

KD11-K

FLOAT PT RAN OPERAND EXE
MD-11-DQFPD-B

EP-DQFPD-B-DL-A
COPYRIGHT © 1977
FICHE 1 OF 1

JUN 1977
digital
MADE IN USA

This microfiche card contains a grid of 100 frames of technical data, arranged in 10 rows and 10 columns. Each frame displays a different set of data, likely related to the MD-11-DQFPD-B processor. The data is presented in a structured, tabular format, with some frames containing headers and sub-headers. The text is small and dense, typical of microfiche storage. The frames contain various alphanumeric strings, possibly representing memory addresses, data values, or control signals. The overall layout is consistent across the grid, suggesting a systematic presentation of related information.

801

EOF1DQFPCBSEQ
PDP10 411

00010000

770608

PDP10 411

HDR1DQFPDBSEQ

00010000

770608

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DQFPD-B-D
PRODUCT NAME: PDP-11/6X - FPII-E FLOATING POINT UNIT
 ADD/SUB/MUL/DIV
 RANDOM OPERAND EXERCISER
DATE : MAY, 1977
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: KEN CHAPMAN
REVISED BY: DON NORTH

COPYRIGHT (C) 1977
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS

THIS SOFTWARE IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM, AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE, AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT NOT SUPPLIED BY DIGITAL.

CONTENTS

- 1. ABSTRACT
- 2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
- 3. LOADING PROCEDURE
- 4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS
 - 4.3 PROGRAM/OPERATOR ACTION
- 5. OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 PROGRAM/OPERATOR ACTION
 - 5.3 HOT (FP11-E) / WARM (PDP-11/6X) SELECTION
- 6. ERRORS
 - 6.1.1 ERROR MESSAGE FORMAT
 - 6.1.2 FLOATING POINT DATA FORMAT
 - 6.2 RECOVERY
 - 6.3 CAUSES
- 7. RESTRICTIONS
 - 7.1 STARTING
 - 7.2 OPERATIONAL
- 8. MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 STACK POINTER
 - 8.3 POWER FAIL
- 9. PROGRAM DESCRIPTION
 - 9.1 ORGANIZATION
 - 9.2 TEST DESCRIPTION
 - 9.3 SUBROUTINE ABSTRACTS
- 10. ACT/APT/XXDP

1. ABSTRACT

THIS PROGRAM IS AN EXERCISER FOR THE PDP-11/6X FLOATING POINT ADD, SUBTRACT, MULTIPLY, AND DIVIDE INSTRUCTIONS. RANDOM NUMBER PATTERNS ARE USED AS THE OPERANDS, AND THE HARDWARE GENERATED RESULTS ARE CHECKED AGAINST RESULTS OBTAINED FROM FLOATING POINT SOFTWARE ROUTINES TO INSURE CORRECTNESS. THE PDP-11/6X IS OPERATED IN DOUBLE AND SINGLE FLOATING MODE, ROUND AND TRUNCATE MODE, AND WITH UNDERFLOW AND OVERFLOW CONDITIONS ENABLED AND DISABLED. THE PROGRAM WILL RUN FOR 400(8) "SUBPASSES" BEFORE GIVING AN "END OF PASS" INDICATION, SO THAT A SUFFICIENT NUMBER OF RANDOM PATTERNS ARE OBTAINED FOR USE AS OPERANDS. ALSO AT THIS TIME, OPTIONAL STATUS INFORMATION ON THE TYPES OF RANDOM OPERANDS SELECTED CAN BE PRINTED ON THE CONSOLE. BOTH "HOT" (FP11-E OPTION) AND "WARM" (PDP-11/6X MICROCODE) FLOATING POINT UNITS CAN BE SELECTED FOR TESTING.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11/6X STANDARD COMPUTER WITH MINIMUM 16K OF MEMORY. OPTIONAL FP11-E FLOATING POINT UNIT, IF SELECTED.

2.2 STORAGE

THE PROGRAM USES MEMORY 0-34120(8). THE UPPER 2.0K WORDS ARE RESERVED FOR THE XXDP MONITOR, IF EMPLOYED.

2.3 PRELIMINARY PROGRAMS

THE CPU, CACHE, AND MEMORY TEST PROGRAMS MUST BE RUN FIRST TO VERIFY THE CORRECT OPERATION OF THE BASE MACHINE.

THE PDP-11/6X - FP11-E FLOATING POINT PROCESSOR INSTRUCTION SET TESTS SHOULD THEN BE RUN IN THE FOLLOWING ORDER:

- (1) DQFPA FPU BASIC INSTRUCTION TESTS
- (2) DQFPB FPU ADVANCED INSTRUCTION TESTS
- (3) DQFPC FPU INSTRUCTION EXERCISER
- (4) DQFPD FPU ADD/SUB/MUL/DIV RANDOM EXERCISER

3. LOADING PROCEDURE

USE THE STANDARD PROCEDURE FOR ABSOLUTE TAPES, OR LOAD VIA XXDP MEDIA.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE SECTION 5.1
SWITCH REGISTER (000000) IS WORST CASE TEST.

4.2 STARTING ADDRESS

THE PROGRAM MUST ALWAYS BE STARTED AT LOCATION 200(8).

4.3 PROGRAM/OPERATOR ACTION

LOADING VIA ABSOLUTE PAPERTAPE:

- (1) LOAD PROGRAM INTO MEMORY USING ABS LOADER.
- (2) LOAD ADDRESS 200 (8).
- (3) SET SWITCHES (SEE SECTION 5.1)
SR=(000000) IS WORST CASE TEST.
- (4) PRESS CONTROL/START TO BEGIN.
- (5) PROGRAM TYPES IDENTIFICATION HEADER (VERIFY THAT THE CORRECT PROGRAM HAS BEEN LOADED!), AND EXECUTION BEGINS.

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

THE DEFINITION OF THE SPECIFIC BITS IN THE SWITCH REGISTER (EITHER HARDWARE OR SOFTWARE) ARE AS FOLLOWS:

- | | | |
|--------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SW15=1 | 100000 | HALT ON ERROR |
| SW14=1 | 040000 | LOOP ON CURRENTLY EXECUTING TEST |
| SW13=1 | 020000 | INHIBIT ERROR TIMEOUTS (WHICH IS AN "ERROR MESSAGE" RESULTING FROM AN ERROR DETECTED IN THE HARDWARE) |
| SW12=1 | 010000 | INHIBIT STATUS TIMEOUTS (WHICH IS A NON-ERROR RELATED INFORMATIVE MESSAGE, SUCH AS "END PASS #XXX") |
| SW11=1 | 004000 | INHIBIT ITERATIONS PER TEST |
| SW10 | 002000 | SET=BELL ON ERROR/CLEAR=BELL ON PASS END |
| SW09=1 | 001000 | LOOP ON ERROR |
| SW08=1 | 000400 | LOOP ON TEST NUMBER IN "SLPTST" IF SET, THEN THE TEST SPECIFIED BY THE TEST NUMBER CONTAINED IN THE MEMORY WORD "SLPTST" (SEE PROGRAM LISTING) WILL SPECIFY THE DESIRED TEST ON WHICH TO LOOP. |
| SW01 | 000002 | CLEAR=TEST HOT-FP/WARM-FP ALTERNATELY EACH PASS (IE, PASS#1 HFP, PASS#1 WFP, PASS#2 HFP, PASS#2 WFP, ETC) |
| SW00 | 000001 | SET=TEST ONLY UNIT SPECIFIED IN SW00
SET=SELECT WARM FP, IF SW01=1 |

CLEAR=SELECT HOT FP, IF SW01=1

NOTE FOR SW01, SW00 - IF NO HOT FP (FP11-E) IS PRESENT, THEN WARM FP (PDP-11/6X MICROCODE) IS AUTOMATICALLY SELECTED.

5.2 PROGRAM/OPERATOR ACTION

ONCE EXECUTION HAS BEGUN, MINIMAL OPERATOR INTERVENTION IS REQUIRED, UNLESS THE PROGRAM DETECTS AN ERROR IN THE HARDWARE.

IF ALL IS WELL, THE PROGRAM TYPES ITS NAME UPON BEGINNING; AND AT THE START OF EACH PASS, THE CURRENT PASS NUMBER (IN OCTAL) IS ECHOED. NOTE THAT SETTING SW<12>=1 WILL INHIBIT THE TYPEOUT OF THE BEGIN AND END PASS MESSAGES.

IF SW<10>=0, THE CONSOLE BELL WILL BE RUNG AT THE END OF EACH PASS. NOTE THAT ONLY SW<10> AFFECTS THE BELL RINGING AT END OF PASS - SW<12> HAS NO EFFECT ON THIS FUNCTION.

IF AN ERROR OCCURS DURING EXECUTION, MANY VARIATIONS IN ACTION ARE POSSIBLE DEPENDING UPON THE SWITCH SETTINGS.

SW<15>=1 WILL CAUSE THE CPU TO HALT AFTER AN ERROR.

SW<13>=1 WILL ALSO INHIBIT ANY ERROR MESSAGE TYPEOUT THAT WOULD OCCUR AT THIS TIME.

SW<10>=1 WILL CAUSE THE CONSOLE BELL TO BE RUNG ONLY WHEN AN ERROR IS DETECTED (AND NOT AT THE END OF A PASS).

SW<9>=1 CAUSES THE PROGRAM TO LOOP ON THE MOST RECENT ERROR, AS LONG AS IT CONTINUES TO OCCUR.

THERE ARE ALSO SEVERAL OTHER GENERAL USE FUNCTIONS DEFINED BY THE SWITCHES:

SW<11>=1 WILL INHIBIT THE ITERATIONS (=2000(10)) PERFORMED OF EACH TEST ON PASSES 2, 3, 4 THRU THE PROGRAM.

SW<14>=1 CAUSES THE PROGRAM TO LOOP INDEFINATELY ON THE CURRENTLY EXECUTING TEST.

SW<8>=1 CAUSES THE PROGRAM TO CONTINUE EXECUTION AS NORMAL, EXCEPT WHEN THE CONTENTS OF MEMORY WORD "SLPTST" MATCHES THE NUMBER OF THE TEST CURRENTLY EXECUTING. AT THIS POINT, THE TEST IS LOOPED ON INDEFINATELY, UNTIL EITHER SW<8>=0 OR "SLPTST" IS CHANGED. NOTE THAT IF "SLPTST" DOES NOT MATCH THE TEST NUMBER OF ANY TEST, THE CONTENTS OF "SLPTST" ARE EFFECTIVELY IGNORED, AND EXECUTION PROCEEDS NORMALLY.

5.3 HOT (FP11-E) / WARM (PDP-11/6X) SELECTION

WHEN THE PROGRAM IS STARTED (AT 200(8)), A MESSAGE IS OPTIONALLY PRINTED INDICATING THE PRESCENCE/ABSCENCE OF AN FP11-E HOT FLOATING POINT UNIT OPTION (BASED UPON WHETHER "WHAMI" BIT<04> IS 1/0 RESPECTIVELY).

IF NO FP11-E HOT FP OPTION IS PRESENT, THE MESSAGE IS TYPED,

AND ANY ATTEMPTS TO SELECT IT FOR TESTING VIA SW01 AND SW00 ARE IGNORED. ONLY WARM FP (PDP-11/6X MICROCODE) FLOATING POINT CAN BE TESTED/SELECTED.

IF THE FP11-E IS PRESENT, TEST SELECTION IS AS FOLLOWS:

WHEN SW01=0, THE HOT AND WARM FLOATING POINT UNITS ARE TESTED ALTERNATELY EACH PASS - IN THE ORDER (1) HOT, THEN (2) WARM. NOTE THAT EACH "PASS" NOW CONSISTS OF TWO SEPARATE SUB-PASSES.

WHEN SW01=1, THEN DEDICATED SELECTION OF A PARTICULAR UNIT IS SPECIFIED IN SW00:

SW00=0 --> TEST WFP FP11-E OPTION ONLY
SW00=1 --> TEST WFP PDP-11/6X MICROCODE ONLY

5. ERRORS

6.1 FORMAT OF MESSAGES

6.1.1 ALL ERROR MESSAGES CONSIST OF THREE LINES OF DATA:

THE FIRST LINE IS A BRIEF MESSAGE WHICH EXPLAINS WHAT ERROR WAS DETECTED (EG, THE RESULT OF THE "ABSF" INSTRUCTION WAS BAD).

THE PREFIX "HOT:" OR "WARM:" IS ALSO ATTACHED TO THE MESSAGE TO INDICATE THE SOURCE OF THE ERROR; THE FP11-E UNIT OR THE PDP-11/6X RESPECTIVELY.

THE SECOND LINE CONSISTS OF DATA HEADERS TO IDENTIFY THE VALUES TYPED OUT ON LINE THREE. THESE HEADERS WILL EITHER BE OF THE FORM "EXPECTED" AND "RECEIVED" DATA, OR WILL BE A MNEMONIC NAME OF A WORD LOCATION IN MEMORY OR REGISTERS.

THE THIRD LINE DISPLAYS THE CONTENTS OF THE LOCATIONS SPECIFIED BY LINE TWO AS SIX DIGIT OCTAL NUMBERS. NOTE THAT ALL DATA DISPLAYED IN ANY MESSAGES ARE OCTAL NUMBERS.

AS EXPLAINED IN SECTION 5.2, SETTING SW<13>=1 WILL SUPPRESS THE TYPING OF THESE MESSAGES.

6.1.2 FLOATING POINT UNIT DATA FORMATS:

FLOATING POINT STATUS WORD (FPS):

BIT#	OCTAL	FUNCTION
15	100000	FER - FLOATING ERROR FLAG SET WHEN EITHER FIUV, FIU, FIV, FIC ENABLED AND APPROPRIATE EXCEPTION OCCURRED.
14	040000	FID - FLOATING DISABLE INTERRUPTS NO FP INTERRUPTS TO VECTOR 244(8) IF SET.
13, 12		NOT USED
11	004000	FIUV - FLOATING UNDEFINED VARIABLE INTERRUPT IF SET, (-0) MEMORY DATA IS ERROR
10	002000	FIU - FLOATING INTR UNDERFLOW IF SET AND UNDERFLOW, SET FER, STORE ANSWER, EXPONENT WRONG BY +400(8) IF CLEAR AND UNDERFLOW, ANSWER <-- ZERO
9	001000	FIV - FLOATING OVERFLOW INTERRUPT IF SET AND OVERFLOW, SET FER, STORE ANSWER, EXPONENT WRONG BY -400(8) IF CLEAR AND OVERFLOW, ANSWER <-- ZERO
8	000400	FIC - FLOATING INTEGER CONVERSION INTERRUPT IF SET AND "STCFI" ERROR, ANSWER <-- ZERO, SET ERROR IF CLEAR AND "STCFI" ERROR, ANSWER <-- ZERO
7	000200	FD - FLOATING MODE 1=DOUBLE, 64 BIT OPERANDS (4W) 0=SINGLE, 32 BIT OPERANDS (2W)
6	000100	FL - INTEGER MODE 1=LONG, 32 BIT INTEGERS (2W) 0=SHORT, 16 BIT INTEGERS (1W)
5	000040	FT - ROUND/TRUNCATE MODE 1=TRUNCATE RESULTS 0=ROUND RESULTS
4	000020	FMM - PUT FP11-E ONLY IN MAINTENANCE MODE
3:0	000017	FN-FZ-FV-FC - FLOATING CONDITION CODES

FLOATING EXCEPTION CODES (FEC):

OCTAL	ENABLE	FUNCTION
00	(NONE)	(NOT USED)
02	(NONE)	FP OPCODE ERROR
04	(NONE)	FP DIVIDE-BY-ZERO ERROR
06	W/FIC	FP INTEGER CONVERSION ERROR
10	W/FIV	FP OVERFLOW ERROR
12	W/FIU	FP UNDERFLOW ERROR
14	W/FIUV	FP UNDEFINED-VARIABLE/(-0) ERROR
16	W/FMM	FP MAINTENANCE TRAP

NOTE - IN "FEC" CODE TYPEOUTS IN ERROR MESSAGES ONLY THE LOW ORDER BYTE IS USED - IGNORE THE PROGRAM FLAG BIT IN THE UPPER BYTE.

FLOATING POINT DATA:

IN FLOAT MODE (FD=0), IS 2-16. BIT WORDS, 32. BITS
 IN DOUBLE MODE (FD=1), IS 4-16. BIT WORDS, 64. BITS

FIRST WORD: (BOTH F, D MODES)

B15=SIGN OF NUMBER (1/-, 0/+)
 B14:07=EXPONENT, 8.BITS, FROM -128./+127.
 B06:00=FRACTION, 7.BITS

SECOND WORD: (BOTH F, D MODES)

B15:00=FRACTION, 16.BITS

THIRD, FOURTH WORDS: (ONLY D MODE)

B15:00, B15:00=FRACTION, 32. BITS

IN F MODE, THE COMPOSITE 24. BIT FRACTION
 IS FORMED BY:

.1#(WORD1-BIT<06:00>)#(WORD2-BIT<15:00>)

IN D MODE, THE COMPOSITE 56. BIT FRACTION
 IS FORMED BY:

.1#(WORD1-BIT<06:00>)#(WORD2-BIT<15:00>)
 #[(WORD3-BIT<15:00>)#(WORD4-BIT<15:00>)]

FOR A MORE DETAILED OPERATION/EXPLANATION OF FLOATING POINT
 DATA FORMATS AND OPERATIONS, SEE THE PDP-11/6X PROCESSOR
 HANDBOOK SECTION ON THE FLOATING POINT INSTRUCTION SET.

6.2 RECOVERY

RECOVERY FROM ERRORS HAS BEEN ATTEMPTED TO BE MADE AS
 AUTOMATIC AND EFFORTLESS AS POSSIBLE. HOWEVER, IN MANY CASES,
 DUE TO THE NATURE OF THE ERROR, THE PROGRAM MAY NOT EVEN BE
 ABLE TO BE RUN (EG, IF THE FLOATING POINT MODULE IS IN A HUNG
 STATE, AND CAN NEVER ENTER THE READY STATE TO ACCEPT A NEW FPP
 INSTRUCTION). AT THIS POINT, SOLVING THE PROBLEM IS A DIRECT
 FUNCTION OF THE OPERATORS INGENUITY. THIS TEST SERIES HAS
 BEEN DESIGNED TO TEST THE FLOATING POINT PROCESSOR SO THAT
 THESE TYPES OF FAILURES TO RUN WILL BE MINIMAL. THE TESTS
 HAVE BEEN PLACED IN A SPECIFICALLY STRUCTURED SEQUENCE IN THE
 PROGRAM TO IMPLEMENT THIS STRATEGY: TESTING THE MOST BASIC
 ELEMENTS FIRST, PROCEEDING UPWARD IN COMPLEXITY AFTER
 ESTABLISHING THEIR CORRECT OPERATION. THIS IS WHY IT IS
 EXTREMELY IMPORTANT THAT THE FLOATING POINT TEST PROGRAMS BE
 (1) RUN IN THE PRESCRIBED ORDER, AND (2) ONLY BE STARTED AT
 THEIR BEGINNING ADDRESS (USUALLY 200(8)). THE PROGRAM WILL
 DISPLAY, AT AN ERROR, THE MOST PERTINENT INFORMATION RELATING
 TO THE ERROR, AND A BRIEF EXPLANATION OF THE FAILING FUNCTION.

6.3 CAUSES

THESE TEST PROGRAMS ARE NOT HARDWARE ORIENTED, AND AS SUCH IT IS NOT POSSIBLE TO CALL OUT PARTICULAR HARDWARE AREAS AND MODULES RELATING TO A GIVEN FUNCTIONAL FAILURE. HARDWARE DIAGNOSIS FOR A PARTICULAR MACHINE MUST BE DONE USING THE APPROPRIATE ENGINEERING ROM FLOWS AND PRINTS, ALONG WITH THE KNOWN FUNCTIONAL ERRORS (AS DETECTED BY THE PROGRAMS). THIS IS THE INTENT UNDER WHICH THESE INSTRUCTION TESTS WERE DESIGNED AND CODED.

7. RESTRICTIONS

7.1 STARTING

THE PROGRAM MUST BE STARTED AT LOCATION 200(8) ALWAYS.

7.2 OPERATIONAL

THERE ARE NO OPERATIONAL RESTRICTIONS.

8. MISCELLANEOUS

8.1 EXECUTION TIME

MODEL	AVERAGE EXECUTION TIME PER PASS	
	SHORTEST PASS	LONGEST PASS
PDP-11/6X MICROCODE	0:20	9:30
PDP-11/6X W/FP11-E	0:15	5:15

TIMES SPECIFIED AS (M. WTES):(SECONDS)

SHORTEST PASS ::= PASS=1, NO ITERATIONS, USING:
SWR=(004003) FOR PDP-11/6X MICROCODE
SWR=(004002) FOR PDP-11/6X W/FP11-E

LONGEST PASS ::= PASS=2, 2000. ITERATIONS/TEST, USING:
SWR=(000003) FOR PDP-11/6X MICROCODE
SWR=(000002) FOR PDP-11/6X W/FP11-E

8.2 STACK POINTER

THE STACK POINTER IS SET TO 1100(8) AT THE START OF EACH PASS. IF ALL IS OPERATING CORRECTLY, IT SHOULD ALSO BE THIS VALUE AT

THE START OF EACH TEST, AND AT THE END OF A PASS.

