KE11-F

EXERCISER

MD-11-DBKEB-A

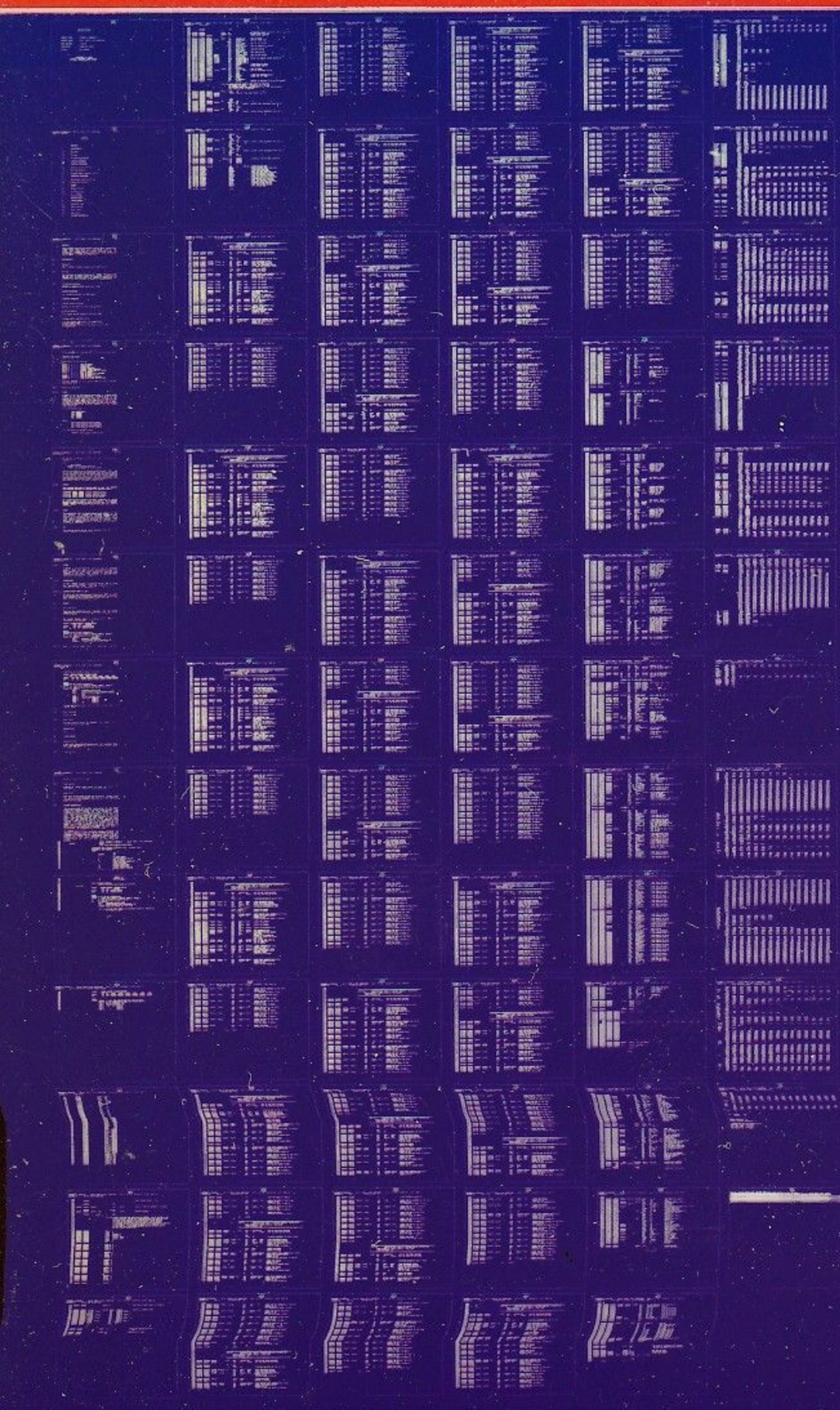
EP-DBKEB-A-DL-A

FICHE 1 OF 1

NOV 1976

digital

MADE IN USA



IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-D8KEB-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-D8KEA-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
          0 ......  BELL ON PASS COMPLETE
SW<09> =  1 ......  LOOP ON ERROR
SW<08> =  1 ......  LOOP ON TEST IN SW<6:0>
SW<07> =  1 ......  INPUT DATA FROM THE TELETYPE
```

Caution: SW<8:0> 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 0, 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 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 "LADS". 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 LADS 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 variety 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 interupte

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 ta 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-D8KEB-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)!
```

```
511             104400                  SCOPE=  TRAP
512             104000                  HLT=    EMT
513             000004                  TYPE=   IOT
514             177776                  PS=     177776
515             177570                  SWR=    177570
516             177570                  DISPLAY=SWR
517             000007                  BELL=   7
518             000000                  R0=     %0
519             000001                  R1=     %1
520             000002                  R2=     %2
521             000003                  R3=     %3
522             000004                  R4=     %4
523             000005                  R5=     %5
524             000005                  TTY=    %5
525             000006                  SP=     %6
526             000007                  PC=     %7
527             100000                  SW15=   100000
528             040000                  SW14=    40000
529             020000                  SW13=    20000
530             010000                  SW12=    10000
531             004000                  SW11=     4000
532             002000                  SW10=     2000
533             001000                  SW09=     1000
534             000400                  SW08=      400
535             000001                  BIT0 =  000001
536             000002                  BIT1 =  000002
537             000004                  BIT2 =  000004
538             000010                  BIT3 =  000010
539             000020                  BIT4 =  000020
540             000040                  BIT5 =  000040
541             000100                  BIT6 =  000100
542             000200                  BIT7 =  000200
543             000400                  BIT8 =  000400
544             001000                  BIT9 =  001000
545             002000                  BIT10 = 002000
546             004000                  BIT11 = 004000
547             010000                  BIT12 = 010000
548             020000                  BIT13 = 020000
549             040000                  BIT14 = 040000
550             100000                  BIT15 = 100000
551             000000                  LEVEL0 = 000
552             000040                  LEVEL1 = 040
553             000100                  LEVEL2 = 100
554             000140                  LEVEL3 = 140
555             000200                  LEVEL4 = 200
556             000240                  LEVEL5 = 240
557             000300                  LEVEL6 = 300
558             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

        000670  000000                  SAVSTK: 0
```

```
                                        RNDFLG: 0                       ;FOR FLAGS TO KEEP TRACK OF ROUNDING

000672  000000

                                        RAND4$: DECB    RAND.D          ;INSURE ALL ZEROES WORKS
000674  105367  177726
000700  066767  177716  177712                ADD     RAND.B, RAND.A
000706  005567  177714                        ADC     RAND.D
000712  066767  177706  177702                ADD     RAND.C, RAND.B
000720  005567  177700                        ADC     RAND.C
000724  066767  177676  177672                ADD     RAND.D, RAND.C
000732  005567  177664                        ADC     RAND.B
000736  066767  177656  177662                ADD     RAND.A, RAND.D
000744  005567  177650                        ADC     RAND.A
000750  000207                                RTS     PC

000752  000006                  YESRT:  RTT                             ;TRACE TRAP SERVICE ROUTINE

000754  104000                  FISTRP: HLT                             ;ERRONIOUS FIS TRAP
000756  000002                          RTI
```

```
                    001000                          .=      1000
                    000000                  ICNT:   0                       ;ITTERATION COUNT (HI BYTE); TEST # (LO BYTE)
                    000000                  ERRORS: 0                       ;ERROR COUNT LOCATION
                    000000  000000          PCNT:   0,0                     ;PASS COUNT LOCATION

                    012706  000604  BEGIN:  MOV     #STACKO,SP              ;SET UP STACK
                    012737  000752  000014  MOV     #YESRT,@#14             ;SET UP TRACE TRAP
                    012700  000020          MOV     #20,R0
                    012720  015256          MOV     #.IOT,(R0)+             ;SET UP IOT VECTOR
                    012720  000340          MOV     #340,(R0)+
                    012720  015536          MOV     #PDOWN$,(R0)+           ;SET UP POWER FAIL VECTOR
                    012720  000340          MOV     #340,(R0)+
                    012720  014020          MOV     #HLT$,(R0)+             ;SET EMT VECTOR
                    012720  000340          MOV     #340,(R0)+
                    012720  013644          MOV     #SCOPE$,(R0)+           ;SET TRAP VECTOR
                    012720  000340          MOV     #340,   (R0)+
                    012737  000754  000244  MOV     #FISTRP,@#244           ;SET UP FIS VECTOR
                    012737  000340  000246  MOV     #340,@#246
                    012767  123456  177510  MOV     #123456,RAND.A          ;PRIME THE RANDOM NUMBER GENERATOR
                    012767  107654  177504  MOV     #107654,RAND.B
                    012767  070707  177500  MOV     #070707,RAND.C
                    012767  125252  177474  MOV     #125252,RAND.D
                    005067  177644          CLR     ERRORS                  ;CLEAR ERROR COUNTER
                    005067  177642          CLR     PCNT                    ;CLEAR PASS COUNTER
                    005067  177640          CLR     PCNT+2
                    012706  000604  START:  MOV     #STACKO,SP              ;SET UP STACK
                    012737  000140  177776  MOV     #140,   @#PS            ;SET UP PROCESSOR STATUS
                    005067  177614          CLR     ICNT
                    005067  012622          CLR     LAD$
                    005067  177476          CLR     RNDFLG                  ;CLEAR THE ROUNDING FLAGS
                    105737  177570          TSTB    @#SWR                   ;CHECK FOR TTY INPUT
                    100403                  BMI     TYPIN
                    004767  177466          JSR     PC,RAND4$
                    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 ARGUEMENT AT A TIME, AND RE-ASKES
                                    ;       WHEN INVALID DATA IS ENTERED.

                    000004  001214  TYPIN:  TYPE,   .+2
                    005015  054524  042520          .ASCIZ  <15><12>"TYPE INPUT DATA:"<15><12>
                    044440  050116  052125
                    042040  052101  035101
                    005015     000
                    001242          .EVEN
                    000004  001246  1$:     TYPE,   .+2
                    030501  020072  000040          .ASCIZ  "A1: "
                    004567  011502          JSR     R5,     READIN  ;ACCEPT FIRST ARGUEMENT FROM THE TTY
                    000620                  RAND.A
                    103752                  BCS     TYPIN
                    000004  001270  2$:     TYPE,   .+2
                    031101  020072  000040          .ASCIZ  "A2: "
                    004567  011460          JSR     R5,     READIN  ;ACCEPT SECOND ARGUEMENT FROM THE TTY
```

```
                                                RAND.B
                                                BCS     2$
                                                BNE     TYPIN
                                        3$:     TYPE,   .+2
                                                .ASCIZ  "B1:  "
                                                JSR     R5,     READIN  ;ACCEPT THIRD ARGUEMENT FROM THE TTY
                                                RAND.C
                                                BCS     3$
                                                BNE     TYPIN
                                        4$:     TYPE,   .+2
                                                .ASCIZ  "B2:  "
                                                JSR     R5,     READIN  ;ACCEPT FOURTH ARGUEMENT FROM THE TTY
                                                RAND.D
                                                BCS     4$
                                                BNE     TYPIN

                                FORTAN: CLR     $ADDPS          ;CLEAR ALL THE PS SAVE LOCATIONS
                                        CLR     $SUBPS
                                        CLR     $MULPS
                                        CLR     $DIVPS

                                        JSR     %4,     $POLSH  ;ENTER POLISH MODE
                                        $PUSH                   ;PUSH THE DATA ONTO THE STACK
                                        $ADR                    ;FORTAN ADD ROUTINE
                                        $POPAD                  ;SAVE THE ADD ANSWERS
                                        $PUSH                   ;PUSH THE DATA ONTO THE STACK
                                        $SBR                    ;FORTRAN SUBTRACT ROUTINE
                                        $POPSB                  ;SAVE THE SUBTRACT ANSWERS
                                        $PUSH                   ;PUSH THE DATA ONTO THE STACK
                                        $MLR                    ;FORTRAN MULTIPLY ROUTINE
                                        $POPML                  ;SAVE THE MULTIPLY ANSWERS
                                        $PUSH                   ;PUSH THE DATA ONTO THE STACK
                                        $DVR                    ;FORTRAN DIVIDE ROUTINE
                                        $POPDV                  ;SAVE THE DIVIDE ANSWERS
                                        $EXIT                   ;EXIT POLISH MODE

                                        SCOPE
```

```
                                ;*************************************************************************
                                ;TEST 1:        EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
                                ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
                                ;       STACK POINTER = R0
                                ;*************************************************************************

001440  012700  000604          TST1:   MOV     #STACK0,R0      ;SET UP THE STACK POINTER
001444  004767  012130                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

001450  000240                          NOP
001452  075000                          FADD+   R0              ;FLOATING ADD ON THE R0 STACK

001454  013767  177776  177132  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
001462  010067  177130                  MOV     R0,     $SP     ;SAVE THE STACK POINTER
001466  026767  177136  177120  6$:     CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
001474  001023                          BNE     4$              ;GO CHECK FOR ROUNDING ERROR

001476  105767  177112                  TSTB    $PSW            ;CHECK FOR ERROR
001502  100464                          BMI     2$              ;BRANCH IF ERROR

001504  012767  000610  177156          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
001512  026767  177152  177076          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
001520  001401                          BEQ     .+4             ;BRANCH IF OK
001522  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

001524  026767  177102  177056          CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
001532  001004                          BNE     4$
001534  026767  177074  177050          CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
001542  001515                          BEQ     3$
001544  032767  000002  177120  4$:     BIT     #BIT1,  RNDFLG  ;CHECK THE ROUNDING FLAG
001552  001022                          BNE     5$
001554  052767  000002  177110          BIS     #BIT1,  RNDFLG  ;SET ROUNDING FLAG
001562  062767  000001  177044          ADD     #1,     $ADD2   ;INCREMENT FORTRAN ANSWER
001570  005567  177036                  ADC     $ADD1           ;ADD CARRY
001574  102334                          BVC     6$              ;BRANCH IF NO OVERFLOW
001576  000257                          CCC                     ;CLEAR ALL CONDITION CODES
001600  000262                          SEV                     ;SET V-BIT
001602  013767  177776  177026          MOV     @#PS,   $ADDER  ;SET UP PSW FOR OVERFLOW
001610  012767  000340  177012          MOV     #340,   $ADDPS  ;SET UP TRAP PSW
001616  000723                          BR      6$              ;TRY IT AGAIN

001620  132767  000002  177045  5$:     BITB    #BIT1,  RNDFLG+1 ;CHECK "DEROUNDING" FLAG
001626  001010                          BNE     7$              ;BRANCH IF SET
001630  152767  000002  177035          BISB    #BIT1,  RNDFLG+1 ;SET "DEROUNDING" FLAG
001636  162767  000001  176770          SUB     #1,     $ADD2   ;RESTORE ORIGINAL ANSWER
001644  005667  176762                  SBC     $ADD1           ;SUBTRACT CARRY
001650  104000                  7$:     HLT                     ;WRONG PSW OR ANSWER

001652  000451                          BR      3$

001654  012767  000604  177006  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
001662  026767  177002  176726          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
001670  001401                          BEQ     .+4             ;BRANCH IF OK
001672  104000                          HLT                     ;STACK POINTER FOULED UP
```

```
  784  001674  022767  001454  176676          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
  785  001702  001401                          BEQ     .+4             ;BRANCH IF OK
  786  001704  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
  787
  788  001706  026767  176724  176666          CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
  789  001714  001401                          BEQ     .+4             ;BRANCH IF OK
  790  001716  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
  791
  792  001720  026767  176700  176656          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
  793  001726  001401                          BEQ     .+4             ;BRANCH IF OK
  794  001730  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
  795
  796  001732  026767  176670  176646          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
  797  001740  001401                          BEQ     .+4             ;BRANCH IF OK
  798  001742  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
  799
  800  001744  026767  176650  176636          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
  801  001752  001401                          BEQ     .+4             ;BRANCH IF OK
  802  001754  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
  803
  804  001756  026767  176640  176626          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
  805  001764  001401                          BEQ     .+4             ;BRANCH IF OK
  806  001766  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
  807
  808  001770  012716  001776                  MOV     #3$,    (SP)    ;RESET THE STACK
  809  001774  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
  810
  811  001776  104400          3$:     SCOPE
  812
```

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

002000  012701  000604          TST2:   MOV     #STACK0,R1      ;SET UP THE STACK POINTER
002004  004767  011570                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

002010  000240                          NOP
002012  075011                          FSUB+   R1              ;FLOATING SUBTRACT ON THE R1 STACK

002014  013767  177776  176572  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
002022  010167  176570                  MOV     R1,     $SP     ;SAVE THE STACK POINTER
002026  026767  176606  176560  6$:     CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
002034  001023                          BNE     4$              ;GO CHECK FOR ROUNDING ERROR

002036  105767  176552                  TSTB    $PSW            ;CHECK FOR ERROR
002042  100464                          BMI     2$              ;BRANCH IF ERROR

002044  012767  000610  176616          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
002052  026767  176612  176536          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
002060  001401                          BEQ     .+4             ;BRANCH IF OK
002062  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

002064  026767  176552  176516          CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
002072  001004                          BNE     4$
002074  026767  176544  176510          CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
002102  001515                          BEQ     3$
002104  032767  000004  176560  4$:     BIT     #BIT2,  RNDFLG  ;CHECK THE ROUNDING FLAG
002112  001022                          BNE     5$
002114  052767  000004  176550          BIS     #BIT2,  RNDFLG  ;SET ROUNDING FLAG
002122  062767  000001  176514          ADD     #1,     $SUB2   ;INCREMENT FORTRAN ANSWER
002130  005567  176506                  ADC     $SUB1           ;ADD CARRY
002134  102334                          BVC     6$              ;BRANCH IF NO OVERFLOW
002136  000257                          CCC                     ;CLEAR ALL CONDITION CODES
002140  000262                          SEV                     ;SET V-BIT
002142  013767  177776  176476          MOV     @#PS,   $SUBER  ;SET UP PSW FOR OVERFLOW
002150  012767  000340  176462          MOV     #340,   $SUBPS  ;SET UP TRAP PSW
002156  000723                          BR      6$              ;TRY IT AGAIN

002160  132767  000004  176505  5$:     BITB    #BIT2,  RNDFLG+1 ;CHECK "DEROUNDING" FLAG
002166  001010                          BNE     7$              ;BRANCH IF SET
002170  152767  000004  176475          BISB    #BIT2,  RNDFLG+1 ;SET "DEROUNDING" FLAG
002176  162767  000001  176440          SUB     #1,     $SUB2   ;RESTORE ORIGINAL ANSWER
002204  005667  176432                  SBC     $SUB1           ;SUBTRACT CARRY
002210  104000                  7$:     HLT                     ;WRONG PSW OR ANSWER

002212  000451                          BR      3$

002214  012767  000604  176446  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
002222  026767  176442  176366          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
002230  001401                          BEQ     .+4             ;BRANCH IF OK
002232  104000                          HLT                     ;STACK POINTER FOULED UP
```

```
869  002234  022767  002014  176336          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
870  002242  001401                          BEQ     .+4             ;BRANCH IF OK
871  002244  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
872
873  002246  026767  176374  176326          CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
874  002254  001401                          BEQ     .+4             ;BRANCH IF OK
875  002256  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
876
877  002260  026767  176340  176316          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
878  002266  001401                          BEQ     .+4             ;BRANCH IF OK
879  002270  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
880
881  002272  026767  176330  176306          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
882  002300  001401                          BEQ     .+4             ;BRANCH IF OK
883  002302  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
884
885  002304  026767  176310  176276          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
886  002312  001401                          BEQ     .+4             ;BRANCH IF OK
887  002314  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
888
889  002316  026767  176300  176266          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
890  002324  001401                          BEQ     .+4             ;BRANCH IF OK
891  002326  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
892
893  002330  012716  002336                  MOV     #3$,    (SP)    ;RESET THE STACK
894  002334  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
895
896  002336  104400                  3$:     SCOPE
897
```

```
                        ;*****************************************************************************
                        ;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
```

