MODE
                             ;       THE NUMERATOR IS THE SECOND ITEM ON THE STACK
                             ;       AND THE DENOMINATOR IS ON TOP,
                             ;       TAKES THE QUOTIENT AND PUTS IT ON TOP
                             ;       OF THE STACK IN THEIR PLACE
            ØØØØØØ                   RØ=%Ø
            ØØØØØ1                   R1=%1
            ØØØØØ2                   R2=%2
            ØØØØØ3                   R3=%3
            ØØØØØ4                   R4=%4
            ØØØØØ5                   R5=%5
            ØØØØØ6                   SP=%6
            ØØØØØ7                   PC=%7
            1773Ø4                   MQ=1773Ø4
            177312                   NOR=177312
            177314                   LSH=177314
            177316                   ASH=177316
            ØØØØØØ                   FØ=%Ø
            ØØØØØ1                   F1=%1
            ØØØØ1Ø                   D=8.
            ØØØØ14                   N=12.
            ØØØØ14                   Q=12.
               ØØ1                   .IFDF   FPU
                             $DVR:   .WORD   17ØØØ1  ;;SETF
                                     .WORD   172526  ;;LDF   (SP)+,F1        ;GET DIVISOR
                                     .WORD   172426  ;;LDF   (SP)+,FØ        ;GET DIVIDEND
                                     .WORD   1744Ø1  ;;DIVF  F1,FØ   ;DIVIDE
                                     .WORD   174Ø46  ;;STF   FØ,-(SP)        ;QUOTIENT TO STCK
                                     JMP     @(R4)+
               ØØØ                   .ENDC
               ØØ1                   .IFNDF  FPU
   ØØØØØØ' Ø1Ø446            $DVR:   MOV     R4,-(SP)
   ØØØØØ2' Ø1Ø546                    MOV     R5,-(SP)
   ØØØØØ4' ØØ5ØØØ                    CLR     RØ
   ØØØØØ6' ØØ5ØØ1                    CLR     R1
   ØØØØ1Ø' ØØ5Ø46                    CLR     -(SP)
   ØØØØ12' ØØ6366  ØØØØ12            ASL     N+Ø-2(SP)       ;SHIFT NUMERATOR
   ØØØØ16' ØØ6116                    ROL     @SP     ;GET NUMERATOR SIGN
   ØØØØ2Ø' ØØ5Ø46                    CLR     -(SP)
   ØØØØ22' 156616  ØØØØ15            BISB    N+1(SP),@SP     ;GET NUMERATOR EXPONENT
   ØØØØ26' ØØ1452                    BEQ     ZERO    ;JUMP IF NUMERATOR IS ZERO
   ØØØØ3Ø' 1566ØØ  ØØØØ14            BISB    N(SP),RØ
   ØØØØ34' ØØØ3ØØ                    SWAB    RØ      ;LEFT JUSTIFY NUMERATOR FRACTION
   ØØØØ36' ØØØ261                    SEC             ;INSERT NORMAL BIT
```