8.3 POWER FAIL

THE TESTS MAY BE POWER FAILED AT ANY TIME. SPURIOUS ERROR MESSAGES MAY OCCUR IF THE FAILURE OCCURRED WHILE THE F.P.U. WAS EXECUTING A FUNCTION, AS NONE OF ITS REGISTERS (FPS, FEC, FEA, ACCUMULATORS) ARE SAVED IN THE EVENT OF A POWER FAILURE. HOWEVER, THESE MESSAGES SHOULD ONLY OCCUR ONCE (IF AT ALL) IMMEDIATELY AFTER POWER IS RESTORED. WHEN POWER IS RESTORED, "POWER" IS TYPED ON THE CONSOLE AND EXECUTION CONTINUES WHERE IT WAS INTERRUPTED.

NOTE THAT THE "VOLATILE" SWITCH REGISTER CONTENTS ARE SAVED AND RESTORED FROM THE STACK IN A POWER FAIL SEQUENCE; THEREFORE THE SWITCH REGISTER SETTINGS SHOULD NOT BE LOST OVER A POWER FAIL.

9. PROGRAM DESCRIPTION

9.1 ORGANIZATION

THESE PROGRAMS ARE ORGANIZED AS MUCH AS POSSIBLE IN A STRAIGHTFORWARD, LINEAR MANNER. THE MAIN BODY OF CODE IS STRUCTURED AS FOLLOWS:

- (1) INITIALIZATION ROUTINE
 - SETS UP VECTORS, TYPES HEADER, ETC.
- (2) MAIN BODY OF TESTS
 - INLINE TEST CODE, INLINE TEST CALLS
- (3) END OF PASS ROUTINE
 - END OF PASS PROCESSING
- (4) TEST SUBROUTINES
 - SUBROUTINES CONTAINING COMMON TEST CODE
- (5) OVERHEAD ROUTINES
 - SERVICE SUBROUTINES (TYPEOUT, ETC.)

WHEREVER FEASIBLE, COMMON SECTIONS OF CODE FOR WIDELY USED FUNCTIONS ARE CONDENSED INTO SUBROUTINES TO CONSERVE MEMORY. THIS INCLUDES NOT ONLY STANDARD SERVICE ROUTINES (SUCH AS SCOPE, ERROR, AND ASCII TYPEOUT), BUT ALSO TESTING ROUTINES WHICH PERFORM VERY SIMILAR FUNCTIONS. THUS IN MANY CASES (THE "ADDF" INSTRUCTION TESTING, FOR EXAMPLE) A SINGLE BODY OF CODE (A SUBROUTINE) IS USED TO PERFORM ALL THE FUNCTIONAL TESTS, WITH A VARIABLE PARAMETER LIST PASSED AT EACH CALL CONTAINING THE DATA OPERANDS AND EXPECTED RESULT FOR EACH INDIVIDUAL TEST. THIS CONSTRUCTION FACILITATES THE ADDITION/DELETION OF TESTS (SHOULD THAT EVER BE NECESSARY), AND ALSO GREATLY CONSERVES MEMORY SPACE REQUIREMENTS WHEN A LARGE NUMBER OF CALLS TO A GIVEN BODY OF CODE ARE REQUIRED.

THE INDIVIDUAL TESTS WITHIN EACH PROGRAM HAVE ALSO BEEN SEQUENCED IN A PARTICULAR ORDER TO FACILITATE THE DETECTION AND RESOLUTION OF ERRORS AS QUICKLY AS POSSIBLE. EACH OF THE TESTS BEGINS AS SIMPLY AS POSSIBLE, FIRST TESTING THE MOST BASIC ELEMENTS. MORE COMPLEX ELEMENTS ARE TESTED AFTERWARDS, EMPLOYING A PHILOSOPHY THAT THE SIMPLER THE TEST, THE BETTER THE RESOLUTION. ALL FUNCTIONS ARE EVENTUALLY TESTED, BUT HOPEFULLY MOST ERRORS WILL BE CAUGHT AND CORRECTED EARLY. A MUCH MORE DETAILED ANALYSIS OF THE SEQUENCE OF TESTS PERFORMED IS PRESENTED IN SECTION 9.2.

9.2 TEST DESCRIPTION

THIS DIAGNOSTIC CONTAINS TESTS FOR THE FLOATING POINT 'ADD-', 'SUB-', 'MUL-', AND 'DIV-' INSTRUCTIONS. ALL COMBINATIONS OF THE SINGLE/DOUBLE, ROUND/TRUNCATE, AND OVERFLOW-UNDERFLOW INTERRUPTS ENABLED/DISABLED MODES ARE EMPLOYED. EACH TEST GENERATES A PAIR OF RANDOM NUMBER OPERANDS, THEN USES BOTH THE HARDWARE AND SOFTWARE ROUTINES TO GENERATE AN ANSWER: EACH SHOULD GENERATE THE SAME ANSWER (WITH A +/- 1 DEVIATION IN THE 'LSB' ALLOWED). FLOATING POINT 'LD-', 'ST-', 'CMP-', AND STATUS INSTRUCTIONS ARE ALSO USED FOR MANIPULATING THE OPERANDS AND RESULTS.

THE PURPOSE OF THESE TESTS IS TO EXERCISE BOTH THE DATA PATH AND CONTROL PORTIONS OF THE FLOATING POINT UNIT SELECTED FOR TESTING WITH AN 'UNLIMITED' SUPPLY OF VARYING OPERANDS, AS MIGHT BE ENCOUNTERED IN A USER/APPLICATION PROGRAM TYPE ENVIRONMENT.

9.3 SUBROUTINE ABSTRACTS

9.3.1 TRAPCATCHER

THE TRAPCATCHER IS A SERIES OF INSTRUCTIONS OCCUPYING THE INTERRUPT VECTOR AREA OF MEMORY. IT CONSISTS OF THE SEQUENCE:

```
.WORD    +2    ;PC AFTER TRAP
.WORD    0     ;PS AFTER TRAP
```

PLACED AT EACH VECTOR ADDRESS IN LOCATIONS 4-776(8) OF MEMORY. THE FIRST WORD OF EACH PAIR ("PC AFTER TRAP") POINTS TO THE SECOND WORD, WHICH SERVES A DUAL PURPOSE AS
(1) THE NEW LOADED PS (ALL ZEROS), AND (2) THE NEXT INSTRUCTION TO EXECUTE (0=HALT).

WHEN THE PROGRAM IS EXECUTING, ANY REQUIRED VECTORS ARE SET UP IN THE VECTOR AREA WITH APPROPRIATE VALUES; THE OTHERS BEING LEFT IN THE "TRAPCATCHER" STATE. THUS, IF AN UNEXPECTED TRAP EVER OCCURS IN THE MACHINE, IT WILL BE CAUGHT, AND THE MACHINE SUBSEQUENTLY HALTED, DISPLAYING THE VECTOR ADDRESS * PLUS FOUR

* IN THE ADDRESS LIGHTS.

9.3.2 SCOPE ROUTINE - \$SCOPE

THE SCOPE ROUTINE IS ENTERED FROM THE FIRST INSTRUCTION OF EACH TEST IN THE PROGRAM. (NOTE THAT BY DEFINITION, A "TEST" WILL BE DESIGNATED AS THE SECTION OF CODE BETWEEN TWO "SCOPE" STATEMENTS.) THIS ROUTINE PROVIDES THE OVERHEAD CODE NECESSARY TO IMPLEMENT SEVERAL OF THE SWITCH REGISTER CONTROL OPTIONS. UPON ENTRANCE TO A TEST, THE SCOPE STATEMENT AT THE BEGINNING SETS UP CERTAIN LOCATIONS (SEE BELOW) TO SPECIFY THE CURRENT TEST NUMBER AND LOOPING ADDRESS (FOR ITERATIONS). CONTROL IS THEN PASSED TO THE ACTUAL TEST CODE, PERFORMING THE DESIRED TEST. UPON EXIT, THE SCOPE STATEMENT OF THE NEXT TEST IS ENTERED, WHICH DETERMINES WHETHER TO (1) LOOP BACK TO THE PREVIOUS TEST (EG, FOR ITERATIONS) OR (2) INITIALIZE FOR THE NEXT TEST (AS DESCRIBED EARLIER, ABOVE).

ENTRANCE TO THE SCOPE ROUTINE IS VIA AN "IOT" TRAP CALL THROUGH LOCATION 20(8). (FROM THE SCOPE=IOT EQUATE). DEPENDING UPON THE SWITCH SETTINGS (SEE 5.2), CODE IS PRESENT TO: LOAD THE FPII MICRO BREAK REGISTER, LOOP ON THE CURRENTLY EXECUTING TEST, LOOP ON A SPECIFIC TEST, PERFORM ITERATIONS OF EACH TEST, AND SET UP ADDRESSES FOR POSSIBLE LOOPING ON ERRORS. IMPORTANT VALUES USED IN THIS ROUTINE ARE:

- \$MXCNT - MAXIMUM NUMBER OF ITERATIONS PER TEST (GENERALLY WILL BE 2000(10))
- \$STNM - A COUNTER INDICATING THE NUMBER (1-377(8)) OF THE TEST CURRENTLY BEING EXECUTED
- \$LPADR - CONTAINS THE ADDRESS TO WHICH THE SCOPE ROUTINE WILL LOOP, IF THE CURRENT TEST IS BEING LOOPEd UPON
- \$LPERR - CONTAINS THE ADDRESS TO WHICH THE ERROR ROUTINE (SEE 9.3.3) WILL LOOP, IF AN ERROR OCCURS AND THE LOOPING ON AN ERROR OPTION IS SPECIFIED IN THE SWITCHES. SET UP BY SCOPE, GENERALLY WILL BE THE SAME AS \$LPADR, ABOVE.

9.3.3 ERROR ROUTINE - \$ERROR

THE ERROR ROUTINE IS ENTERED WHEN THE TEST CODE HAS DETERMINED THAT AN ERROR HAS OCCURRED AS PART OF A TEST. THROUGH USE OF THIS ROUTINE, THE TEST HAS A MEANS OF SIGNALING AN ERROR TO THE 10520 OPERATOR/MONITOR; AND IMPLEMENTING THE CONTROL FUNCTIONS FOR HALTING ON ERROR, BELL ON ERROR, AND LOOPING ON ERROR. IN ADDITION, THE ERROR ROUTINE HAS THE PROVISION TO TYPE OUT ON THE OPERATOR'S CONSOLE A MESSAGE BRIEFLY EXPLAINING THE ERROR, AND SOME OF THE MOST PERTINENT DATA VALUES TO HELP DIAGNOSE THE CAUSE (SEE SECTION 6.2).

THE CALLING MECHANISM IS SIMILAR TO THAT EMPLOYED FOR THE SCOPE ROUTINE (VIA A TRAP), EXCEPT IN THIS INSTANCE, THE "EMT"

INSTRUCTION IS USED, TRAPPING THROUGH LOCATION 30(8). (NOTE THE EQUATE ERROR N=EMT N). THE LOWER BYTE OF THE EMT INSTRUCTION IS CAPABLE OF TRANSMITTING A NUMBER FROM 0-377(8) WHICH WILL BE TERMED THE "ERROR ITEM NUMBER." THIS NUMBER DETERMINES WHICH ERROR MESSAGE, AND ASSOCIATED DATA VALUES WILL BE TYPED OUT WHEN A PARTICULAR ERROR IS SIGNALLED. IF THIS NUMBER IS ZERO, JUST THE PC OF THE CALLING "ERROR" INSTRUCTION WILL BE TYPED, OTHERWISE, THE NUMBER IS USED AS AN INDEX THROUGH THE ERROR TABLE (SERRTB) TO FIND THE APPROPRIATE VALUES TO TYPE (SEE PROGRAM LISTING FOR FURTHER DETAILS).

IMPORTANT VALUES USED IN THIS ROUTINE ARE:

- EREG0 THRU EREG7 - CONTENTS OF GENERAL REGISTERS R0 THRU R7 JUST BEFORE ERROR CALL
- SERTTL - CUMULATIVE NUMBER OF ERRORS ENCOUNTERED TO DATE
- SERRPC - CONTAINS THE PC OF THE "ERROR" INSTRUCTION JUST EXECUTED
- SLPERR - CONTAINS THE ADDRESS WHICH WILL BE LOOPED UPON FOR THE ERROR LOOPING FACILITY

9.3.4 ERROR MESSAGE TIMEOUT ROUTINE - \$STYERR

THIS ROUTINE (\$STYERR ENTRY POINT) IS CALLED BY THE ERROR PROCESSING ROUTINE DESCRIBED IN 9.3.3 ABOVE. ITS PURPOSE IS TO IMPLEMENT THE ERROR MESSAGE/DATA VALUE ERROR TIMEOUT FACILITY. THE SUBROUTINE WILL, GIVEN THE INDEXING BYTE FROM THE ERROR CALL INSTRUCTION, PICK UP THE CORRECT ERROR MESSAGE VECTOR FROM SERRTB (ERROR TABLE), AND TYPE OUT THE ERROR MESSAGE, DATA HEADER, AND DATA VALUES ON THE CONSOLE.

9.3.5 TYPE ROUTINE - \$STYPE

THIS ROUTINE IS THE STANDARD SYSTEM TIMEOUT ROUTINE FOR ASCII SINGLE-CHARACTER-PER-BYTE STRINGS. IT IS CALLED THROUGH A TRAP INSTRUCTION WITH THE NEXT WORD CONTAINING THE ADDRESS OF THE FIRST CHARACTER IN THE STRING. TYPING TERMINATES WHEN AN ALL-ZERO BYTE IS FOUND. HORIZONTAL TAB STOPS ARE ALSO AUTOMATICALLY PLACED.

9.3.6 OCTAL NUMBER TYPE ROUTINE - \$STYPOC

THIS ROUTINE CONVERTS THE TOP NUMBER ON THE STACK TO A 6-DIGIT OCTAL REPRESENTATION, AND TYPES IT ON THE CONSOLE USING THE TYPE ROUTINE \$STYPE. SEE LISTING FOR OPTIONS AND FURTHER DETAILS.

9.3.7 POWER UP AND DOWN ROUTINES - \$SPWRUP AND \$SPWRDN

THESE TWO ROUTINES ARE ENTERED FOR THE POWER UP AND DOWN CONDITIONS, RESPECTIVELY. THE POWER DOWN ROUTINE (\$SPWRDN) SAVES THE GENERAL REGISTERS AND STACK POINTER. THE POWER UP

MAINDEC-11-DGFPD-B

ROUTINE (SPWRUP) CORRESPONDINGLY RESTORES THE REGISTERS, STACK POINTER, AND TYPES THE MESSAGE "POWER" WHEN POWER IS RESTORED. THE VOLATILE INTERNAL SWITCH REGISTER IS ALSO SAVED/RESTORED BY THIS ROUTINE.

9.3.8 END OF PASS ROUTINE - SEOP

THE END OF PASS ROUTINE COUNTS THE NUMBER OF PASSES PERFORMED, DINGS THE BELL/TYPES A MESSAGE (IF ENABLED), SETS/CLEARs THE T-BIT (IF ENABLED), AND ALSO INTERFACES TO THE MONITOR, IF PRESENT. IT ALSO OPTIONALLY LOOPS FOR A NUMBER OF SUBPASSES BEFORE SIGNALLING AN END OF PASS CONDITION.

10. ACT/APT/XXDP

10.1 ACT COMPATIBILITY

THIS PROGRAM WILL RUN UNDER THE ACT SYSTEM.

10.2 APT COMPATIBILITY

THIS PROGRAM WILL RUN UNDER THE APT SYSTEM MONITOR. ALL NECESSARY SOFTWARE COMMUNICATION HOOKS ARE PRESENT.

10.3 XXDP COMPATIBILITY

FOR XXDP MEDIA COMPATIBILITY, THE TOP 2K WORDS OF THE 16K WORD MINIMUM MEMORY AREA ARE NOT DISTURBED DURING EXECUTION.

13	OPERATIONAL SWITCH SETTINGS
31	BASIC DEFINITIONS
163	TRAP CATCHER
172	STARTING ADDRESS(ES)
175	ACT11 HOOKS
186	APT PARAMETER BLOCK
209	COMMON TAGS
256	APT MAILBOX-ETABLE
283	ERROR POINTER TABLE
429	PROGRAM DEFINED COMMON TAGS
514	START OF PASS ROUTINE
522	INITIALIZE THE COMMON TAGS
656	T1 EXERCISE ADD, ALL INTERRUPTS ON, ROUNDING MODE
723	T2 EXERCISE ADD, ALL INTERRUPTS ON, ROUNDING MODE
793	T3 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
849	T4 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
907	T5 EXERCISE ADD, ALL INTERRUPTS ON, TRUNCATE MODE
981	T6 EXERCISE ADD, ALL INTERRUPTS ON, TRUNCATE MODE
1059	T7 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
1120	T10 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
1182	T11 EXERCISE SUB, ALL INTERRUPTS ON, ROUNDING MODE
1250	T12 EXERCISE SUB, ALL INTERRUPTS ON, ROUNDING MODE
1320	T13 EXERCISE SUB, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
1376	T14 EXERCISE SUB, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
1434	T15 EXERCISE SUB, ALL INTERRUPTS ON, TRUNCATE MODE
1507	T16 EXERCISE SUB, ALL INTERRUPTS ON, TRUNCATE MODE
1584	T17 EXERCISE SUB, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
1645	T20 EXERCISE SUB, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
1707	T21 EXERCISE MUL, ALL INTERRUPTS ON, ROUNDING MODE
1775	T22 EXERCISE MUL, ALL INTERRUPTS ON, ROUNDING MODE
1845	T23 EXERCISE MUL, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
1901	T24 EXERCISE MUL, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
1958	T25 EXERCISE MUL, ALL INTERRUPTS ON, TRUNCATE MODE
2031	T26 EXERCISE MUL, ALL INTERRUPTS ON, TRUNCATE MODE
2108	T27 EXERCISE MUL, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
2169	T30 EXERCISE MUL, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
2231	T31 EXERCISE DIV, ALL INTERRUPTS ON, ROUNDING MODE
2299	T32 EXERCISE DIV, ALL INTERRUPTS ON, ROUNDING MODE
2369	T33 EXERCISE DIV, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
2437	T34 EXERCISE DIV, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
2507	T35 EXERCISE DIV, ALL INTERRUPTS ON, TRUNCATE MODE
2580	T36 EXERCISE DIV, ALL INTERRUPTS ON, TRUNCATE MODE
2657	T37 EXERCISE DIV, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
2730	T40 EXERCISE DIV, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
2807	T41 EXERCISE DIV, INTERRUPT DISABLE SET, ROUNDING MODE
2874	T42 EXERCISE DIV, INTERRUPT DISABLE SET, ROUNDING MODE
2944	T43 EXERCISE DIV, INTERRUPT DISABLE SET, TRUNCATE MODE
3016	T44 EXERCISE DIV, INTERRUPT DISABLE SET, TRUNCATE MODE
3089	T45 ADD, SUB, MUL, DIV EXERCISER
3178	T46 ADD, SUB, MUL, DIV EXERCISER
3281	SUB PASS END CONTROL
3321	END OF PASS ROUTINE (MODIFIED SYMAC)
3369	STATISTICS TIMEOUT SUBROUTINE
3511	FPP TRAP CATCHER
3529	RANDOM NUMBER GENERATOR
3584	POLISH EXPRESSION ROUTINES

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18
D8FP08.P11 04-MAY-77 17:30 TABLE OF CONTENTS

SEQ 0002

3649	FLOATING POINT SOFTWARE ROUTINES
4530	SCOPE HANDLER ROUTINE
4596	ERROR HANDLER ROUTINE
4659	ERROR MESSAGE TIMEOUT ROUTINE (MODIFIED SYSMAC)
4744	TYPE ROUTINE
4823	APT COMMUNICATIONS ROUTINE
4880	BINARY TO OCTAL (ASCII) AND TYPE
4957	TRAP DECODER
4980	TRAP TABLE
4994	POWER DOWN AND UP ROUTINES
5041	ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```

.TITLE FPU ADD/SUB/MUL/DIV RANDOM EXER
*COPYRIGHT (C) 1976
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*PROGRAM BY DONALD NORTH
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
*
.SBTTL OPERATIONAL SWITCH SETTINGS
*
*      SWITCH  OCTAL      USE
*      -----  -----  -----
*      15      100000     HALT ON ERROR
*      14      040000     LOOP ON CURRENTLY EXECUTING TEST
*      13      020000     INHIBIT ERROR TYPEOUTS
*      12      010000     INHIBIT STATUS TYPEOUTS
*      11      004000     INHIBIT ITERATIONS
*      10      000000     0=BELL ON PASS END
*                       1=BELL ON ERROR
*      9       001000     LOOP ON ERROR
*      8       000400     LOOP ON TEST NUMBER IN "SLPTST"
*      7       000000     0=TEST WFP/WFP ALTERNATELY EACH PASS
*                       1=TEST ONLY UNIT SPECIFIED IN SW<00>
*      6       000002     0=SELECT WFP, IF SW<01>=1
*                       1=SELECT WFP, IF SW<01>=1
*      5       000002
*      4       000003
*
.SBTTL BASIC DEFINITIONS
*
*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK= 1100
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
*
*
.MISCELLANEOUS DEFINITIONS
000011 HT= 11      ;;CODE FOR HORIZONTAL TAB
000012 LF= 12      ;;CODE FOR LINE FEED
000015 CR= 15      ;;CODE FOR CARRIAGE RETURN
000200 CRLF= 200    ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776 PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
177774 STKLM= 177774 ;;STACK LIMIT REGISTER
177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
*
*
*GENERAL PURPOSE REGISTER DEFINITIONS
000000 R0= %0      ;;GENERAL REGISTER
000001 R1= %1      ;;GENERAL REGISTER
000002 R2= %2      ;;GENERAL REGISTER
000003 R3= %3      ;;GENERAL REGISTER
000004 R4= %4      ;;GENERAL REGISTER
000005 R5= %5      ;;GENERAL REGISTER
000006 R6= %6      ;;GENERAL REGISTER

```