```
954  002574  022767  002354  175776          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
955  002602  001401                          BEQ     .+4             ;BRANCH IF OK
956  002604  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
957
958  002606  026767  176044  175766          CMP     $MULER, STK2    ;CHECK THE PSW ON THE STACK
959  002614  001401                          BEQ     .+4             ;BRANCH IF OK
960  002616  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
961
962  002620  026767  176000  175756          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
963  002626  001401                          BEQ     .+4             ;BRANCH IF OK
964  002630  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
965
966  002632  026767  175770  175746          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
967  002640  001401                          BEQ     .+4             ;BRANCH IF OK
968  002642  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
969
970  002644  026767  175750  175736          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
971  002652  001401                          BEQ     .+4             ;BRANCH IF OK
972  002654  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
973
974  002656  026767  175740  175726          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
975  002664  001401                          BEQ     .+4             ;BRANCH IF OK
976  002666  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
977
978  002670  012716  002676                  MOV     #3$,    (SP)    ;RESET THE STACK
979  002674  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
980
981  002676  104400                  3$:     SCOPE
982
```

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

002700  012703  000604          TST4:   MOV     #STACK0,R3      ;SET UP THE STACK POINTER
002704  004767  010670                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

002710  000240                          NOP
002712  075033                          FDIV+   R3              ;FLOATING DIVIDE ON THE R3 STACK

002714  013767  177776  175672  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
002722  010367  175670                  MOV     R3,     $SP     ;SAVE THE STACK POINTER
002726  026767  175726  175660  6$:     CMP     $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
002734  001023                          BNE     4$              ;GO CHECK FOR ROUNDING ERROR

002736  105767  175652                  TSTB    $PSW            ;CHECK FOR ERROR
002742  100464                          BMI     2$              ;BRANCH IF ERROR

002744  012767  000610  175716          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
002752  026767  175712  175636          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
002760  001401                          BEQ     .+4             ;BRANCH IF OK
002762  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

002764  026767  175672  175616          CMP     $DIV1,  ANS1    ;CHECK THE ANSWER
002772  001004                          BNE     4$
002774  026767  175664  175610          CMP     $DIV2,  ANS2    ;CHECK THE ANSWER
003002  001515                          BEQ     3$
003004  032767  000020  175660  4$:     BIT     #BIT4,  RNDFLG  ;CHECK THE ROUNDING FLAG
003012  001022                          BNE     5$
003014  052767  000020  175650          BIS     #BIT4,  RNDFLG  ;SET ROUNDING FLAG
003022  062767  000001  175634          ADD     #1,     $DIV2   ;INCREMENT FORTRAN ANSWER
003030  005567  175626                  ADC     $DIV1           ;ADD CARRY
003034  102334                          BVC     6$              ;BRANCH IF NO OVERFLOW
003036  000257                          CCC                     ;CLEAR ALL CONDITION CODES
003040  000262                          SEV                     ;SET V-BIT
003042  013767  177776  175616          MOV     @#PS,   $DIVER  ;SET UP PSW FOR OVERFLOW
003050  012767  000340  175602          MOV     #340,   $DIVPS  ;SET UP TRAP PSW
003056  000723                          BR      6$              ;TRY IT AGAIN

003060  132767  000020  175605  5$:     BITB    #BIT4,  RNDFLG+1 ;CHECK "DEROUNDING" FLAG
003066  001010                          BNE     7$              ;BRANCH IF SET
003070  152767  000020  175575          BISB    #BIT4,  RNDFLG+1 ;SET "DEROUNDING" FLAG
003076  162767  000001  175560          SUB     #1,     $DIV2   ;RESTORE ORIGINAL ANSWER
003104  005667  175552                  SBC     $DIV1           ;SUBTRACT CARRY
003110  104000                  7$:     HLT                     ;WRONG PSW OR ANSWER

003112  000451                          BR      3$

003114  012767  000604  175546  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
003122  026767  175542  175466          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
003130  001401                          BEQ     .+4             ;BRANCH IF OK
003132  104000                          HLT                     ;STACK POINTER FOULED UP
```

```
1039  003134  022767  002714  175436          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1040  003142  001401                          BEQ     .+4             ;BRANCH IF OK
1041  003144  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1042
1043  003146  026767  175514  175426          CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
1044  003154  001401                          BEQ     .+4             ;BRANCH IF OK
1045  003156  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
1046
1047  003160  026767  175440  175416          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1048  003166  001401                          BEQ     .+4             ;BRANCH IF OK
1049  003170  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
1050
1051  003172  026767  175430  175406          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1052  003200  001401                          BEQ     .+4             ;BRANCH IF OK
1053  003202  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1054
1055  003204  026767  175410  175376          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1056  003212  001401                          BEQ     .+4             ;BRANCH IF OK
1057  003214  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1058
1059  003216  026767  175400  175366          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1060  003224  001401                          BEQ     .+4             ;BRANCH IF OK
1061  003226  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1062
1063  003230  012716  003236                  MOV     #3$,    (SP)    ;RESET THE STACK
1064  003234  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1065
1066  003236  104400          3$:     SCOPE
1067
```

```
1068
1069                                    ;*************************************************************************
1070                                    ;TEST 5:        EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
1071                                    ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
1072                                    ;       STACK POINTER = R4
1073                                    ;*************************************************************************
1074
1075  003240  012704  000604        TST5:   MOV     #STACK0,R4      ;SET UP THE STACK POINTER
1076  003244  004767  010330                JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1077
1078  003250  000240                        NOP
1079  003252  075004                        FADD+   R4              ;FLOATING ADD ON THE R4 STACK
1080
1081  003254  013767  177776  175332  1$:   MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1082  003262  010467  175330                MOV     R4,     $SP     ;SAVE THE STACK POINTER
1083  003266  026767  175336  175320        CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
1084  003274  001401                        BEQ     .+4             ;BRANCH IF OK
1085  003276  104000                        HLT                     ;PSW NOT EQUAL TO $ADDPS
1086
1087  003300  105767  175310                TSTB    $PSW            ;CHECK FOR ERROR
1088  003304  100423                        BMI     2$              ;BRANCH IF ERROR
1089
1090  003306  012767  000610  175354        MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1091  003314  026767  175350  175274        CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1092  003322  001401                        BEQ     .+4             ;BRANCH IF OK
1093  003324  104000                        HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
1094
1095  003326  026767  175300  175254        CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
1096  003334  001401                        BEQ     .+4             ;BRANCH IF OK
1097  003336  104000                        HLT                     ;LEFT HALF OF ANSWER WRONG
1098
1099  003340  026767  175270  175244        CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
1100  003346  001401                        BEQ     .+4             ;BRANCH IF OK
1101  003350  104000                        HLT                     ;RIGHT HALF OF ANSWER WRONG
1102
1103  003352  000451                        BR      3$
1104
1105  003354  012767  000604  175306  2$:   MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1106  003362  026767  175302  175226        CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1107  003370  001401                        BEQ     .+4             ;BRANCH IF OK
1108  003372  104000                        HLT                     ;STACK POINTER FOULED UP
1109
1110  003374  022767  003254  175176        CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1111  003402  001401                        BEQ     .+4             ;BRANCH IF OK
1112  003404  104000                        HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1113
1114  003406  026767  175224  175166        CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
1115  003414  001401                        BEQ     .+4             ;BRANCH IF OK
1116  003416  104000                        HLT                     ;RTI PSW NOT EQUAL TO 200
1117
1118  003420  026767  175200  175156        CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1119  003426  001401                        BEQ     .+4             ;BRANCH IF OK
1120  003430  104000                        HLT                     ;STK3 NOT EQUAL TO RAND.C
1121
1122  003432  026767  175170  175146        CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1123  003440  001401                        BEQ     .+4             ;BRANCH IF OK
```

```
1124  003442  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1125
1126  003444  026767  175150  175136          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1127  003452  001401                          BEQ     .+4             ;BRANCH IF OK
1128  003454  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1129
1130  003456  026767  175140  175126          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1131  003464  001401                          BEQ     .+4             ;BRANCH IF OK
1132  003466  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1133
1134  003470  012716  003476                  MOV     #3$,    (SP)    ;RESET THE STACK
1135  003474  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1136
1137  003476  104400                  3$:     SCOPE
1138
1139
1140                                  ;*************************************************************************
1141                                  ;TEST 6:        EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
1142                                  ;       RAND.A,RAND.B - RAND.C,RAND.D = ANS1,ANS2
1143                                  ;       STACK POINTER = R5
1144                                  ;*************************************************************************
1145
1146  003500  012705  000604          TST6:   MOV     #STACK0,R5      ;SET UP THE STACK POINTER
1147  003504  004767  010070                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1148
1149  003510  000240                          NOP
1150  003512  075015                          FSUB+   R5              ;FLOATING SUBTRACT ON THE R5 STACK
1151
1152  003514  013767  177776  175072  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1153  003522  010567  175070                  MOV     R5,     $SP     ;SAVE THE STACK POINTER
1154  003526  026767  175106  175060          CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
1155  003534  001401                          BEQ     .+4             ;BRANCH IF OK
1156  003536  104000                          HLT                     ;PSW NOT EQUAL TO $SUBPS
1157
1158  003540  105767  175050                  TSTB    $PSW            ;CHECK FOR ERROR
1159  003544  100423                          BMI     2$              ;BRANCH IF ERROR
1160
1161  003546  012767  000610  175114          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1162  003554  026767  175110  175034          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1163  003562  001401                          BEQ     .+4             ;BRANCH IF OK
1164  003564  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
1165
1166  003566  026767  175050  175014          CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
1167  003574  001401                          BEQ     .+4             ;BRANCH IF OK
1168  003576  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
1169
1170  003600  026767  175040  175004          CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
1171  003606  001401                          BEQ     .+4             ;BRANCH IF OK
1172  003610  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
1173
1174  003612  000451                          BR      3$
1175
1176  003614  012767  000604  175046  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1177  003622  026767  175042  174766          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1178  003630  001401                          BEQ     .+4             ;BRANCH IF OK
1179  003632  104000                          HLT                     ;STACK POINTER FOULED UP
```

```
1180
1181  003634  022767  003514  174736          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1182  003642  001401                          BEQ     .+4             ;BRANCH IF OK
1183  003644  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1184
1185  003646  026767  174774  174726          CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
1186  003654  001401                          BEQ     .+4             ;BRANCH IF OK
1187  003656  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
1188
1189  003660  026767  174740  174716          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1190  003666  001401                          BEQ     .+4             ;BRANCH IF OK
1191  003670  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
1192
1193  003672  026767  174730  174706          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1194  003700  001401                          BEQ     .+4             ;BRANCH IF OK
1195  003702  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1196
1197  003704  026767  174710  174676          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1198  003712  001401                          BEQ     .+4             ;BRANCH IF OK
1199  003714  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1200
1201  003716  026767  174700  174666          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1202  003724  001401                          BEQ     .+4             ;BRANCH IF OK
1203  003726  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1204
1205  003730  012716  003736                  MOV     #3$,    (SP)    ;RESET THE STACK
1206  003734  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1207
1208  003736  104400              3$:         SCOPE
1209
1210
1211                        ;*****************************************************************
1212                        ;TEST 7:        EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
1213                        ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
1214                        ;       STACK POINTER = SP
1215                        ;*****************************************************************
1216
1217  003740  012706  000604      TST7:       MOV     #STACK0,SP      ;SET UP THE STACK POINTER
1218  003744  004767  007630                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1219
1220  003750  000240                          NOP
1221  003752  075026                          FMUL+   SP              ;FLOATING MULTIPLY ON THE SP STACK
1222
1223  003754  013767  177776  174632  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1224  003762  010667  174630                  MOV     SP,     $SP     ;SAVE THE STACK POINTER
1225  003766  026767  174656  174620          CMP     $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
1226  003774  001401                          BEQ     .+4             ;BRANCH IF OK
1227  003776  104000                          HLT                     ;PSW NOT EQUAL TO $MULPS
1228
1229  004000  105767  174610                  TSTB    $PSW            ;CHECK FOR ERROR
1230  004004  100424                          BMI     2$              ;BRANCH IF ERROR
1231
1232  004006  012767  000610  174654          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1233  004014  026767  174650  174574          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1234  004022  001401                          BEQ     .+4             ;BRANCH IF OK
1235  004024  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
```

```
1236
1237  004026  026767  174620  174554        CMP     $MUL1,  ANS1    ;CHECK THE ANSWER
1238  004034  001401                        BEQ     .+4             ;BRANCH IF OK
1239  004036  104000                        HLT                     ;LEFT HALF OF ANSWER WRONG
1240
1241  004040  026767  174610  174544        CMP     $MUL2,  ANS2    ;CHECK THE ANSWER
1242  004046  001401                        BEQ     .+4             ;BRANCH IF OK
1243  004050  104000                        HLT                     ;RIGHT HALF OF ANSWER WRONG
1244
1245  004052  024646                        CMP     -(SP),  -(SP)   ;RESTORE THE STACK
1246  004054  000451                        BR      3$
1247
1248  004056  012767  000600  174604  2$:   MOV     #STK1,  SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1249  004064  026767  174600  174524        CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1250  004072  001401                        BEQ     .+4             ;BRANCH IF OK
1251  004074  104000                        HLT                     ;STACK POINTER FOULED UP
1252
1253  004076  022767  003754  174474        CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1254  004104  001401                        BEQ     .+4             ;BRANCH IF OK
1255  004106  104000                        HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1256
1257  004110  026767  174542  174464        CMP     $MULER, STK2    ;CHECK THE PSW ON THE STACK
1258  004116  001401                        BEQ     .+4             ;BRANCH IF OK
1259  004120  104000                        HLT                     ;RTI PSW NOT EQUAL TO 200
1260
1261  004122  026767  174476  174454        CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1262  004130  001401                        BEQ     .+4             ;BRANCH IF OK
1263  004132  104000                        HLT                     ;STK3 NOT EQUAL TO RAND.C
1264
1265  004134  026767  174466  174444        CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1266  004142  001401                        BEQ     .+4             ;BRANCH IF OK
1267  004144  104000                        HLT                     ;STK4 NOT EQUAL TO RAND.D
1268
1269  004146  026767  174446  174434        CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1270  004154  001401                        BEQ     .+4             ;BRANCH IF OK
1271  004156  104000                        HLT                     ;STK5 NOT EQUAL TO RAND.A
1272
1273  004160  026767  174436  174424        CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1274  004166  001401                        BEQ     .+4             ;BRANCH IF OK
1275  004170  104000                        HLT                     ;STK6 NOT EQUAL TO RAND.B
1276
1277  004172  012716  004200                MOV     #3$,    (SP)    ;RESET THE STACK
1278  004176  000002                        RTI                     ;RESTORE THE STATUS (T-BIT)
1279
1280  004200  104400            3$:         SCOPE
1281
```

```
1282
1283                                    ;*************************************************************************
1284                                    ;TEST 10:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
1285                                    ;       RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
1286                                    ;       STACK POINTER = RO
1287                                    ;*************************************************************************
1288
1289  004202  012700  000604           TST10:  MOV     #STACK0,RO      ;SET UP THE STACK POINTER
1290  004206  004767  007366                   JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1291
1292  004212  000240                           NOP
1293  004214  075030                           FDIV+   RO              ;FLOATING DIVIDE ON THE RO STACK
1294
1295  004216  013767  177776  174370   1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1296  004224  010067  174366                   MOV     RO,     $SP     ;SAVE THE STACK POINTER
1297  004230  026767  174424  174356           CMP     $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
1298  004236  001401                           BEQ     .+4             ;BRANCH IF OK
1299  004240  104000                           HLT                     ;PSW NOT EQUAL TO $DIVPS
1300
1301  004242  105767  174346                   TSTB    $PSW            ;CHECK FOR ERROR
1302  004246  100423                           BMI     2$              ;BRANCH IF ERROR
1303
1304  004250  012767  000610  174412           MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1305  004256  026767  174406  174332           CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1306  004264  001401                           BEQ     .+4             ;BRANCH IF OK
1307  004266  104000                           HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
1308
1309  004270  026767  174366  174312           CMP     $DIV1,  ANS1    ;CHECK THE ANSWER
1310  004276  001401                           BEQ     .+4             ;BRANCH IF OK
1311  004300  104000                           HLT                     ;LEFT HALF OF ANSWER WRONG
1312
1313  004302  026767  174356  174302           CMP     $DIV2,  ANS2    ;CHECK THE ANSWER
1314  004310  001401                           BEQ     .+4             ;BRANCH IF OK
1315  004312  104000                           HLT                     ;RIGHT HALF OF ANSWER WRONG
1316
1317  004314  000451                           BR      3$
1318
1319  004316  012767  000604  174344   2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1320  004324  026767  174340  174264           CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1321  004332  001401                           BEQ     .+4             ;BRANCH IF OK
1322  004334  104000                           HLT                     ;STACK POINTER FOULED UP
1323
1324  004336  022767  004216  174234           CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1325  004344  001401                           BEQ     .+4             ;BRANCH IF OK
1326  004346  104000                           HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1327
1328  004350  026767  174312  174224           CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
1329  004356  001401                           BEQ     .+4             ;BRANCH IF OK
1330  004360  104000                           HLT                     ;RTI PSW NOT EQUAL TO 200
1331
1332  004362  026767  174236  174214           CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1333  004370  001401                           BEQ     .+4             ;BRANCH IF OK
1334  004372  104000                           HLT                     ;STK3 NOT EQUAL TO RAND.C
1335
1336  004374  026767  174226  174204           CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1337  004402  001401                           BEQ     .+4             ;BRANCH IF OK
```

```
1338  004404  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1339
1340  004406  026767  174206  174174          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1341  004414  001401                          BEQ     .+4             ;BRANCH IF OK
1342  004416  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1343
1344  004420  026767  174176  174164          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1345  004426  001401                          BEQ     .+4             ;BRANCH IF OK
1346  004430  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1347
1348  004432  012716  004440                  MOV     #3$,    (SP)    ;RESET THE STACK
1349  004436  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1350
1351  004440  104400              3$:         SCOPE
1352
1353
1354                                ;*************************************************************************
1355                                ;TEST 11:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
1356                                ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
1357                                ;       STACK POINTER = R1
1358                                ;*************************************************************************
1359
1360  004442  012701  000604      TST11:      MOV     #STACK0,R1      ;SET UP THE STACK POINTER
1361  004446  004767  007126                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1362
1363  004452  000240                          NOP
1364  004454  075001                          FADD+   R1              ;FLOATING ADD ON THE R1 STACK
1365
1366  004456  013767  177776  174130  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1367  004464  010167  174126                  MOV     R1,     $SP     ;SAVE THE STACK POINTER
1368  004470  026767  174134  174116          CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
1369  004476  001401                          BEQ     .+4             ;BRANCH IF OK
1370  004500  104000                          HLT                     ;PSW NOT EQUAL TO $ADDPS
1371
1372  004502  105767  174106                  TSTB    $PSW            ;CHECK FOR ERROR
1373  004506  100423                          BMI     2$              ;BRANCH IF ERROR
1374
1375  004510  012767  000610  174152          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1376  004516  026767  174146  174072          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1377  004524  001401                          BEQ     .+4             ;BRANCH IF OK
1378  004526  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
1379
1380  004530  026767  174076  174052          CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
1381  004536  001401                          BEQ     .+4             ;BRANCH IF OK
1382  004540  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
1383
1384  004542  026767  174066  174042          CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
1385  004550  001401                          BEQ     .+4             ;BRANCH IF OK
1386  004552  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
1387
1388  004554  000451                          BR      3$
1389
1390  004556  012767  000604  174104  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1391  004564  026767  174100  174024          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1392  004572  001401                          BEQ     .+4             ;BRANCH IF OK
1393  004574  104000                          HLT                     ;STACK POINTER FOULED UP
```

```
1394
1395  004576  022767  004456  173774          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1396  004604  001401                          BEQ     .+4             ;BRANCH IF OK
1397  004606  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1398
1399  004610  026767  174022  173764          CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
1400  004616  001401                          BEQ     .+4             ;BRANCH IF OK
1401  004620  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
1402
1403  004622  026767  173776  173754          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1404  004630  001401                          BEQ     .+4             ;BRANCH IF OK
1405  004632  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
1406
1407  004634  026767  173766  173744          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1408  004642  001401                          BEQ     .+4             ;BRANCH IF OK
1409  004644  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1410
1411  004646  026767  173746  173734          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1412  004654  001401                          BEQ     .+4             ;BRANCH IF OK
1413  004656  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1414
1415  004660  026767  173736  173724          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1416  004666  001401                          BEQ     .+4             ;BRANCH IF OK
1417  004670  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1418
1419  004672  012716  004700                  MOV     #3$,    (SP)    ;RESET THE STACK
1420  004676  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1421
1422  004700  104400          3$:     SCOPE
1423
1424
1425                          ;*************************************************************************
1426                          ;TEST 12:       EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
1427                          ;       RAND.A,RAND.B - RAND.C,RAND.D = ANS1,ANS2
1428                          ;       STACK POINTER = R2
1429                          ;*************************************************************************
1430
1431  004702  012702  000604  TST12:  MOV     #STACK0,R2      ;SET UP THE STACK POINTER
1432  004706  004767  006666          JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1433
1434  004712  000240                  NOP
1435  004714  075012                  FSUB+   R2              ;FLOATING SUBTRACT ON THE R2 STACK
1436
1437  004716  013767  177776  173670  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1438  004724  010267  173666                  MOV     R2,     $SP     ;SAVE THE STACK POINTER
1439  004730  026767  173704  173656          CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