```
$DVRØ5  MACY11.62Ø  22-AUG-72  11:41  PAGE 2
DVRØ5.PAL

ØØØØ4Ø' ØØ6Ø2Ø                            ROR     RØ
ØØØØ42' 15662Ø  ØØØØ17                    BISB    N+3(SP),RØ
ØØØØ46' 156621  ØØØØ16                    BISB    N+2(SP),R1
ØØØØ52' ØØØ3Ø1                            SWAB    R1
ØØØØ54' ØØ5ØØ2                            CLR     R2
ØØØØ56' ØØ5ØØ3                            CLR     R3
ØØØØ6Ø' ØØ6366  ØØØØ1Ø                    ASL     D(SP)   ;SHIFT DENOMINATOR
ØØØØ64' ØØ5566  ØØØØØ2                    ADC     2(SP)   ;GET RESULT SIGN
ØØØØ7Ø' 1566Ø2  ØØØØ11                    BISB    D+1(SP),R2      ;GET DIVISOR EXPONENT
ØØØØ74' ØØ1431                            BEQ     DCHK    ;JUMP IF DIVISOR IS ZERO
ØØØØ76' 16Ø216                            SUB     R2,@SP  ;SUBTRACT EXPONENTS
ØØØ1ØØ' ØØ5ØØ2                            CLR     R2
ØØØ1Ø2' 1566Ø2  ØØØØ1Ø                    BISB    D(SP),R2        ;GET HIGH ORDER FRACTION
ØØØ1Ø6' ØØØ3Ø2                            SWAB    R2
ØØØ11Ø' ØØØ261                            SEC             ;INSERT NORMAL BIT
ØØØ112' ØØ6ØØ2                            ROR     R2
ØØØ114' 1566Ø2  ØØØØ13                    BISB    D+3(SP),R2
ØØØ12Ø' 1566Ø3  ØØØØ12                    BISB    D+2(SP),R3
ØØØ124' ØØØ3Ø3                            SWAB    R3
ØØØ126' Ø2ØØØ2                            CMP     RØ,R2   ;COMPARE HIGH NUMERATOR AND DENOMINATOR
ØØØ13Ø' 10344Ø                            BLO     DHI     ;JUMP IF DENOMINATOR HIGH
                   ØØ2                    .IFNDF  EAE&MULDIV
ØØØ132' 1Ø1Ø34                            BHI     DLOW    ;JUMP IF DENOMINATOR LOW
ØØØ134' Ø2Ø1Ø3                            CMP     R1,R3   ;COMPARE LOW ORDER PARTS
ØØØ136' 1Ø1Ø32                            BHI     DLOW
ØØØ14Ø' ØØ1Ø34                            BNE     DHI
ØØØ142' ØØ5Ø66  ØØØØ14                    CLR     Q(SP)   ;QUOTIENT FRACTION IS 1
ØØØ146' ØØ5216                            INC     @SP     ;BUMP EXPONENT
ØØØ15Ø' ØØ5ØØ5                            CLR     R5
ØØØ152' ØØØ445                            BR      FLOAT
                   ØØ1                    .ENDC
                   ØØ2                    .IFDF   EAE!MULDIV
                                          BHIS    DLOW    ;JUMP IF DENOMINATOR LOW OR SAME
                   ØØ1                    .ENDC
ØØØ154' Ø22626                    ZERO:   CMP     (SP)+,(SP)+     ;FLUSH EXP AND SIGN
ØØØ156' ØØØ415                            BR      ECALL1
ØØØ16Ø' ØØ5726                    DCHK:   TST     (SP)+   ;FLUSH EXP
ØØØ162' Ø127ØØ  ØØ4ØØ3                    MOV     #4ØØ3,RØ        ;ERROR 3,8
ØØØ166' ØØØ4Ø6                            BR      ECALL
ØØØ17Ø' ØØ5746                    OVER1:  TST     -(SP)   ;FAKE SIGN
ØØØ172' Ø127ØØ  ØØ3ØØ3            OVER:   MOV     #3ØØ3,RØ        ;ERROR 3,6
ØØØ176' ØØØ4Ø2                            BR      ECALL
ØØØ2ØØ' Ø127ØØ  ØØ24Ø5            UNDER:  MOV     #24Ø5,RØ        ;ERROR 5,3
ØØØ2Ø4' ØØ5726                    ECALL:  TST     (SP)+   ;FLUSH SIGN
ØØØ2Ø6' ØØ4567  ØØØØØØG                   JSR     R5,$ERRA
ØØØ212' ØØ5Ø66  ØØØØ1Ø            ECALL1: CLR     Q+Ø-4(SP)       ;RETURN Ø
ØØØ216' ØØ5Ø66  ØØØØ12                    CLR     Q+2-4(SP)
ØØØ222' ØØØ445                            BR      RTN
ØØØ224' ØØ6ØØØ                    DLOW:   ROR     RØ      ;HALVE NUMERATOR  (C=Ø)
ØØØ226' ØØ6ØØ1                            ROR     R1      ;TO ENSURE THAT N<D
ØØØ23Ø' ØØ5216                            INC     @SP     ;COMPENSATE EXPONENT
                   ØØ2                    .IFNDF  EAE&MULDIV
ØØØ232' Ø127Ø4  ØØØØ11            DHI:    MOV     #9.,R4  ;GO DO FIRST 9 QUOTIENT BITS
ØØØ236' ØØ4767  ØØØ1Ø4                    JSR     PC,DIV1
```