```

57      000007      R7=      %7      ;; GENERAL REGISTER
58      000006      SP=      %6      ;; STACK POINTER
59      000007      PC=      %7      ;; PROGRAM COUNTER
60
61      ;*PRIORITY LEVEL DEFINITIONS
62      000000      PR0=      0      ;; PRIORITY LEVEL 0
63      000040      PR1=      40     ;; PRIORITY LEVEL 1
64      000100      PR2=      100    ;; PRIORITY LEVEL 2
65      000140      PR3=      140    ;; PRIORITY LEVEL 3
66      000200      PR4=      200    ;; PRIORITY LEVEL 4
67      000240      PR5=      240    ;; PRIORITY LEVEL 5
68      000300      PR6=      300    ;; PRIORITY LEVEL 6
69      000340      PR7=      340    ;; PRIORITY LEVEL 7
70
71      ;*"SWITCH REGISTER" SWITCH DEFINITIONS
72      100000      SW15=     100000
73      040000      SW14=     40000
74      020000      SW13=     20000
75      010000      SW12=     10000
76      004000      SW11=     4000
77      002000      SW10=     2000
78      001000      SW09=     1000
79      000400      SW08=     400
80      000200      SW07=     200
81      000100      SW06=     100
82      000040      SW05=     40
83      000020      SW04=     20
84      000010      SW03=     10
85      000004      SW02=     4
86      000002      SW01=     2
87      000001      SW00=     1
88      .EQUIV     SW09, SW9
89      .EQUIV     SW08, SW8
90      .EQUIV     SW07, SW7
91      .EQUIV     SW06, SW6
92      .EQUIV     SW05, SW5
93      .EQUIV     SW04, SW4
94      .EQUIV     SW03, SW3
95      .EQUIV     SW02, SW2
96      .EQUIV     SW01, SW1
97      .EQUIV     SW00, SW0
98
99      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100     100000     BIT15=    100000
101     040000     BIT14=    40000
102     020000     BIT13=    20000
103     010000     BIT12=    10000
104     004000     BIT11=    4000
105     002000     BIT10=    2000
106     001000     BIT09=    1000
107     000400     BIT08=    400
108     000200     BIT07=    200
109     000100     BIT06=    100
110     000040     BIT05=    40
111     000020     BIT04=    20
112     000010     BIT03=    10

```

```

113      000004      BIT02= 4
114      000002      BIT01= 2
115      000001      BIT00= 1
116      .EQUIV      BIT09,BIT9
117      .EQUIV      BIT08,BIT8
118      .EQUIV      BIT07,BIT7
119      .EQUIV      BIT06,BIT6
120      .EQUIV      BIT05,BIT5
121      .EQUIV      BIT04,BIT4
122      .EQUIV      BIT03,BIT3
123      .EQUIV      BIT02,BIT2
124      .EQUIV      BIT01,BIT1
125      .EQUIV      BIT00,BIT0
126
127      .#BASIC "CPU" TRAP VECTOR ADDRESSES
128      000004      ERRVEC= 4          ;: TIME OUT AND OTHER ERRORS
129      000010      RESVEC= 10         ;: RESERVED AND ILLEGAL INSTRUCTIONS
130      000014      TBITVEC=14        ;: "T" BIT
131      000014      TRTVEC= 14        ;: TRACE TRAP
132      000014      BPTVEC= 14        ;: BREAKPOINT TRAP (BPT)
133      000020      IOTVEC= 20        ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
134      000024      PWRVEC= 24        ;: POWER FAIL
135      000030      EMTVEC= 30        ;: EMULATOR TRAP (EMT) **ERROR**
136      000034      TRAPVEC=34        ;: "TRAP" TRAP
137      000060      TKVEC= 60         ;: TTY KEYBOARD VECTOR
138      000064      TPVEC= 64         ;: TTY PRINTER VECTOR
139      000240      PIRQVEC=240       ;: PROGRAM INTERRUPT REQUEST VECTOR
140
141      .#MED INSTR CODES
142      076600      MED= 076600       ;: OPCODE
143
144      000022      RWHAMI= 022        ;: READ WHAMI
145
146      000144      RFLAG= 144         ;: READ FLAGS
147      000344      WFLAG= 344        ;: WRITE FLAGS
148
149      .#FLOATING POINT INTERRUPT VECTOR
150      000244      FPPVEC= 244
151
152      .#FLOATING POINT REGISTER DEFINITIONS
153      000000      AC0= %0
154      000001      AC1= %1
155      000002      AC2= %2
156      000003      AC3= %3
157      000004      AC4= %4
158      000005      AC5= %5
159
160
161      .SBTTL TRAP CATCHER
162
163
164      000000      . = 0
165      ;#ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
166      ;#SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
167      ;#LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
168      000174      . = 174

```

H02

```

169 000174 000000  DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
170 000176 000000  SWREG: .WORD 0           ;; SOFTWARE SWITCH REGISTER
171                                     .SBTTL STARTING ADDRESS(ES)
172 000200 000137 003000  JMP @#START ;; JUMP TO STARTING ADDRESS OF PROGRAM
173
174                                     .SBTTL ACT11 HOOKS
175
176                                     ;; *****
177                                     ;HOOKS REQUIRED BY ACT11
178                                     $SVPC=. ;SAVE PC
179                                     .=46
180 000046 022106  SENDAD ;; 1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
181                                     .=52
182 000052 000000  .WORD 0 ;; 2)SET LOC.52 TO ZERO
183                                     .=$SVPC ;; RESTORE PC
184                                     .=1000
185                                     .SBTTL APT PARAMETER BLOCK
186
187                                     ;; *****
188                                     ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
189                                     ;; *****
190                                     .SX=. ;; SAVE CURRENT LOCATION
191                                     .=24 ;; SET POWER FAIL TO POINT TO START OF PROGRAM
192 000024 000200  200 ;; FOR APT START UP
193                                     .=44 ;; POINT TO APT INDIRECT ADDRESS PNTR.
194 000044 001000  $APTHDR ;; POINT TO APT HEADER BLOCK
195                                     .=.SX ;; RESET LOCATION COUNTER
196                                     ;; *****
197                                     ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
198                                     ;INTERFACE SPEC.
199
200 001000  $APTHD:
201 001000 000000  $SHIBTS: .WORD 0 ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
202 001002 001202  $MBAOR: .WORD $MAIL ;; ADDRESS OF APT MAILBOX (BITS 0-15)
203 001004 000012  $TSTM: .WORD 10. ;; RUN TIM OF LONGEST TEST
204 001006 000055  $PASTM: .WORD 45. ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
205 001010 000000  $UNITM: .WORD 0 ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
206 001012 000014  .WORD SETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)
207

```

208
209
210
211
212
213
214
215 001100 001100
216
217 001100 000000
218 001102 000000
219 001104 000000
220 001106 000000
221 001110 000000
222 001112 000000
223 001114 000000
224 001116 000000
225 001120 000001
226 001122 000000
227 001124 000000
228 001126 000000
229 001130 000000
230 001132 000000
231 001134 000000
232 001136 000000
233 001140 000
234 001141 000
235 001142 000000
236
237 001144 177570
238 001146 177570
239 001150 000000
240 001152 177560
241 001154 177562
242 001156 177564
243 001160 177566
244 001162 000
245 001163 002
246 001164 012
247 001165 000
248 001166 000000
249 001170 000000
250 001172 177607 000377
251 001176 077
252 001177 015
253 001200 000012
254
255
256
257
258
259 001202
260 001202 000000
261 001204 000000
262 001206 000000
263 001210 000000

```
.SBTTL COMMON TAGS

:*****
:THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
:USED IN THE PROGRAM.

      . =1100
$CHTAG: .WORD 0 ;: START OF COMMON TAGS
;:-----START OF CLEAR COMMON TAGS-----
      .WORD 0
$STNM: .WORD 0 ;: CONTAINS THE TEST NUMBER
$ERFLG: .WORD 0 ;: CONTAINS ERROR FLAG
$ICNT: .WORD 0 ;: CONTAINS SUBTEST ITERATION COUNT
$LPADR: .WORD 0 ;: CONTAINS SCOPE LOOP ADDRESS
$LPERR: .WORD 0 ;: CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL: .WORD 0 ;: CONTAINS TOTAL ERRORS DETECTED
$ITEMB: .WORD 0 ;: CONTAINS ITEM CONTROL BYTE
$ERMAX: .WORD 1 ;: CONTAINS MAX. ERRORS PER TEST
$ERRPC: .WORD 0 ;: CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD 0 ;: CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD 0 ;: CONTAINS ADDRESS OF 'BAD' DATA
$GD0AT: .WORD 0 ;: CONTAINS 'GOOD' DATA
$BD0AT: .WORD 0 ;: CONTAINS 'BAD' DATA
      .WORD 0 ;: RESERVED--NOT TO BE USED
      .WORD 0
$AUTOB: .BYTE 0 ;: AUTOMATIC MODE INDICATOR
$INTAG: .BYTE 0 ;: INTERRUPT MODE INDICATOR
      .WORD 0
;:-----END OF CLEAR COMMON TAGS-----
$SWR: .WORD DSWR ;: ADDRESS OF SWITCH REGISTER
DISPLA: .WORD DDISP ;: ADDRESS OF DISPLAY REGISTER
$LPST: .WORD 0 ;: CONTAINS TEST NUMBER TO LOOP UPON
$TKS: 177560 ;: TTY KBD STATUS
$TKB: 177562 ;: TTY KBD BUFFER
$TPS: 177564 ;: TTY PRINTER STATUS REG. ADDRESS
$TPB: 177566 ;: TTY PRINTER BUFFER REG. ADDRESS
$NULL: .BYTE 0 ;: CONTAINS NULL CHARACTER FOR FILLS
$FILLS: .BYTE 2 ;: CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC: .BYTE 12 ;: INSERT FILL CHARS. AFTER A "LINE FEED"
$TPFLG: .BYTE 0 ;: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
$TIMES: 0 ;: MAX. NUMBER OF ITERATIONS
$ESCAPE: 0 ;: ESCAPE ON ERROR ADDRESS
$BELL: .ASCIZ <207><377><377> ;: CODE FOR BELL
$QUES: .ASCII '?' ;: QUESTION MARK
$CRLF: .ASCII <15> ;: CARRIAGE RETURN
$LF: .ASCIZ <12> ;: LINE FEED

:*****
.SBTTL APT MAILBOX-ETABLE

:*****
.EVEN
$MAIL: ;: APT MAILBOX
$MSGTY: .WORD AMSQTY ;: MESSAGE TYPE CODE
$FATAL: .WORD AFATAL ;: FATAL ERROR NUMBER
$TESTN: .WORD ATESTN ;: TEST NUMBER
$PASS: .WORD APASS ;: PASS COUNT
```


338	001532	000000	000000	000000						
339	001540	000000								
340	001542	032405	033220	033514	EMV013: .WORD	EMH,DHB,DTD,LOF,HIF,0,0,0,0,0	;DIVF	*		
341	001550	033622	033632	000000						
342	001556	000000	000000	000000						
343	001564	000000								
344	001566	032405	033256	033526	EMV014: .WORD	EMH,DHC,DTE,LOO,HID,0,0,0,0,0	;DIVD	*		
345	001574	033572	033606	000000						
346	001602	000000	000000	000000						
347	001610	000000								
348										
349	001612	032435	033354	033550	EMV015: .WORD	EMI,DHD,DTF,0,0,0,0,0,0,0	;ILLEGAL FPP TRAP			
350	001620	000000	000000	000000						
351	001626	000000	000000	000000						
352	001634	000000								
353										
354	001636	032607	033204	033474	EMV016: .WORD	EMM,DHA,DTA,0,0,0,0,0,0,0	;DIVF	*		
355	001644	000000	000000	000000						
356	001652	000000	000000	000000						
357	001660	000000								
358	001662	032630	033204	033474	EMV017: .WORD	EMN,DHA,DTA,0,0,0,0,0,0,0	;ADD	*		
359	001670	000000	000000	000000						
360	001676	000000	000000	000000						
361	001704	000000								
362	001706	032651	033204	033474	EMV020: .WORD	EMO,DHA,DTA,0,0,0,0,0,0,0	;SUBD	*	FPS ERRORS	
363	001714	000000	000000	000000						
364	001722	000000	000000	000000						
365	001730	000000								
366	001732	032672	033204	033474	EMV021: .WORD	EMP,DHA,DTA,0,0,0,0,0,0,0	;MULD	*		
367	001740	000000	000000	000000						
368	001746	000000	000000	000000						
369	001754	000000								
370	001756	032713	033204	033474	EMV022: .WORD	EMQ,DHA,DTA,0,0,0,0,0,0,0	;DIVD	*		
371	001764	000000	000000	000000						
372	001772	000000	000000	000000						
373	002000	000000								
374										
375	002002	032734	033433	033502	EMV023: .WORD	EMR,DHE,DTB,0,0,0,0,0,0,0	;ADDF	*		
376	002010	000000	000000	000000						
377	002016	000000	000000	000000						
378	002024	000000								
379	002026	032761	033433	033502	EMV024: .WORD	EMS,DHE,DTB,0,0,0,0,0,0,0	;SUBF	*		
380	002134	000000	000000	000000						
381	002142	000000	000000	000000						
382	002050	000000								
383	002052	033006	033433	033502	EMV025: .WORD	EMT,DHE,DTB,0,0,0,0,0,0,0	;MULF	*		
384	002060	000000	000000	000000						
385	002066	000000	000000	000000						
386	002074	000000								
387	002076	033033	033433	033502	EMV026: .WORD	EMU,DHE,DTB,0,0,0,0,0,0,0	;DIVF	*	FEC/FEA ERRORS	
388	002104	000000	000000	000000						
389	002112	000000	000000	000000						
390	002120	000000								
391	002122	033060	033433	033502	EMV027: .WORD	EMV,DHE,DTB,0,0,0,0,0,0,0	;ADD	*		
392	002130	000000	000000	000000						
393	002136	000000	000000	000000						


```

448 002446 175463 030712 105726 OP1: .WORD 175463,030712,105726,124064 ;
449 002454 124064
450 002456 156607 002361 070707 OP2: .WORD 156607,002361,070707,061111 ;
451 002464 061111
452 002466 003505 134261 062451 OP3: .WORD 003505,134261,062451,052525 ;EXTRA RANDOM OPERANDS
453 002474 052525
454 002476 102466 123456 111111 OP4: .WORD 102466,123456,111111,101010 ;
455 002504 101010
456 002506 076103 000346 060612 OP5: .WORD 076103,000346,060612,125252 ;
457 002514 125252
458 002516 044100 112400 177777 OP6: .WORD 044100,112400,177777,000006 ;
459 002524 000006
  
```

```

461 :*ADDED COUNTERS FOR TRACING WHERE WE'VE BEEN
462 002526 000000 ADDC0: .WORD 0 ;TOTAL NUMBER *
463 002530 000000 ADDC1: .WORD 0 ;A=0, B#0 *
464 002532 000000 ADDC2: .WORD 0 ;A#0, B=0 *
465 002534 000000 ADDC3: .WORD 0 ;A=0, B=0 * FOR
466 002536 000000 ADDC4: .WORD 0 ;NO SHIFT REQ'D * ADD
467 002540 000000 ADDC5: .WORD 0 ;OVERFLOW * AND
468 002542 000000 ADDC6: .WORD 0 ;OVERFLOW, ENABLED * SUBTRACT
469 002544 000000 ADDC7: .WORD 0 ;UNDERFLOW *
470 002546 000000 ADDC8: .WORD 0 ;UNDERFLOW, ENABLED *
471
472 002550 000000 MULC0: .WORD 0 ;TOTAL NUMBER *
473 002552 000000 MULC1: .WORD 0 ;A=0 AND/OR B=0 *
474 002554 000000 MULC2: .WORD 0 ;OVERFLOW * FOR
475 002556 000000 MULC3: .WORD 0 ;OVERFLOW, ENABLED * MULTIPLY
476 002560 000000 MULC4: .WORD 0 ;UNDERFLOW *
477 002562 000000 MULC5: .WORD 0 ;UNDERFLOW, ENABLED *
478
479 002564 000000 DIVC0: .WORD 0 ;TOTAL NUMBER *
480 002566 000000 DIVC1: .WORD 0 ;NUM A=0 *
481 002570 000000 DIVC2: .WORD 0 ;DEN B=0 * FOR
482 002572 000000 DIVC3: .WORD 0 ;OVERFLOW * DIVIDE
483 002574 000000 DIVC4: .WORD 0 ;OVERFLOW, ENABLED *
484 002576 000000 DIVC5: .WORD 0 ;UNDERFLOW *
485 002600 000000 DIVC6: .WORD 0 ;UNDERFLOW, ENABLED *
486
  
```

```

487 :*REGISTER CONTENTS, AT ERROR, STORED HERE
488 002602 000000 EREG0: .WORD 0
489 002604 000000 EREG1: .WORD 0
490 002606 000000 EREG2: .WORD 0
491 002610 000000 EREG3: .WORD 0
492 002612 000000 EREG4: .WORD 0
493 002614 000000 EREG5: .WORD 0
494 002616 000000 EREG6: .WORD 0
495 002620 000000 EREG7: .WORD 0
  
```

```

497 :*MESSAGES FOR BEGIN PROGRAM/START OF PASS
498 002622 005015 005012 042115 BGNMES: .ASCII <CR><LF><LF><LF>"MD-11-DQFPD-"
499 002630 030455 026461 050504
500 002636 050106 026504
501 002642 027102 .ASCII "B."
502 002644 027056 .ASCII " "
503 002646 042120 026520 030461 .ASCIIZ "POP-11/6X F.P.U. ADD/SUB/MUL/DIV EXERCISER"<CR><LF>
  
```

B03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 12
DQFP08.P11 04-MAY-77 17:30 PROGRAM DEFINED COMMON TAGS

SEQ 0013

504	002654	032057	020130	027106	
505	002662	027120	027125	040440	
506	002670	042104	051457	041125	
507	002676	046457	046125	042057	
508	002704	053111	042440	042530	
509	002712	041522	051511	051105	
510	002720	005015	000		
511	002723	015	050012	051501	NWPAS1: .ASCIZ <CR><LF>"PASS #"
512	002730	020123	000043		

```

513 .SBTTL START OF PASS ROUTINE
514
515
516 ;;*****
517 .ENABL AMA ; ASSEMBLE ALL RELATIVE REFERENCES AS ABSOLUTE
518 ;;*****
519
520 003000
521
522 START:
523 .SBTTL INITIALIZE THE COMMON TAGS
524 ;;CLEAR THE COMMON TAGS (SCHTAG) AREA
525 MOV #SCHTAG,R6 ;:FIRST LOCATION TO BE CLEARED
526 CLR (R6)+ ;:CLEAR MEMORY LOCATION
527 CMP #SWR,R6 ;:DONE?
528 BNE .-6 ;:LOOP BACK IF NO
529 MOV #STACK,SP ;:SETUP THE STACK POINTER
530 ;;INITIALIZE A FEW VECTORS
531 MOV #SCOPE,#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
532 MOV #340,#IOTVEC+2 ;:LEVEL 7
533 MOV #ERROR,#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
534 MOV #340,#EMTVEC+2 ;:LEVEL 7
535 MOV #TRAP,#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
536 MOV #340,#TRAPVEC+2 ;:LEVEL 7
537 MOV #SPWRON,#PWAVEC ;:POWER FAILURE VECTOR
538 MOV #340,#PWAVEC+2 ;:LEVEL 7
539 MOV #ENDCT,#EOPCT ;:SETUP END-OF-PROGRAM COUNTER
540 CLR #TIMES ;:INITIALIZE NUMBER OF ITERATIONS
541 CLR #ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
542 MOV #1,#SEMAX ;:ALLOW ONE ERROR PER TEST
543 MOV #0,#SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
544 MOV #0,#SLPERR ;:SETUP THE ERROR LOOP ADDRESS
545 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
546 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
547 MOV #ERRVEC,-(SP) ;:SAVE ERROR VECTOR
548 MOV #64S,#ERRVEC ;:SET UP ERROR VECTOR
549 MOV #OSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
550 MOV #DISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
551 CMP #-1,OSWR ;:TRY TO REFERENCE HARDWARE SWR
552 BNE 66S ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
553 ;:AND THE HARDWARE SWR IS NOT = -1
554 BR 65S ;:BRANCH IF NO TIMEOUT
555 MOV #65S,(SP) ;:SET UP FOR TRAP RETURN
556 RTI
557 MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
558 MOV #DISPREG,DISPLAY ;:RESTORE ERROR VECTOR
559 MOV (SP)+,#ERRVEC
560 CLR #PASS ;:CLEAR PASS COUNT
561 BITB #APTSIZE,#ENVM ;:TEST USER SIZE UNDER APT
562 BEQ 67S ;:YES,USE NON-APT SWITCH
563 MOV #SSWREG,SWR ;:NO,USE APT SWITCH REGISTER
564
565 ;SET UP FPP UNEXPECTED TRAP CATCHER - - - - -
566 MOV #FPPILT,#FPPVEC ;:NEW PC AT FPP TRAP
567 CLR #FPPVEC+2 ;:NEW PS AT FPP TRAP
568

```