1440  004736  001401                          BEQ     .+4             ;BRANCH IF OK
1441  004740  104000                          HLT                     ;PSW NOT EQUAL TO $SUBPS
1442
1443  004742  105767  173646                  TSTB    $PSW            ;CHECK FOR ERROR
1444  004746  100423                          BMI     2$              ;BRANCH IF ERROR
1445
1446  004750  012767  000610  173712          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1447  004756  026767  173706  173632          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1448  004764  001401                          BEQ     .+4             ;BRANCH IF OK
1449  004766  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
```

```
1450
1451  004770  026767  173646  173612          CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
1452  004776  001401                          BEQ     .+4             ;BRANCH IF OK
1453  005000  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
1454
1455  005002  026767  173636  173602          CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
1456  005010  001401                          BEQ     .+4             ;BRANCH IF OK
1457  005012  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
1458
1459  005014  000451                          BR      3$
1460
1461  005016  012767  000604  173644  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1462  005024  026767  173640  173564          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1463  005032  001401                          BEQ     .+4             ;BRANCH IF OK
1464  005034  104000                          HLT                     ;STACK POINTER FOULED UP
1465
1466  005036  022767  004716  173534          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1467  005044  001401                          BEQ     .+4             ;BRANCH IF OK
1468  005046  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1469
1470  005050  026767  173572  173524          CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
1471  005056  001401                          BEQ     .+4             ;BRANCH IF OK
1472  005060  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
1473
1474  005062  026767  173536  173514          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1475  005070  001401                          BEQ     .+4             ;BRANCH IF OK
1476  005072  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
1477
1478  005074  026767  173526  173504          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1479  005102  001401                          BEQ     .+4             ;BRANCH IF OK
1480  005104  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1481
1482  005106  026767  173506  173474          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1483  005114  001401                          BEQ     .+4             ;BRANCH IF OK
1484  005116  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1485
1486  005120  026767  173476  173464          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1487  005126  001401                          BEQ     .+4             ;BRANCH IF OK
1488  005130  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1489
1490  005132  012716  005140                  MOV     #3$,    (SP)    ;RESET THE STACK
1491  005136  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1492
1493  005140  104400                  3$:     SCOPE
1494
```

```
1495
1496                                    ;*************************************************************************
1497                                    ;TEST 13:       EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
1498                                    ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
1499                                    ;       STACK POINTER = R3
1500                                    ;*************************************************************************
1501
1502  005142  012703  000604       TST13:  MOV     #STACK0,R3      ;SET UP THE STACK POINTER
1503  005146  004767  006426               JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1504
1505  005152  000240                       NOP
1506  005154  075023                       FMUL+   R3              ;FLOATING MULTIPLY ON THE R3 STACK
1507
1508  005156  013767  177776  173430  1$:  MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1509  005164  010367  173426               MOV     R3,     $SP     ;SAVE THE STACK POINTER
1510  005170  026767  173454  173416       CMP     $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
1511  005176  001401                       BEQ     .+4             ;BRANCH IF OK
1512  005200  104000                       HLT                     ;PSW NOT EQUAL TO $MULPS
1513
1514  005202  105767  173406               TSTB    $PSW            ;CHECK FOR ERROR
1515  005206  100423                       BMI     2$              ;BRANCH IF ERROR
1516
1517  005210  012767  000610  173452       MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1518  005216  026767  173446  173372       CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1519  005224  001401                       BEQ     .+4             ;BRANCH IF OK
1520  005226  104000                       HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
1521
1522  005230  026767  173416  173352       CMP     $MUL1,  ANS1    ;CHECK THE ANSWER
1523  005236  001401                       BEQ     .+4             ;BRANCH IF OK
1524  005240  104000                       HLT                     ;LEFT HALF OF ANSWER WRONG
1525
1526  005242  026767  173406  173342       CMP     $MUL2,  ANS2    ;CHECK THE ANSWER
1527  005250  001401                       BEQ     .+4             ;BRANCH IF OK
1528  005252  104000                       HLT                     ;RIGHT HALF OF ANSWER WRONG
1529
1530  005254  000451                       BR      3$
1531
1532  005256  012767  000604  173404  2$:  MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1533  005264  026767  173400  173324       CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1534  005272  001401                       BEQ     .+4             ;BRANCH IF OK
1535  005274  104000                       HLT                     ;STACK POINTER FOULED UP
1536
1537  005276  022767  005156  173274       CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1538  005304  001401                       BEQ     .+4             ;BRANCH IF OK
1539  005306  104000                       HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1540
1541  005310  026767  173342  173264       CMP     $MULER, STK2    ;CHECK THE PSW ON THE STACK
1542  005316  001401                       BEQ     .+4             ;BRANCH IF OK
1543  005320  104000                       HLT                     ;RTI PSW NOT EQUAL TO 200
1544
1545  005322  026767  173276  173254       CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1546  005330  001401                       BEQ     .+4             ;BRANCH IF OK
1547  005332  104000                       HLT                     ;STK3 NOT EQUAL TO RAND.C
1548
1549  005334  026767  173266  173244       CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1550  005342  001401                       BEQ     .+4             ;BRANCH IF OK
```

```
1551  005344  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1552
1553  005346  026767  173246  173234          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1554  005354  001401                          BEQ     .+4             ;BRANCH IF OK
1555  005356  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1556
1557  005360  026767  173236  173224          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1558  005366  001401                          BEQ     .+4             ;BRANCH IF OK
1559  005370  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1560
1561  005372  012716  005400                  MOV     #3$,    (SP)    ;RESET THE STACK
1562  005376  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1563
1564  005400  104400          3$:     SCOPE
1565
1566
1567                          ;*************************************************************************
1568                          ;TEST 14:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
1569                          ;       RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
1570                          ;       STACK POINTER = R4
1571                          ;*************************************************************************
1572
1573  005402  012704  000604  TST14:  MOV     #STACK0,R4      ;SET UP THE STACK POINTER
1574  005406  004767  006166          JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1575
1576  005412  000240                  NOP
1577  005414  075034                  FDIV+   R4              ;FLOATING DIVIDE ON THE R4 STACK
1578
1579  005416  013767  177776  173170  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1580  005424  010467  173166                  MOV     R4,     $SP     ;SAVE THE STACK POINTER
1581  005430  026767  173224  173156          CMP     $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
1582  005436  001401                          BEQ     .+4             ;BRANCH IF OK
1583  005440  104000                          HLT                     ;PSW NOT EQUAL TO $DIVPS
1584
1585  005442  105767  173146                  TSTB    $PSW            ;CHECK FOR ERROR
1586  005446  100423                          BMI     2$              ;BRANCH IF ERROR
1587
1588  005450  012767  000610  173212          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1589  005456  026767  173206  173132          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1590  005464  001401                          BEQ     .+4             ;BRANCH IF OK
1591  005466  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
1592
1593  005470  026767  173166  173112          CMP     $DIV1,  ANS1    ;CHECK THE ANSWER
1594  005476  001401                          BEQ     .+4             ;BRANCH IF OK
1595  005500  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
1596
1597  005502  026767  173156  173102          CMP     $DIV2,  ANS2    ;CHECK THE ANSWER
1598  005510  001401                          BEQ     .+4             ;BRANCH IF OK
1599  005512  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
1600
1601  005514  000451                  BR      3$
1602
1603  005516  012767  000604  173144  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1604  005524  026767  173140  173064          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1605  005532  001401                          BEQ     .+4             ;BRANCH IF OK
1606  005534  104000                          HLT                     ;STACK POINTER FOULED UP
```

```
1607
1608  005536  022767  005416  173034          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1609  005544  001401                          BEQ     .+4             ;BRANCH IF OK
1610  005546  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1611
1612  005550  026767  173112  173024          CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
1613  005556  001401                          BEQ     .+4             ;BRANCH IF OK
1614  005560  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
1615
1616  005562  026767  173036  173014          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1617  005570  001401                          BEQ     .+4             ;BRANCH IF OK
1618  005572  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
1619
1620  005574  026767  173026  173004          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1621  005602  001401                          BEQ     .+4             ;BRANCH IF OK
1622  005604  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1623
1624  005606  026767  173006  172774          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1625  005614  001401                          BEQ     .+4             ;BRANCH IF OK
1626  005616  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1627
1628  005620  026767  172776  172764          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1629  005626  001401                          BEQ     .+4             ;BRANCH IF OK
1630  005630  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1631
1632  005632  012716  005640                  MOV     #3$,    (SP)    ;RESET THE STACK
1633  005636  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1634
1635  005640  104400              3$:         SCOPE
1636
1637
1638                                ;*************************************************************************
1639                                ;TEST 15:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
1640                                ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
1641                                ;       STACK POINTER = R5
1642                                ;*************************************************************************
1643
1644  005642  012705  000604      TST15:  MOV     #STACK0,R5      ;SET UP THE STACK POINTER
1645  005646  004767  005726              JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1646
1647  005652  000240                      NOP
1648  005654  075005                      FADD+   R5              ;FLOATING ADD ON THE R5 STACK
1649
1650  005656  013767  177776  172730  1$:  MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1651  005664  010567  172726              MOV     R5,     $SP     ;SAVE THE STACK POINTER
1652  005670  026767  172734  172716      CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
1653  005676  001401                      BEQ     .+4             ;BRANCH IF OK
1654  005700  104000                      HLT                     ;PSW NOT EQUAL TO $ADDPS
1655
1656  005702  105767  172706              TSTB    $PSW            ;CHECK FOR ERROR
1657  005706  100423                      BMI     2$              ;BRANCH IF ERROR
1658
1659  005710  012767  000610  172752      MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1660  005716  026767  172746  172672      CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1661  005724  001401                      BEQ     .+4             ;BRANCH IF OK
1662  005726  104000                      HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
```

```
1663
1664  005730  026767  172676  172652          CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
1665  005736  001401                          BEQ     .+4             ;BRANCH IF OK
1666  005740  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
1667
1668  005742  026767  172666  172642          CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
1669  005750  001401                          BEQ     .+4             ;BRANCH IF OK
1670  005752  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
1671
1672  005754  000451                          BR      3$
1673
1674  005756  012767  000604  172704  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1675  005764  026767  172700  172624          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1676  005772  001401                          BEQ     .+4             ;BRANCH IF OK
1677  005774  104000                          HLT                     ;STACK POINTER FOULED UP
1678
1679  005776  022767  005656  172574          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1680  006004  001401                          BEQ     .+4             ;BRANCH IF OK
1681  006006  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1682
1683  006010  026767  172622  172564          CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
1684  006016  001401                          BEQ     .+4             ;BRANCH IF OK
1685  006020  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
1686
1687  006022  026767  172576  172554          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1688  006030  001401                          BEQ     .+4             ;BRANCH IF OK
1689  006032  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
1690
1691  006034  026767  172566  172544          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1692  006042  001401                          BEQ     .+4             ;BRANCH IF OK
1693  006044  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1694
1695  006046  026767  172546  172534          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1696  006054  001401                          BEQ     .+4             ;BRANCH IF OK
1697  006056  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1698
1699  006060  026767  172536  172524          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1700  006066  001401                          BEQ     .+4             ;BRANCH IF OK
1701  006070  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1702
1703  006072  012716  006100                  MOV     #3$,    (SP)    ;RESET THE STACK
1704  006076  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1705
1706  006100  104400                  3$:     SCOPE
1707
```

```
1708
1709                                    ;*************************************************************************
1710                                    ;TEST 16:       EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
1711                                    ;       RAND.A,RAND.B - RAND.C,RAND.D = ANS1,ANS2
1712                                    ;       STACK POINTER = SP
1713                                    ;*************************************************************************
1714
1715  006102  012706  000604           TST16:  MOV     #STACK0,SP      ;SET UP THE STACK POINTER
1716  006106  004767  005466                   JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1717
1718  006112  000240                           NOP
1719  006114  075016                           FSUB+   SP              ;FLOATING SUBTRACT ON THE SP STACK
1720
1721  006116  013767  177776  172470   1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1722  006124  010667  172466                   MOV     SP,     $SP     ;SAVE THE STACK POINTER
1723  006130  026767  172504  172456           CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
1724  006136  001401                           BEQ     .+4             ;BRANCH IF OK
1725  006140  104000                           HLT                     ;PSW NOT EQUAL TO $SUBPS
1726
1727  006142  105767  172446                   TSTB    $PSW            ;CHECK FOR ERROR
1728  006146  100424                           BMI     2$              ;BRANCH IF ERROR
1729
1730  006150  012767  000610  172512           MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1731  006156  026767  172506  172432           CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1732  006164  001401                           BEQ     .+4             ;BRANCH IF OK
1733  006166  104000                           HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
1734
1735  006170  026767  172446  172412           CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
1736  006176  001401                           BEQ     .+4             ;BRANCH IF OK
1737  006200  104000                           HLT                     ;LEFT HALF OF ANSWER WRONG
1738
1739  006202  026767  172436  172402           CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
1740  006210  001401                           BEQ     .+4             ;BRANCH IF OK
1741  006212  104000                           HLT                     ;RIGHT HALF OF ANSWER WRONG
1742
1743  006214  024646                           CMP     -(SP),  -(SP)   ;RESTORE THE STACK
1744  006216  000451                           BR      3$
1745
1746  006220  012767  000600  172442   2$:     MOV     #STK1,  SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1747  006226  026767  172436  172362           CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1748  006234  001401                           BEQ     .+4             ;BRANCH IF OK
1749  006236  104000                           HLT                     ;STACK POINTER FOULED UP
1750
1751  006240  022767  006116  172332           CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1752  006246  001401                           BEQ     .+4             ;BRANCH IF OK
1753  006250  104000                           HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1754
1755  006252  026767  172370  172322           CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
1756  006260  001401                           BEQ     .+4             ;BRANCH IF OK
1757  006262  104000                           HLT                     ;RTI PSW NOT EQUAL TO 200
1758
1759  006264  026767  172334  172312           CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1760  006272  001401                           BEQ     .+4             ;BRANCH IF OK
1761  006274  104000                           HLT                     ;STK3 NOT EQUAL TO RAND.C
1762
1763  006276  026767  172324  172302           CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
```

```
1764  006304  001401                          BEQ   .+4             ;BRANCH IF OK
1765  006306  104000                          HLT                   ;STK4 NOT EQUAL TO RAND.D
1766
1767  006310  026767  172304  172272         CMP   RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1768  006316  001401                          BEQ   .+4             ;BRANCH IF OK
1769  006320  104000                          HLT                   ;STK5 NOT EQUAL TO RAND.A
1770
1771  006322  026767  172274  172262         CMP   RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1772  006330  001401                          BEQ   .+4             ;BRANCH IF OK
1773  006332  104000                          HLT                   ;STK6 NOT EQUAL TO RAND.B
1774
1775  006334  012716  006342                 MOV   #3$,    (SP)    ;RESET THE STACK
1776  006340  000002                          RTI                   ;RESTORE THE STATUS (T-BIT)
1777
1778  006342  104400              3$:         SCOPE
1779
1780
1781                        ;*************************************************************************
1782                        ;TEST 17:       EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
1783                        ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
1784                        ;       STACK POINTER = R0
1785                        ;*************************************************************************
1786
1787  006344  012700  000604          TST17: MOV   #STACK0,R0      ;SET UP THE STACK POINTER
1788  006350  004767  005224                 JSR   PC,     PUSHR   ;PUT THE DATA ON THE STACK
1789
1790  006354  000240                          NOP
1791  006356  075020                          FMUL+ R0              ;FLOATING MULTIPLY ON THE R0 STACK
1792
1793  006360  013767  177776  172226  1$:     MOV   @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1794  006366  010067  172224                 MOV   R0,     $SP     ;SAVE THE STACK POINTER
1795  006372  026767  172252  172214         CMP   $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
1796  006400  001401                          BEQ   .+4             ;BRANCH IF OK
1797  006402  104000                          HLT                   ;PSW NOT EQUAL TO $MULPS
1798
1799  006404  105767  172204                 TSTB  $PSW            ;CHECK FOR ERROR
1800  006410  100423                          BMI   2$              ;BRANCH IF ERROR
1801
1802  006412  012767  000610  172250         MOV   #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1803  006420  026767  172244  172170         CMP   SAVSTK, $SP     ;CHECK THE STACK POINTER
1804  006426  001401                          BEQ   .+4             ;BRANCH IF OK
1805  006430  104000                          HLT                   ;STACK POINTER NOT EQUAL TO #STACK4
1806
1807  006432  026767  172214  172150         CMP   $MUL1,  ANS1    ;CHECK THE ANSWER
1808  006440  001401                          BEQ   .+4             ;BRANCH IF OK
1809  006442  104000                          HLT                   ;LEFT HALF OF ANSWER WRONG
1810
1811  006444  026767  172204  172140         CMP   $MUL2,  ANS2    ;CHECK THE ANSWER
1812  006452  001401                          BEQ   .+4             ;BRANCH IF OK
1813  006454  104000                          HLT                   ;RIGHT HALF OF ANSWER WRONG
1814
1815  006456  000451                          BR    3$
1816
1817  006460  012767  000604  172202  2$:     MOV   #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1818  006466  026767  172176  172122         CMP   SAVSTK, $SP     ;CHECK THE STACK POINTER
1819  006474  001401                          BEQ   .+4             ;BRANCH IF OK
```

```
1820  006476  104000                          HLT                     ;STACK POINTER FOULED UP
1821
1822  006500  022767  006360  172072          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1823  006506  001401                          BEQ     .+4             ;BRANCH IF OK
1824  006510  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1825
1826  006512  026767  172140  172062          CMP     $MULER, STK2    ;CHECK THE PSW ON THE STACK
1827  006520  001401                          BEQ     .+4             ;BRANCH IF OK
1828  006522  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
1829
1830  006524  026767  172074  172052          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1831  006532  001401                          BEQ     .+4             ;BRANCH IF OK
1832  006534  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
1833
1834  006536  026767  172064  172042          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1835  006544  001401                          BEQ     .+4             ;BRANCH IF OK
1836  006546  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1837
1838  006550  026767  172044  172032          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1839  006556  001401                          BEQ     .+4             ;BRANCH IF OK
1840  006560  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1841
1842  006562  026767  172034  172022          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1843  006570  001401                          BEQ     .+4             ;BRANCH IF OK
1844  006572  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1845
1846  006574  012716  006602                  MOV     #3$,    (SP)    ;RESET THE STACK
1847  006600  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1848
1849  006602  104400                  3$:     SCOPE
1850
1851
1852                                  ;*************************************************************************
1853                                  ;TEST 20:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
1854                                  ;       RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
1855                                  ;       STACK POINTER = R1
1856                                  ;*************************************************************************
1857
1858  006604  012701  000604          TST20:  MOV     #STACK0,R1      ;SET UP THE STACK POINTER
1859  006610  004767  004764                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
1860
1861  006614  000240                          NOP
1862  006616  075031                          FDIV+   R1              ;FLOATING DIVIDE ON THE R1 STACK
1863
1864  006620  013767  177776  171766  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
1865  006626  010167  171764                  MOV     R1,     $SP     ;SAVE THE STACK POINTER
1866  006632  026767  172022  171754          CMP     $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
1867  006640  001401                          BEQ     .+4             ;BRANCH IF OK
1868  006642  104000                          HLT                     ;PSW NOT EQUAL TO $DIVPS
1869
1870  006644  105767  171744                  TSTB    $PSW            ;CHECK FOR ERROR
1871  006650  100423                          BMI     2$              ;BRANCH IF ERROR
1872
1873  006652  012767  000610  172010          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
1874  006660  026767  172004  171730          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1875  006666  001401                          BEQ     .+4             ;BRANCH IF OK
```

```
1876  006670  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
1877
1878  006672  026767  171764  171710          CMP     $DIV1,  ANS1    ;CHECK THE ANSWER
1879  006700  001401                          BEQ     .+4             ;BRANCH IF OK
1880  006702  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
1881
1882  006704  026767  171754  171700          CMP     $DIV2,  ANS2    ;CHECK THE ANSWER
1883  006712  001401                          BEQ     .+4             ;BRANCH IF OK
1884  006714  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
1885
1886  006716  000451                          BR      3$
1887
1888  006720  012767  000604  171742  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
1889  006726  026767  171736  171662          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
1890  006734  001401                          BEQ     .+4             ;BRANCH IF OK
1891  006736  104000                          HLT                     ;STACK POINTER FOULED UP
1892
1893  006740  022767  006620  171632          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
1894  006746  001401                          BEQ     .+4             ;BRANCH IF OK
1895  006750  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
1896
1897  006752  026767  171710  171622          CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
1898  006760  001401                          BEQ     .+4             ;BRANCH IF OK
1899  006762  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
1900
1901  006764  026767  171634  171612          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
1902  006772  001401                          BEQ     .+4             ;BRANCH IF OK
1903  006774  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
1904
1905  006776  026767  171624  171602          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
1906  007004  001401                          BEQ     .+4             ;BRANCH IF OK
1907  007006  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1908
1909  007010  026767  171604  171572          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1910  007016  001401                          BEQ     .+4             ;BRANCH IF OK
1911  007020  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1912
1913  007022  026767  171574  171562          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1914  007030  001401                          BEQ     .+4             ;BRANCH IF OK
1915  007032  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1916
1917  007034  012716  007042                  MOV     #3$,    (SP)    ;RESET THE STACK
1918  007040  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1919
1920  007042  104400                  3$:     SCOPE
1921
```