```
ØØØ242' 11Ø566  ØØØØ14                  MOVB    R5,Q(SP)        ;SAVE ALL HIGH ORDER Q FRACTION
                                                                ;EXCEPT NORMAL BIT
ØØØ246' ØØ57Ø4                          TST     R4      ;SEE IF DONE
ØØØ25Ø' ØØ14Ø2                          BEQ     NOTØ    ;NØ, NUMERATOR NOT Ø
ØØØ252' ØØ5ØØ5                          CLR     R5      ;ALL THE REST OF THE QUOTIENT IS ZERO
ØØØ254' ØØØ4Ø4                          BR      FLOAT
ØØØ256' Ø127Ø4  ØØØØ2Ø          NOTØ:   MOV     #16.,R4 ;GO DO 16 MORE BITS
ØØØ262' ØØ4767  ØØØØ6Ø                  JSR     PC,DIV1
        ØØ1                             .ENDC
        ØØ2                             .IFDF   EAE!MULDIV
                                DHI:    CLC
                                        ROR     RØ      ;ENSURE NUM. AND DENOM. +
                                        ROR     R1
                                        ROR     R2      ;LOW ORDER R1 AND R3 ARE Ø
                                        ROR     R3
                                        ROR     R3
                                        ROR     RØ
                                        ROR     R1
        ØØ1                             .ENDC
        ØØ2                             .IFDF   EAE
                                        MOV     #MQ,R5  ;POINT TO MQ
                                        MOV     R1,@R5  ;NUMERATOR TO AC,MQ
                                        MOV     RØ,-(R5)
                                        MOV     R2,-(R5)        ;(A+S*B)/C
                                        TST     (R5)+   ;POINT TO AC
                                        MOV     (R5)+,R1        ;KEEP REMAINDER
                                        MOV     (R5)+,R4        ;KEEP QUOTIENT
                                        MOV     R3,@R5  ;GET Q*D
                                        TST     -(R5)   ;POINT TO MQ
                                        ASR     R1      ;SCALE R
                                        SUB     R1,-(R5)        ;Q*D-R
                                        DEC     @#ASH
                                        MOV     R2,-(R5)        ;(Q*D-R)/C
                                        CMP     (R5)+,(R5)+     ;MQ
                                        NEG     @R5
                                        MOV     #2,@#ASH        ;MULT BY 4
                                        ADD     R4,-(R5)        ;Q+(Q*D-R)*S/C
                                        CLR     @#NOR   ;NORMALIZE
                                        SUB     @#NOR,@SP       ;APPLY TO EXPONENT
                                        MOV     #-6,@#LSH       ;POSITION NORMAL BIT
                                        MOV     (R5)+,Q(SP)     ;STORE QUOTIENT
                                        MOV     @R5,R5
        ØØ1                             .ENDC
        ØØ2                             .IFDF   MULDIV
                                        MOV     RØ,R4   ;NUMERATOR TO DIVIDEND
                                        MOV     R1,R5
                                        .WORD   Ø714Ø2  ;;      DIV     R2,R4   ;(A+S*B)/C
                                        MOV     R5,R1   ;SAVE REMAINDER
                                        MOV     R4,RØ   ;SAVE QUOTIENT
                                        .WORD   Ø7Ø4Ø3  ;;      MUL     R3,R4   ;GET Q*D
                                        ASR     R1      ;SCALE R
                                        SUB     R1,R4   ;Q*D-R