```

569                                     ; *CLEAR COUNTERS FOR TRACING INFO
570 003262 012700 002526                MOV     $A00C0,RO      ; FIRST LOCATION TO CLEAR
571 003266 012701 000026                MOV     $26,R1        ; 26(8) WORDS
572 003272 005020                        25:    CLR     (RO)+       ; CLEAR AND BUMP
573 003274 077102                        SOB     R1,25         ; DO AGAIN
574
575 003276 104401 002622                TYPE   ,BGNMES       ; ID MESSAGE AT START
576
577                                     ; //////////////////////////////////////
578                                     ; MESSAGE ON WHETHER OR NOT HFP UNIT IS PRESENT
579
580 003302 076600 000022                MED     RWHAMI        ; WHAMI INTO RO
581 003306 032700 000020                BIT     $BIT04,RO     ; IS THERE A HFP UNIT ?
582 003312 001403                        BEQ     70$           ; NO, BR
583 003314 104401 003330                TYPE   68$           ; INDICATE FP11-E PRESENT
584 003320 000453                        BR     NEWPAS        ; GO FOR SUBPASS INIT
585 003322 104401 003370                70$:   TYPE   69$           ; INDICATE NO FP11-E
586 003326 000450                        BR     NEWPAS        ; GO FOR SUBPASS INIT
587
588 003330 005015 020052 050106          68$:   .ASCIZ <15><12>* FP11-E HFP UNIT PRESENT *<15><12>
589 003336 030461 042455 044040
590 003344 050106 052440 044516
591 003352 020124 051120 051505
592 003360 047105 020124 006452
593 003366 000012
594 003370 005015 020052 047516          69$:   .ASCIZ <15><12>* NO FP11-E HFP UNIT - ALL TESTS HFP ONLY *<15><12>
595 003376 043040 030520 026461
596 003404 020105 043110 020120
597 003412 047125 052111 026440
598 003420 040440 046114 052040
599 003426 051505 051524 053440
600 003434 050106 047440 046116
601 003442 020131 006452 000012
602
603                                     .EVEN
604
605                                     ; //////////////////////////////////////
606
607                                     ; *****
608                                     ; NEW PASS ENTERS HERE
609                                     ; *****
610
611 003450 012706 001100                NEWPAS: MOV    $STACK,SP ; RESET STACK PTR
612
613 003454 032777 010000 175462          BIT     $BIT12,$SWR   ; INHIBIT STATUS TIMEOUTS ?
614 003462 001015                        BNE    SUBPAS        ; BR IF YES
615 003464 023737 022042 022034          CMP     $ENDCT,$EOPCT ; TIME FOR A MESSAGE ?
616 003472 001011                        BNE    SUBPAS        ; NO, NOT YET
617
618 003474 104401 002723                TYPE   NEWPAS1       ; "PASS #"  
619 003500 013746 001210                MOV     $PASS,-(SP)   ; PASS COUNT INTO ...  
620 003504 005216                        INC     (SP)          ; 1-N RANGE  
621 003506 104403                        TYPOS   TYPE OCTAL   ; TYPE OCTAL  
622 003510 006 000                        .BYTE  6,0           ; 6 DIGITS, NO LEADING ZEROS  
623 003512 104401 001177                TYPE   , $CRLF       ; END THE LINE
624

```

E03

```

625
626
627
628
629
630 003516 012706 001100 SUBPAS: MOV #STACK, SP ;RESET SP FOR INSURANCE
631
632 003522 076600 000022 MED ,RWHAMI ;GET WHAMI INTO RO
633 003526 032700 000020 BIT #BIT04, RO ;1=HFP PRESENT, 0=NO
634 003532 001430 BEQ 20$ ;IF NO HFP, TEST WARM ONLY
635
636 003534 076600 000144 MED ,RFLAG ;GET FLAGS INTO RO
637
638 003540 032777 000002 175376 BIT #SW01, 2SWR ;SW01: 1=HFP OR WFP TEST ONLY
639 003546 001413 BEQ 1$ ; 0=ALTERNATE HFP/WFP PER PASS
640
641 003550 032777 000001 175366 BIT #SW00, 2SWR ;SW00: 1=HFP ONLY
642 003556 001403 BEQ 2$ ; 0=HFP ONLY
643 003560 042700 010000 BIC #BIT12, RO ;CLEAR HFP ENABLE FLAG<5> FOR WFP
644 003564 000402 BR 3$ ;
645 003566 052700 010000 2$: BIS #BIT12, RO ;SET HFP ENABLE FLAG<5> FOR HFP
646 003572 076600 000344 3$: MED ,WFLAG ;REWRITE FLAGS
647
648 003576 032700 010000 1$: BIT #BIT12, RO ;TEST WHO'S ENABLED: HOT, WARM
649 003602 001404 BEQ 20$ ;SET APPROPRIATE HEADER:
650
651 003604 012737 032066 030414 19$: MOV #ASCHOT, HOTWARM ;"HOT: "
652 003612 000403 BR 21$ ;
653 003614 012737 032074 030414 20$: MOV #ASCWARM, HOTWARM ;"WARM: "
654 003622 21$:
  
```

F03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 16
 DQFP08.P11 04-MAY-77 17:30 T1

SEQ 0017

EXERCISE AODF, ALL INTERRUPTS ON, ROUNDING MODE

```

655
656
657
658 003622 000004
659 003624 012737 003756 002376
660 003632 012737 007400 002400
661 003640 005037 002402
662 003644 005037 002404
663 003650 005037 002362
664 003654 005037 002364
665 003660 005037 002366
666 003664 004737 023342
667 003670 002426 002436
668 003674 004437 023506
669 003700 023510 002426
670 003704 023510 002436
671 003710 023566
672 003712 023540 002416
673
674 003716 013700 002400
675 003722 170127 040000
676 003726 172437 002426
677 003732 172537 002436
678 003736 172737 002416
679 003742 170127 007400
680 003746 012737 003754 001110
681
682
683
684 003754 172600
685 003756 172201
686 003760 170237 002362 002400
687 003764 023737 002362
688 003772 001403
689 003774 174237 002406
690 004000 104001
691
692 004002 005737 002400
693 004006 100014
694 004010 170337 002364
695
696 004014 023737 002364 002402
697 004022 001401
698 004024 104023
699
700 004026 023737 002366 002404
701 004034 001401
702 004036 104023
703
704 004040 173702
705 004042 170000
706 004044 001416
707
708 004046 174237 002406
709 004052 162737 000001 002410
710 004060 005637 002406

```

```

*****
;TEST 1 EXERCISE AODF, ALL INTERRUPTS ON, ROUNDING MODE
*****
TST1: SCOPE
MOV #ARET1,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007400,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL2 ;GET RANDOM INPUT DATA
;WORD
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
$AOD ;ADDRESS OF FORTRAN AOD
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;LOAD FPS, INTERRUPT DISABLE
LDF LONUM,ACO ;LOAD ACO WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007400 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

*****
ARET1: LDF ACO,AC2 ;LOAD ACO INTO AC2
AODF AC1,AC2 ;ADD AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ AERR1 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 1 ;FPS ERROR

AERR1: TST $FPS ;ERROR BIT SET ?
BPL ATST1 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

;CHECK THE FLOATING EXCEPTION CODES
CMP FEC,$FEC
BEQ 1$ ;BRANCH IF OK
ERROR 23 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ ATST1 ;BRANCH IF OK
ERROR 23 ;LONUM ADDRESS IN FEA

ATST1: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND1 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1

```

```

711 004064 173737 002406      CMPF   ANS1,AC3      ;CHECK ANSWERS AGAIN
712 004070 170000              CFCC              ;COPY FLOATING CONDITION CODES
713 004072 001403              BEQ    AEND1        ;BRANCH IF OK
714 004074 174237 002406      STF    AC2,ANS1     ;SAVE FPU ANSWER
715 004100 104005              ERROR   5          ;FPU AND FORTRAN DISAGREE
716
717 004102 005037 002362      AEND1: CLR   FPS      ;CLR FPU FPS BUFFER
718
719
720
721
722
723
724 004106 000004              ;*****
725 004110 012737 004242 002376  ;*TEST 2      EXERCISE ADDF, ALL INTERRUPTS ON, ROUNDING MODE
726 004116 012737 007600 002400  ;*****
727 004124 005037 002402      TST2:  SCOPE
728 004130 005037 002404      MOV    #ARET2,EXPFEA ;ADDR OF INSTR BEING TESTED
729 004134 005037 002362      MOV    #007600,$FPS  ;SET IE BITS IN FORTRAN ANSWER
730 004140 005037 002364      CLR   $FEC          ;CLR FORTRAN FEC
731 004144 005037 002366      CLR   $FEA          ;CLR FORTRAN FEA
732 004150 004737 023332      CLR   FPS           ;CLR FPU FPS BUFFER
733 004154 002426 002436      CLR   FEC           ;CLR FPU FEC BUFFER
734 004160 004437 023506      CLR   FEA          ;CLR FPU FEA BUFFER
735 004164 023510 002426      JSR   PC,RANDL4     ;GET RANDOM INPUT DATA
736 004170 023510 002436      .WORD LONUM,HINUM
737 004174 023566              JSR   R4,$POLSH     ;ENTER POLISH MODE
738 004176 023540 002416      $PUSH ,LONUM        ;PUSH 4 WORDS ON STACK (LONUM)
739              $PUSH ,HINUM     ;PUSH 4 WORDS ON STACK (HINUM)
740 004202 013700 002400      $ADD  ,ANS2         ;ADDRESS OF FORTRAN ADD
741 004206 170127 040200      $POPX ,ANS2         ;POP 4 WORDS AND EXIT POLISH MODE
742 004212 172437 002426      MOV    $FPS,R0      ;DISPLAY FLOATING POINT STATUS
743 004216 172537 002436      LDFPS #040200       ;LOAD FPS, INTERRUPT DISABLE AND FD
744 004222 172737 002416      LDO   LONUM,AC0     ;LOAD AC0 WITH A RANDOM NUMBER
745 004226 170127 007600      LDO   HINUM,AC1     ;LOAD AC1 WITH A RANDOM NUMBER
746 004232 012737 004240 001110  LDO   ANS2,AC3      ;LOAD AC3 WITH THE SUM
747              LDFPS #007600  ;TURN INTERRUPTS ON
748              MOV    #.+6,$LPADR ;RESET LOOP ADDRESS
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766

```

H03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 18
D0FP08.P11 04-MAY-77 17:30 T2

EXERCISE ADD, ALL INTERRUPTS ON, ROUNDING MODE

SEQ 0019

767	004320	001401			BEQ	ATST2			; BRANCH IF OK
768	004322	104027			ERROR	27			; WRONG ADDRESS IN FEA
769									
770	004324	173702			ATST2:	CMPO	AC2, AC3		; COMPARE FPU ANSWER TO FORTRAN ANSWER
771	004326	170000				CFCC			; COPY FLOATING CONDITION CODES
772	004330	001422				BEQ	AREND2		; ANSWERS CHECK
773									; COMPENSATE FOR FORTRAN INACCURACIES
774	004332	174237	002406			STD	AC2, ANS1		; SAVE FPU ANSWER
775	004336	162737	000001	002414		SUB	#1, ANS1+6		; DECREMENT FPU ANSWER
776	004344	005637	002412			SBC	ANS1+4		
777	004350	005637	002410			SBC	ANS1+2		
778	004354	005637	002406			SBC	ANS1		
779	004360	173737	002406			CMPO	ANS1, AC3		; CHECK ANSWERS AGAIN
780	004364	170000				CFCC			; COPY FLOATING CONDITION CODES
781	004366	001403				BEQ	AREND2		; BRANCH IF OK
782	004370	174237	002406			STD	AC2, ANS1		; SAVE FPU ANSWER
783	004374	104006				ERROR	6		; FPU AND FORTRAN DISAGREE
784									
785	004376	005037	002362		AREND2:	CLR	FPS		; CLR FPU FPS BUFFER
786									
787									
788									
789									
790									
791									
792									
793	004402	000004							
794	004404	012737	004536	002376	TEST 3:	MOV	#ARET3, EXPFEA		; ADDR OF INSTR BEING TESTED
795	004412	012737	004400	002400		MOV	#004400, \$FPS		; SET IE BITS IN FORTRAN ANSWER
796	004420	005037	002402			CLR	\$FEC		; CLR FORTRAN FEC
797	004424	005037	002404			CLR	\$FEA		; CLR FORTRAN FEA
798	004430	005037	002362			CLR	FPS		; CLR FPU FPS BUFFER
799	004434	005037	002364			CLR	FEC		; CLR FPU FEC BUFFER
800	004440	005037	002366			CLR	FEA		; CLR FPU FEA BUFFER
801	004444	004737	023342			JSR	PC, RANDL2		; GET RANDOM INPUT DATA
802	004450	002426	002436			.WORD	LONUM, HINUM		
803	004454	004437	023506			JSR	R4, \$POLSH		; ENTER POLISH MODE
804	004460	023510	002426			\$PUSH	, LONUM		; PUSH 2 WORDS ON STACK (LONUM)
805	004464	023510	002436			\$PUSH	, HINUM		; PUSH 2 WORDS ON STACK (HINUM)
806	004470	023566				\$ADD			; ADDRESS OF FORTRAN ADD
807	004472	023540	002416			\$POPX	, ANS2		; POP 2 WORDS AND EXIT POLISH MODE
808									
809	004476	013700	002400			MOV	\$FPS, R0		; DISPLAY FLOATING POINT STATUS
810	004502	170127	040000			LDFPS	#040000		; LOAD FPS, INTERRUPT DISABLE
811	004506	172437	002426			LDF	LONUM, AC0		; LOAD AC0 WITH A RANDOM NUMBER
812	004512	172537	002436			LDF	HINUM, AC1		; LOAD AC1 WITH A RANDOM NUMBER
813	004516	172737	002416			LDF	ANS2, AC3		; LOAD AC3 WITH THE SUM
814	004522	170127	004400			LDFPS	#004400		; TURN INTERRUPTS ON, EXCEPT OVERFLOW AND UNDERFLOW
815	004526	012737	004534	001110		MOV	#, +6, \$LPRDR		; RESET LOOP ADDRESS
816									
817									
818									
819	004534	172600				LDF	AC0, AC2		; LOAD AC0 INTO AC2
820	004536	172201			ARET3:	ADD	AC1, AC2		; ADD AC1 BY AC2
821	004540	170237	002362			STFPS	FPS		; STORE FLOATING POINT STATUS
822	004544	023737	002362	002400		CMF	FPS, \$FPS		; CHECK FPS

```

823 004552 001403          BEQ   ATST3          ; BRANCH IF OK
824 004554 174237 002406   STF   AC2,ANS1      ; SAVE FPU ANSWER
825 004560 104001          ERROR  1            ; FPS ERROR
826
827 004562 173702          ATST3: CMPF   AC2,AC3      ; COMPARE FPU ANSWER TO FORTRAN ANSWER
828 004564 170000          CFCC                          ; COPY FLOATING CONDITION CODES
829 004566 001416          BEQ   AEND3          ; ANSWERS CHECK
830          ; COMPENSATE FOR FORTRAN INACCURACIES
831 004570 174237 002406   STF   AC2,ANS1      ; SAVE FPU ANSWER
832 004574 162737 000001 002410   SUB   #1,ANS1+2     ; DECREMENT FPU ANSWER
833 004602 005637 002406   SBC   ANS1
834 004606 173737 002406   CMPF   ANS1,AC3     ; CHECK ANSWERS AGAIN
835 004612 170000          CFCC                          ; COPY FLOATING CONDITION CODES
836 004614 001403          BEQ   AEND3          ; BRANCH IF OK
837 004616 174237 002406   STF   AC2,ANS1      ; SAVE FPU ANSWER
838 004622 104005          ERROR  5            ; FPU AND FORTRAN DISAGREE
839
840 004624 005037 002362   AEND3: CLR   FPS          ; CLR FPU FPS BUFFER
841
842
843
844
845
846
847
848 004630 000004          ; *****
849 004632 012737 004764 002376 ; *TEST 4 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
850 004640 012737 004600 002400 ; *****
851 004646 005037 002402          ;ST4: SCOPE
852 004652 005037 002404          MOV   #ARET4,EXPFEA ; ADDR OF INSTR BEING TESTED
853 004656 005037 002362          MOV   #004600,$FPS ; SET IE BITS IN FORTRAN ANSWER
854 004662 005037 002364          CLR   $FEC          ; CLR FORTRAN FEC
855 004666 005037 002366          CLR   $FEA          ; CLR FORTRAN FEA
856 004672 004737 023332          CLR   FPS          ; CLR FPU FPS BUFFER
857 004676 002426 002436          CLR   FEC          ; CLR FPU FEC BUFFER
858 004702 004437 023506          CLR   FEA          ; CLR FPU FEA BUFFER
859 004706 023510 002426          JSR   PC,RANDL4     ; GET RANDOM INPUT DATA
860 004712 023510 002436          .WORD LONUM,HINUM ;
861 004716 023566          JSR   R4,$POLSH    ; ENTER POLISH MODE
862 004720 023540 002416          $PUSH ,LONUM       ; PUSH 4 WORDS ON STACK (LONUM)
863          $PUSH ,HINUM    ; PUSH 4 WORDS ON STACK (HINUM)
864 004724 013700 002400          $ADD  ,ANS2        ; ADDRESS OF FORTRAN ADD
865 004730 170127 040200          $POPX ,ANS2        ; POP 4 WORDS AND EXIT POLISH MODE
866 004734 172437 002426          MOV   $FPS,R0      ; DISPLAY FLOATING POINT STATUS
867 004740 172537 002436          LDFPS #040200      ; LOAD FPS, INTERRUPT DISABLE AND FD
868 004744 172737 002416          LDD   LONUM,AC0    ; LOAD AC0 WITH A RANDOM NUMBER
869 004750 170127 004600          LDD   HINUM,AC1    ; LOAD AC1 WITH A RANDOM NUMBER
870 004754 012737 004762 001110   LDD   ANS2,AC3     ; LOAD AC3 WITH THE SUM
871          LDFPS #004600   ; TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
872          MOV   #.+6,$LPAOR ; RESET LOOP ADDRESS
873          ; *****
874 004762 172600          ARET4: LDD   AC0,AC2    ; LOAD AC0 INTO AC2
875 004764 172201          ADDD  AC1,AC2     ; ADD AC1 BY AC2
876 004766 170237 002362          STFPS FPS         ; STORE FLOATING POINT STATUS
877 004772 023737 002362 002400   CMP   FPS,$FPS     ; CHECK FPS
878 005000 001403          BEQ   ATST4        ; BRANCH IF OK

```


J03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 20 SEQ 0021
 DQFP08.P11 04-MAY-77 17:30 T4 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE

```

879 005002 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
880 005006 104017 ERROR 17 ;FPS ERROR
881
882 005010 173702 ATST4: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
883 005012 170000 CFCC ;COPY FLOATING CONDITION CODES
884 005014 001422 BEQ AEND4 ;ANSWERS CHECK
885 ;COMPENSATE FOR FORTRAN INACCURACIES.
886 005016 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
887 005022 162737 000001 002414 SUB #1,ANS1+6 ;DECREMENT FPU ANSWER
888 005030 005637 002412 SBC ANS1+4
889 005034 005637 002410 SBC ANS1+2
890 005040 005637 002406 SBC ANS1
891 005044 173737 002406 CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
892 005050 170000 CFCC ;COPY FLOATING CONDITION CODES
893 005052 001403 BEQ AEND4 ;BRANCH IF OK
894 005054 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
895 005060 104006 ERROR 6 ;FPU AND FORTRAN DISAGREE
896
897 005062 005037 002362 AEND4: CLR FPS ;CLR FPU FPS BUFFER
898
899
900
901
902
903
904

```

```

;*****
;#TEST 5 EXERCISE ADD, ALL INTERRUPTS ON, TRUNCATE MODE
;*****

```

```

905 005066 000004 STS: SCOPE
906 005070 012737 005222 002376 MOV #ARETS,EXPFEA ;ADDR OF INSTR BEING TESTED
907 005076 012737 007440 002400 MOV #007440,$FPS ;SET IE BITS IN FORTRAN ANSWER
908 005104 005037 002402 CLR $FEC ;CLR FORTRAN FEC
909 005110 005037 002404 CLR $FEA ;CLR FORTRAN FEA
910 005114 005037 002362 CLR FPS ;CLR FPU FPS BUFFER
911 005120 005037 002364 CLR FEC ;CLR FPU FEC BUFFER
912 005124 005037 002366 CLR FEA ;CLR FPU FEA BUFFER
913 005130 004737 023342 JSR PC,RANDL2 ;GET RANDOM INPUT DATA
914 005134 002426 002436 .WORD LONUM,HINUM
915 005140 004437 023506 JSR R4,$POLSH ;ENTER POLISH MODE
916 005144 023510 002426 $PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
917 005150 023510 002436 $PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
918 005154 023566 $ADD ;ADDRESS OF FORTRAN ADD
919 005156 023540 002416 $POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE
920
921 005162 013700 002400 MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
922 005166 170127 040000 LDFPS #040000 ;LOAD FPS, INTERRUPT DISABLE
923 005172 172437 002426 LDF LONUM,ACO ;LOAD ACO WITH A RANDOM NUMBER
924 005176 172537 002436 LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
925 005202 172737 002416 LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
926 005206 170127 007440 LDFPS #007440 ;TURN INTERRUPTS ON
927 005212 012737 005220 001110 MOV #.+6,$LPCOR ;RESET LOOP ADDRESS
928
929 ;*****
930

```