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

007044  012702  000604          TST21:  MOV     #STACK0,R2      ;SET UP THE STACK POINTER
007050  004767  004524                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK

007054  000240                          NOP
007056  075002                          FADD+   R2              ;FLOATING ADD ON THE R2 STACK

007060  013767  177776  171526  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
007066  010267  171524                  MOV     R2,     $SP     ;SAVE THE STACK POINTER
007072  026767  171532  171514          CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
007100  001401                          BEQ     .+4             ;BRANCH IF OK
007102  104000                          HLT                     ;PSW NOT EQUAL TO $ADDPS

007104  105767  171504                  TSTB    $PSW            ;CHECK FOR ERROR
007110  100423                          BMI     2$              ;BRANCH IF ERROR

007112  012767  000610  171550          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
007120  026767  171544  171470          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
007126  001401                          BEQ     .+4             ;BRANCH IF OK
007130  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4

007132  026767  171474  171450          CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
007140  001401                          BEQ     .+4             ;BRANCH IF OK
007142  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG

007144  026767  171464  171440          CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
007152  001401                          BEQ     .+4             ;BRANCH IF OK
007154  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG

007156  000451                          BR      3$

007160  012767  000604  171502  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
007166  026767  171476  171422          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
007174  001401                          BEQ     .+4             ;BRANCH IF OK
007176  104000                          HLT                     ;STACK POINTER FOULED UP

007200  022767  007060  171372          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
007206  001401                          BEQ     .+4             ;BRANCH IF OK
007210  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$

007212  026767  171420  171362          CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
007220  001401                          BEQ     .+4             ;BRANCH IF OK
007222  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200

007224  026767  171374  171352          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
007232  001401                          BEQ     .+4             ;BRANCH IF OK
007234  104000                          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
```

```
1978  007246  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
1979
1980  007250  026767  171344  171332          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
1981  007256  001401                          BEQ     .+4             ;BRANCH IF OK
1982  007260  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
1983
1984  007262  026767  171334  171322          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
1985  007270  001401                          BEQ     .+4             ;BRANCH IF OK
1986  007272  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
1987
1988  007274  012716  007302                  MOV     #3$,    (SP)    ;RESET THE STACK
1989  007300  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
1990
1991  007302  104400                  3$:     SCOPE
1992
1993
1994                                  ;*************************************************************************
1995                                  ;TEST 22:       EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
1996                                  ;       RAND.A,RAND.B - RAND.C,RAND.D = ANS1,ANS2
1997                                  ;       STACK POINTER = R3
1998                                  ;*************************************************************************
1999
2000  007304  012703  000604          TST22:  MOV     #STACK0,R3      ;SET UP THE STACK POINTER
2001  007310  004767  004264                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
2002
2003  007314  000240                          NOP
2004  007316  075013                          FSUB+   R3              ;FLOATING SUBTRACT ON THE R3 STACK
2005
2006  007320  013767  177776  171266  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2007  007326  010367  171264                  MOV     R3,     $SP     ;SAVE THE STACK POINTER
2008  007332  026767  171302  171254          CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
2009  007340  001401                          BEQ     .+4             ;BRANCH IF OK
2010  007342  104000                          HLT                     ;PSW NOT EQUAL TO $SUBPS
2011
2012  007344  105767  171244                  TSTB    $PSW            ;CHECK FOR ERROR
2013  007350  100423                          BMI     2$              ;BRANCH IF ERROR
2014
2015  007352  012767  000610  171310          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2016  007360  026767  171304  171230          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2017  007366  001401                          BEQ     .+4             ;BRANCH IF OK
2018  007370  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
2019
2020  007372  026767  171244  171210          CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
2021  007400  001401                          BEQ     .+4             ;BRANCH IF OK
2022  007402  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
2023
2024  007404  026767  171234  171200          CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
2025  007412  001401                          BEQ     .+4             ;BRANCH IF OK
2026  007414  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
2027
2028  007416  000451                          BR      3$
2029
2030  007420  012767  000604  171242  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
2031  007426  026767  171236  171162          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2032  007434  001401                          BEQ     .+4             ;BRANCH IF OK
2033  007436  104000                          HLT                     ;STACK POINTER FOULED UP
```

```
2034
2035  007440  022767  007320  171132          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2036  007446  001401                          BEQ     .+4             ;BRANCH IF OK
2037  007450  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2038
2039  007452  026767  171170  171122          CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
2040  007460  001401                          BEQ     .+4             ;BRANCH IF OK
2041  007462  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
2042
2043  007464  026767  171134  171112          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2044  007472  001401                          BEQ     .+4             ;BRANCH IF OK
2045  007474  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
2046
2047  007476  026767  171124  171102          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2048  007504  001401                          BEQ     .+4             ;BRANCH IF OK
2049  007506  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2050
2051  007510  026767  171104  171072          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2052  007516  001401                          BEQ     .+4             ;BRANCH IF OK
2053  007520  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2054
2055  007522  026767  171074  171062          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2056  007530  001401                          BEQ     .+4             ;BRANCH IF OK
2057  007532  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2058
2059  007534  012716  007542                  MOV     #3$,    (SP)    ;RESET THE STACK
2060  007540  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2061
2062  007542  104400          3$:     .      SCOPE
2063
2064
2065                          ;*****************************************************************************
2066                          ;TEST 23:       EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
2067                          ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
2068                          ;       STACK POINTER = R4
2069                          ;*****************************************************************************
2070
2071  007544  012704  000604  TST23:  MOV     #STACK0,R4      ;SET UP THE STACK POINTER
2072  007550  004767  004024          JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
2073
2074  007554  000240                  NOP
2075  007556  075024                  FMUL+   R4              ;FLOATING MULTIPLY ON THE R4 STACK
2076
2077  007560  013767  177776  171026  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2078  007566  010467  171024                  MOV     R4,     $SP     ;SAVE THE STACK POINTER
2079  007572  026767  171052  171014          CMP     $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
2080  007600  001401                          BEQ     .+4             ;BRANCH IF OK
2081  007602  104000                          HLT                     ;PSW NOT EQUAL TO $MULPS
2082
2083  007604  105767  171004                  TSTB    $PSW            ;CHECK FOR ERROR
2084  007610  100423                          BMI     2$              ;BRANCH IF ERROR
2085
2086  007612  012767  000610  171050          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2087  007620  026767  171044  170770          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2088  007626  001401                          BEQ     .+4             ;BRANCH IF OK
2089  007630  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
```

```
2090
2091  007632  026767  171014  170750          CMP     $MUL1,  ANS1    ;CHECK THE ANSWER
2092  007640  001401                          BEQ     .+4             ;BRANCH IF OK
2093  007642  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
2094
2095  007644  026767  171004  170740          CMP     $MUL2,  ANS2    ;CHECK THE ANSWER
2096  007652  001401                          BEQ     .+4             ;BRANCH IF OK
2097  007654  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
2098
2099  007656  000451                          BR      3$
2100
2101  007660  012767  000604  171002  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
2102  007666  026767  170776  170722          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2103  007674  001401                          BEQ     .+4             ;BRANCH IF OK
2104  007676  104000                          HLT                     ;STACK POINTER FOULED UP
2105
2106  007700  022767  007560  170672          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2107  007706  001401                          BEQ     .+4             ;BRANCH IF OK
2108  007710  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2109
2110  007712  026767  170740  170662          CMP     $MULER, STK2    ;CHECK THE PSW ON THE STACK
2111  007720  001401                          BEQ     .+4             ;BRANCH IF OK
2112  007722  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
2113
2114  007724  026767  170674  170652          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2115  007732  001401                          BEQ     .+4             ;BRANCH IF OK
2116  007734  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
2117
2118  007736  026767  170664  170642          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2119  007744  001401                          BEQ     .+4             ;BRANCH IF OK
2120  007746  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2121
2122  007750  026767  170644  170632          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2123  007756  001401                          BEQ     .+4             ;BRANCH IF OK
2124  007760  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2125
2126  007762  026767  170634  170622          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2127  007770  001401                          BEQ     .+4             ;BRANCH IF OK
2128  007772  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2129
2130  007774  012716  010002                  MOV     #3$,    (SP)    ;RESET THE STACK
2131  010000  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2132
2133  010002  104400          3$:     SCOPE
2134
```

```
                                ;*************************************************************************
                                ;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
```

```
2191  010206  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2192
2193  010210  026767  170404  170372          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2194  010216  001401                          BEQ     .+4             ;BRANCH IF OK
2195  010220  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2196
2197  010222  026767  170374  170362          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2198  010230  001401                          BEQ     .+4             ;BRANCH IF OK
2199  010232  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2200
2201  010234  012716  010242                  MOV     #3$,    (SP)    ;RESET THE STACK
2202  010240  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2203
2204  010242  104400                  3$:     SCOPE
2205
2206
2207                                  ;*************************************************************************
2208                                  ;TEST 25:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
2209                                  ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
2210                                  ;       STACK POINTER = SP
2211                                  ;*************************************************************************
2212
2213  010244  012706  000604          TST25:  MOV     #STACK0,SP      ;SET UP THE STACK POINTER
2214  010250  004767  003324                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
2215
2216  010254  000240                          NOP
2217  010256  075006                          FADD+   SP              ;FLOATING ADD ON THE SP STACK
2218
2219  010260  013767  177776  170326  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2220  010266  010667  170324                  MOV     SP,     $SP     ;SAVE THE STACK POINTER
2221  010272  026767  170332  170314          CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
2222  010300  001401                          BEQ     .+4             ;BRANCH IF OK
2223  010302  104000                          HLT                     ;PSW NOT EQUAL TO $ADDPS
2224
2225  010304  105767  170304                  TSTB    $PSW            ;CHECK FOR ERROR
2226  010310  100424                          BMI     2$              ;BRANCH IF ERROR
2227
2228  010312  012767  000610  170350          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2229  010320  026767  170344  170270          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2230  010326  001401                          BEQ     .+4             ;BRANCH IF OK
2231  010330  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
2232
2233  010332  026767  170274  170250          CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
2234  010340  001401                          BEQ     .+4             ;BRANCH IF OK
2235  010342  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
2236
2237  010344  026767  170264  170240          CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
2238  010352  001401                          BEQ     .+4             ;BRANCH IF OK
2239  010354  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
2240
2241  010356  024646                          CMP     -(SP),  -(SP)   ;RESTORE THE STACK
2242  010360  000451                          BR      3$
2243
2244  010362  012767  000600  170300  2$:     MOV     #STK1,  SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2245  010370  026767  170274  170220          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2246  010376  001401                          BEQ     .+4             ;BRANCH IF OK
```

```
2247  010400  104000                          HLT                     ;STACK POINTER FOULED UP
2248
2249  010402  022767  010260  170170          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2250  010410  001401                          BEQ     .+4             ;BRANCH IF OK
2251  010412  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2252
2253  010414  026767  170216  170160          CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
2254  010422  001401                          BEQ     .+4             ;BRANCH IF OK
2255  010424  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
2256
2257  010426  026767  170172  170150          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2258  010434  001401                          BEQ     .+4             ;BRANCH IF OK
2259  010436  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
2260
2261  010440  026767  170162  170140          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2262  010446  001401                          BEQ     .+4             ;BRANCH IF OK
2263  010450  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2264
2265  010452  026767  170142  170130          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2266  010460  001401                          BEQ     .+4             ;BRANCH IF OK
2267  010462  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2268
2269  010464  026767  170132  170120          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2270  010472  001401                          BEQ     .+4             ;BRANCH IF OK
2271  010474  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2272
2273  010476  012716  010504                  MOV     #3$,    (SP)    ;RESET THE STACK
2274  010502  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2275
2276  010504  104400                  3$:     SCOPE
2277
2278
2279                                  ;*****************************************************************************
2280                                  ;TEST 26:       EXERCISE FSUB (PDP-11 FLOATING SUBTRACT INSTRUCTION)
2281                                  ;       RAND.A,RAND.B - RAND.C,RAND.D = ANS1,ANS2
2282                                  ;       STACK POINTER = R0
2283                                  ;*****************************************************************************
2284
2285  010506  012700  000604          TST26:  MOV     #STACK0,R0      ;SET UP THE STACK POINTER
2286  010512  004767  003062                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
2287
2288  010516  000240                          NOP
2289  010520  075010                          FSUB+   R0              ;FLOATING SUBTRACT ON THE R0 STACK
2290
2291  010522  013767  177776  170064  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2292  010530  010067  170062                  MOV     R0,     $SP     ;SAVE THE STACK POINTER
2293  010534  026767  170100  170052          CMP     $SUBPS, $PSW    ;CHECK THE PROCESSOR STATUS
2294  010542  001401                          BEQ     .+4             ;BRANCH IF OK
2295  010544  104000                          HLT                     ;PSW NOT EQUAL TO $SUBPS
2296
2297  010546  105767  170042                  TSTB    $PSW            ;CHECK FOR ERROR
2298  010552  100423                          BMI     2$              ;BRANCH IF ERROR
2299
2300  010554  012767  000610  170106          MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2301  010562  026767  170102  170026          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2302  010570  001401                          BEQ     .+4             ;BRANCH IF OK
```

```
2303  010572  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
2304
2305  010574  026767  170042  170006          CMP     $SUB1,  ANS1    ;CHECK THE ANSWER
2306  010602  001401                          BEQ     .+4             ;BRANCH IF OK
2307  010604  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
2308
2309  010606  026767  170032  167776          CMP     $SUB2,  ANS2    ;CHECK THE ANSWER
2310  010614  001401                          BEQ     .+4             ;BRANCH IF OK
2311  010616  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
2312
2313  010620  000451                          BR      3$
2314
2315  010622  012767  000604  170040  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
2316  010630  026767  170034  167760          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2317  010636  001401                          BEQ     .+4             ;BRANCH IF OK
2318  010640  104000                          HLT                     ;STACK POINTER FOULED UP
2319
2320  010642  022767  010522  167730          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2321  010650  001401                          BEQ     .+4             ;BRANCH IF OK
2322  010652  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2323
2324  010654  026767  167766  167720          CMP     $SUBER, STK2    ;CHECK THE PSW ON THE STACK
2325  010662  001401                          BEQ     .+4             ;BRANCH IF OK
2326  010664  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
2327
2328  010666  026767  167732  167710          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2329  010674  001401                          BEQ     .+4             ;BRANCH IF OK
2330  010676  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
2331
2332  010700  026767  167722  167700          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2333  010706  001401                          BEQ     .+4             ;BRANCH IF OK
2334  010710  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2335
2336  010712  026767  167702  167670          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2337  010720  001401                          BEQ     .+4             ;BRANCH IF OK
2338  010722  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2339
2340  010724  026767  167672  167660          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2341  010732  001401                          BEQ     .+4             ;BRANCH IF OK
2342  010734  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2343
2344  010736  012716  010744                  MOV     #3$,    (SP)    ;RESET THE STACK
2345  010742  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2346
2347  010744  104400                  3$:     SCOPE
2348
```

```
2349
2350                                    ;*****************************************************************************
2351                                    ;TEST 27:       EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
2352                                    ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
2353                                    ;       STACK POINTER = R1
2354                                    ;*****************************************************************************
2355
2356  010746  012701  000604            TST27:  MOV     #STACK0,R1      ;SET UP THE STACK POINTER
2357  010752  004767  002622                    JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
2358
2359  010756  000240                            NOP
2360  010760  075021                            FMUL+   R1              ;FLOATING MULTIPLY ON THE R1 STACK
2361
2362  010762  013767  177776  167624    1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2363  010770  010167  167622                    MOV     R1,     $SP     ;SAVE THE STACK POINTER
2364  010774  026767  167650  167612            CMP     $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
2365  011002  001401                            BEQ     .+4             ;BRANCH IF OK
2366  011004  104000                            HLT                     ;PSW NOT EQUAL TO $MULPS
2367
2368  011006  105767  167602                    TSTB    $PSW            ;CHECK FOR ERROR
2369  011012  100423                            BMI     2$              ;BRANCH IF ERROR
2370
2371  011014  012767  000610  167646            MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2372  011022  026767  167642  167566            CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2373  011030  001401                            BEQ     .+4             ;BRANCH IF OK
2374  011032  104000                            HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
2375
2376  011034  026767  167612  167546            CMP     $MUL1,  ANS1    ;CHECK THE ANSWER
2377  011042  001401                            BEQ     .+4             ;BRANCH IF OK
2378  011044  104000                            HLT                     ;LEFT HALF OF ANSWER WRONG
2379
2380  011046  026767  167602  167536            CMP     $MUL2,  ANS2    ;CHECK THE ANSWER
2381  011054  001401                            BEQ     .+4             ;BRANCH IF OK
2382  011056  104000                            HLT                     ;RIGHT HALF OF ANSWER WRONG
2383
2384  011060  000451                            BR      3$
2385
2386  011062  012767  000604  167600    2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
2387  011070  026767  167574  167520            CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2388  011076  001401                            BEQ     .+4             ;BRANCH IF OK
2389  011100  104000                            HLT                     ;STACK POINTER FOULED UP
2390
2391  011102  022767  010762  167470            CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2392  011110  001401                            BEQ     .+4             ;BRANCH IF OK
2393  011112  104000                            HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2394
2395  011114  026767  167536  167460            CMP     $MULER, STK2    ;CHECK THE PSW ON THE STACK
2396  011122  001401                            BEQ     .+4             ;BRANCH IF OK
2397  011124  104000                            HLT                     ;RTI PSW NOT EQUAL TO 200
2398
2399  011126  026767  167472  167450            CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2400  011134  001401                            BEQ     .+4             ;BRANCH IF OK
2401  011136  104000                            HLT                     ;STK3 NOT EQUAL TO RAND.C
2402
2403  011140  026767  167462  167440            CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2404  011146  001401                            BEQ     .+4             ;BRANCH IF OK
```

```
2405  011150  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2406
2407  011152  026767  167442  167430         CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2408  011160  001401                          BEQ     .+4             ;BRANCH IF OK
2409  011162  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2410
2411  011164  026767  167432  167420         CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2412  011172  001401                          BEQ     .+4             ;BRANCH IF OK
2413  011174  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2414
2415  011176  012716  011204                  MOV     #3$,    (SP)    ;RESET THE STACK
2416  011202  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2417
2418  011204  104400                  3$:     SCOPE
2419
2420
2421                                  ;*************************************************************************
2422                                  ;TEST 30:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
2423                                  ;       RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
2424                                  ;       STACK POINTER = R2
2425                                  ;*************************************************************************
2426
2427  011206  012702  000604          TST30:  MOV     #STACK0,R2      ;SET UP THE STACK POINTER
2428  011212  004767  002362                  JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
2429
2430  011216  000240                          NOP
2431  011220  075032                          FDIV+   R2              ;FLOATING DIVIDE ON THE R2 STACK
2432
2433  011222  013767  177776  167364  1$:     MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2434  011230  010267  167362                  MOV     R2,     $SP     ;SAVE THE STACK POINTER
2435  011234  026767  167420  167352         CMP     $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
2436  011242  001401                          BEQ     .+4             ;BRANCH IF OK
2437  011244  104000                          HLT                     ;PSW NOT EQUAL TO $DIVPS
2438
2439  011246  105767  167342                  TSTB    $PSW            ;CHECK FOR ERROR
2440  011252  100423                          BMI     2$              ;BRANCH IF ERROR
2441
2442  011254  012767  000610  167406         MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2443  011262  026767  167402  167326         CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2444  011270  001401                          BEQ     .+4             ;BRANCH IF OK
2445  011272  104000                          HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
2446
2447  011274  026767  167362  167306         CMP     $DIV1,  ANS1    ;CHECK THE ANSWER
2448  011302  001401                          BEQ     .+4             ;BRANCH IF OK
2449  011304  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
2450
2451  011306  026767  167352  167276         CMP     $DIV2,  ANS2    ;CHECK THE ANSWER
2452  011314  001401                          BEQ     .+4             ;BRANCH IF OK
2453  011316  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
2454
2455  011320  000451                          BR      3$
2456
2457  011322  012767  000604  167340  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
2458  011330  026767  167334  167260         CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2459  011336  001401                          BEQ     .+4             ;BRANCH IF OK
2460  011340  104000                          HLT                     ;STACK POINTER FOULED UP
```