                                        .WORD   Ø73427,-1       ;;      ASHC    #-1,R4  ;SCALE
                                        .WORD   Ø714Ø2  ;;      DIV     R2,R4   ;GET (Q*D-R)/C
                                NEG     R4      ;(R-Q*D)/C
                                .WORD   Ø73427,-14.     ;;      ASHC    #14.,R4 ;UNSCALE
                                ADD     RØ,R4   ;Q+(R-Q*D)*S/C
                        NBTST:  .WORD   Ø73427,1        ;;      ASHC    #1,R4   ;SHIFT
                                BMI     NBIT    ;CHECK FOR NORMAL BIT
                                DEC     @SP     ;COMPENSATE EXPONENT
                                BR      NBTST   ;GO AGAIN
                        NBIT:   .WORD   Ø73427,-7       ;;ASHC  #-8,R4  ;ALIGN FRACTION
                                MOV     R4,Q(SP)        ;STORE HIGH ORDER
           ØØ1                  .ENDC
ØØØ266' Ø126Ø4          FLOAT:  MOV     (SP)+,R4        ;PUSH UP EXPONENT
ØØØ27Ø' Ø627Ø4  ØØØ2ØØ          ADD     #2ØØ,R4 ;ADD IN EXCESS 2ØØ
ØØØ274' ØØ3741                  BLE     UNDER   ;UNDERFLOW
ØØØ276' Ø227Ø4  ØØØ377          CMP     #377,R4
ØØØ3Ø2' ØØ2733                  BLT     OVER    ;OVERFLOW
ØØØ3Ø4' 11Ø466  ØØØØ13          MOVB    R4,Q+1-2(SP)    ;INSERT EXPONENT IN RESULT
ØØØ31Ø' ØØ6Ø26          SIGN:   ROR     (SP)+   ;INSERT QUOTIENT SIGN
ØØØ312' ØØ6Ø66  ØØØØ1Ø          ROR     Q+Ø-4(SP)
ØØØ316' ØØ6ØØ5                  ROR     R5
ØØØ32Ø' ØØ55Ø5                  ADC     R5      ;ROUND
ØØØ322' ØØ5566  ØØØØ1Ø          ADC     Q+Ø-4(SP)
ØØØ326' Ø1Ø566  ØØØØ12          MOV     R5,Q+2-4(SP)    ;INSERT LOW ORDER FRACTION
ØØØ332' 1Ø3716                  BCS     OVER1
ØØØ334' 1Ø2715                  BVS     OVER1
ØØØ336' Ø126Ø5          RTN:    MOV     (SP)+,R5
ØØØ34Ø' Ø126Ø4                  MOV     (SP)+,R4
ØØØ342' Ø22626                  CMP     (SP)+,(SP)+     ;FLUSH FIRST ARGUMENT
ØØØ344' ØØØ134                  JMP     @(R4)+
           ØØ2                  .IFNDF  EAE&MULDIV
ØØØ346' ØØ63Ø5          DIV1:   ASL     R5      ;SHIFT QUOTIENT
ØØØ35Ø' ØØ63Ø1                  ASL     R1      ;SHIFT NUMERATOR
ØØØ352' ØØ61ØØ                  ROL     RØ
ØØØ354' 1Ø34Ø6                  BCS     GO      ;GUARANTEED TO GO
ØØØ356' Ø2Ø2ØØ                  CMP     R2,RØ   ;COMPARE HIGH DIVISOR AND DIVIDEND
ØØØ36Ø' 1Ø1Ø1Ø                  BHI     NOGO    ;JUMP IF DIVISOR BIGGER
ØØØ362' 1Ø34Ø3                  BLO     GO      ;JUMP IF DIVISOR SMALLER
ØØØ364' Ø2Ø3Ø1                  CMP     R3,R1   ;CHECK THE LOW ORDERS
ØØØ366' 1Ø1ØØ5                  BHI     NOGO
ØØØ37Ø' ØØ14Ø7                  BEQ     NEQD    ;JUMP IF NUMERATOR =DENOMINATOR
ØØØ372' 16Ø3Ø1          GO:     SUB     R3,R1   ;N=N-D
ØØØ374' ØØ56ØØ                  SBC     RØ
ØØØ376' 16Ø2ØØ                  SUB     R2,RØ
ØØØ4ØØ' ØØ52Ø5                  INC     R5      ;INSERT QUOTIENT BIT
ØØØ4Ø2' ØØ53Ø4          NOGO:   DEC     R4      ;COUNT LOOP
ØØØ4Ø4' ØØ336Ø                  BGT     DIV1
ØØØ4Ø6' ØØØ2Ø7                  RTS     PC
ØØØ41Ø' ØØ52Ø5          NEQD:   INC     R5      ;INSERT LAST 1 BIT IN QUOTIENT
ØØØ412' ØØØ4Ø1                  BR      EQ1
ØØØ414' ØØ63Ø5          EQ2:    ASL     R5      ;FINISH OUT QUOTIENT WITH Ø'S
ØØØ416' ØØ53Ø4          EQ1:    DEC     R4
ØØØ42Ø' ØØ3375                  BGT     EQ2
ØØØ422' ØØ52Ø4                  INC     R4      ;FLAG NO MORE NUMERATOR
ØØØ424' ØØØ2Ø7          RTS:    RTS     PC      ;RETURN TO CALLER
           ØØ1                  .ENDC
```