```

931 005220 172600 ARETS: LDF ACO,AC2 ;LOAD ACO INTO AC2
932 005222 172201 ADF AC1,AC2 ;ADD AC1 BY AC2
933 005224 170237 002362 STFPS FPS ;STORE FLOATING POINT STATUS
934 005230 023737 002362 002400 CMP FPS,$FPS ;CHECK FPS

```

K03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 16.18 PAGE 21
 DAFP08.P11 04-MAY-77 17:30 T5

EXERCISE ADD, ALL INTERRUPTS ON, TRUNCATE MODE

SEQ 0022

935	005236	001403				BEQ	AERRS		: BRANCH IF OK
936	005240	174237	002406			STF	AC2,ANS1		: SAVE FPU ANSWER
937	005244	104001				ERROR	1		: FPS ERROR
938									
939	005246	005737	002400		AERRS:	TST	\$FPS		: ERROR BIT SET ?
940	005252	100014				BPL	ATSTS		: NO, DONT GET FEC/FEA
941	005254	170337	002364			STST	FEC		: YES, CHECK STATUS
942									
943	005260	023737	002364	002402		CMP	FEC,\$FEC		: CHECK THE FLOATING EXCEPTION CODES
944	005266	001401				BEQ	15		: BRANCH IF OK
945	005270	104023				ERROR	23		: FEC IS WRONG
946									
947	005272	023737	002366	002404	15:	CMP	FEA,\$FEA		: CHECK FLOATING PC
948	005300	001401				BEQ	ATSTS		: BRANCH IF OK
949	005302	104023				ERROR	21		: WRONG ADDRESS IN FEA
950									
951	005304	173702			ATSTS:	CMPF	AC2,AC3		: COMPARE FPU ANSWER TO FORTRAN ANSWER
952	005306	170000				CFCC			: COPY FLOATING CONDITION CODES
953	005310	001427				BEQ	AENDS		: ANSWERS CHECK
954									: COMPENSATE FOR FORTRAN INACCURACIES.
955	005312	174237	002406			STF	AC2,ANS1		: SAVE FPU ANSWER
956	005316	062737	000001	002410		ADD	#1,ANS1+2		: INCREMENT FPU ANSWER
957	005324	005537	002406			RDC	ANS1		
958	005330	173737	002406			CMPF	ANS1,AC3		: CHECK ANSWERS AGAIN
959	005334	170000				CFCC			: COPY FLOATING CONDITION CODES
960	005336	001414				BEQ	AENDS		: BRANCH IF OK
961	005340	162737	000002	002410		SUB	#2,ANS1+2		: DECREMENT FPU ANSWER
962	005346	005637	002406			SBC	ANS1		
963	005352	173737	002406			CMPF	ANS1,AC3		: CHECK ANSWERS AGAIN
964	005356	170000				CFCC			: COPY FLOATING CONDITION CODES
965	005360	001403				BEQ	AENDS		: BRANCH IF OK
966	005362	174237	002406			STF	AC2,ANS1		: SAVE FPU ANSWER
967	005366	104005				ERROR	5		: FPU AND FORTRAN DISAGREE
968									
969	005370	005037	002362		AENDS:	CLR	FPS		: CLR FPU FPS BUFFER
970									
971									
972									
973									
974									
975									
976									
977									
978	005374	000004							
979	005376	012737	005530	002376	TST6:	SCOPE			
980	005404	012737	007640	002400		MOV	#ARET6,EXPFEA		: ADDR OF INSTR BEING TESTED
981	005412	005037	002402			MOV	#007640,\$FPS		: SET IE BITS IN FORTRAN ANSWER
982	005416	005037	002404			CLR	\$FEC		: CLR FORTRAN FEC
983	005422	005037	002362			CLR	\$FEA		: CLR FORTRAN FEA
984	005426	005037	002364			CLR	FPS		: CLR FPU FPS BUFFER
985	005432	005037	002366			CLR	FEC		: CLR FPU FEC BUFFER
986	005436	004737	023332			CLR	FEA		: CLR FPU FEA BUFFER
987	005442	002426	002436			JSR	PC,RANDL4		: GET RANDOM INPUT DATA
988	005446	004437	023506			.WORD	LONUM,HINUM		
989	005452	023510	002426			JSR	R4,\$POLSH		: ENTER POLISH MODE
990	005456	023510	002436			\$PUSH	,LONUM		: PUSH 4 WORDS ON STACK (LONUM)
						\$PUSH	,HINUM		: PUSH 4 WORDS ON STACK (HINUM)

 : TEST 6 EXERCISE ADD, ALL INTERRUPTS ON, TRUNCATE MODE
 : *****

L03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 22
 DQFP08.P11 04-MAY-77 17:30 T6

EXERCISE ADD, ALL INTERRUPTS ON, TRUNCATE MODE

SEQ 0023

991	005462	023566			\$ADD		; ADDRESS OF FORTRAN ADD
992	005464	023540	002416		\$POPX	,ANS2	; POP 4 WORDS AND EXIT POLISH MODE
993							
994	005470	013700	002400		MOV	\$FPS, R0	; DISPLAY FLOATING POINT STATUS
995	005474	170127	040200		LDFPS	#040200	; LOAD FPS, INTERRUPT DISABLE AND FD
996	005500	172437	002426		LDO	LONUM, ACO	; LOAD ACO WITH A RANDOM NUMBER
997	005504	172537	002436		LDO	HINUM, AC1	; LOAD AC1 WITH A RANDOM NUMBER
998	005510	172737	002416		LDO	ANS2, AC3	; LOAD AC3 WITH THE SUM
999	005514	170127	007640		LDFPS	#007640	; TURN INTERRUPTS ON
1000	005520	012737	005526	001110	MOV	#.+6, \$LPAOR	; RESET LOOP ADDRESS
1001							
1002							;*****
1003							
1004	005526	172600			LDO	ACO, AC2	; LOAD ACO INTO AC2
1005	005530	172201			ARET6:	ADD AC1, AC2	; ADD AC1 BY AC2
1006	005532	170237	002362		STFPS	FPS	; STORE FLOATING POINT STATUS
1007	005536	023737	002362	002400	CMP	FPS, \$FPS	; CHECK FPS
1008	005544	001403			BEQ	AERR6	; BRANCH IF OK
1009	005546	174237	002406		STD	AC2, ANS1	; SAVE FPU ANSWER
1010	005552	104017			ERROR	17	; FPS ERROR
1011							
1012	005554	005737	002400		AERR6:	TST \$FPS	; ERROR BIT SET ?
1013	005560	100014			BPL	ATST6	; NO, DONT GET FEC/FEA
1014	005562	170337	002364		STST	FEC	; YES, CHECK STATUS
1015							
1016	005566	023737	002364	002402	CMP	FEC, \$FEC	; CHECK THE FLOATING EXCEPTION CODES
1017	005574	001401			BEQ	15	; BRANCH IF OK
1018	005576	104027			ERROR	27	; FEC IS WRONG
1019							
1020	005600	023737	002366	002404	15:	CMP FEA, \$FEA	; CHECK FLOATING PC
1021	005606	001401			BEQ	ATST6	; BRANCH IF OK
1022	005610	104027			ERROR	27	; WRONG ADDRESS IN FEA
1023							
1024	005612	173702			ATST6:	CMPO AC2, AC3	; COMPARE FPU ANSWER TO FORTRAN ANSWER
1025	005614	170000			CFCC		; COPY FLOATING CONDITION CODES
1026	005616	001437			BEQ	AEND6	; ANSWERS CHECK
1027							; COMPENSATE FOR FORTRAN INACCURACIES.
1028	005620	174237	002406		STD	AC2, ANS1	; SAVE FPU ANSWER
1029	005624	062737	000001	002414	ADD	#1, ANS1+6	; INCREMENT FPU ANSWER
1030	005632	005537	002412		ADC	ANS1+4	
1031	005636	005537	002410		ADC	ANS1+2	
1032	005642	005537	002406		ADC	ANS1	
1033	005646	173737	002406		CMPO	ANS1, AC3	; CHECK ANSWERS AGAIN
1034	005652	170000			CFCC		; COPY FLOATING CONDITION CODES
1035	005654	001420			BEQ	AEND6	; BRANCH IF OK
1036	005656	162737	000002	002414	SUB	#2, ANS1+6	; DECREMENT FPU ANSWER
1037	005664	005637	002412		SBC	ANS1+4	
1038	005670	005637	002410		SBC	ANS1+2	
1039	005674	005637	002406		SBC	ANS1	
1040	005700	173737	002406		CMPO	ANS1, AC3	; CHECK ANSWERS AGAIN
1041	005704	170000			CFCC		; COPY FLOATING CONDITION CODES
1042	005706	001403			BEQ	AEND6	; BRANCH IF OK
1043	005710	174237	002406		STD	AC2, ANS1	; SAVE FPU ANSWER
1044	005714	104006			ERROR	6	; FPU AND FORTRAN DISAGREE
1045							
1046	005716	005037	002362		AEND6:	CLR FPS	; CLR FPU FPS BUFFER

1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102

005722 000004
005724 012737 006056 002376
005732 012737 004440 002400
005740 005037 002402
005744 005037 002404
005750 005037 002362
005754 005037 002364
005760 005037 002366
005764 004737 023342
005770 002426 002436
005774 004437 023506
006000 023510 002426
006004 023510 002436
006010 023566
006012 023540 002416
006016 013700 002400
006022 170127 040000
006026 172437 002426
006032 172537 002436
006036 172737 002416
006042 170127 004440
006046 012737 006054 001110
006054 172600
006056 172201
006060 170237 002362
006064 023737 002362 002400
006072 001403
006074 174237 002406
006100 104001
006102 173702
006104 170000
006106 001427
006110 174237 002406
006114 062737 000001 002410
006122 005537 002406
006126 173737 002406
006132 170000
006134 001414
006136 162737 000002 002410
006144 005637 002406
006150 173737 002406
006154 170000

```
*****  
: TEST 7 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE  
*****  
↑ST7: SCOPE  
MOV #ARET7, EXPFEA ; ADDR OF INSTR BEING TESTED  
MOV #004440, SFPS ; SET IE BITS IN FORTRAN ANSWER  
CLR SFEC ; CLR FORTRAN FEC  
CLR SFEA ; CLR FORTRAN FEA  
CLR FPS ; CLR FPU FPS BUFFER  
CLR FEC ; CLR FPU FEC BUFFER  
CLR FEA ; CLR FPU FEA BUFFER  
JSR PC, RANDL2 ; GET RANDOM INPUT DATA  
WORD LONUM, HINUM  
JSR R4, SPOLSH ; ENTER POLISH MODE  
SPUSH LONUM ; PUSH 2 WORDS ON STACK (LONUM)  
SPUSH HINUM ; PUSH 2 WORDS ON STACK (HINUM)  
SADD ; ADDRESS OF FORTRAN ADD  
SPOPX ,ANS2 ; POP 2 WORDS AND EXIT POLISH MODE  
  
MOV SFPS, R0 ; DISPLAY FLOATING POINT STATUS  
LDFPS #040000 ; LOAD FPS, INTERRUPT DISABLE  
LDF LONUM, AC0 ; LOAD AC0 WITH A RANDOM NUMBER  
LDF HINUM, AC1 ; LOAD AC1 WITH A RANDOM NUMBER  
LDF ANS2, AC3 ; LOAD AC3 WITH THE SUM  
LDFPS #004440 ; TURN INTERRUPTS ON, EXCEPT OVERFLOW AND UNDERFLOW  
MOV #. +6, SLPADR ; RESET LOOP ADDRESS  
  
; *****  
ARET7: LDF AC0, AC2 ; LOAD AC0 INTO AC2  
ADD AC1, AC2 ; ADD AC1 BY AC2  
STFPS FPS ; STORE FLOATING POINT STATUS  
CMP FPS, SFPS ; CHECK FPS  
BEQ ATST7 ; BRANCH IF OK  
STF AC2, ANS1 ; SAVE FPU ANSWER  
ERROR 1 ; FPS ERROR  
  
ATST7: CMPF AC2, AC3 ; COMPARE FPU ANSWER TO FORTRAN ANSWER  
CFCC ; COPY FLOATING CONDITION CODES  
BEQ REND7 ; ANSWERS CHECK  
; COMPENSATE FOR FORTRAN INACCURACIES.  
STF AC2, ANS1 ; SAVE FPU ANSWER  
ADD #1, ANS1+2 ; INCREMENT FPU ANSWER  
ADC ANS1  
CMPF ANS1, AC3 ; CHECK ANSWERS AGAIN  
CFCC ; COPY FLOATING CONDITION CODES  
BEQ REND7 ; BRANCH IF OK  
SUB #2, ANS1+2 ; DECREMENT FPU ANSWER  
SBC ANS1  
CMPF ANS1, AC3 ; CHECK ANSWERS AGAIN  
CFCC ; COPY FLOATING CONDITION CODES
```

```

1103 006156 001403          BEQ      AEND7          ;BRANCH IF OK
1104 006160 174237 002406   STF      AC2,ANS1      ;SAVE FPU ANSWER
1105 006164 104005          ERROR    5              ;FPU AND FORTRAN DISAGREE
1106
1107 006166 005037 002362   AEND7:  CLR      FPS          ;CLR FPU FPS BUFFER
1108
1109
1110
1111
1112
1113
1114
1115 006172 000004          ;*****
1116 006174 012737 006326 002376 ;*TEST 10 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
1117 006202 012737 004640 002400 ;*****
1118 006210 005037 002402          †ST10: SCOPE
1119 006214 005037 002404          MOV      #ARET10,EXPFEA ;ADDR OF INSTR BEING TESTED
1120 006220 005037 002362          MOV      #004640,$FPS   ;SET IE BITS IN FORTRAN ANSWER
1121 006224 005037 002364          CLR      $FEC           ;CLR FORTRAN FEC
1122 006230 005037 002366          CLR      $FEA           ;CLR FORTRAN FEA
1123 006234 004737 023332          CLR      FPS           ;CLR FPU FPS BUFFER
1124 006240 002426 002436          CLR      FEC           ;CLR FPU FEC BUFFER
1125 006244 004437 023506          CLR      FEA           ;CLR FPU FEA BUFFER
1126 006250 023510 002426          JSR      PC,RANDL4     ;GET RANDOM INPUT DATA
1127 006254 023510 002436          .WORD   LONUM,HINUM   ;
1128 006260 023566          JSR      R4,$POLSH     ;ENTER POLISH MODE
1129 006262 023540 002416          SPUSH   ,LONUM         ;PUSH 4 WORDS ON STACK (LONUM)
1130
1131 006266 013700 002400          SPUSH   ,HINUM         ;PUSH 4 WORDS ON STACK (HINUM)
1132 006272 170127 040200          SADD    ,ANS2          ;ADDRESS OF FORTRAN ADD
1133 006276 172437 002426          SPOPX   ,ANS2          ;POP 4 WORDS AND EXIT POLISH MODE
1134 006302 172537 002436          MOV      $FPS,R0       ;DISPLFY FLOATING POINT STATUS
1135 006306 172737 002416          LDFPS  #040200        ;LOAD FPS, INTERRUPT DISABLE AND FD
1136 006312 170127 004640          LDD    LONUM,AC0      ;LOAD AC0 WITH A RANDOM NUMBER
1137 006316 012737 006324 001110   LDD    HINUM,AC1      ;LOAD AC1 WITH A RANDOM NUMBER
1138
1139
1140
1141 006324 172600          LDD    ANS2,AC3       ;LOAD AC3 WITH THE SUM
1142 006326 172201          LDFPS  #004640        ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
1143 006330 170237 002362          MOV      #.+6,$LPADR   ;RESET LOOP ADDRESS
1144 006334 023737 002362 002400   ;*****
1145 006342 001403          ARET10: LDD    AC0,AC2   ;LOAD AC0 INTO AC2
1146 006344 174237 002406          ADD    AC1,AC2        ;ADD AC1 BY AC2
1147 006350 104017          STFPS  FPS            ;STORE FLOATING POINT STATUS
1148
1149 006352 173702          CMP    FPS,$FPS       ;CHECK FPS
1150 006354 170000          BEQ    ATST10         ;BRANCH IF OK
1151 006356 001437          STD    AC2,ANS1       ;SAVE FPU ANSWER
1152
1153 006360 174237 002406          ERROR  17            ;FPS ERROR
1154 006364 062737 000001 002414   ATST10: CMPD   AC2,AC3   ;COMPARE FPU ANSWER TO FORTRAN ANSWER
1155 006372 005537 002412          CFCC   ;COPY FLOATING CONDITION CODES
1156 006376 005537 002410          BEQ    AEND10         ;ANSWERS CHECK
1157 006402 005537 002406          ;COMPENSATE FOR FORTRAN INACCURACIES.
1158 006406 173737 002406          STD    AC2,ANS1       ;SAVE FPU ANSWER
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
324
```


1173					*****		
1174					TEST 11	EXERCISE SUBF, ALL INTERRUPTS ON, ROUNDING MODE	
1175					*****		
1176	006462	000004			TST11: SCOPE		
1177	006464	012737	006616	002376	MOV #ARET11,EXPFEA	: ADDR OF INSTR BEING TESTED	
1178	006472	012737	007400	002400	MOV #007400,\$FPS	: SET IE BITS IN FORTRAN ANSWER	
1179	006500	005037	002402		CLR \$FEC	: CLR FORTRAN FEC	
1180	006504	005037	002404		CLR \$FEA	: CLR FORTRAN FEA	
1181	006510	005037	002362		CLR FPS	: CLR FPU FPS BUFFER	
1182	006514	005037	002364		CLR FEC	: CLR FPU FEC BUFFER	
1183	006520	005037	002366		CLR FEA	: CLR FPU FEA BUFFER	
1184	006524	004737	023342		JSR PC,RANDL2	: GET RANDOM INPUT DATA	
1185	006530	002426	002436		WORD LONUM,HINUM		
1186	006534	004437	023506		JSR R4,\$POLSH	: ENTER POLISH MODE	
1187	006540	023510	002426		\$PUSH ,LONUM	: PUSH 2 WORDS ON STACK (LONUM)	
1188	006544	023510	002436		\$PUSH ,HINUM	: PUSH 2 WORDS ON STACK (HINUM)	
1189	006550	023562			\$SUB	: ADDRESS OF FORTRAN SUBTRACT	
1190	006552	023540	002416		\$POPY ,ANS2	: POP 2 WORDS AND EXIT POLISH MODE	
1191							
1192	006556	013700	002400		MOV \$FPS,R0	: DISPLAY FLOATING POINT STATUS	
1193	006562	170127	040000		LDFPS #040000	: LOAD FPS, INTERRUPT DISABLE	
1194	006566	172437	002426		LDF LONUM,AC0	: LOAD AC0 WITH A RANDOM NUMBER	
1195	006572	172537	002436		LDF HINUM,AC1	: LOAD AC1 WITH A RANDOM NUMBER	
1196	006576	172737	002416		LDF ANS2,AC3	: LOAD AC3 WITH THE SUM	
1197	006602	170127	007400		LDFPS #007400	: TURN INTERRUPTS ON	
1198	006606	012737	006614	001110	MOV #.+6,\$LPADR	: RESET LOOP ADDRESS	
1199							
1200					*****		
1201							
1202	006614	172600			LDF AC0,AC2	: LOAD AC0 INTO AC2	
1203	006616	173201			ARET11: SUBF AC1,AC2	: SUBTRACT AC1 BY AC2	
1204	006620	170237	002362		STFPS FPS	: STORE FLOATING POINT STATUS	
1205	006624	023737	002362	002400	CMP FPS,\$FPS	: CHECK FPS	
1206	006632	001403			BEQ AERR11	: BRANCH IF OK	
1207	006634	174237	002406		STF AC2,ANS1	: SAVE FPU ANSWER	
1208	006640	104002			ERROR 2	: FPS ERROR	
1209							
1210	006642	005737	002400		AERR11: TST \$FPS	: ERROR BIT SET ?	
1211	006646	100014			BPL ATST11	: NO, DONT GET FEC/FEA	
1212	006650	170337	002364		STST FEC	: YES, CHECK STATUS	
1213							
1214	006654	023737	002364	002402	CMP FEC,\$FEC	: CHECK THE FLOATING EXCEPTION CODES	
1215	006662	001401			BEQ 18	: BRANCH IF OK	
1216	006664	104024			ERROR 24	: FEC IS WRONG	
1217							
1218	006666	023737	002366	002404	18: CMP FEA,\$FEA	: CHECK FLOATING PC	
1219	006674	001401			BEQ ATST11	: BRANCH IF OK	
1220	006676	104024			ERROR 24	: WRONG ADDRESS IN FEA	
1221							
1222	006700	173702			ATST11: CMPF AC2,AC3	: COMPARE FPU ANSWER TO FORTRAN ANSWER	
1223	006702	170000			CFCC	: COPY FLOATING CONDITION CODES	
1224	006704	001416			BEQ AEND11	: ANSWERS CHECK	
1225					: COMPENSATE FOR FORTRAN	: INACCURACIES.	
1226	006706	174237	002406		STF AC2,ANS1	: SAVE FPU ANSWER	
1227	006712	162737	000001	002410	SUB #1,ANS1+2	: DECREMENT FPU ANSWER	
1228	006720	005637	002406		SBC ANS1		