```
2461
2462  011342  022767  011222  167230          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2463  011350  001401                          BEQ     .+4             ;BRANCH IF OK
2464  011352  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2465
2466  011354  026767  167306  167220          CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
2467  011362  001401                          BEQ     .+4             ;BRANCH IF OK
2468  011364  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
2469
2470  011366  026767  167232  167210          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2471  011374  001401                          BEQ     .+4             ;BRANCH IF OK
2472  011376  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
2473
2474  011400  026767  167222  167200          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2475  011406  001401                          BEQ     .+4             ;BRANCH IF OK
2476  011410  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2477
2478  011412  026767  167202  167170          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2479  011420  001401                          BEQ     .+4             ;BRANCH IF OK
2480  011422  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2481
2482  011424  026767  167172  167160          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2483  011432  001401                          BEQ     .+4             ;BRANCH IF OK
2484  011434  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2485
2486  011436  012716  011444                  MOV     #3$,    (SP)    ;RESET THE STACK
2487  011442  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2488
2489  011444  104400                3$:       SCOPE
2490
2491
2492                                ;*************************************************************************
2493                                ;TEST 31:       EXERCISE FADD (PDP-11 FLOATING ADD INSTRUCTION)
2494                                ;       RAND.A,RAND.B + RAND.C,RAND.D = ANS1,ANS2
2495                                ;       STACK POINTER = R3
2496                                ;*************************************************************************
2497
2498  011446  012703  000604        TST31:  MOV     #STACK0,R3      ;SET UP THE STACK POINTER
2499  011452  004767  002122                JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
2500
2501  011456  000240                        NOP
2502  011460  075003                        FADD+   R3              ;FLOATING ADD ON THE R3 STACK
2503
2504  011462  013767  177776  167124  1$:   MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2505  011470  010367  167122                MOV     R3,     $SP     ;SAVE THE STACK POINTER
2506  011474  026767  167130  167112        CMP     $ADDPS, $PSW    ;CHECK THE PROCESSOR STATUS
2507  011502  001401                        BEQ     .+4             ;BRANCH IF OK
2508  011504  104000                        HLT                     ;PSW NOT EQUAL TO $ADDPS
2509
2510  011506  105767  167102                TSTB    $PSW            ;CHECK FOR ERROR
2511  011512  100423                        BMI     2$              ;BRANCH IF ERROR
2512
2513  011514  012767  000610  167146        MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2514  011522  026767  167142  167066        CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2515  011530  001401                        BEQ     .+4             ;BRANCH IF OK
2516  011532  104000                        HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
```

```
2517
2518  011534  026767  167072  167046          CMP     $ADD1,  ANS1    ;CHECK THE ANSWER
2519  011542  001401                          BEQ     .+4             ;BRANCH IF OK
2520  011544  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
2521
2522  011546  026767  167062  167036          CMP     $ADD2,  ANS2    ;CHECK THE ANSWER
2523  011554  001401                          BEQ     .+4             ;BRANCH IF OK
2524  011556  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
2525
2526  011560  000451                          BR      3$
2527
2528  011562  012767  000604  167100  2$:     MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
2529  011570  026767  167074  167020          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2530  011576  001401                          BEQ     .+4             ;BRANCH IF OK
2531  011600  104000                          HLT                     ;STACK POINTER FOULED UP
2532
2533  011602  022767  011462  166770          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2534  011610  001401                          BEQ     :+4             ;BRANCH IF OK
2535  011612  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2536
2537  011614  026767  167016  166760          CMP     $ADDER, STK2    ;CHECK THE PSW ON THE STACK
2538  011622  001401                          BEQ     .+4             ;BRANCH IF OK
2539  011624  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
2540
2541  011626  026767  166772  166750          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2542  011634  001401                          BEQ     .+4             ;BRANCH IF OK
2543  011636  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
2544
2545  011640  026767  166762  166740          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2546  011646  001401                          BEQ     .+4             ;BRANCH IF OK
2547  011650  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2548
2549  011652  026767  166742  166730          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2550  011660  001401                          BEQ     .+4             ;BRANCH IF OK
2551  011662  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2552
2553  011664  026767  166732  166720          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2554  011672  001401                          BEQ     .+4             ;BRANCH IF OK
2555  011674  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2556
2557  011676  012716  011704                  MOV     #3$,    (SP)    ;RESET THE STACK
2558  011702  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2559
2560  011704  104400                  3$:     SCOPE
2561
```

```
                                ;*************************************************************************
                                ;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
```

```
2618  012110  104000                        HLT                     ;STK4 NOT EQUAL TO RAND.D
2619
2620  012112  026767  166502  166470        CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2621  012120  001401                        BEQ     .+4             ;BRANCH IF OK
2622  012122  104000                        HLT                     ;STK5 NOT EQUAL TO RAND.A
2623
2624  012124  026767  166472  166460        CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2625  012132  001401                        BEQ     .+4             ;BRANCH IF OK
2626  012134  104000                        HLT                     ;STK6 NOT EQUAL TO RAND.B
2627
2628  012136  012716  012144                MOV     #3$,    (SP)    ;RESET THE STACK
2629  012142  000002                        RTI                     ;RESTORE THE STATUS (T-BIT)
2630
2631  012144  104400              3$:       SCOPE
2632
2633
2634                                ;*************************************************************************
2635                                ;TEST 33:       EXERCISE FMUL (PDP-11 FLOATING MULTIPLY INSTRUCTION)
2636                                ;       RAND.A,RAND.B * RAND.C,RAND.D = ANS1,ANS2
2637                                ;       STACK POINTER = R5
2638                                ;*************************************************************************
2639
2640  012146  012705  000604        TST33:  MOV     #STACK0,R5      ;SET UP THE STACK POINTER
2641  012152  004767  001422                JSR     PC,     PUSHR   ;PUT THE DATA ON THE STACK
2642
2643  012156  000240                        NOP
2644  012160  075025                        FMUL+   R5              ;FLOATING MULTIPLY ON THE R5 STACK
2645
2646  012162  013767  177776  166424  1$:   MOV     @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2647  012170  010567  166422                MOV     R5,     $SP     ;SAVE THE STACK POINTER
2648  012174  026767  166450  166412        CMP     $MULPS, $PSW    ;CHECK THE PROCESSOR STATUS
2649  012202  001401                        BEQ     .+4             ;BRANCH IF OK
2650  012204  104000                        HLT                     ;PSW NOT EQUAL TO $MULPS
2651
2652  012206  105767  166402                TSTB    $PSW            ;CHECK FOR ERROR
2653  012212  100423                        BMI     2$              ;BRANCH IF ERROR
2654
2655  012214  012767  000610  166446        MOV     #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2656  012222  026767  166442  166366        CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2657  012230  001401                        BEQ     .+4             ;BRANCH IF OK
2658  012232  104000                        HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
2659
2660  012234  026767  166412  166346        CMP     $MUL1,  ANS1    ;CHECK THE ANSWER
2661  012242  001401                        BEQ     .+4             ;BRANCH IF OK
2662  012244  104000                        HLT                     ;LEFT HALF OF ANSWER WRONG
2663
2664  012246  026767  166402  166336        CMP     $MUL2,  ANS2    ;CHECK THE ANSWER
2665  012254  001401                        BEQ     .+4             ;BRANCH IF OK
2666  012256  104000                        HLT                     ;RIGHT HALF OF ANSWER WRONG
2667
2668  012260  000451                        BR      3$
2669
2670  012262  012767  000604  166400  2$:   MOV     #STACK0,SAVSTK  ;SAVE STACK ADDRESS FOR TYPING
2671  012270  026767  166374  166320        CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2672  012276  001401                        BEQ     .+4             ;BRANCH IF OK
2673  012300  104000                        HLT                     ;STACK POINTER FOULED UP
```

```
2674
2675  012302  022767  012162  166270          CMP   #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2676  012310  001401                          BEQ   .+4               ;BRANCH IF OK
2677  012312  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2678
2679  012314  026767  166336  166260          CMP   $MULER, STK2    ;CHECK THE PSW ON THE STACK
2680  012322  001401                          BEQ   .+4               ;BRANCH IF OK
2681  012324  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
2682
2683  012326  026767  166272  166250          CMP   RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2684  012334  001401                          BEQ   .+4               ;BRANCH IF OK
2685  012336  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
2686
2687  012340  026767  166262  166240          CMP   RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2688  012346  001401                          BEQ   .+4               ;BRANCH IF OK
2689  012350  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2690
2691  012352  026767  166242  166230          CMP   RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2692  012360  001401                          BEQ   .+4               ;BRANCH IF OK
2693  012362  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2694
2695  012364  026767  166232  166220          CMP   RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2696  012372  001401                          BEQ   .+4               ;BRANCH IF OK
2697  012374  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2698
2699  012376  012716  012404                  MOV   #3$,    (SP)      ;RESET THE STACK
2700  012402  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2701
2702  012404  104400                3$:       SCOPE
2703
2704
2705                                ;*****************************************************************************
2706                                ;TEST 34:       EXERCISE FDIV (PDP-11 FLOATING DIVIDE INSTRUCTION)
2707                                ;       RAND.A,RAND.B / RAND.C,RAND.D = ANS1,ANS2
2708                                ;       STACK POINTER = SP
2709                                ;*****************************************************************************
2710
2711  012406  012706  000604        TST34:  MOV   #STACK0,SP      ;SET UP THE STACK POINTER
2712  012412  004767  001162                JSR   PC,     PUSHR   ;PUT THE DATA ON THE STACK
2713
2714  012416  000240                        NOP
2715  012420  075036                        FDIV+ SP              ;FLOATING DIVIDE ON THE SP STACK
2716
2717  012422  013767  177776  166164  1$:   MOV   @#PS,   $PSW    ;SAVE PROCESSOR STATUS
2718  012430  010667  166162                MOV   SP,     $SP     ;SAVE THE STACK POINTER
2719  012434  026767  166220  166152        CMP   $DIVPS, $PSW    ;CHECK THE PROCESSOR STATUS
2720  012442  001401                        BEQ   .+4               ;BRANCH IF OK
2721  012444  104000                        HLT                     ;PSW NOT EQUAL TO $DIVPS
2722
2723  012446  105767  166142                TSTB  $PSW            ;CHECK FOR ERROR
2724  012452  100424                        BMI   2$                ;BRANCH IF ERROR
2725
2726  012454  012767  000610  166206        MOV   #STACK4,SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2727  012462  026767  166202  166126        CMP   SAVSTK, $SP     ;CHECK THE STACK POINTER
2728  012470  001401                        BEQ   .+4               ;BRANCH IF OK
2729  012472  104000                        HLT                     ;STACK POINTER NOT EQUAL TO #STACK4
```

```
2730
2731  012474  026767  166162  166106          CMP     $DIV1,  ANS1    ;CHECK THE ANSWER
2732  012502  001401                          BEQ     .+4             ;BRANCH IF OK
2733  012504  104000                          HLT                     ;LEFT HALF OF ANSWER WRONG
2734
2735  012506  026767  166152  166076          CMP     $DIV2,  ANS2    ;CHECK THE ANSWER
2736  012514  001401                          BEQ     .+4             ;BRANCH IF OK
2737  012516  104000                          HLT                     ;RIGHT HALF OF ANSWER WRONG
2738
2739  012520  024646                          CMP     -(SP),  -(SP)   ;RESTORE THE STACK
2740  012522  000451                          BR      3$
2741
2742  012524  012767  000600  166136  2$:     MOV     #STK1,  SAVSTK  ;SAVE PROPER STACK ADDRESS FOR TYPING
2743  012532  026767  166132  166056          CMP     SAVSTK, $SP     ;CHECK THE STACK POINTER
2744  012540  001401                          BEQ     .+4             ;BRANCH IF OK
2745  012542  104000                          HLT                     ;STACK POINTER FOULED UP
2746
2747  012544  022767  012422  166026          CMP     #1$,    STK1    ;CHECK THE RTI ADDRESS ON THE STACK
2748  012552  001401                          BEQ     .+4             ;BRANCH IF OK
2749  012554  104000                          HLT                     ;RTI ADDRESS NOT EQUAL TO #1$
2750
2751  012556  026767  166104  166016          CMP     $DIVER, STK2    ;CHECK THE PSW ON THE STACK
2752  012564  001401                          BEQ     .+4             ;BRANCH IF OK
2753  012566  104000                          HLT                     ;RTI PSW NOT EQUAL TO 200
2754
2755  012570  026767  166030  166006          CMP     RAND.C, STK3    ;CHECK THE DATA ON THE STACK
2756  012576  001401                          BEQ     .+4             ;BRANCH IF OK
2757  012600  104000                          HLT                     ;STK3 NOT EQUAL TO RAND.C
2758
2759  012602  026767  166020  165776          CMP     RAND.D, STK4    ;CHECK THE DATA ON THE STACK
2760  012610  001401                          BEQ     .+4             ;BRANCH IF OK
2761  012612  104000                          HLT                     ;STK4 NOT EQUAL TO RAND.D
2762
2763  012614  026767  166000  165766          CMP     RAND.A, STK5    ;CHECK THE DATA ON THE STACK
2764  012622  001401                          BEQ     .+4             ;BRANCH IF OK
2765  012624  104000                          HLT                     ;STK5 NOT EQUAL TO RAND.A
2766
2767  012626  026767  165770  165756          CMP     RAND.B, STK6    ;CHECK THE DATA ON THE STACK
2768  012634  001401                          BEQ     .+4             ;BRANCH IF OK
2769  012636  104000                          HLT                     ;STK6 NOT EQUAL TO RAND.B
2770
2771  012640  012716  012646                  MOV     #3$,    (SP)    ;RESET THE STACK
2772  012644  000002                          RTI                     ;RESTORE THE STATUS (T-BIT)
2773
2774  012646  104400                  3$:     SCOPE
2775
```

```
2776
2777  012650  062767  000001  166130          ADD     #1,     PCNT+2  ;COUNT PASSES
2778  012656  005567  166122                  ADC     PCNT
2779
2780  012662      001             DONE:
2781  012662  032737  002000  177570          BIT     #SW10,@#SWR     ;RING THE BELL?
2782  012670  001002                          BNE     1$              ;NO!
2783  012672  000004  000007                  TYPE    ,BELL
2784  012676  005046                  1$:     CLR     -(6)            ;CLEAR TRACE TRAP
2785  012700  032737  010000  177570          BIT     #SW12,@#SWR     ;RUN WITH TRT?
2786  012706  001010                          BNE     2$
2787  012710  005167  000044                  COM     .TBIT
2788  012714  100005                          BPL     2$
2789  012716  052716  000020                  BIS     #20,(6)         ;SET TRACE TRAP
2790  012722  012746  012754                  MOV     #3$,-(6)        ;JUMP TO START OF TEST
2791  012726  000002                          RTI
2792  012730  012746  012736          2$:     MOV     #4$,-(6)        ;JUMP TO START OF TEST
2793  012734  000002                          RTI                     ;RETURN
2794  012736  013700  000042          4$:     MOV     @#42,R0         ;GET MONITOR ADDRESS
2795  012742  001404                          BEQ     3$              ;IF NONE
2796  012744  004710                          JSR     7,(0)           ;GO TO MONITOR
2797  012746  000240                          NOP
2798  012750  000240                          NOP
2799  012752  000240                          NOP
2800  012754  000137  001146          3$:     JMP     @#START         ;RETURN
2801
2802  012760  000000          .TBIT:  0
2803
2804
2805                          ;SUBROUTINE TO READ TTY INPUT AND SAVE OCTAL NUMBER
2806
2807  012762  004767  002124  READIN: JSR     PC,READ$
2808  012766  012702  015212          MOV     #INPUT,R2
2809  012772  012501                  MOV     (R5)+,R1
2810  012774  005011                  CLR     (R1)
2811  012776  112203          1$:     MOVB    (R2)+,R3        ;STORE DATA
2812  013000  001420                  BEQ     4$              ;BRANCH IF DONE
2813  013002  162703  000060          SUB     #60,R3
2814  013006  000241                  CLC
2815  013010  032703  177770          BIT     #177770,R3
2816  013014  001010                  BNE     2$
2817  013016  006311                  ASL     (R1)
2818  013020  103407                  BCS     3$
2819  013022  006311                  ASL     (R1)
2820  013024  103405                  BCS     3$
2821  013026  006311                  ASL     (R1)
2822  013030  103403                  BCS     3$
2823  013032  050311                  BIS     R3,(R1)
2824  013034  000760                  BR      1$
2825  013036  000261          2$:     SEC                     ;SET C-BIT IF NOT
2826  013040  000244          3$:     CLZ
2827  013042  000205          4$:     RTS     R5
```

```
2828
2829  013044  016746  165552          $PUSH:  MOV     RAND.B,-(SP)
2830  013050  016746  165544                  MOV     RAND.A,-(SP)
2831  013054  016746  165546                  MOV     RAND.D,-(SP)
2832  013060  016746  165540                  MOV     RAND.C,-(SP)
2833  013064  000134                  $POLSH: JMP     @(R4)+
2834
2835  013066  005767  165536          $POPAD: TST     $ADDPS          ;CHECK FOR ERROR
2836  013072  001145                          BNE     $SKIP           ;BRANCH IF PS SET
2837  013074  032716  077600                  BIT     #77600, (SP)    ;CHECK FOR  ZERO
2838  013100  001010                          BNE     1$              ;BRANCH IF NOT
2839  013102  013767  177776  165520          MOV     @#PS,   $ADDPS  ;Z-BIT IN PSW
2840  013110  005067  165516                  CLR     $ADD1           ;ZERO ANSWER
2841  013114  005067  165514                  CLR     $ADD2
2842  013120  000532                          BR      $SKIP
2843
2844  013122  005716                  1$:     TST     (SP)            ;GET N-BIT, CLEAR C-BIT, V-BIT
2845  013124  013767  177776  165476          MOV     @#PS,   $ADDPS  ;SET THE PSW SAVE
2846  013132  012667  165474                  MOV     (SP)+,  $ADD1
2847  013136  012667  165472                  MOV     (SP)+,  $ADD2
2848  013142  000134                          JMP     @(R4)+
2849
2850  013144  005767  165470          $POPSB: TST     $SUBPS          ;CHECK FOR ERROR
2851  013150  001116                          BNE     $SKIP           ;BRANCH IF PS SET
2852  013152  032716  077600                  BIT     #77600, (SP)    ;CHECK FOR  ZERO
2853  013156  001010                          BNE     1$              ;BRANCH IF NOT
2854  013160  013767  177776  165452          MOV     @#PS,   $SUBPS  ;Z-BIT IN PSW
2855  013166  005067  165450                  CLR     $SUB1           ;ZERO ANSWER
2856  013172  005067  165446                  CLR     $SUB2
2857  013176  000503                          BR      $SKIP
2858
2859  013200  005716                  1$:     TST     (SP)            ;GET N-BIT, CLEAR C-BIT, V-BIT
2860  013202  013767  177776  165430          MOV     @#PS,   $SUBPS  ;SET THE PSW SAVE
2861  013210  012667  165426                  MOV     (SP)+,  $SUB1
2862  013214  012667  165424                  MOV     (SP)+,  $SUB2
2863  013220  000134                          JMP     @(R4)+
2864
2865  013222  005767  165422          $POPML: TST     $MULPS          ;CHECK FOR ERROR
2866  013226  001067                          BNE     $SKIP           ;BRANCH IF PS SET
2867  013230  032716  077600                  BIT     #77600, (SP)    ;CHECK FOR  ZERO
2868  013234  001010                          BNE     1$              ;BRANCH IF NOT
2869  013236  013767  177776  165404          MOV     @#PS,   $MULPS  ;Z-BIT IN PSW
2870  013244  005067  165402                  CLR     $MUL1           ;ZERO ANSWER
2871  013250  005067  165400                  CLR     $MUL2
2872  013254  000454                          BR      $SKIP
2873
2874  013256  005716                  1$:     TST     (SP)            ;GET N-BIT, CLEAR C-BIT, V-BIT
2875  013260  013767  177776  165362          MOV     @#PS,   $MULPS  ;SET THE PSW SAVE
2876  013266  012667  165360                  MOV     (SP)+,  $MUL1
2877  013272  012667  165356                  MOV     (SP)+,  $MUL2
2878  013276  000134                          JMP     @(R4)+
```