```
                ØØØ                                 .ENDC
             ØØØØØ1                                 .END
```

```
$DVRØ5  MACY11.62Ø  22-AUG-72  11:41  PAGE 6
DVRØ5.PAL       SYMBOL TABLE

ASH     = 177316        D       = ØØØØ1Ø        DCHK      ØØØ16ØR       DHI       ØØØ232R
DIV1      ØØØ346R       DLOW      ØØØ224R       ECALL     ØØØ2Ø4R       ECALL1    ØØØ212R
EQ1       ØØØ416R       EQ2       ØØØ414R       FLOAT     ØØØ266R       FØ      =%ØØØØØØ
F1      =%ØØØØØ1        GO        ØØØ372R       LSH     = 177314        MQ      = 1773Ø4
N       = ØØØØ14        NEQD      ØØØ41ØR       NOGO      ØØØ4Ø2R       NOR     = 177312
NOTØ      ØØØ256R       OVER      ØØØ172R       OVER1     ØØØ17ØR       PC      =%ØØØØØ7
Q       = ØØØØ14        RTN       ØØØ336R       RTS       ØØØ424R       RØ      =%ØØØØØØ
R1      =%ØØØØØ1        R2      =%ØØØØØ2        R3      =%ØØØØØ3        R4      =%ØØØØØ4
R5      =%ØØØØØ5        SIGN      ØØØ31ØR       SP      =%ØØØØØ6        UNDER     ØØØ2ØØR
ZERO      ØØØ154R       $DVR      ØØØØØØRG      $ERRA   = ****** G      .       = ØØØ426R

          ØØØ426

 ERRORS DETECTED:  Ø
```

```
   217           ØØØ                          .ENDC
   218        ØØØØØ1                          .END
```

```
$DVR05  MACY11,620  22-AUG-72  11:41  PAGE 6
DVR05.PAL       SYMBOL TABLE

ASH     = 177316        D       = 000010        DCHK      000160R       DHI       000232R
DIV1      000346R       DLOW      000224R       ECALL     000204R       ECALL1    000212R
EQ1       000416R       EQ2       000414R       FLOAT     000266R       F0      =%000000
F1      =%000001        GO        000372R       LSH     = 177314        MQ      = 177304
N       = 000014        NEQD      000410R       NOGO      000402R       NOR     = 177312
NOT0      000256R       OVER      000172R       OVER1     000170R       PC      =%000007
Q       = 000014        RTN       000336R       RTS       000424R       R0      =%000000
R1      =%000001        R2      =%000002        R3      =%000003        R4      =%000004
R5      =%000005        SIGN      000310R       SP      =%000006        UNDER     000200R
ZERO      000154R       $DVR      000000RG      $ERRA   = ****** G      .       = 000426R

          000426

 ERRORS DETECTED:  0
```

```
LNKX11   V021   22-AUG-72   11:41

#DBKEBA/T:17440,DBKEBA<DBKEBA,ADR04,MLR05,DVR05,/E


LOAD MAP

TRANSFER ADDRESS: 000001
LOW LIMIT: 015700
HIGH LIMIT: 017440
**********
MODULE  MAINDE
SECTION ENTRY   ADDRESS SIZE
<, ABS.>        000000  000300
        $ERR    013414
        $ERRA   013420
<      >        015700  000000
**********
MODULE  $ADR
SECTION ENTRY   ADDRESS SIZE
<      >        015700  000540
        $ADR    015704
        $SBR    015700
**********
MODULE  $MLR05
SECTION ENTRY   ADDRESS SIZE
<      >        016440  000352
        $MLR    016440
**********
MODULE  $DVR05
SECTION ENTRY   ADDRESS SIZE
<      >        017012  000426
        $DVR    017012

 RUN-TIME:  2 SECONDS
 2K CORE USED
```