```

1229 006724 173737 002406      CMPF      ANS1,AC3      ;CHECK ANSWERS AGAIN
1230 006730 170000              CFCC              ;COPY FLOATING CONDITION CODES
1231 006732 001403              BEQ        AEND11     ;BRANCH IF OK
1232 006734 174237 002406      STF        AC2,ANS1   ;SAVE FPU ANSWER
1233 006740 104007              ERROR       7         ;FPU AND FORTRAN DISAGREE
1234
1235 006742 005037 002362      AEND11: CLR      FPS      ;CLR FPU FPS BUFFER
1236
1237
1238
1239
1240
1241
1242
1243 006746 000004              ;*****
1244 006750 012737 007102 002376 ;*TEST 12      EXERCISE SUBO, ALL INTERRUPTS ON, ROUNDING MODE
1245 006756 012737 007600 002400 ;*****
1246 006764 005037 002402      AET12: SCOPE
1247 006770 005037 002404      MOV        #ARET12,EXPFEA ;ADDR OF INSTR BEING TESTED
1248 006774 005037 002362      MOV        #007600,$FPS   ;GET IE BITS IN FORTRAN ANSWER
1249 007000 005037 002364      CLR        $FEC          ;CLR FORTRAN FEC
1250 007004 005037 002366      CLR        $FEA          ;CLR FORTRAN FEA
1251 007010 004737 023332      CLR        FPS           ;CLR FPU FPS BUFFER
1252 007014 002426 002436      CLR        FEC           ;CLR FPU FEC BUFFER
1253 007020 004437 023506      CLR        FEA           ;CLR FPU FEA BUFFER
1254 007024 023510 002426      JSR        PC,RANDL4     ;GET RANDOM INPUT DATA
1255 007030 023510 002436      .WORD      LONUM,HINUM   ;
1256 007034 023562              JSR        R4,$POLSH     ;ENTER POLISH MODE
1257 007036 023540 002416      $PUSH      ,LONUM        ;PUSH 4 WORDS ON STACK (LONUM)
1258
1259 007042 013700 002400      $PUSH      ,HINUM        ;PUSH 4 WORDS ON STACK (HINUM)
1260 007046 170127 040200      $SUB       ,ANS2        ;ADDRESS OF FORTRAN SUBTRACT
1261 007052 172437 002426      $POPX     ,ANS2         ;POP 4 WORDS AND EXIT POLISH MODE
1262 007056 172537 002436      MOV        $FPS,RO       ;DISPLAY FLOATING POINT STATUS
1263 007062 172737 002416      LDFPS     #040200       ;LOAD FPS, INTERRUPT DISABLE AND FD
1264 007066 170127 007600      LDD       LONUM,AC0     ;LOAD AC0 WITH A RANDOM NUMBER
1265 007072 012737 007100 001110 ;LOAD AC1 WITH A RANDOM NUMBER
1266
1267
1268
1269 007100 172600              LDD       HINUM,AC1     ;LOAD AC1 WITH A RANDOM NUMBER
1270 007102 173201              LDD       ANS2,AC3     ;LOAD AC3 WITH THE SUM
1271 007104 170237 002362      LDFPS     #007600       ;TURN INTERRUPTS ON
1272 007110 023737 002362 002400 ;MOV        #.+6,$LPADR  ;RESET LOOP ADDRESS
1273 007116 001403              ;*****
1274 007120 174237 002406      ARET12: LDD       AC0,AC2 ;LOAD AC0 INTO AC2
1275 007124 104020              SUBO      AC1,AC2      ;SUBTRACT AC1 BY AC2
1276
1277 007126 005737 002400      STFPS     FPS           ;STORE FLOATING POINT STATUS
1278 007132 100014              CMP       FPS,$FPS     ;CHECK FPS
1279 007134 170337 002364      BEQ       AERR12       ;BRANCH IF OK
1280
1281 007140 023737 002364 002402 ;STD       AC2,ANS1     ;SAVE FPU ANSWER
1282 007146 001401              ERROR     20          ;FPS ERROR
1283 007150 104030              AERR12: TST       $FPS   ;ERROR BIT SET ?
1284
1285
1286
1287
1288
1289
1290
1291 007140 023737 002364 002402 ;BPL      ATST12        ;NO, DONT GET FEC/FEA
1292 007146 001401              STST     FEC           ;YES, CHECK STATUS
1293 007150 104030              CMP       FEC,$FEC     ;CHECK THE FLOATING EXCEPTION CODES
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

E04

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 28
 DDFP08.P11 04-MAY-77 17:30 T12

EXERCISE SUBD, ALL INTERRUPTS ON, ROUNDING MODE

SEQ 0029

```

1285 007152 023737 002366 002404 15:  CMP      FEA,$FEA      ;CHECK FLOATING PC
1286 007160 001401          BEQ      ATST12      ;BRANCH IF OK
1287 007162 104030          ERROR    30         ;WRONG ADDRESS IN FEA
1288
1289 007164 173702          ATST12:  CMPD     AC2,AC3      ;COMPARE FPU ANSWER TO FORTRAN ANSWER
1290 007166 170000          CFCC                    ;COPY FLOATING CONDITION CODES
1291 007170 001422          BEQ      REND12      ;ANSWERS CHECK
1292          ;COMPENSATE FOR FORTRAN INACCURACIES.
1293 007172 174237 002406          STD     AC2,ANS1     ;SAVE FPU ANSWER
1294 007176 162737 000001 002414          SUB     #1,ANS1+6    ;DECREMENT FPU ANSWER
1295 007204 005637 002412          SBC     ANS1+4
1296 007210 005637 002410          SBC     ANS1+2
1297 007214 005637 002406          SBC     ANS1
1298 007220 173737 002406          CMPD   ANS1,AC3     ;CHECK ANSWERS AGAIN
1299 007224 170000          CFCC                    ;COPY FLOATING CONDITION CODES
1300 007226 001403          BEQ      REND12      ;BRANCH IF OK
1301 007230 174237 002406          STD     AC2,ANS1     ;SAVE FPU ANSWER
1302 007234 104010          ERROR    10         ;FPU AND FORTRAN DISAGREE
1303
1304 007236 005037 002362          REND12: CLR     FPS      ;CLR FPU FPS BUFFER
1305
1306
1307
1308
1309
1310
1311
1312 007242 000004          ;*****
1313 007244 012737 007376 002376 ;*TEST 13      EXERCISE SUBF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
1314 007252 012737 004400 002400 ;*****
1315 007260 005037 002402          †ST13:  SCOPE
1316 007264 005037 002404          MOV     #ARET13,EXPFEA ;ADDR OF INSTR BEING TESTED
1317 007270 005037 002362          MOV     #004400,$FPS   ;SET IE BITS IN FORTRAN ANSWER
1318 007274 005037 002364          CLR     $FEC          ;CLR FORTRAN FEC
1319 007300 005037 002366          CLR     $FEA          ;CLR FORTRAN FEA
1320 007304 004737 023342          CLR     FPS           ;CLR FPU FPS BUFFER
1321 007310 002426 002436          CLR     FEC           ;CLR FPU FEC BUFFER
1322 007314 004437 023506          CLR     FEA          ;CLR FPU FEA BUFFER
1323 007320 023510 002426          JSR     PC,RANDL2     ;GET RANDOM INPUT DATA
1324 007324 023510 002436          .WORD  LONUM,HINUM
1325 007330 023562          JSR     R4,$POLSH    ;ENTER POLISH MODE
1326 007332 023540 002416          $PUSH  ,LONUM        ;PUSH 2 WORDS ON STACK (LONUM)
1327          $PUSH  ,HINUM        ;PUSH 2 WORDS ON STACK (HINUM)
1328          $SUB   ,ANS2         ;ADDRESS OF FORTRAN SUBTRACT
1329          $POPX ,ANS2         ;POP 2 WORDS AND EXIT POLISH MODE
1330 007336 013700 002400          MOV     $FPS,R0       ;DISPLAY FLOATING POINT STATUS
1331 007342 170127 040000          LDFPS  #040000       ;LOAD FPS, INTERRUPT DISABLE
1332 007346 172437 002426          LDF    LONUM,AC0     ;LOAD AC0 WITH A RANDOM NUMBER
1333 007352 172537 002436          LDF    HINUM,AC1    ;LOAD AC1 WITH A RANDOM NUMBER
1334 007356 172737 002416          LDF    ANS2,AC3     ;LOAD AC3 WITH THE SUM
1335 007362 170127 004400          LDFPS  #004400       ;TURN INTERRUPTS ON, EXCEPT OVERFLOW AND UNDERFLOW
1336 007366 012737 007374 001110          MOV     #.+6,$LPADR  ;RESET LOOP ADDRESS
1337
1338 007374 172600          ;*****
1339 007376 173201          ARET13: LDF     AC0,AC2     ;LOAD AC0 INTO AC2
1340 007400 170237 002362          SUBF   AC1,AC2      ;SUBTRACT AC1 BY AC2
          STFPS  FPS      ;STORE FLOATING POINT STATUS
  
```

F04

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 29 SEQ 0030
 DDFP08.P11 04-MAY-77 17:30 T13 EXERCISE SUBF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE

1341	007404	023737	002362	002400	CMP	FPS,\$FPS	:CHECK FPS
1342	007412	001403			BEQ	ATST13	:BRANCH IF OK
1343	007414	174237	002406		STF	AC2,ANS1	:SAVE FPU ANSWER
1344	007420	104002			ERROR	2	:FPS ERROR
1345							
1346	007422	173702			ATST13: CMPF	AC2,AC3	:COMPARE FPU ANSWER TO FORTRAN ANSWER
1347	007424	170000			CFCC		:COPY FLOATING CONDITION CODES
1348	007426	001416			BEQ	AREND13	:ANSWERS CHECK
1349							:COMPENSATE FOR FORTRAN INACCURACIES.
1350	007430	174237	002406		STF	AC2,ANS1	:SAVE FPU ANSWER
1351	007434	162737	000001	002410	SUB	#1,ANS1+2	:DECREMENT FPU ANSWER
1352	007442	005637	002406		SBC	ANS1	
1353	007446	173737	002406		CMPF	ANS1,AC3	:CHECK ANSWERS AGAIN
1354	007452	170000			CFCC		:COPY FLOATING CONDITION CODES
1355	007454	001403			BEQ	AREND13	:BRANCH IF OK
1356	007456	174237	002406		STF	AC2,ANS1	:SAVE FPU ANSWER
1357	007462	104007			ERROR	7	:FPU AND FORTRAN DISAGREE
1358							
1359	007464	005037	002362		AREND13: CLR	FPS	:CLR FPU FPS BUFFER
1360							
1361							
1362							
1363							
1364							
1365							
1366							
1367	007470	000004			†ST14: SCOPE		
1368	007472	012737	007624	002376	MOV	#ARET14,EXPFEA	:ADDR OF INSTR BEING TESTED
1369	007500	012737	004600	002400	MOV	#004600,\$FPS	:SET IE BITS IN FORTRAN ANSWER
1370	007506	005037	002402		CLR	\$FEC	:CLR FORTRAN FEC
1371	007512	005037	002404		CLR	\$FEA	:CLR FORTRAN FEA
1372	007516	005037	002362		CLR	FPS	:CLR FPU FPS BUFFER
1373	007522	005037	002364		CLR	FEC	:CLR FPU FEC BUFFER
1374	007526	005037	002366		CLR	FEA	:CLR FPU FEA BUFFER
1375	007532	004737	023332		JSR	PC,RANDL4	:GET RANDOM INPUT DATA
1376	007536	002426	002436		.WORD	LONUM,HINUM	
1377	007542	004437	023506		JSR	R4,\$POLSH	:ENTER POLISH MODE
1378	007546	023510	002426		\$PUSH	,LONUM	:PUSH 4 WORDS ON STACK (LONUM)
1379	007552	023510	002436		\$PUSH	,HINUM	:PUSH 4 WORDS ON STACK (HINUM)
1380	007556	023562			\$SUB		:ADDRESS OF FORTRAN SUBTRACT
1381	007560	023540	002416		\$POPX	,ANS2	:POP 4 WORDS AND EXIT POLISH MODE
1382							
1383	007564	013700	002400		MOV	\$FPS,R0	:DISPLAY FLOATING POINT STATUS
1384	007570	170127	040200		LDFPS	#040200	:LOAD FPS, INTERRUPT DISABLE AND FD
1385	007574	172437	002426		LDD	LONUM,AC0	:LOAD AC0 WITH A RANDOM NUMBER
1386	007600	172537	002436		LDD	HINUM,AC1	:LOAD AC1 WITH A RANDOM NUMBER
1387	007604	172737	002416		LDD	ANS2,AC3	:LOAD AC3 WITH THE SUM
1388	007610	170127	004600		LDFPS	#004600	:TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
1389	007614	012737	007622	001110	MOV	#.+6,\$LPRDR	:RESET LOOP ADDRESS
1390							
1391							
1392							
1393	007622	172600			ARET14: LDD	AC0,AC2	:LOAD AC0 INTO AC2
1394	007624	173201			SUBD	AC1,AC2	:SUBTRACT AC1 BY AC2
1395	007626	170237	002362		STFPS	FPS	:STORE FLOATING POINT STATUS
1396	007632	023737	002362	002400	CMP	FPS,\$FPS	:CHECK FPS

```

1397 007640 001403 BEQ ATST14 ; BRANCH IF OK
1398 007642 174237 002406 STD AC2,ANS1 ; SAVE FPU ANSWER
1399 007646 104020 ERROR 20 ; FPS ERROR
1400
1401 007650 173702 ATST14: CMPD AC2,AC3 ; COMPARE FPU ANSWER TO FORTRAN ANSWER
1402 007652 170000 CFCC ; COPY FLOATING CONDITION CODES
1403 007654 001422 BEQ AEND14 ; ANSWERS CHECK
1404 ; COMPENSATE FOR FORTRAN INACCURACIES.
1405 007656 174237 002406 STD AC2,ANS1 ; SAVE FPU ANSWER
1406 007662 162737 000001 002414 SUB #1,ANS1+6 ; DECREMENT FPU ANSWER
1407 007670 005637 002412 SBC ANS1+4
1408 007674 005637 002410 SBC ANS1+2
1409 007700 005637 002406 SBC ANS1
1410 007704 173737 002406 CMPD ANS1,AC3 ; CHECK ANSWERS AGAIN
1411 007710 170000 CFCC ; COPY FLOATING CONDITION CODES
1412 007712 001403 BEQ AEND14 ; BRANCH IF OK
1413 007714 174237 002406 STD AC2,ANS1 ; SAVE FPU ANSWER
1414 007720 104010 ERROR 10 ; FPU AND FORTRAN DISAGREE
1415
1416 007722 005037 002362 AEND14: CLR FPS ; CLR FPU FPS BUFFER
1417
1418
1419
1420

```

```

;*****
;TEST 15 EXERCISE SUBF, ALL INTERRUPTS ON, TRUNCATE MODE
;*****

```

```

1421
1422
1423
1424 007726 000004 ST15: SCOPE
1425 007730 012737 010062 002371, MOV #ARET15,EXPFEA ; ADDR OF INSTR BEING TESTED
1426 007736 012737 007440 002401 MOV #007440,$FPS ; SET IE BITS IN FORTRAN ANSWER
1427 007744 005037 002402 CLR $FEC ; CLR FORTRAN FEC
1428 007750 005037 002404 CLR $FEA ; CLR FORTRAN FEA
1429 007754 005037 002362 CLR FPS ; CLR FPU FPS BUFFER
1430 007760 005037 002364 CLR FEC ; CLR FPU FEC BUFFER
1431 007764 005037 002366 CLR FEA ; CLR FPU FEA BUFFER
1432 007770 004737 023342 JSR PC,RANDL2 ; GET RANDOM INPUT DATA
1433 007774 002426 002436 .WORD LONUM,HINUM
1434 010000 004437 023506 JSR R4,$POLSH ; ENTER POLISH MODE
1435 010004 023510 002426 $PUSH ,LONUM ; PUSH 2 WORDS ON STACK (LONUM)
1436 010010 023510 002436 $PUSH ,HINUM ; PUSH 2 WORDS ON STACK (HINUM)
1437 010014 023552 $SUB ; ADDRESS OF FORTRAN SUBTRACT
1438 010016 023540 002416 $POPX ,ANS2 ; POP 2 WORDS AND EXIT POLISH MODE
1439
1440 010022 013700 002400 MOV $FPS,RO ; DISPLAY FLOATING POINT STATUS
1441 010026 170127 040000 LDFPS #040000 ; LOAD FPS, INTERRUPT DISABLE
1442 010032 172437 002426 LDF LONUM,AC0 ; LOAD AC0 WITH A RANDOM NUMBER
1443 010036 172537 002436 LDF HINUM,AC1 ; LOAD AC1 WITH A RANDOM NUMBER
1444 010042 172737 002416 LDF ANS2,AC3 ; LOAD AC3 WITH THE SUM
1445 010046 170127 007440 LDFPS #007440 ; TURN INTERRUPTS ON
1446 010052 012737 010060 001110 MOV #.+6,$LPADR ; RESET LOOP ADDRESS
1447
1448
1449

```

```

;*****

```

```

1450 010060 172600 ARET15: LDF AC0,AC2 ; LOAD AC0 INTO AC2
1451 010062 173201 SUBF AC1,AC2 ; SUBTRACT AC1 BY AC2
1452 010064 170237 002362 STFPS FPS ; STORE FLOATING POINT STATUS

```

```

1453 010070 023737 002362 002400      CMP      FPS,$FPS      ;CHECK FPS
1454 010076 001403                BEQ      AERR15        ;BRANCH IF OK
1455 010100 174237 002406                STF      AC2,ANS1     ;SAVE FPU ANSWER
1456 010104 104002                ERROR   2              ;FPS ERROR
1457
1458 010106 005737 002400      AERR15: TST      $FPS      ;ERROR BIT SET ?
1459 010112 100014                BPL     ATST15        ;NO, DONT GET FEC/FEA
1460 010114 170337 002364                STST    FEC           ;YES, CHECK STATUS
1461
1462 010120 023737 002364 002402      CMP      FEC,$FEC     ;CHECK THE FLOATING EXCEPTION CODES
1463 010126 001401                BEQ      1$           ;BRANCH IF OK
1464 010130 104024                ERROR   24           ;FEC IS WRONG
1465
1466 010132 023737 002366 002404      1$:     CMP      FEA,$FEA ;CHECK FLOATING PC
1467 010140 001401                BEQ      ATST15      ;BRANCH IF OK
1468 010142 104024                ERROR   24           ;WRONG ADDRESS IN FEA
1469
1470 010144 173702                ATST15: CMPF     AC2,AC3   ;COMPARE FPU ANSWER TO FORTRAN ANSWER
1471 010146 170000                CFCC                    ;COPY FLOATING CONDITION CODES
1472 010150 001427                BEQ      AEND15      ;ANSWERS CHECK
1473                ;COMPENSATE FOR FORTRAN INACCURACIES.
1474 010152 174237 002406                STF      AC2,ANS1     ;SAVE FPU ANSWER
1475 010156 062737 000001 002410      ADD     #1,ANS1+2     ;INCREMENT FPU ANSWER
1476 010164 005537 002406                ADC     ANS1
1477 010170 173737 002406                CMPF    ANS1,AC3     ;CHECK ANSWERS AGAIN
1478 010174 170000                CFCC                    ;COPY FLOATING CONDITION CODES
1479 010176 001414                BEQ      AEND15      ;BRANCH IF OK
1480 010200 162737 000002 002410      SUB     #2,ANS1+2     ;DECREMENT FPU ANSWER
1481 010206 005637 002406                SBC     ANS1
1482 010212 173737 002406                CMPF    ANS1,AC3     ;CHECK ANSWERS AGAIN
1483 010216 170000                CFCC                    ;COPY FLOATING CONDITION CODES
1484 010220 001403                BEQ      AEND15      ;BRANCH IF OK
1485 010222 174237 002406                STF      AC2,ANS1     ;SAVE FPU ANSWER
1486 010226 104007                ERROR   7              ;FPU AND FORTRAN DISAGREE
1487
1488 010230 005037 002362      AEND15: CLR     FPS      ;CLR FPU FPS BUFFER
1489
1490
1491
1492
1493
1494
1495
1496 010234 000004                ;*****
1497 010236 012737 010370 002376      ;*TEST 16      EXERCISE SUBD, ALL INTERRUPTS ON, TRUNCATE MODE
1498 010244 012737 007640 002400      ;*****
1499 010252 005037 002402                ;*ST16: SCOPE
1500 010256 005037 002404                MOV     #ARET16,EXPFEA ;ADDR OF INSTR BEING TESTED
1501 010262 005037 002362                MOV     #007640,$FPS   ;SET IE BITS IN FORTRAN ANSWER
1502 010266 005037 002364                CLR     $FEC          ;CLR FORTRAN FEC
1503 010272 005037 002366                CLR     $FEA          ;CLR FORTRAN FEA
1504 010276 004737 023332                CLR     FPS          ;CLR FPU FPS BUFFER
1505 010302 002426 002436                CLR     FEC          ;CLR FPU FEC BUFFER
1506 010306 004437 023506                CLR     FEA          ;CLR FPU FEA BUFFER
1507 010312 023510 002426                JSR     PC,RANDL4     ;GET RANDOM INPUT DATA
1508 010316 023510 002436                .WORD  LONUM,HINUM
                JSR     R4,$POLSH ;ENTER POLISH MODE
                $PUSH ,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
                $PUSH ,HINUM ;PUSH 4 WORDS ON STACK (HINUM)