```
2879
2880  013300  032767  077600  165316  $POPDV:  BIT   #77600,RAND.C  ;CHECK FOR DIVIDED BY ZERO
2881  013306  001010                   BNE     1$
2882  013310  000277                   SCC                         ;SET ALL CONDITION CODES
2883  013312  000244                   CLZ                         ;CLEAR THE Z-BIT
2884  013314  013767  177776  165344           MOV     @#PS,   $DIVER  ;SET UP DIVIDE BY ZERO CC'S
2885  013322  012767  000340  165330           MOV     #340,   $DIVPS  ;SET UP PSW
2886  013330  005767  165324          1$:      TST     $DIVPS          ;CHECK FOR ERROR
2887  013334  001024                   BNE     $SKIP                   ;BRANCH IF PS SET
2888  013336  032716  077600                   BIT     #77600, (SP)    ;CHECK FOR  ZERO
2889  013342  001010                   BNE     2$                      ;BRANCH IF NOT
2890  013344  013767  177776  165306           MOV     @#PS,   $DIVPS  ;Z-BIT IN PSW
2891  013352  005067  165304                   CLR     $DIV1           ;ZERO ANSWER
2892  013356  005067  165302                   CLR     $DIV2
2893  013362  000411                   BR      $SKIP
2894
2895  013364  005716          2$:      TST     (SP)                    ;GET N-BIT, CLEAR C-BIT, V-BIT
2896  013366  013767  177776  165264           MOV     @#PS,   $DIVPS  ;SET THE PSW SAVE
2897  013374  012667  165262                   MOV     (SP)+,  $DIV1
2898  013400  012667  165260                   MOV     (SP)+,  $DIV2
2899  013404  000134                   JMP     @(R4)+
2900
2901  013406  022626          $SKIP:   CMP     (SP)+,  (SP)+   ;POP GARBAGE OFF THE STACK
2902  013410  000134                   JMP     @(R4)+
2903
2904  013412  000204          $EXIT:   RTS     R4              ;EXIT POLISH MODE
2905
2906  013414  016500  000002  $ERR:    MOV     2(5),   R0      ;PUT CODE INTO R0
2907  013420  022700  004003  $ERRA:   CMP     #4003,  R0      ;CHECK FOR DIVIDE BY ZERO
2908  013424  001464                   BEQ     8$              ;SKIP OUT
2909
2910  013426  122700  000003                   CMPB    #3,     R0      ;CHECK FOR OVERFLOW
2911  013432  001006                   BNE     2$              ;BRANCH IF NOT
2912  013434  000257                   CCC                     ;CLEAR ALL CONDITION CODES
2913  013436  000262                   SEV                     ;SET THE V-BIT
2914  013440  013767  177776  165146           MOV     @#PS,   $PSW    ;SET UP PSW FOR OVERFLOW
2915  013446  000405                   BR      3$
2916
2917  013450  000257          2$:      CCC                     ;CLEAR ALL CONDITION CODES
2918  013452  000272                   SNV                     ;SET N-BIT AND V-BIT
2919  013454  013767  177776  165132           MOV     @#PS,   $PSW    ;SET UP PSW FOR UNDERFLOW
2920  013462  105000          3$:      CLRB    R0              ;CLEAR LOW BYTE
2921  013464  000300                   SWAB    R0              ;HIGH BYTE INTO LOW
2922  013466  162700  000002                   SUB     #2,     R0      ;CHECK FOR ADD/SUB
2923  013472  001021                   BNE     5$              ;BRANCH IF NOT
2924  013474  005767  165130                   TST     $ADDPS          ;CHECK FOR ADD
2925  013500  001007                   BNE     4$              ;BRANCH IF NOT
2926  013502  016767  165106  165126           MOV     $PSW,   $ADDER  ;SET UP ADD ERROR PSW
2927  013510  012767  000340  165112           MOV     #340,   $ADDPS  ;SET UP ADD PSW
2928  013516  000427                   BR      8$
2929
2930  013520  016767  165070  165120  4$:      MOV     $PSW,   $SUBER  ;SET UP SUBTRACT ERROR PSW
2931  013526  012767  000340  165104           MOV     #340,   $SUBPS  ;SET UP SUBTRACT PSW
2932  013534  000420                   BR      8$
2933
2934  013536  162700  000004          5$:      SUB     #4,     R0      ;CHECK FOR MUL
```

```
2935  013542  003407                          BLE     6$                      ;BRANCH IF NOT
2936  013544  016767  165044  165104          MOV     $PSW,   $MULER          ;SET UP MULTIPLY ERROR PSW
2937  013552  012767  000340  165070          MOV     #340,   $MULPS          ;SET UP MULTIPLY PSW
2938  013560  000406                          BR      8$
2939
2940  013562  016767  165026  165076  6$:     MOV     $PSW,   $DIVER          ;SET UP DIVIDE ERROR PSW
2941  013570  012767  000340  165062  7$:     MOV     #340,   $DIVPS          ;SET UP DIVIDE PSW
2942  013576  000205                  8$:     RTS     R5                      ;RETURN TO FORTRAN
2943
2944                                  ;SUBROUTINE TO PUSH DATA ONTO STACK
2945
2946  013600  016767  165016  165004  PUSHR:  MOV     RAND.B, STACK6  ;PUT DATA ON THE STACK
2947  013606  016767  165006  164774          MOV     RAND.A, STACK4
2948  013614  016767  165006  164764          MOV     RAND.D, STACK2
2949  013622  016767  164776  164754          MOV     RAND.C, STACK0
2950  013630  011637  000244                  MOV     (SP),   @#244           ;SET UP TRAP VECTOR
2951  013634  062737  000004  000244          ADD     #4,     @#244
2952  013642  000207                          RTS     PC
2953
2954  013644  032737  000400  177570  SCOPE$: BIT     #SW08,@#SWR     ;KILL LDUB OR LOOP ON SPEC. TEST
2955  013652  001412                          BEQ     1$
2956  013654  013767  177570  000134          MOV     @#SWR,  SCOTMP  ;SAVE SWR
2957  013662  042767  177600  000126          BIC     #177600,SCOTMP  ;CLR ALL BUT TEST NO.
2958  013670  126767  000122  165102          CMPB    SCOTMP,ICNT     ;ON RIGHT TEST?   *SW6-0*
2959  013676  001434                          BEQ     OVER$
2960  013700  032737  040000  177570  1$:     BIT     #SW14,@#SWR     ;LOOP ON TEST
2961  013706  001026                          BNE     KIT$
2962  013710  032737  004000  177570          BIT     #SW11,@#SWR     ;KILL ITERATIONS
2963  013716  001012                          BNE     SVLAD$
2964  013720  105767  165055                  TSTB    ICNT+1
2965  013724  001404                          BEQ     2$              ;BRANCH IF FIRST
2966  013726  126767  000062  165045          CMPB    TIMES,ICNT+1    ;DONE?
2967  013734  001013                          BNE     KIT$            ;BRANCH IF NOT
2968  013736  112767  000001  165035  2$:     MOVB    #1,ICNT+1       ;FIRST ITERATION
2969  013744  105267  165030          SVLAD$: INCB    ICNT            ;COUNT TEST NUMBERS
2970  013750  011667  000036                  MOV     (6),LAD$        ;SAVE LOOP ADDRESS
2971  013754  016737  165020  177570          MOV     ICNT,@#DISPLAY  ;DISPLAY TEST NO. AND ITERATION COUNT
2972  013762  000002                          RTI                     ;RETURN
2973
2974  013764  105267  165011          KIT$:   INCB    ICNT+1
2975  013770  016737  165004  177570  OVER$:  MOV     ICNT,@#DISPLAY  ;SET UP DISPLAY
2976  013776  005767  000010                  TST     LAD$            ;FIRST ONE?
2977  014002  001760                          BEQ     SVLAD$
2978  014004  016716  000002                  MOV     LAD$,(6)        ;FUDGE RETURN ADDRESS
2979  014010  000002                          RTI                     ;FIXES PS
2980
2981  014012  000000          LAD$:   0               ;LOOP ADDRESS
2982  014014  000377          TIMES:  377             ;RUN 377 TIMES
2983  014016  000000          SCOTMP: 0
```

```
 2984  014020  032737  002000  177570  HLT$:   BIT     #SW10,@#SWR     ;BELL ON ERROR?
 2985  014026  001402                          BEQ     1$              ;NO - SKIP
 2986  014030  000004  000007                  TYPE    ,BELL           ;RING BELL
 2987  014034  005267  164742          1$:     INC     ERRORS          ;COUNT THE NUMBER OF ERRORS
 2988  014040  032737  020000  177570          BIT     #SW13,@#SWR     ;SKIP TYPEOUT IF SET
 2989  014046  001017                          BNE     2$              ;SKIP TYPEOUTS
 2990  014050  000004  015362                  TYPE    ,RETURN
 2991  014054  011667  000060                  MOV     (6),HLTAD$      ;PUT ADDRESS OF INSTRUCTION ON STACK
 2992  014060  162767  000002  000052          SUB     #2,HLTAD$
 2993  014066  016705  000046                  MOV     HLTAD$,TTY      ;TYPE HLTAD$ IN OCTAL
 2994  014072  004767  001300                  JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
 2995  014076  000004  015370                  TYPE    ,SPACE+3
 2996  014102  004767  000034                  JSR     PC,     ERROR$  ;GO TO USER ERROR ROUTINE
 2997  014106  005737  177570          2$:     TST     @#SWR           ;HALT ON ERROR
 2998  014112  100001                          BPL     .+4             ;SKIP IF CONTINUE
 2999  014114  000000                          HALT                    ;HALT ON ERROR!
 3000  014116  032737  001000  177570          BIT     #SW09,@#SWR     ;CHECK FOR INHIBIT LOOP ON ERROR
 3001  014124  001001                          BNE     .+4             ;SKIP IF LOOP ON ERROR
 3002  014126  000002                          RTI
 3003  014130  105067  164645                  CLRB    ICNT+1
 3004  014134  000167  177624                  JMP     KIT$            ;LOOP ON TEST UNTIL NO ERRORS
 3005
 3006  014140  000000          HLTAD$: 0
 3007
 3008  014142  010046          ERROR$: MOV     R0,     -(SP)   ;SAVE R0
 3009  014144  010146                  MOV     R1,     -(SP)   ;SAVE R1
 3010  014146  000004  015370          TYPE,   SPACE+3
 3011  014152  016705  164442          MOV     RAND.A,TTY      ;TYPE RAND.A IN OCTAL
 3012  014156  004767  001214          JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
 3013  014162  000004  014702          TYPE,   COMMA
 3014  014166  016705  164430          MOV     RAND.B,TTY      ;TYPE RAND.B IN OCTAL
 3015  014172  004767  001200          JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
 3016  014176  013700  000244          MOV     @#244,  R0      ;GET PC+2 OF INSTRUCTION
 3017  014202  014001                  MOV     -(R0),  R1      ;GET THE INSTRUCTION
 3018  014204  042701  177747          BIC     #177747,R1      ;MASK ALL BUT TYPE (+,-,*,/)
 3019  014210  006201                  ASR     R1              ;DIV BY 2
 3020  014212  012767  014662  000006  MOV     #SIGN$, 1$      ;SET TO TOP OF SIGN TABLE
 3021  014220  060167  000002          ADD     R1,     1$      ;ADD OFFSET
 3022  014224  000004                  TYPE
 3023  014226  014662          1$:     SIGN$                   ;TYPE THE RIGHT SIGN
 3024  014230  016705  164370          MOV     RAND.C,TTY      ;TYPE RAND.C IN OCTAL
 3025  014234  004767  001136          JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
 3026  014240  000004  014702          TYPE,   COMMA
 3027  014244  016705  164356          MOV     RAND.D,TTY      ;TYPE RAND.D IN OCTAL
 3028  014250  004767  001122          JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
 3029  014254  006301                  ASL     R1              ;RESET TABLE POINTER
 3030  014256  062701  000630          ADD     #$ADDPS,R1
 3031  014262  105767  164326          TSTB    $PSW            ;CHECK FOR ERROR CONDITIONS
 3032  014266  100460                  BMI     3$              ;BRANCH IF ERROR
 3033  014270  000004  014704          TYPE,   HEAD1
 3034  014274  000004  015062          TYPE,   EXPECT
 3035  014300  012105                  MOV     (R1)+,TTY       ;TYPE (R1)+ IN OCTAL
 3036  014302  004767  001070          JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
 3037  014306  000004  015370          TYPE,   SPACE+3
 3038  014312  012705  000610          MOV     #STACK4,TTY     ;TYPE #STACK4 IN OCTAL
 3039  014316  004767  001054          JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
```

. . .

```
3040  014322  000004  015370                TYPE,   SPACE+3
3041  014326  012105                        MOV     (R1)+,TTY       ;TYPE (R1)+ IN OCTAL
3042  014330  004767  001042                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3043  014334  000004  014702                TYPE,   COMMA
3044  014340  011105                        MOV     (R1),TTY        ;TYPE (R1) IN OCTAL
3045  014342  004767  001030                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3046  014346  000004  015076                TYPE,   GOT
3047  014352  016705  164236                MOV     $PSW,TTY        ;TYPE $PSW IN OCTAL
3048  014356  004767  001014                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3049  014362  000004  015370                TYPE,   SPACE+3
3050  014366  016705  164224                MOV     $SP,TTY         ;TYPE $SP IN OCTAL
3051  014372  004767  001000                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3052  014376  000004  015370                TYPE,   SPACE+3
3053  014402  016705  164202                MOV     ANS1,TTY        ;TYPE ANS1 IN OCTAL
3054  014406  004767  000764                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3055  014412  000004  014702                TYPE,   COMMA
3056  014416  016705  164170                MOV     ANS2,TTY        ;TYPE ANS2 IN OCTAL
3057  014422  004767  000750                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3058  014426  000510                        BR      7$
3059
3060  014430  000004  014751        3$:     TYPE,   HEAD2
3061  014434  000004  015062                TYPE,   EXPECT
3062  014440  012105                        MOV     (R1)+,TTY       ;TYPE (R1)+ IN OCTAL
3063  014442  004767  000730                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3064  014446  000004  015370                TYPE,   SPACE+3
3065  014452  016705  164212                MOV     SAVSTK,TTY      ;TYPE SAVSTK IN OCTAL
3066  014456  004767  000714                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3067  014462  000004  015370                TYPE,   SPACE+3
3068  014466  005720                        TST     (R0)+           ;UPDATE R0 TO RIGHT ADDRESS
3069  014470  010005                        MOV     R0,TTY          ;TYPE R0 IN OCTAL
3070  014472  004767  000700                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3071  014476  000004  015370                TYPE,   SPACE+3
3072  014502  022121                        CMP     (R1)+,  (R1)+   ;ADD 4 TO R1
3073  014504  011105                        MOV     (R1),TTY        ;TYPE (R1) IN OCTAL
3074  014506  004767  000664                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3075  014512  000004  015370                TYPE,   SPACE+3
3076  014516  016705  164102                MOV     RAND.C,TTY      ;TYPE RAND.C IN OCTAL
3077  014522  004767  000650                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3078  014526  000004  015370                TYPE,   SPACE+3
3079  014532  016705  164070                MOV     RAND.D,TTY      ;TYPE RAND.D IN OCTAL
3080  014536  004767  000634                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3081  014542  000004  015370                TYPE,   SPACE+3
3082  014546  016705  164046                MOV     RAND.A,TTY      ;TYPE RAND.A IN OCTAL
3083  014552  004767  000620                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3084  014556  000004  015370                TYPE,   SPACE+3
3085  014562  016705  164034                MOV     RAND.B,TTY      ;TYPE RAND.B IN OCTAL
3086  014566  004767  000604                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3087  014572  000004  015076                TYPE,   GOT
3088  014576  016705  164012                MOV     $PSW,TTY        ;TYPE $PSW IN OCTAL
3089  014602  004767  000570                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3090  014606  000004  015370                TYPE,   SPACE+3
3091  014612  016705  164000                MOV     $SP,TTY         ;TYPE $SP IN OCTAL
3092  014616  004767  000554                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3093  014622  012701  000600                MOV     #STK1,  R1      ;SET UP TABLE POINTER
3094  014626  012700  000006                MOV     #6,     R0
3095  014632  000004  015370        6$:     TYPE,   SPACE+3
```

```
3096  014636  012105                        MOV     (R1)+,TTY       ;TYPE (R1)+ IN OCTAL
3097  014640  004767  000532                JSR     %7,PRINTR       ;TYPE LEADING ZERO'S
3098  014644  005300                        DEC     R0              ;COUNT DOWN
3099  014646  001371                        BNE     6$              ;KEEP TYPING
3100  014650  000004  015362        7$:     TYPE,   RETURN
3101  014654  012601                        MOV     (SP)+,  R1      ;RESTORE R1
3102  014656  012600                        MOV     (SP)+,  R0      ;RESTORE R0
3103  014660  000207                        RTS     PC
3104
3105  014662  025440  000040        SIGN$:  .ASCIZ  " + "
3106  014666  026440  000040                .ASCIZ  " - "
3107  014672  025040  000040                .ASCIZ  " * "
3108  014676  027440  000040                .ASCIZ  " / "
3109
3110  014702  000054                COMMA:  .ASCIZ  ","
3111
3112  014704  005015  020040  020040  HEAD1:  .ASCIZ  <15><12>"          PSW     SP      ANSWER"
3113  014712  020040  020040  020040
3114  014720  050040  053523  020040
3115  014726  020040  051440  020120
3116  014734  020040  020040  020040
3117  014742  047101  053523  051105
3118  014750     000
3119  014751     015  020012  020040  HEAD2:  .ASCIZ  <15><12>"          PSW     SP      STK1    STK2    STK3    STK4    STK5
3120  014756  020040  020040  020040
3121  014764  020040  051520  020127
3122  014772  020040  020040  050123
3123  015000  020040  020040  051440
3124  015006  045524  020061  020040
3125  015014  051440  045524  020062
3126  015022  020040  051440  045524
3127  015030  020063  020040  051440
3128  015036  045524  020064  020040
3129  015044  051440  045524  020065
3130  015052  020040  051440  045524
3131  015060  000066
3132  015062  005015  054105  042520  EXPECT: .ASCIZ  <15><12>"EXPECT:   "
3133  015070  052103  020072  000040
3134  015076  005015  047507  035124  GOT:    .ASCIZ  <15><12>"GOT:      "
3135  015104  020040  020040  000040
3136                                  .EVEN
```