```

```

1509 010322 023562          $SUB          ;ADDRESS OF FORTRAN SUBTRACT
1510 010324 023540 002416 $POP          ;POP 4 WORDS AND EXIT POLISH MODE
1511
1512 010330 013700 002400 MOV $FPS, R0 ;DISPLAY FLOATING POINT STATUS
1513 010334 170127 040200 LDFFS #040200 ;LOAD FPS, INTERRUPT DISABLE AND FD
1514 010340 172437 002426 LDD LONUM, AC0 ;LOAD AC0 WITH A RANDOM NUMBER
1515 010344 172537 002436 LDD HINUM, AC1 ;LOAD AC1 WITH A RANDOM NUMBER
1516 010350 172737 002416 LDD ANS2, AC3 ;LOAD AC3 WITH THE SUM
1517 010354 170127 007640 LDFFS #007640 ;TURN INTERRUPTS ON
1518 010360 012737 010366 001110 MOV #.+6, $LPROR ;RESET LOOP ADDRESS
1519
1520 ;*****
1521
1522 010366 172600          LDD AC0, AC2 ;LOAD AC0 INTO AC2
1523 010370 173201          ARET16: SUBO AC1, AC2 ;SUBTRACT AC1 BY AC2
1524 010372 170237 002362 STFPS ;STORE FLOATING POINT STATUS
1525 010376 023737 002362 002400 CMP FPS, $FPS ;CHECK FPS
1526 010404 001403          BEQ AERR16 ;BRANCH IF OK
1527 010406 174237 002406 STD AC2, ANS1 ;SAVE FPU ANSWER
1528 010412 104020          ERROR 20 ;FPS ERROR
1529
1530 010414 005737 002400 AERR16: TST $FPS ;ERROR BIT SET ?
1531 010420 100014          BPL ATST16 ;NO, DONT GET FEC/FEA
1532 010422 170337 002364 STST FEC ;YES, CHECK STATUS
1533
1534 010426 023737 002364 002402 CMP FEC, $FEC ;CHECK THE FLOATING EXCEPTION CODES
1535 010434 001401          BEQ 15 ;BRANCH IF OK
1536 010436 104030          ERROR 30 ;FEC IS WRONG
1537
1538 010440 023737 002366 002404 15: CMP FEA, $FEA ;CHECK FLOATING PC
1539 010446 001401          BEQ ATST16 ;BRANCH IF OK
1540 010450 104030          ERROR 30 ;WRONG ADDRESS IN FEA
1541
1542 010452 173702          ATST16: CMPD AC2, AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
1543 010454 170000          CFCC ;COPY FLOATING CONDITION CODES
1544 010456 001437          BEQ AEND16 ;ANSWERS CHECK
1545 ;COMPENSATE FOR FORTRAN INACCURACIES.
1546 010460 174237 002406 STD AC2, ANS1 ;SAVE FPU ANSWER
1547 010464 062737 000001 002414 ADD #1, ANS1+6 ;INCREMENT FPU ANSWER
1548 010472 005537 002412 ADC ANS1+4
1549 010476 005537 002410 ADC ANS1+2
1550 010502 005537 002406 ADC ANS1
1551 010506 173737 002406 CMPD ANS1, AC3 ;CHECK ANSWERS AGAIN
1552 010512 170000          CFCC ;COPY FLOATING CONDITION CODES
1553 010514 001420          BEQ AEND16 ;BRANCH IF OK
1554 010516 162737 000002 002414 SUB #2, ANS1+6 ;DECREMENT FPU ANSWER
1555 010524 005637 002412 SBC ANS1+4
1556 010530 005637 002410 SBC ANS1+2
1557 010534 005637 002406 SBC ANS1
1558 010540 173737 002406 CMPD ANS1, AC3 ;CHECK ANSWERS AGAIN
1559 010544 170000          CFCC ;COPY FLOATING CONDITION CODES
1560 010546 001403          BEQ AEND16 ;BRANCH IF OK
1561 010550 174237 002406 STD AC2, ANS1 ;SAVE FPU ANSWER
1562 010554 104010          ERROR 10 ;FPU AND FORTRAN DISAGREE
1563
1564 010556 005037 002362 AEND16: CLR FPS ;CLR FPU FPS BUFFER

```

```

1565
1566
1567
1568
1569
1570
1571
1572 010562 000004
1573 010564 012737 010716 002376
1574 010572 012737 004440 002400
1575 010600 005037 002402
1576 010604 005037 002404
1577 010610 005037 002362
1578 010614 005037 002364
1579 010620 005037 002366
1580 010624 004737 023342
1581 010630 002426 002436
1582 010634 004437 023506
1583 010640 023510 002426
1584 010644 023510 002436
1585 010650 023562
1586 010652 023540 002416
1587
1588 010656 013700 002400
1589 010662 170127 040000
1590 010666 172437 002426
1591 010672 172537 002436
1592 010676 172737 002416
1593 010702 170127 004440
1594 010706 012737 010714 001110
1595
1596
1597
1598 010714 172600
1599 010716 173201
1600 010720 170237 002362 002400
1601 010724 023737 002362
1602 010732 001403
1603 010734 174237 002406
1604 010740 104002
1605
1606 010742 173702
1607 010744 170000
1608 010746 001427
1609
1610 010750 174237 002406
1611 010754 062737 000001 002410
1612 010762 005537 002406
1613 010766 173737 002406
1614 010772 170000
1615 010774 001414
1616 010776 162737 000002 002410
1617 011004 005637 002406
1618 011010 173737 002406
1619 011014 170000
1620 011016 001403

```

```

;*****
;TEST 17 EXERCISE SUBD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
;*****
↑ST17: SCOPE
MOV #ARET17,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #004440,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL2 ;GET RANDOM INPUT DATA
;WORD LONUM,HINUM
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
$SUB ;ADDRESS OF FORTRAN SUBTRACT
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;LOAD FPS, INTERRUPT DISABLE
LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #004440 ;TURN INTERRUPTS ON, EXCEPT OVERFLOW AND UNDERFLOW
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
ARET17: LDF AC0,AC2 ;LOAD AC0 INTO AC2
SUBF AC1,AC2 ;SUBTRACT AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ ATST17 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 2 ;FPS ERROR

ATST17: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND17 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
ADC #1,ANS1+2 ;INCREMENT FPU ANSWER
ADC ANS1
CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND17 ;BRANCH IF OK
SUB #2,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1
CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND17 ;BRANCH IF OK

```


K04

```

1621 011020 174237 002406          STF      AC2,ANS1          ;SAVE FPU ANSWER
1622 011024 104007          ERROR      7              ;FPU AND FORTRAN DISAGREE
1623
1624 011026 005037 002362          AEND17: CLR      FPS              ;CLR FPU FPS BUFFER
1625
1626
1627
1628
1629
1630
1631
1632 011032 000004          ;*****
1633 011034 012737 011166 002376  ;*TEST 20      EXERCISE SUBD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
1634 011042 012737 004640 002400  ;*****
1635 011050 005037 002402          ;*ST20: SCOPE
1636 011054 005037 002404          MOV      #ARET20,EXPFEA    ;ADDR OF INSTR BEING TESTED
1637 011060 005037 002362          MOV      #004640,$FPS     ;SET IE BITS IN FORTRAN ANSWER
1638 011064 005037 002364          CLR      $FEC             ;CLR FORTRAN FEC
1639 011070 005037 002366          CLR      $FEA             ;CLR FORTRAN FEA
1640 011074 004737 023332          CLR      FPS              ;CLR FPU FPS BUFFER
1641 011100 002426 002436          CLR      FEC             ;CLR FPU FEC BUFFER
1642 011104 004437 023506          CLR      FEA             ;CLR FPU FEA BUFFER
1643 011110 023510 002426          JSR      PC,RANDL4        ;GET RANDOM INPUT DATA
1644 011114 023510 002436          WORD    LONUM,HINUM      ;
1645 011120 023562          JSR      R4,$POLSH        ;ENTER POLISH MODE
1646 011122 023540 002416          $PUSH   ,LONUM            ;PUSH 4 WORDS ON STACK (LONUM)
1647                                     $PUSH   ,HINUM            ;PUSH 4 WORDS ON STACK (HINUM)
1648                                     $SUB    ,ANS2             ;ADDRESS OF FORTRAN SUBTRACT
1649                                     $POPX   ,ANS2             ;POP 4 WORDS AND EXIT POLISH MODE
1650 011136 172437 002426          MOV      $FPS,R0          ;DISPLAY FLOATING POINT STATUS
1651 011142 172537 002436          LDFPS   #040200          ;LOAD FPS, INTERRUPT DISABLE AND FD
1652 011146 172737 002416          LOD     LONUM,AC0         ;LOAD AC0 WITH A RANDOM NUMBER
1653 011152 170127 004640          LOD     HINUM,AC1         ;LOAD AC1 WITH A RANDOM NUMBER
1654 011156 012737 011164 001110  ;LOAD AC3 WITH THE SUM
1655                                     LDFPS   #004640          ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
1656                                     MOV      #.+6,$LPADR     ;RESET LOOP ADDRESS
1657                                     ;*****
1658 011164 172600          ARET20: LOD     AC0,AC2    ;LOAD AC0 INTO AC2
1659 011166 173201          SUBD    AC1,AC2          ;SUBTRACT AC1 BY AC2
1660 011170 170237 002362          STFPS   FPS              ;STORE FLOATING POINT STATUS
1661 011174 023737 002362 002400  ;CHECK FPS
1662 011202 001403          BEQ     ATST20           ;BRANCH IF OK
1663 011204 174237 002406          ST      AC2,ANS1         ;SAVE FPU ANSWER
1664 011210 104020          ST      20              ;FPS ERROR
1665
1666 011212 173702          ATST20: CM      AC2,AC3   ;COMPARE FPU ANSWER TO FORTRAN ANSWER
1667 011214 170000          CF      ;COPY FLOATING CONDITION CODES
1668 011216 001437          AEND20 ;ANSWERS CHECK
1669                                     MPENSATE FOR FORTRAN INACCURACIES
1670 011220 174237 002406          LD      AC2,ANS1         ;SAVE FPU ANSWER
1671 011224 062737 000001 002414  ;INCREMENT FPU ANSWER
1672 011232 005537 002412          ADD     #1,ANS1+6
1673 011236 005537 002410          ADD     ANS1+4
1674 011242 005537 002406          ADD     ANS1+2
1675 011246 173737 002406          ADD     ANS1
1676 011252 170000          CMPD   ANS1,AC3          ;CHECK ANSWERS AGAIN
1676                                     CFCC   ;COPY FLOATING CONDITION CODES

```



```

1690
1691
1692
1693 011322 000004
1694 011324 012737 011456 002376
1695 011332 012737 007400 002400
1696 011340 005037 002402
1697 011344 005037 002404
1698 011350 005037 002362
1699 011354 005037 002364
1700 011360 005037 002366
1701 011364 004737 023342
1702 011370 002426 002436
1703 011374 004437 023506
1704 011400 023510 002426
1705 011404 023510 002436
1706 011410 025244
1707 011412 023540 002416
1708
1709 011416 013700 002400
1710 011422 170127 040000
1711 011426 172437 002426
1712 011432 172537 002436
1713 011436 172737 002416
1714 011442 170127 007400
1715 011446 012737 011454 001110
1716
1717
1718
1719 011454 172600
1720 011456 171201
1721 011460 170237 002362
1722 011464 023737 002362 002400
1723 011472 001403
1724 011474 174237 002406
1725 011500 104003
1726
1727 011502 005737 002400
1728 011506 100014
1729 011510 170337 002364
1730
1731 011514 023737 002364 002402
1732 011522 001401
1733 011524 104025
1734
1735 011526 023737 002366 002404
1736 011534 001401
1737 011536 104025
1738
1739 011540 173702
1740 011542 170000
1741 011544 001416
1742
1743 011546 174237 002406
1744 011552 162737 000001 002410
1745 011560 005637 002406

```

```

;*****
;#TEST 21 EXERCISE MULF, ALL INTERRUPTS ON, ROUNDING MODE
;*****
↑ST21: SCOPE
MOV #MRET1,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007400,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL2 ;GET RANDOM INPUT DATA
;WORD
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
$MUL ;ADDRESS OF FORTRAN MULTIPLY
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;CLEAR THE FPS, INTERRUPT DISABLE
LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007400 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
MRET1: LDF AC0,AC2 ;LOAD AC0 INTO AC2
MULF AC1,AC2 ;MULTIPLY AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ MERR1 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 3 ;FPS ERROR

MERR1: TST $FPS ;ERROR BIT SET ?
BPL MTST1 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

1$: CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 25 ;FEC IS WRONG

2$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ MTST1 ;BRANCH IF OK
ERROR 25 ;WRONG ADDRESS IN FEA

MTST1: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ MEND1 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1

```

1746	011564	173737	002406		CMPF	ANS1,AC3	;CHECK ANSWERS AGAIN
1747	011570	170000			CFCC		;COPY FLOATING CONDITION CODES
1748	011572	001403			BEQ	MEND1	;BRANCH IF OK
1749	011574	174237	002406		STF	AC2,ANS1	;SAVE FPU ANSWER
1750	011600	104011			ERROR	11	;FPU AND FORTRAN DISAGREE
1751							
1752	011602	005037	002362		MEND1:	CLR FPS	;CLEAR FPP FPS BUFFER
1753							
1754							
1755							
1756							
1757							
1758							
1759							
1760	011606	000004			†ST22:	SCOPE	
1761	011610	012737	011742	002376	MOV	#MRET2,EXPFEA	;ADDR OF INSTR BEING TESTED
1762	011616	012737	007600	002400	MOV	#007600,\$FPS	;SET IE BITS IN FORTRAN ANSWER
1763	011624	005037	002402		CLR	\$FEC	;CLR FORTRAN FEC
1764	011630	005037	002404		CLR	\$FEA	;CLR FORTRAN FEA
1765	011634	005037	002362		CLR	FPS	;CLR FPU FPS BUFFER
1766	011640	005037	002364		CLR	FEC	;CLR FPU FEC BUFFER
1767	011644	005037	002366		CLR	FEA	;CLR FPU FEA BUFFER
1768	011650	004737	023332		JSR	PC,RANDL4	;GET RANDOM INPUT DATA
1769	011654	002426	002436		.WORD	LONUM,HINUM	
1770	011660	004437	023506		JSR	R4,\$POLSH	;ENTER POLISH MODE
1771	011664	023510	002426		\$PUSH	,LONUM	;PUSH 4 WORDS ON STACK (LONUM)
1772	011670	023510	002436		\$PUSH	,HINUM	;PUSH 4 WORDS ON STACK (HINUM)
1773	011674	025244			\$MUL		;ADDRESS OF FORTRAN MULTIPLY
1774	011676	023540	002416		\$POPX	,ANS2	;POP 4 WORDS AND EXIT POLISH MODE
1775							
1776	011702	013700	002400		MOV	\$FPS,RO	;DISPLAY FLOATING POINT STATUS
1777	011706	170127	040200		LDFPS	#040200	;SET FD OF FPS ONLY, INTERRUPT DISABLE
1778	011712	172437	002426		LDD	LONUM,ACO	;LOAD ACO WITH A RANDOM NUMBER
1779	011716	172537	002436		LDD	HINUM,AC1	;LOAD AC1 WITH A RANDOM NUMBER
1780	011722	172737	002416		LDD	ANS2,AC3	;LOAD AC3 WITH THE SUM
1781	011726	170127	007600		LDFPS	#007600	;TURN INTERRUPTS ON
1782	011732	012737	011740	001110	MOV	#+6,\$LPADR	;RESET LOOP ADDRESS
1783							
1784							
1785							
1786	011740	172600					
1787	011742	171201			MRET2:	LDD ACO,AC2	;LOAD ACO INTO AC2
1788	011744	170237	002362		MULD	AC1,AC2	;MULTIPLY AC1 BY AC2
1789	011750	023737	002362	002400	STFPS	FPS	;STORE FLOATING POINT STATUS
1790	011756	001403			CMP	FPS,\$FPS	;CHECK FPS
1791	011760	174237	002406		BEQ	MERR2	;BRANCH IF OK
1792	011764	104021			STD	AC2,ANS1	;SAVE FPU ANSWER
1793					ERROR	21	;FPS ERROR
1794	011766	005737	002400		MERR2:	TST \$FPS	;ERROR BIT SET ?
1795	011772	100014			BPL	MTST2	;NO, DONT GET FEC/FEA
1796	011774	170337	002364		STST	FEC	;YES, CHECK STATUS
1797							
1798	012000	023737	002364	002402	CMP	FEC,\$FEC	;CHECK THE FLOATING EXCEPTION CODES
1799	012006	001401			BEQ	15	;BRANCH IF OK
1800	012010	104031			ERROR	31	;FEC IS WRONG
1801							

```

1802 012012 023737 002366 002404 18:  CMP      FEA,SFEA      ;CHECK FLOATING PC
1803 012020 001401          BEQ      MTST2        ;BRANCH IF OK
1804 012022 104031          ERROR    31          ;WRONG ADDRESS IN FEA
1805
1806 012024 173702          MTST2:  CMPO     AC2,AC3      ;COMPARE FPU ANSWER TO FORTRAN ANSWER
1807 012026 170000          CFCC                    ;COPY FLOATING CONDITION CODES
1808 012030 001422          BEQ      MEND2        ;ANSWERS CHECK
1809          ;COMPENSATE FOR FORTRAN INACCURACIES.
1810 012032 174237 002406          STD     AC2,ANS1      ;SAVE FPU ANSWER
1811 012036 162737 000001 002414          SUB     #1,ANS1+6     ;DECREMENT FPU ANSWER
1812 012044 005637 002412          SBC     ANS1+4
1813 012050 005637 002410          SBC     ANS1+2
1814 012054 005637 002406          SBC     ANS1
1815 012060 173737 002406          CMPO     ANS1,AC3      ;CHECK ANSWERS AGAIN
1816 012064 170000          CFCC                    ;COPY FLOATING CONDITION CODES
1817 012066 001403          BEQ      MEND2        ;BRANCH IF OK
1818 012070 174237 002406          STD     AC2,ANS1      ;SAVE FPU ANSWER
1819 012074 104012          ERROR    12          ;FPU AND FORTRAN DISAGREE
1820
1821 012076 005037 002362          MEND2:  CLR     FPS      ;CLEAR FPP FPS BUFFER
1822
1823
1824
1825
1826
1827
1828
1829 012102 000004          ;*****
1830 012104 012737 012236 002376          ;*TEST 23 EXERCISE MULF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
1831 012112 012737 004400 002400          ;*****
1832 012120 005037 002402          ;ST23:  SCOPE
1833 012124 005037 002404          MOV     #MRET3,EXPFEA ;ADDR OF INSTR BEING TESTED
1834 012130 005037 002362          MOV     #004400,$FPS  ;SET IE BITS IN FORTRAN ANSWER
1835 012134 005037 002364          CLR     $FEC          ;CLR FORTRAN FEC
1836 012140 005037 002366          CLR     $FEA          ;CLR FORTRAN FEA
1837 012144 004737 023342          CLR     FPS          ;CLR FPU FPS BUFFER
1838 012150 002426 002436          CLR     FEC          ;CLR FPU FEC BUFFER
1839 012154 004437 023506          CLR     FEA          ;CLR FPU FEA BUFFER
1840 012160 023510 002426          JSR     PC,RANDL2     ;GET RANDOM INPUT DATA
1841 012164 023510 002436          .WORD  LONUM,HINUM
1842 012170 025244          JSR     R4,$POLSH    ;ENTER POLISH MODE
1843 012172 023540 002416          $PUSH  ,LONUM        ;PUSH 2 WORDS ON STACK (LONUM)
1844          $PUSH  ,HINUM        ;PUSH 2 WORDS ON STACK (HINUM)
1845 012176 013700 002400          $MUL   ,ANS2         ;ADDRESS OF FORTRAN MULTIPLY
1846 012202 170127 040000          $POPX  ,ANS2         ;POP 2 WORDS AND EXIT POLISH MODE
1847 012206 172437 002426          MOV     $FPS,R0      ;DISPLAY FLOATING POINT STATUS
1848 012212 172537 002436          LDFPS  #040000       ;CLEAR THE FPS, INTERRUPT DISABLE
1849 012216 172737 002416          LDF    LONUM,AC0     ;LOAD AC0 WITH A RANDOM NUMBER
1850 012222 170127 004400          LDF    HINUM,AC1     ;LOAD AC1 WITH A RANDOM NUMBER
1851 012226 012737 012234 001110          LDF    ANS2,AC3      ;LOAD AC3 WITH THE SUM
1852          LDFPS  #004400     ;TURN INTERRUPTS ON, EXCEPT OVERFLOW AND UNDERFLOW
1853          MOV     #. +6,$LPAOR ;RESET LOOP ADDRESS
1854          ;*****
1855 012234 172600          MRET3:  LDF     AC0,AC2      ;LOAD AC0 INTO AC2
1856 012236 171201          MULF   AC1,AC2       ;MULTIPLY AC1 BY AC2
1857 012240 170237 002362          STFPS  FPS          ;STORE FLOATING POINT STATUS