```
3137
3138  015112  010346                  READ$:  MOV     R3,-(6)         ;SAVE R3
3139  015114  012703  015212          1$:     MOV     #INPUT,R3       ;GET ADDRESS
3140  015120  022703  015252          2$:     CMP     #.QUES, R3      ;CHECK FOR BUFFER OVERFLOW
3141  015124  001412                          BEQ     4$              ;ABORT
3142  015126  105737  177560                  TSTB    @#177560        ;WAIT FOR
3143  015132  100375                          BPL     .-4             ;A CHARACTER
3144  015134  113713  177562                  MOVB    @#177562,(3)    ;GET CHARACTER
3145  015140  142713  000200                  BICB    #200,(3)        ;GET RID OF JUNK
3146  015144  122713  000177                  CMPB    #177,(3)        ;IS IT A RUBOUT
3147  015150  001003                          BNE     3$              ;SKIP IF NOT
3148  015152  000004  015252          4$:     TYPE    .QUES           ;TYPE A '?'
3149  015156  000756                          BR      1$              ;ZAP THE BUFFER AND LOOP
3150  015160  111367  000210          3$:     MOVB    (3),.TYPE       ;SET UP FOR TYPING
3151  015164  000004  015374                  TYPE    .TYPE           ;ECHO IT
3152  015170  122723  000015                  CMPB    #15,(3)+        ;CHECK FOR RETURN
3153  015174  001351                          BNE     2$              ;LOOP IF NOT RETURN
3154  015176  105063  177777                  CLRB    -1(3)           ;ZAP RETURN (THE 15)
3155  015202  000004  000012                  TYPE    ,12             ;TYPE A LINE FEED
3156  015206  012603                          MOV     (6)+,R3         ;RESTORE R3
3157  015210  000207                          RTS     PC              ;RETURN
3158
3159  015212  000020                  INPUT:  .BLKW   20
3160  015252  006477  000012          .QUES:  .ASCIZ  "?"<15><12>
3161
3162  015256  010546                  .IOT:   MOV     TTY,-(6)        ;SAVE TTY
3163  015260  017605  000002                  MOV     @2(6),TTY       ;GET ADDRESS TO BE TYPED
3164  015264  032705  177400                  BIT     #177400,TTY     ;IS IT A TYPEM?
3165  015270  001004                          BNE     1$              ;NO
3166  015272  010567  000076                  MOV     TTY,.TYPE       ;GET THE CHARACTER
3167  015276  012705  015374                  MOV     #.TYPE,TTY      ;FUDGE THE ADDRESS
3168  015302  105715                  1$:     TSTB    (TTY)           ;TERMINATOR?
3169  015304  001406                          BEQ     2$              ;GET OUT IF SO
3170  015306  112537  177566                  MOVB    (TTY)+,@#177566 ;LOAD AND TYPE THE CHARACTER
3171  015312  105737  177564                  TSTB    @#177564        ;IS THE PRINTER READY
3172  015316  100375                          BPL     .-4             ;WAIT UNTIL IT IS
3173  015320  000770                          BR      1$              ;GET THE NEXT CHARACTER
3174  015322  017646  000002          2$:     MOV     @2(6),-(6)      ;GET ADDRESS TO BE TYPED
3175  015326  062766  000002  000004          ADD     #2,4(6)         ;ADD 2 TO THE ADDRESS
3176  015334  022666  000002                  CMP     (6)+,2(6)       ;IS IT .+2?
3177  015340  001006                          BNE     3$              ;NO
3178  015342  062705  000002                  ADD     #2,TTY          ;ADD 2 TO THE ADDRESS
3179  015346  042705  000001                  BIC     #1,TTY          ;BACK UP TO AN EVEN BYTE
3180  015352  010566  000002                  MOV     TTY,2(6)        ;RESTORE ADDRESS
3181  015356  012605                  3$:     MOV     (6)+,TTY        ;RESTORE TTY
3182  015360  000002                          RTI                     ;RETURN
3183
3184  015362  005015     000          RETURN: .ASCIZ  <15><12>        ;RETURN AND LINEFEED
3185  015365     015  020012  020040  SPACE:  .ASCIZ  <15><12>"   "   ;RETURN AND 3 SPACES
3186  015372     000
3187          015374                  .EVEN
3188  015374  000000                  .TYPE:  0                       ;CHARACTER TYPE LOCATION
```

```
3189
3190  015376  112767  000001  000130  PRINTR: MOVB    #1,.PR          ;SET ZERO FILL SWITCH
3191  015404  000402                  BR      .+6             ;SKIP
3192  015406  005067  000122          PRINTS: CLR     .PR             ;SUPRESS LEADING ZERO'S
3193  015412  112767  177772  000115          MOVB    #-6,.PR+1       ;SET COUNT
3194  015420  010446                  MOV     R4,-(6)         ;SAVE R4
3195  015422  012704  015524                  MOV     #.PRBUF,R4      ;SET POINTER TO FIRST ASCII CHAR.
3196  015426  105014                  CLRB    (4)             ;CLEAR FIRST BYTE
3197  015430  000405                  BR      .PRF            ;ROTATE FIRST BIT
3198  015432  105014          .PRL:   CLRB    (4)             ;CLEAR BYTE OF CHARACTER
3199  015434  006105                  ROL     TTY             ;ROTATE BIT INTO C
3200  015436  106114                  ROLB    (4)             ;PACK IT
3201  015440  006105                  ROL     TTY             ;ROTATE BIT INTO C
3202  015442  106114                  ROLB    (4)             ;PACK IT
3203  015444  006105          .PRF:   ROL     TTY             ;ROTATE BIT INTO C
3204  015446  106114                  ROLB    (4)             ;PACK IT
3205  015450  105714                  TSTB    (4)             ;IS IT ZERO?
3206  015452  001402                  BEQ     .+6             ;SKIP INC
3207  015454  105267  000054          INCB    .PR             ;SET FILL SWITCH
3208  015460  105767  000050          TSTB    .PR             ;CHECK FILL SWITCH
3209  015464  001402                  BEQ     .+6             ;SKIP BITSET
3210  015466  152724  000060          BISB    #'0,(4)+        ;MAKE INTO ASCII CHAR
3211  015472  105267  000037          INCB    .PR+1           ;INC COUNT
3212  015476  001355                  BNE     .PRL            ;REPEAT
3213  015500  022704  015524          CMP     #.PRBUF,R4      ;EMPTY BUFFER?
3214  015504  001002                  BNE     .+6             ;SKIP IF NOT
3215  015506  112724  000060          MOVB    #'0,(4)+        ;LOAD 1 ZERO
3216  015512  105014                  CLRB    (4)             ;NULL TERMINATOR
3217  015514  000004  015524          TYPE    .PRBUF          ;TYPE IT
3218  015520  012604                  MOV     (6)+,R4         ;RESTORE R4
3219  015522  000207                  RTS     PC              ;RETURN
3220
3221  015524  000004          .PRBUF: .BLKW   4               ;OUTPUT BUFFER
3222  015534  000000          .PR:    0                       ;COUNT AND SWITCH
```

```

015536  012777  015652  000120  PDOWN$:  MOV     #ILLUP,@PUVEC$          ;SET FOR FAST UP
015544  012777  000340  000114           MOV     #340,@PUVEC$+2          ;PRIO:7
015552  010046                           MOV     R0,-(6)                 ;PUSH R0 ON STACK
015554  010146                           MOV     R1,-(6)                 ;PUSH R1 ON STACK
015556  010246                           MOV     R2,-(6)                 ;PUSH R2 ON STACK
015560  010346                           MOV     R3,-(6)                 ;PUSH R3 ON STACK
015562  010446                           MOV     R4,-(6)                 ;PUSH R4 ON STACK
015564  010546                           MOV     R5,-(6)                 ;PUSH R5 ON STACK
015566  010667  000064                   MOV     SP,.SAVR6               ;SAVE SP
015572  012777  015602  000064           MOV     #PUP$,@PUVEC$           ;SET UP VECTOR
015600  000000                           HALT

015602  016706  000050          PUP$:    MOV     .SAVR6,SP               ;GET SP
015606  005001                           CLR     R1                      ;WAIT LOOP FOR THE TTY
015610  005201                  1$:      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     #PDOWN$,@PDVEC$         ;SET UP THE POWER DOWN VECTOR
015636  012777  000340  000016           MOV     #340,@PDVEC$+2          ;PRIO:7
015644  000004  015670                   TYPE    ,POWER$
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          PDVEC$:  24,26                           ;POWER DOWN VECTOR
015664  000024  000026          PUVEC$:  24,26                           ;POWER UP VECTOR
015670  005015  047520  042527  POWER$:  .ASCIZ  <15><12>"POWER"
015676  000122
                                         .EVEN

        000001                  .END
```

```
ANS1    000610          584#    754     839     924    1009    1095    1166    1237    1309    1380    1451    1522    1593
                       1664    1735    1807    1878    1949    2020    2091    2162    2233    2305    2376    2447    2518
                       2589    2660    2731    3053
ANS2    000612          585#    756     841     926    1011    1099    1170    1241    1313    1384    1455    1526    1597
                       1668    1739    1811    1882    1953    2024    2095    2166    2237    2309    2380    2451    2522
                       2593    2664    2735    3056
BEGIN   001010           564     642#
BELL  = 000007           517#   2783    2986
BIT0  = 000001           535#
BIT1  = 000002           536#    758     760     770     772
BIT10 = 002000           545#
BIT11 = 004000           546#
BIT12 = 010000           547#
BIT13 = 020000           548#
BIT14 = 040000           549#
BIT15 = 100000           550#
BIT2  = 000004           537#    843     845     855     857
BIT3  = 000010           538#    928     930     940     942
BIT4  = 000020           539#   1013    1015    1025    1027
BIT5  = 000040           540#
BIT6  = 000100           541#
BIT7  = 000200           542#
BIT8  = 000400           543#
BIT9  = 001000           544#
CCC   = 000257           511#
COMMA   014702          3013    3026    3043    3055    3110#
DISPLA= 177570           516#   2971*   2975*
DONE    012662          2780#
ERRORS  001002           639#    659*   2987*
ERRORS  014142          2996    3008#
EXPECT  015062          3034    3061    3132#
FADD  = 075000           511#
FDIV  = 075030           511#
FISTRP  000754           632#    653
FMUL  = 075020           511#
FORTAN  001360           670     707#
FSUB  = 075010           511#
GOT     015076          3046    3087    3134#
HEAD1   014704          3033    3112#
HEAD2   014751          3060    3119#
HLT   = 104000           512#    632     752     775     782     786     790     794     798     802     806     837     860
                          867     871     875     879     883     887     891     922     945     952     956     960     964
                          968     972     976    1007    1030    1037    1041    1045    1049    1053    1057    1061    1085
                         1093    1097    1101    1108    1112    1116    1120    1124    1128    1132    1156    1164    1168
                         1172    1179    1183    1187    1191    1195    1199    1203    1227    1235    1239    1243    1251
                         1255    1259    1263    1267    1271    1275    1299    1307    1311    1315    1322    1326    1330
                         1334    1338    1342    1346    1370    1378    1382    1386    1393    1397    1401    1405    1409
                         1413    1417    1441    1449    1453    1457    1464    1468    1472    1476    1480    1484    1488
                         1512    1520    1524    1528    1535    1539    1543    1547    1551    1555    1559    1583    1591
                         1595    1599    1606    1610    1614    1618    1622    1626    1630    1654    1662    1666    1670
                         1677    1681    1685    1689    1693    1697    1701    1725    1733    1737    1741    1749    1753
                         1757    1761    1765    1769    1773    1797    1805    1809    1813    1820    1824    1828    1832
                         1836    1840    1844    1868    1876    1880    1884    1891    1895    1899    1903    1907    1911
                         1915    1939    1947    1951    1955    1962    1966    1970    1974    1978    1982    1986    2010
                         2018    2022    2026    2033    2037    2041    2045    2049    2053    2057    2081    2089    2093
                         2097    2104    2108    2112    2116    2120    2124    2128    2152    2160    2164    2168    2175
```

```
                    2179    2183    2187    2191    2195    2199    2223    2231    2235    2239    2247    2251    2255
                    2259    2263    2267    2271    2295    2303    2307    2311    2318    2322    2326    2330    2334
                    2338    2342    2366    2374    2378    2382    2389    2393    2397    2401    2405    2409    2413
                    2437    2445    2449    2453    2460    2464    2468    2472    2476    2480    2484    2508    2516
                    2520    2524    2531    2535    2539    2543    2547    2551    2555    2579    2587    2591    2595
                    2602    2606    2610    2614    2618    2622    2626    2650    2658    2662    2666    2673    2677
                    2681    2685    2689    2693    2697    2721    2729    2733    2737    2745    2749    2753    2757
                    2761    2765    2769
HLTADS  014140      2991*   2992*   2993    3006#
HLTS    014020       649    2984#
ICNT    001000       638#    664*   2958    2964    2966    2968*   2969*   2971    2974*   2975    3003*
ILLUP   015652      3224    3251#
INPUT   015212      2808    3139    3159#
KITS    013764      2961    2967    2974#   3004
LADS    014012       665*   2970*   2976    2978    2981#
LEVEL0= 000000       551#
LEVEL1= 000040       552#
LEVEL2= 000100       553#
LEVEL3= 000140       554#
LEVEL4= 000200       555#
LEVEL5= 000240       556#
LEVEL6= 000300       557#
LEVEL7= 000340       558#
N     = 000035       458#     728     813#    898#    983#   1068#   1139#   1210#   1282#   1353#   1424#   1495#   1566#
                    1637#   1708#   1780#   1851#   1922#   1993#   2064#   2135#   2206#   2278#   2349#   2420#   2491#
                    2562#   2633#   2704#   2776#
OVERS   013770      2959    2975#
PC    =%000007       526#    627*    669*    736*    821*    906*    991*   1076*   1147*   1218*   1290*   1361*   1432*
                    1503*   1574*   1645*   1716*   1788*   1859*   1930*   2001*   2072*   2143*   2214*   2286*   2357*
                    2428*   2499*   2570*   2641*   2712*   2807*   2952*   2996*   3103*   3157*   3219*
PCNT    001004       640#    660*    661*    2777*   2778*
PDOWNS  015536       647    3224#   3246
PDVECS  015660      3246*   3247*   3255#
POWERS  015670      3248    3257#
PRINTR  015376      2994    3012    3015    3025    3028    3036    3039    3042    3045    3048    3051    3054    3057
                    3063    3066    3070    3074    3077    3080    3083    3086    3089    3092    3097    3190#
PRINTS  015406      3192#
PS    = 177776       514#    663*    741     766     826     851     911     936     996    1021    1081    1152    1223
                    1295    1366    1437    1508    1579    1650    1721    1793    1864    1935    2006    2077    2148
                    2219    2291    2362    2433    2504    2575    2646    2717    2839    2845    2854    2860    2869
                    2875    2884    2890    2896    2914    2919
PUPS    015602      3233    3236#
PUSHR   013600       736     821     906     991    1076    1147    1218    1290    1361    1432    1503    1574    1645
                    1716    1788    1859    1930    2001    2072    2143    2214    2286    2357    2428    2499    2570
                    2641    2712    2946#
PUVECS  015664      3224*   3225*   3233*   3256#
RAND.A  000620       589#    619*    625     626*    655*    686     800     885     970    1055    1126    1197    1269
                    1340    1411    1482    1553    1624    1695    1767    1838    1909    1980    2051    2122    2193
                    2265    2336    2407    2478    2549    2620    2691    2763    2830    2947    3011    3082
RAND.B  000622       590#    619     621*    624*    656*    691     804     889     974    1059    1130    1201    1273
                    1344    1415    1486    1557    1628    1699    1771    1842    1913    1984    2055    2126    2197
                    2269    2340    2411    2482    2553    2624    2695    2767    2829    2946    3014    3085
RAND.C  000624       591#    621     622*    623*    657*    697     792     877     962    1047    1118    1189    1261
                    1332    1403    1474    1545    1616    1687    1759    1830    1901    1972    2043    2114    2185
                    2257    2328    2399    2470    2541    2612    2683    2755    2832    2880    2949    3024    3076
RAND.D  000626       592#    618*    620*    623     625*    658*    703     796     881     966    1051    1122    1193
```

```
                    1265    1336    1407    1478    1549    1620    1691    1763    1834    1905    1976    2047    2118
                    2189    2261    2332    2403    2474    2545    2616    2687    2759    2831    2948    3027    3079
RAND4$  000674       618#    669
READIN  012762       685     690     696     702    2807#
READ$   015112      2807    3138#
RETURN  015362      2990    3100    3184#
RNDFLG  000672       615#    666*    758     760*    770     772*    843     845*    855     857*    928     930*    940
                     942*   1013    1015*   1025    1027*
R0      =%000000     518#    644*    645*    646*    647*    648*    649*    650*    651*    652*    735*    739*    742
                     777    1289*   1293*   1296    1317    1787*   1791*   1794    1815    2285*   2289*   2292    2313
                    2794*   2906*   2907    2910    2920*   2921*   2922*   2934*   3008    3016*   3017    3068    3069
                    3094*   3098*   3102*   3226    3245*
R1      =%000001     519#    820*    824*    827     862    1360*   1364*   1367    1388    1858*   1862*   1865    1886
                    2356*   2360*   2363    2384    2809*   2810*   2817*   2819*   2821*   2823*   3009    3017*   3018*
                    3019*   3021    3029*   3030*   3035    3041    3044    3062    3072    3073    3093*   3096    3101*
                    3227    3237*   3238*   3244*
R2      =%000002     520#    905*    909*    912     947    1431*   1435*   1438    1459    1929*   1933*   1936    1957
                    2427*   2431*   2434    2455    2808*   2811    3228    3243*
R3      =%000003     521#    990*    994*    997    1032    1502*   1506*   1509    1530    2000*   2004*   2007    2028
                    2498*   2502*   2505    2526    2811*   2813*   2815    2823    3138    3139*   3140    3156*   3229
                    3242*
R4      =%000004     522#   1075*   1079*   1082    1103    1573*   1577*   1580    1601    2071*   2075*   2078    2099
                    2569*   2573*   2576    2597    2833    2848    2863    2878    2899    2902    2904*   3194    3195*
                    3213    3218*   3230    3241*
R5      =%000005     523#    685*    690*    696*    702*   1146*   1150*   1153    1174    1644*   1648*   1651    1672
                    2142*   2146*   2149    2170    2640*   2644*   2647    2668    2809    2827*   2942*   3231    3240*
SAVSTK  000670       614#    749*    750     779*    780     834*    835     864*    865     919*    920     949*    950
                    1004*   1005    1034*   1035    1090*   1091    1105*   1106    1161*   1162    1176*   1177    1232*
                    1233    1248*   1249    1304*   1305    1319*   1320    1375*   1376    1390*   1391    1446*   1447
                    1461*   1462    1517*   1518    1532*   1533    1588*   1589    1603*   1604    1659*   1660    1674*
                    1675    1730*   1731    1746*   1747    1802*   1803    1817*   1818    1873*   1874    1888*   1889
                    1944*   1945    1959*   1960    2015*   2016    2030*   2031    2086*   2087    2101*   2102    2157*
                    2158    2172*   2173    2228*   2229    2244*   2245    2300*   2301    2315*   2316    2371*   2372
                    2386*   2387    2442*   2443    2457*   2458    2513*   2514    2528*   2529    2584*   2585    2599*
                    2600    2655*   2656    2670*   2671    2726*   2727    2742*   2743    3065
SCC     = 000277     511#
SCOPE   = 104400     511#     727     811     896     981    1066    1137    1208    1280    1351    1422    1493    1564
                    1635    1706    1778    1849    1920    1991    2062    2133    2204    2276    2347    2418    2489
                    2560    2631    2702    2774
SCOPE$  013644       651    2954#
SCOTMP  014016      2956*   2957*   2958    2983#
SIGN$   014662      3020    3023    3105#
SNV     = 000272     511#    2918
SP      =%000006     525#    642*    662*    808*    893*    978*   1063*   1134*   1205*   1217*   1221*   1224    1245
                    1277*   1348*   1419*   1490*   1561*   1632*   1703*   1715*   1719*   1722    1743    1775*   1846*
                    1917*   1988*   2059*   2130*   2201*   2213*   2217*   2220    2241    2273*   2344*   2415*   2486*
                    2557*   2628*   2699*   2711*   2715*   2718    2739    2771*   2829*   2830*   2831*   2832*   2837
                    2844    2846    2847    2852    2859    2861    2862    2867    2874    2876    2877    2888    2895
                    2897    2898    2901    2950    3008*   3009*   3101    3102    3232    3236*
SPACE   015365      2995    3010    3037    3040    3049    3052    3064    3067    3071    3075    3078    3081    3084
                    3090    3095    3185#
STACK0  000604       582#    642     662     735     779     820     864     905     949     990    1034    1075    1105
                    1146    1176    1217    1289    1319    1360    1390    1431    1461    1502    1532    1573    1603
                    1644    1674    1715    1787    1817    1858    1888    1929    1959    2000    2030    2071    2101
                    2142    2172    2213    2285    2315    2356    2386    2427    2457    2498    2528    2569    2599
                    2640    2670    2711    2949*
```

```
STACK2  000606          583#  2948*
STACK4  000610          584#   749   834   919  1004  1090  1161  1232  1304  1375  1446  1517  1588
                       1659  1730  1802  1873  1944  2015  2086  2157  2228  2300  2371  2442  2513
                       2584  2655  2726  2947* 3038
STACK6  000612          585#  2946*
START   001146          567   662#  2800
STK1    000600          580#   784   869   954  1039  1110  1181  1248  1253  1324  1395  1466  1537
                       1608  1679  1746  1751  1822  1893  1964  2035  2106  2177  2244  2249  2320
                       2391  2462  2533  2604  2675  2742  2747  3093
STK2    000602          581#   788   873   958  1043  1114  1185  1257  1328  1399  1470  1541  1612
                       1683  1755  1826  1897  1968  2039  2110  2181  2253  2324  2395  2466  2537
                       2608  2679  2751
STK3    000604          582#   792   877   962  1047  1118  1189  1261  1332  1403  1474  1545  1616
                       1687  1759  1830  1901  1972  2043  2114  2185  2257  2328  2399  2470  2541
                       2612  2683  2755
STK4    000606          583#   796   881   966  1051  1122  1193  1265  1336  1407  1478  1549  1620
                       1691  1763  1834  1905  1976  2047  2118  2189  2261  2332  2403  2474  2545
                       2616  2687  2759
STK5    000610          584#   800   885   970  1055  1126  1197  1269  1340  1411  1482  1553  1624
                       1695  1767  1838  1909  1980  2051  2122  2193  2265  2336  2407  2478  2549
                       2620  2691  2763
STK6    000612          585#   804   889   974  1059  1130  1201  1273  1344  1415  1486  1557  1628
                       1699  1771  1842  1913  1984  2055  2126  2197  2269  2340  2411  2482  2553
                       2624  2695  2767
SVLADS  013744          2963  2969#  2977
SWR   = 177570          515#   516   667  2781  2785  2954  2956  2960  2962  2984  2988  2997  3000
SW08  = 000400          534#  2954
SW09  = 001000          533#  3000
SW10  = 002000          532#  2780  2781  2984
SW11  = 004000          531#  2962
SW12  = 010000          530#  2782  2784  2785
SW13  = 020000          529#  2988
SW14  = 040000          528#  2960
SW15  = 100000          527#
TIMES   014014         2966  2982#
TST1    001440          735#
TST10   004202         1289#
TST11   004442         1360#
TST12   004702         1431#
TST13   005142         1502#
TST14   005402         1573#
TST15   005642         1644#
TST16   006102         1715#
TST17   006344         1787#
TST2    002000          820#
TST20   006604         1858#
TST21   007044         1929#
TST22   007304         2000#
TST23   007544         2071#
TST24   010004         2142#
TST25   010244         2213#
TST26   010506         2285#
TST27   010746         2356#
TST3    002340          905#
TST30   011206         2427#
TST31   011446         2498#
```

```
TST32   011706          2569#
TST33   012146          2640#
TST34   012406          2711#
TST4    002700           990#
TST5    003240          1075#
TST6    003500          1146#
TST7    003740          1217#
TTY    =%000005          524#  2993*  3011*  3014*  3024*  3027*  3035*  3038*  3041*  3044*  3047*  3050*  3053*
                        3056*  3062*  3065*  3069*  3073*  3076*  3079*  3082*  3085*  3088*  3091*  3096*  3162
                        3163*  3164   3166   3167*  3168   3170   3178*  3179*  3180   3181*  3199*  3201*  3203*
TYPE   = 000004          513#   677    683    688    694    700   2783   2986   2990   2995   3010   3013   3022
                        3026   3033   3034   3037   3040   3043   3046   3049   3052   3055   3060   3061   3064
                        3067   3071   3075   3078   3081   3084   3087   3090   3095   3100   3148   3151   3155
                        3217   3248
TYPIN   001210           668    677#   687    693    699    705
YESRT   000752           630#   643
$ADDER  000636           597#   766*   788   1114   1399   1683   1968   2253   2537   2926*
$ADDPS  000630           594#   707*   743    767*  1083   1368   1652   1937   2221   2506   2835   2839*  2845*
                        2924   2927*  3030
$ADD1   000632           595#   754    762*   774*  1095   1380   1664   1949   2233   2518   2840*  2846*
$ADD2   000634           596#   756    761*   773*  1099   1384   1668   1953   2237   2522   2841*  2847*
$ADR   = ******   G      460#   714
$DIVER  000666           612#  1021*  1043   1328   1612   1897   2181   2466   2751   2884*  2940*
$DIVPS  000660           609#   710*   998   1022*  1297   1581   1866   2150   2435   2719   2885*  2886   2890*
                        2896*  2941*
$DIV1   000662           610#  1009   1017*  1029*  1309   1593   1878   2162   2447   2731   2891*  2897*
$DIV2   000664           611#  1011   1016*  1028*  1313   1597   1882   2166   2451   2735   2892*  2898*
$DVR   = ******   G      460#   723
$ERR    013414   G       460#  2906#
$ERRA   013420   G       460#  2907#
$EXIT   013412           725   2904#
$MLR   = ******   G      460#   720
$MULER  000656           607#   936*   958   1257   1541   1826   2110   2395   2679   2936*
$MULPS  000650           604#   709*   913    937*  1225   1510   1795   2079   2364   2648   2865   2869*  2875*
                        2937*
$MUL1   000652           605#   924    932*   944*  1237   1522   1807   2091   2376   2660   2870*  2876*
$MUL2   000654           606#   926    931*   943*  1241   1526   1811   2095   2380   2664   2871*  2877*
$POLSH  013064           712   2833#
$POPAD  013066           715   2835#
$POPDV  013300           724   2880#
$POPML  013222           721   2865#
$POPSB  013144           718   2850#
$PSW    000614           586#   741*   743    746    826*   828    831    911*   913    916    996*   998    1001
                        1081*  1083   1087   1152*  1154   1158   1223*  1225   1229   1295*  1297   1301   1366*
                        1368   1372   1437*  1439   1443   1508*  1510   1514   1579*  1581   1585   1650*  1652
                        1656   1721*  1723   1727   1793*  1795   1799   1864*  1866   1870   1935*  1937   1941
                        2006*  2008   2012   2077*  2079   2083   2148*  2150   2154   2219*  2221   2225   2291*
                        2293   2297   2362*  2364   2368   2433*  2435   2439   2504*  2506   2510   2575*  2577
                        2581   2646*  2648   2652   2717*  2719   2723   2914*  2919*  2926   2930   2936   2940
                        3031   3047   3088
$PUSH   013044           713    716    719    722   2829#
$SBR   = ******   G      460#   717
$SKIP   013406          2836   2842   2851   2857   2866   2872   2887   2893   2901#
$SP     000616           587#   742*   750    780    827*   835    865    912*   920    950    997*   1005   1035
                        1082*  1091   1106   1153*  1162   1177   1224*  1233   1249   1296*  1305   1320   1367*
                        1376   1391   1438*  1447   1462   1509*  1518   1533   1580*  1589   1604   1651*  1660
```

```
                    1675   1722*  1731   1747   1794*  1803   1818   1865*  1874   1889   1936*  1945   1960
                    2007*  2016   2031   2078*  2087   2102   2149*  2158   2173   2220*  2229   2245   2292*
                    2301   2316   2363*  2372   2387   2434*  2443   2458   2505*  2514   2529   2576*  2585
                    2600   2647*  2656   2671   2718*  2727   2743   3050   3091
$SUBER  000646       602#   851*   873   1185   1470   1755   2039   2324   2608   2930*
$SUBPS  000640       599#   708*   828    852*  1154   1439   1723   2008   2293   2577   2850   2854*  2860*
                    2931*
$SUB1   000642       600#   839    847*   859*  1166   1451   1735   2020   2305   2589   2855*  2861*
$SUB2   000644       601#   841    846*   858*  1170   1455   1739   2024   2309   2593   2856*  2862*
.       = 015700     560#   561    562#   566#   569#   636#   677    682#   683    688    694    700    751
                     781    785    789    793    797    801    805    836    866    870    874    878    882
                     886    890    921    951    955    959    963    967    971    975   1006   1036   1040
                    1044   1048   1052   1056   1060   1084   1092   1096   1100   1107   1111   1115   1119
                    1123   1127   1131   1155   1163   1167   1171   1178   1182   1186   1190   1194   1198
                    1202   1226   1234   1238   1242   1250   1254   1258   1262   1266   1270   1274   1298
                    1306   1310   1314   1321   1325   1329   1333   1337   1341   1345   1369   1377   1381
                    1385   1392   1396   1400   1404   1408   1412   1416   1440   1448   1452   1456   1463
                    1467   1471   1475   1479   1483   1487   1511   1519   1523   1527   1534   1538   1542
                    1546   1550   1554   1558   1582   1590   1594   1598   1605   1609   1613   1617   1621
                    1625   1629   1653   1661   1665   1669   1676   1680   1684   1688   1692   1696   1700
                    1724   1732   1736   1740   1748   1752   1756   1760   1764   1768   1772   1796   1804
                    1808   1812   1819   1823   1827   1831   1835   1839   1843   1867   1875   1879   1883
                    1890   1894   1898   1902   1906   1910   1914   1938   1946   1950   1954   1961   1965
                    1969   1973   1977   1981   1985   2009   2017   2021   2025   2032   2036   2040   2044
                    2048   2052   2056   2080   2088   2092   2096   2103   2107   2111   2115   2119   2123
                    2127   2151   2159   2163   2167   2174   2178   2182   2186   2190   2194   2198   2222
                    2230   2234   2238   2246   2250   2254   2258   2262   2266   2270   2294   2302   2306
                    2310   2317   2321   2325   2329   2333   2337   2341   2365   2373   2377   2381   2388
                    2392   2396   2400   2404   2408   2412   2436   2444   2448   2452   2459   2463   2467
                    2471   2475   2479   2483   2507   2515   2519   2523   2530   2534   2538   2542   2546
                    2550   2554   2578   2586   2590   2594   2601   2605   2609   2613   2617   2621   2625
                    2649   2657   2661   2665   2672   2676   2680   2684   2688   2692   2696   2720   2728
                    2732   2736   2744   2748   2752   2756   2760   2764   2768   2998   3001   3143   3159#
                    3172   3187#  3191   3206   3209   3214   3221#  3252
.BIT    = 177777     458#   2780   2782   2784   2984   2988   3000
.IOT    015256       645    3162#
.PR     015534      3190*  3192*  3193*  3207*  3208   3211*  3222#
.PRBUF  015524      3195   3213   3217   3221#
.PRF    015444      3197   3203#
.PRL    015432      3198#  3212
.QUES   015252      3140   3148   3160#
.SAVR6  015656      3232*  3236   3254#
.TBIT   012760      2787*  2802#
.TYPE   015374      3150*  3151   3166*  3167   3188#
```

H06


```
DUMP      458#  2993  3011  3014  3024  3027  3035  3038  3041  3044  3047  3050  3053  3056  3062
        3065    3069  3073  3076  3079  3082  3085  3088  3091  3096
POP       458#  3240
PUSH      458#  3226
$CATCH    458#   560
$END      458#  2780
$EQUAT    458#   511
$EXER     728#   813   898   983  1068  1139  1210  1282  1353  1424  1495  1566  1637  1708  1780
         1851   1922  1993  2064  2135  2206  2278  2349  2420  2491  2562  2633  2704
$OCTAL    458#  3190
$POWER    458#  3224
$RAND4    458#   617
$READ     458#  3138
$TYPE     458#  3162
```

DBKEBA.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

DBKEBA.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
        1181  1185  1189  1193  1197  1201  1225  1233  1237  1241  1245  1249  1253  1257  1261
        1265  1269  1273  1297  1305  1309  1313  1320  1324  1328  1332  1336  1340  1344  1368
        1376  1380  1384  1391  1395  1399  1403  1407  1411  1415  1439  1447  1451  1455  1462
        1466  1470  1474  1478  1482  1486  1510  1518  1522  1526  1533  1537  1541  1545  1549
        1553  1557  1581  1589  1593  1597  1604  1608  1612  1616  1620  1624  1628  1652  1660
        1664  1668  1675  1679  1683  1687  1691  1695  1699  1723  1731  1735  1739  1743  1747
        1751  1755  1759  1763  1767  1771  1795  1803  1807  1811  1818  1822  1826  1830  1834
        1838  1842  1866  1874  1878  1882  1889  1893  1897  1901  1905  1909  1913  1937  1945
        1949  1953  1960  1964  1968  1972  1976  1980  1984  2008  2016  2020  2024  2031  2035
        2039  2043  2047  2051  2055  2079  2087  2091  2095  2102  2106  2110  2114  2118  2122
        2126  2150  2158  2162  2166  2173  2177  2181  2185  2189  2193  2197  2221  2229  2233
        2237  2241  2245  2249  2253  2257  2261  2265  2269  2293  2301  2305  2309  2316  2320
        2324  2328  2332  2336  2340  2364  2372  2376  2380  2387  2391  2395  2399  2403  2407
        2411  2435  2443  2447  2451  2458  2462  2466  2470  2474  2478  2482  2506  2514  2518
        2522  2529  2533  2537  2541  2545  2549  2553  2577  2585  2589  2593  2600  2604  2608
        2612  2616  2620  2624  2648  2656  2660  2664  2671  2675  2679  2683  2687  2691  2695
        2719  2727  2731  2735  2739  2743  2747  2751  2755  2759  2763  2767  2901  2907  3072
        3140  3176  3213
CMPB    2910  2958  2966  3146  3152
COM     2787
DEC     3098
DECB     618
EMT      512
FADD     739  1079  1364  1648  1933  2217  2502
FDIV     994  1293  1577  1862  2146  2431  2715
FMUL     909  1221  1506  1791  2075  2360  2644
FSUB     824  1150  1435  1719  2004  2289  2573
HALT     561  2999  3234  3251
INC     2987  3238
INCB    2969  2974  3207  3211
IOT      513
JMP      564   567  2800  2833  2848  2863  2878  2899  2902  3004
JSR      669   685   690   696   702   712   736   821   906   991  1076  1147  1218  1290  1361
        1432  1503  1574  1645  1716  1788  1859  1930  2001  2072  2143  2214  2286  2357  2428
        2499  2570  2641  2712  2796  2807  2994  2996  3012  3015  3025  3028  3036  3039  3042
        3045  3048  3051  3054  3057  3063  3066  3070  3074  3077  3080  3083  3086  3089  3092
        3097
MOV      642   643   644   645   646   647   648   649   650   651   652   653   654   655   656
         657   658   662   663   735   741   742   749   766   767   779   808   820   826   827
         834   851   852   864   893   905   911   912   919   936   937   949   978   990   996
         997  1004  1021  1022  1034  1063  1075  1081  1082  1090  1105  1134  1146  1152  1153
        1161  1176  1205  1217  1223  1224  1232  1248  1277  1289  1295  1296  1304  1319  1348
        1360  1366  1367  1375  1390  1419  1431  1437  1438  1446  1461  1490  1502  1508  1509
        1517  1532  1561  1573  1579  1580  1588  1603  1632  1644  1650  1651  1659  1674  1703
        1715  1721  1722  1730  1746  1775  1787  1793  1794  1802  1817  1846  1858  1864  1865
        1873  1888  1917  1929  1935  1936  1944  1959  1988  2000  2006  2007  2015  2030  2059
        2071  2077  2078  2086  2101  2130  2142  2148  2149  2157  2172  2201  2213  2219  2220
        2228  2244  2273  2285  2291  2292  2300  2315  2344  2356  2362  2363  2371  2386  2415
        2427  2433  2434  2442  2457  2486  2498  2504  2505  2513  2528  2557  2569  2575  2576
        2584  2599  2628  2640  2646  2647  2655  2670  2699  2711  2717  2718  2726  2742  2771
        2790  2792  2794  2808  2809  2829  2830  2831  2832  2839  2845  2846  2847  2854  2860
        2861  2862  2869  2875  2876  2877  2884  2885  2890  2896  2897  2898  2906  2914  2919
        2926  2927  2930  2931  2936  2937  2940  2941  2946  2947  2948  2949  2950  2956  2970
        2971  2975  2978  2991  2993  3008  3009  3011  3014  3016  3017  3020  3024  3027  3035
        3038  3041  3044  3047  3050  3053  3056  3062  3065  3069  3073  3076  3079  3082  3085
        3088  3091  3093  3094  3096  3101  3102  3138  3139  3156  3162  3163  3166  3167  3174
```

DBKEBA.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
          3180  3181  3194  3195  3218  3224  3225  3226  3227  3228  3229  3230  3231  3232  3233
          3236  3240  3241  3242  3243  3244  3245  3246  3247
MOVB      2811  2968  3144  3150  3170  3190  3193  3215
NOP        738   823   908   993  1078  1149  1220  1292  1363  1434  1505  1576  1647  1718  1790
          1861  1932  2003  2074  2145  2216  2288  2359  2430  2501  2572  2643  2714  2797  2798
          2799
ROL       3199  3201  3203
ROLB      3200  3202  3204
RTI        633   809   894   979  1064  1135  1206  1278  1349  1420  1491  1562  1633  1704  1776
          1847  1918  1989  2060  2131  2202  2274  2345  2416  2487  2558  2629  2700  2772  2791
          2793  2972  2979  3002  3182  3249
RTS        627  2827  2904  2942  2952  3103  3157  3219
RTT        630
SBC        774   859   944  1029
SCC       2882
SEC       2825
SEV        765   850   935  1020  2913
SUB        773   858   943  1028  2813  2922  2934  2992
SWAB      2921
TRAP       511
TST       2835  2844  2850  2859  2865  2874  2886  2895  2924  2976  2997  3068
TSTB       667   746   831   916  1001  1087  1158  1229  1301  1372  1443  1514  1585  1656  1727
          1799  1870  1941  2012  2083  2154  2225  2297  2368  2439  2510  2581  2652  2723  2964
          3031  3142  3168  3171  3205  3208
.ASCIZ     678   684   689   695   701  3105  3106  3107  3108  3110  3112  3119  3132  3134  3160
          3184  3185  3257
.ASECT     459
.BLKW     3159  3221
.END      3261
.ENDC      565   745   776   780   830   861   865   915   946   950  1000  1031  1035  1086  1102
          1106  1157  1173  1177  1228  1244  1249  1300  1316  1320  1371  1387  1391  1442  1458
          1462  1513  1529  1533  1584  1600  1604  1655  1671  1675  1726  1742  1747  1798  1814
          1818  1869  1885  1889  1940  1956  1960  2011  2027  2031  2082  2098  2102  2153  2169
          2173  2224  2240  2245  2296  2312  2316  2367  2383  2387  2438  2454  2458  2509  2525
          2529  2580  2596  2600  2651  2667  2671  2722  2738  2743  2783  2784  2794  2988  2990
          3005  3168  3183  3198  3232  3240  3249
.EVEN      682  3136  3187  3259
.GLOBL     460
.IF        561   743   754   777   828   839   862   913   924   947   998  1009  1032  1083  1095
          1103  1154  1166  1174  1225  1237  1245  1297  1309  1317  1368  1380  1388  1439  1451
          1459  1510  1522  1530  1581  1593  1601  1652  1664  1672  1723  1735  1743  1795  1807
          1815  1866  1878  1886  1937  1949  1957  2008  2020  2028  2079  2091  2099  2150  2162
          2170  2221  2233  2241  2293  2305  2313  2364  2376  2384  2435  2447  2455  2506  2518
          2526  2577  2589  2597  2648  2660  2668  2719  2731  2739  2780  2782  2784  2984  2988
          3000  3164  3174  3198  3232  3240  3248
.IFF       745   776   777   830   861   862   915   946   947  1000  1031  1032  1083  1095  1103
          1154  1166  1174  1225  1237  1249  1297  1309  1317  1368  1380  1388  1439  1451  1459
          1510  1522  1530  1581  1593  1601  1652  1664  1672  1723  1735  1747  1795  1807  1815
          1866  1878  1886  1937  1949  1957  2008  2020  2028  2079  2091  2099  2150  2162  2170
          2221  2233  2245  2293  2305  2313  2364  2376  2384  2435  2447  2455  2506  2518  2526
          2577  2589  2597  2648  2660  2668  2719  2731  2743  2783  2984  3000  3183
.IIF       3223
.IRP       3226  3240
.LIST      458   511   559   561   616   617   635   728   813   898   983  1068  1139  1210  1282
          1353  1424  1495  1566  1637  1708  1780  1851  1922  1993  2064  2135  2206  2278  2349
          2420  2491  2562  2633  2704  2776  2803  2828  2943  2953  2984  3138  3162  3190  3224
```

DBKEBA.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
.MACRO    728
.MCALL    458
.NLIST    458    511    559    561    616    617    635    728    813    898    983   1068   1139   1210   1282
         1353   1424   1495   1566   1637   1708   1780   1851   1922   1993   2064   2135   2206   2278   2349
         2420   2491   2562   2633   2704   2776   2803   2828   2943   2953   2984   3138   3162   3190   3224
.PAGE     511    635    813    898    983   1068   1282   1495   1708   1922   2135   2349   2562   2776   2879
         3137   3189   3223
.REM        1    464
.REPT     561
.SBTTL    458    511    559    616    617    635    728    813    898    983   1068   1139   1210   1282   1353
         1424   1495   1566   1637   1708   1780   1851   1922   1993   2064   2135   2206   2278   2349   2420
         2491   2562   2633   2704   2776   2803   2828   2943   2953   2984   3138   3162   3190   3224
.TITLE    458

. ABS.  015700    000
        000000    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 *,DBKEBA.SEQ/SOL/CRF/PAGNUM=DBKEBA
 RUN-TIME: 15 24 4 SECONDS
 RUN-TIME RATIO: 177/44=3.9
 CORE USED:  9K  (17 PAGES)
```

M06

Spooler runtime 10 Seconds, 48 KCS, 300 disk reads, 0 disk writes, 76 pages

[redacted]