```

```

1858 012244 023737 002362 002400      CMP      FPS,$FPS      ;CHECK FPS
1859 012252 001403          BEQ      MTST3         ;BRANCH IF OK
1860 012254 174237 002406          STF      AC2,ANS1     ;SAVE FPU ANSWER
1861 012260 104003          ERROR   3            ;FPS ERROR
1862
1863 012262 173702          MTST3:  CMPF     AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
1864 012264 170000          CFCC          ;COPY FLOATING CONDITION CODES
1865 012266 001416          BEQ      MEND3        ;ANSWERS CHECK
1866          ;COMPENSATE FOR FORTRAN INACCURACIES.
1867 012270 174237 002406          STF      AC2,ANS1     ;SAVE FPU ANSWER
1868 012274 162737 000001 002410          SUB      #1,ANS1+2    ;DECREMENT FPU ANSWER
1869 012302 005637 002406          SBC      ANS1
1870 012306 173737 002406          CMPF     ANS1,AC3     ;CHECK ANSWERS AGAIN
1871 012312 170000          CFCC          ;COPY FLOATING CONDITION CODES
1872 012314 001403          BEQ      MEND3        ;BRANCH IF OK
1873 012316 174237 002406          STF      AC2,ANS1     ;SAVE FPU ANSWER
1874 012322 104011          ERROR   11          ;FPU AND FORTRAN DISAGREE
1875
1876 012324 005037 002362          MEND3:  CLR      FPS      ;CLEAR FPP FPS BUFFER
1877
1878
1879
1880
1881
1882
1883          ;*****
1884          ;TEST 24 EXERCISE MULF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
1885          ;*****
1886          ;ST24: SCOPE
1887          MOV      #MRET4,EXPFEA ;ADDR OF INSTR BEING TESTED
1888          MOV      #004600,$FPS ;SET IE BITS IN FORTRAN ANSWER
1889          CLR      $FEC          ;CLR FORTRAN FEC
1890          CLR      $FEA          ;CLR FORTRAN FEA
1891          CLR      FPS          ;CLR FPU FPS BUFFER
1892          CLR      FEC          ;CLR FPU FEC BUFFER
1893          CLR      FEA          ;CLR FPU FEA BUFFER
1894          JSR      PC,RANDL4     ;GET RANDOM INPUT DATA
1895          .WORD  LONUM,HINUM
1896          JSR      R4,SPOLSH    ;ENTER POLISH MODE
1897          SPUSH  ,LONUM         ;PUSH 4 WORDS ON STACK (LONUM)
1898          SPUSH  ,HINUM         ;PUSH 4 WORDS ON STACK (HINUM)
1899          SMUL   ,ANS2          ;ADDRESS OF FORTRAN MULTIPLY
1900          SPOPX  ,ANS2          ;POP 4 WORDS AND EXIT POLISH MODE
1901
1902          MOV      $FPS,R0       ;DISPLAY FLOATING POINT STATUS
1903          LDFPS  #040200        ;SET FD OF FPS ONLY, INTERRUPT DISABLE
1904          LDD   LONUM,AC0       ;LOAD AC0 WITH A RANDOM NUMBER
1905          LDD   HINUM,AC1       ;LOAD AC1 WITH A RANDOM NUMBER
1906          LDD   ANS2,AC3        ;LOAD AC3 WITH THE SUM
1907          LDFPS  #004600        ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
1908          MOV      #.+6,$LPAOR   ;RESET LOOP ADDRESS
1909
1910          ;*****
1911          LDD   AC0,AC2          ;LOAD AC0 INTO AC2
1912          MULD  AC1,AC2          ;MULTIPLY AC1 BY AC2
1913          STFPS FPS             ;STORE FLOATING POINT STATUS
1914          CMP   FPS,$FPS        ;CHECK FPS

```

```

1914 012500 001403 BEQ MTST4 ; BRANCH IF OK
1915 012502 174237 002406 STD AC2,ANS1 ; SAVE FPU ANSWER
1916 012506 104021 ERROR 21 ; FPS ERROR
1917
1918 012510 173702 MTST4: CMPD AC2,AC3 ; COMPARE FPU ANSWER TO FORTRAN ANSWER
1919 012512 170000 CFCC ; COPY FLOATING CONDITION CODES
1920 012514 001422 BEQ MEND4 ; ANSWERS CHECK
1921 ; COMPENSATE FOR FORTRAN INACCURACIES.
1922 012516 174237 002406 STD AC2,ANS1 ; SAVE FPU ANSWER
1923 012522 162737 000001 002414 SUB #1,ANS1+6 ; DECREMENT FPU ANSWER
1924 012530 005637 002412 SBC ANS1+4
1925 012534 005637 002410 SBC ANS1+2
1926 012540 005637 002406 SBC ANS1
1927 012544 173737 002406 CMPD ANS1,AC3 ; CHECK ANSWERS AGAIN
1928 012550 170000 CFCC ; COPY FLOATING CONDITION CODES
1929 012552 001403 BEQ MEND4 ; BRANCH IF OK
1930 012554 174237 002406 STD AC2,ANS1 ; SAVE FPU ANSWER
1931 012560 104012 ERROR 12 ; FPU AND FORTRAN DISAGREE
1932
1933 012562 005037 002362 MEND4: CLR FPS ; CLEAR FPP FPS BUFFER

```

```

; *****
; *TEST 25 EXERCISE MULF, ALL INTERRUPTS ON, TRUNCATE MODE
; *****
↑ST25:

```

```

1940 012566 000004 SCOPE
1941 012570 012737 012722 002376 MOV #MRETS,EXPFEA ; ADDR OF INSTR BEING TESTED
1942 012576 012737 007440 002400 MOV #007440,$FPS ; SET IE BITS IN FORTRAN ANSWER
1943 012604 005037 002402 CLR $FEC ; CLR FORTRAN FEC
1944 012610 005037 002404 CLR $FEA ; CLR FORTRAN FEA
1945 012614 005037 002362 CLR FPS ; CLR FPU FPS BUFFER
1946 012620 005037 002364 CLR FEC ; CLR FPU FEC BUFFER
1947 012624 005037 002366 CLR FEA ; CLR FPU FEA BUFFER
1948 012630 004737 023342 JSR PC,RANDL2 ; GET RANDOM INPUT DATA
1949 012634 002426 002436 .WORD LONUM,HINUM
1950 012640 004437 023506 JSR #M,$POLSH ; ENTER POLISH MODE
1951 .44 023510 002426 SPUSH ,LONUM ; PUSH 2 WORDS ON STACK (LONUM)
1952 .650 023510 002436 SPUSH ,HINUM ; PUSH 2 WORDS ON STACK (HINUM)
1953 012654 025244 SMUL ; ADDRESS OF FORTRAN MULTIPLY
1954 012656 023540 002416 SPOPX ,ANS2 ; POP 2 WORDS AND EXIT POLISH MODE
1955
1956 012662 013700 002400 MOV $FPS,R0 ; DISPLAY FLOATING POINT STATUS
1957 012666 170127 040000 LDFPS #040000 ; CLEAR THE FPS, INTERRUPT DISABLE
1958 012672 172437 002426 LDF LONUM,AC0 ; LOAD AC0 WITH A RANDOM NUMBER
1959 012676 172537 002436 LDF HINUM,AC1 ; LOAD AC1 WITH A RANDOM NUMBER
1960 012702 172737 002416 LDF ANS2,AC3 ; LOAD AC3 WITH THE SUM
1961 012706 170127 007440 LDFPS #007440 ; TURN INTERRUPTS ON
1962 012712 012737 012720 001110 MOV #. +6,SLPADR ; RESET LOOP ADDRESS

```

```

; *****

```

```

1966 012720 172600 MRETS: LDF AC0,AC2 ; LOAD AC0 INTO AC2
1967 012722 171201 MULF AC1,AC2 ; MULTIPLY AC1 BY AC2
1968 012724 170237 002362 STFPS FPS ; STORE FLOATING POINT STATUS
1969 012730 023737 002362 002400 CMP FPS,$FPS ; CHECK FPS

```


E05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 17:30
DQFPD1.P11 04-MAY-77 17:30

04-MAY-77 18:18 PAGE 41
EXERCISE MULF, ALL INTERRUPTS ON, TRUNCATE MODE

SEQ 0042

1970	012736	001403			BEQ	MERRS		; BRANCH IF OK
1971	012740	174237	002406		STF	AC2,ANS1		; SAVE FPU ANSWER
1972	012744	104003			ERROR	3		; FPS ERROR
1973								
1974	012746	005737	002400		MERRS:	TST	\$FPS	; ERROR BIT SET ?
1975	012752	100014				BPL	MTSTS	; NO, DONT GET FEC/FEA
1976	012754	170337	002364			STST	FEC	; YES, CHECK STATUS
1977								
1978	012760	023737	002364	002402		CHP	FEC,\$FEC	; CHECK THE FLOATING EXCEPTION CODES
1979	012766	001401				BEQ	1\$; BRANCH IF OK
1980	012770	104025				ERROR	2\$; FEC IS WRONG
1981								
1982	012772	023737	002366	002404	1\$:	CHP	FEA,\$FEA	; CHECK FLOATING PC
1983	013000	001401				BEQ	MTSTS	; BRANCH IF OK
1984	013002	104025				ERROR	2\$; WRONG ADDRESS IN FEA
1985								
1986	013004	173702			MTSTS:	CHPF	AC2,AC3	; COMPARE FPU ANSWER TO FORTRAN ANSWER
1987	013006	170000				CFCC		; COPY FLOATING CONDITION CODES
1988	013010	001427				BEQ	MENDS	; ANSWERS CHECK
1989								; COMPENSATE FOR FORTRAN INACCURACIES.
1990	013012	174237	002406			STF	AC2,ANS1	; SAVE FPU ANSWER
1991	013016	062737	000001	002410		ADD	#1,ANS1+2	; INCREMENT FPU ANSWER
1992	013024	005537	002406			ADC	ANS1	
1993	013030	173737	002406			CHPF	ANS1,AC3	; CHECK ANSWERS AGAIN
1994	013034	170000				CFCC		; COPY FLOATING CONDITION CODES
1995	013036	001414				BEQ	MENDS	; BRANCH IF OK
1996	013040	162737	000002	002410		SUB	#2,ANS1+2	; DECREMENT FPU ANSWER
1997	013046	005637	002406			SBC	ANS1	
1998	013052	173737	002406			CHPF	ANS1,AC3	; CHECK ANSWERS AGAIN
1999	013056	170000				CFCC		; COPY FLOATING CONDITION CODES
2000	013060	001403				BEQ	MENDS	; BRANCH IF OK
2001	013062	174237	002406			STF	AC2,ANS1	; SAVE FPU ANSWER
2002	013066	104011				ERROR	11	; FPU AND FORTRAN DISAGREE
2003								
2004	013070	005037	002362		MENDS:	CLR	FPS	; CLEAR FPP FPS BUFFER
2005								
2006								
2007								
2008								
2009								
2010								
2011								
2012	013074	000004						
2013	013076	012737	0132 J0	002376				
2014	013104	012737	007540	002400				
2015	013112	005037	002402					
2016	013116	005037	002404					
2017	013122	005037	002362					
2018	013126	005037	002364					
2019	013132	005037	002366					
2020	013136	004737	023332					
2021	013142	002426	002436					
2022	013146	004437	023506					
2023	013152	023510	002426					
2024	013156	023510	002436					
2025	013162	025244						

```

;*****
;TEST 26 EXERCISE MULD, ALL INTERRUPTS ON, TRUNCATE MODE
;*****
†ST26: SCOPE
MOV #MRET6,EXPFEA ; ADDR OF INSTR BEING TESTED
MOV #007640,$FPS ; SET IE BITS IN FORTRAN ANSWER
CLR $FEC ; CLR FORTRAN FEC
CLR $FEA ; CLR FORTRAN FEA
CLR FPS ; CLR FPU FPS BUFFER
CLR FEC ; CLR FPU FEC BUFFER
CLR FEA ; CLR FPU FEA BUFFER
JSR PC,RANDL4 ; GET RANDOM INPUT DATA
;
; WORD
JSR R4,$POLSH ; ENTER POLISH MODE
$PUSH ,LONUM ; PUSH 4 WORDS ON STACK (LONUM)
$PUSH ,HINUM ; PUSH 4 WORDS ON STACK (HINUM)
$MUL ; ADDRESS OF FORTRAN MULTIPLY

```

F05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 42
 DDFP08.P11 04-MAY-77 17:30 T26

EXERCISE MULD, ALL INTERRUPTS ON, TRUNCATE MODE

SEQ 0043

```

2026 013164 023540 002416 SPOPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE
2027
2028 013170 013700 002400 MOV SFPS,RO ;DISPLAY FLOATING POINT STATUS
2029 013174 170127 040200 LDFPS #040200 ;SET FD OF FPS ONLY, INTERRUPT DISABLE
2030 013200 172437 002426 LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
2031 013204 172537 002436 LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
2032 013210 172737 002416 LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
2033 013214 170127 007640 LDFPS #007640 ;TURN INTERRUPTS ON
2034 013220 012737 013226 001110 MOV #. +6,SLPADR ;RESET LOOP ADDRESS
2035
2036 ;*****
2037
2038 013226 172600 LDD AC0,AC2 ;LOAD AC0 INTO AC2
2039 013230 171201 MRET6: MULD AC1,AC2 ;MULTIPLY AC1 BY AC2
2040 013232 170237 002362 STFPS FPS ;STORE FLOATING POINT STATUS
2041 013236 023737 002362 002400 CMP FPS,SFPS ;CHECK FPS
2042 013244 001403 BEQ MERR6 ;BRANCH IF OK
2043 013246 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
2044 013252 104021 ERROR 21 ;FPS ERROR
2045
2046 013254 005737 002400 MERR6: TST SFPS ;ERROR BIT SET ?
2047 013260 100014 BPL MTST6 ;NO, DONT GET FEC/FEA
2048 013262 170337 002364 STST FEC ;YES, CHECK STATUS
2049
2050 013266 023737 002364 002402 CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
2051 013274 001401 BEQ 1$ ;BRANCH IF OK
2052 013276 104031 ERROR 31 ;FEC IS WRONG
2053
2054 013300 023737 002366 002404 1$: CMP FEA,$FEA ;CHECK FLOATING PC
2055 013306 001401 BEQ MTST6 ;BRANCH IF OK
2056 013310 104031 ERROR 31 ;WRONG ADDRESS IN FEA
2057
2058 013312 173702 MTST6: CMPO AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2059 013314 170000 CFCC ;COPY FLOATING CONDITION CODES
2060 013316 001437 BEQ MEND6 ;ANSWERS CHECK
2061 ;COMPENSATE FOR FORTRAN INACCURACIES.
2062 013320 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
2063 013324 062737 000001 002414 ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
2064 013332 005537 002412 ADC ANS1+4
2065 013336 005537 002410 ADC ANS1+2
2066 013342 005537 002406 ADC ANS1
2067 013346 173737 002406 CMPO ANS1,AC3 ;CHECK ANSWERS AGAIN
2068 013352 170000 CFCC ;COPY FLOATING CONDITION CODES
2069 013354 001420 BEQ MEND6 ;BRANCH IF OK
2070 013356 162737 000002 002414 SUB #2,ANS1+6 ;DECREMENT FPU ANSWER
2071 013364 005637 002412 SBC ANS1+4
2072 013370 005637 002410 SBC ANS1+2
2073 013374 005637 002406 SBC ANS1
2074 013400 173737 002406 CMPO ANS1,AC3 ;CHECK ANSWERS AGAIN
2075 013404 170000 CFCC ;COPY FLOATING CONDITION CODES
2076 013406 001403 BEQ MEND6 ;BRANCH IF OK
2077 013410 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
2078 013414 104012 ERROR 12 ;FPU AND FORTRAN DISAGREE
2079
2080 013416 005037 002362 MEND6: CLR FPS ;CLEAR FPP FPS BUFFER
2081
  
```

G05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 43
 DAFP08.P11 04-MAY-77 17:30 T26

EXERCISE MULF, ALL INTERRUPTS ON, TRUNCATE MODE

SEQ 0044

2082
 2083
 2084
 2085
 2086
 2087
 2088 013422 000004
 2089 013424 012737 013556 002376
 2090 013432 012737 004440 002400
 2091 013440 005037 002402
 2092 013444 005037 002404
 2093 013450 005037 002362
 2094 013454 005037 002364
 2095 013460 005037 002366
 2096 013464 004737 023342
 2097 013470 002426 002436
 2098 013474 004437 023506
 2099 013500 023510 002426
 2100 013504 023510 002436
 2101 013510 025244
 2102 013512 023540 002416
 2103
 2104 013516 013700 002400
 2105 013522 170127 040000
 2106 013526 172437 002426
 2107 013532 172537 002436
 2108 013536 172737 002416
 2109 013542 170127 004440
 2110 013546 012737 013554 001110
 2111
 2112
 2113
 2114 013554 172600
 2115 013556 171201
 2116 013560 170237 002362
 2117 013564 023737 002362 002400
 2118 013572 001403
 2119 013574 174237 002406
 2120 013600 104003
 2121
 2122 013602 173702
 2123 013604 170000
 2124 013606 001427
 2125
 2126 013610 174237 002406
 2127 013614 062737 000001 002410
 2128 013622 005537 002406
 2129 013626 173737 002406
 2130 013632 170000
 2131 013634 001414
 2132 013636 162737 000002 002410
 2133 013644 005637 002406
 2134 013650 173737 002406
 2135 013654 170000
 2136 013656 001403
 2137 013660 174237 002406

 *TEST 27 EXERCISE MULF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

†ST27: SCOPE
 MOV #MRET7,EXPFEA ; ADDR OF INSTR BEING TESTED
 MOV #004440,\$FPS ; SET IE BITS IN FORTRAN ANSWER
 CLR \$FEC ; CLR FORTRAN FEC
 CLR \$FEA ; CLR FORTRAN FEA
 CLR FPS ; CLR FPU FPS BUFFER
 CLR FEC ; CLR FPU FEC BUFFER
 CLR FEA ; CLR FPU FEA BUFFER
 JSR PC,RANDL2 ; GET RANDOM INPUT DATA
 .WORD LONUM,HINUM
 JSR R4,\$POLSH ; ENTER POLISH MODE
 \$PUSH ,LONUM ; PUSH 2 WORDS ON STACK (LONUM)
 \$PUSH ,HINUM ; PUSH 2 WORDS ON STACK (HINUM)
 \$MUL ; ADDRESS OF FORTRAN MULTIPLY
 \$POPX ,ANS2 ; POP 2 WORDS AND EXIT POLISH MODE
 MOV \$FPS,R0 ; DISPLAY FLOATING POINT STATUS
 LDFPS #040000 ; CLEAR THE FPS, INTERRUPT DISABLE
 LDF LONUM,AC0 ; LOAD AC0 WITH A RANDOM NUMBER
 LDF HINUM,AC1 ; LOAD AC1 WITH A RANDOM NUMBER
 LDF ANS2,AC3 ; LOAD AC3 WITH THE SUM
 LDFPS #004440 ; TURN INTERRUPTS ON, EXCEPT OVERFLOW AND UNDERFLOW
 MOV #.+6,\$LPADR ; RESET LOOP ADDRESS

MRET7: LDF AC0,AC2 ; LOAD AC0 INTO AC2
 MULF AC1,AC2 ; MULTIPLY AC1 BY AC2
 STFPS FPS ; STORE FLOATING POINT STATUS
 CMP FPS,\$FPS ; CHECK FPS
 BEQ MTST7 ; BRANCH IF OK
 STF AC2,ANS1 ; SAVE FPU ANSWER
 ERROR 3 ; FPS ERROR

MTST7: CMPF AC2,AC3 ; COMPARE FPU ANSWER TO FORTRAN ANSWER
 CFCC ; COPY FLOATING CONDITION CODES
 BEQ MEND7 ; ANSWERS CHECK
 ; COMPENSATE FOR FORTRAN INACCURACIES.
 STF AC2,ANS1 ; SAVE FPU ANSWER
 ADD #1,ANS1+2 ; INCREMENT FPU ANSWER
 ADC ANS1
 CMPF ANS1,AC3 ; CHECK ANSWERS AGAIN
 CFCC ; COPY FLOATING CONDITION CODES
 BEQ MEND7 ; BRANCH IF OK
 SUB #2,ANS1+2 ; DECREMENT FPU ANSWER
 SBC ANS1
 CMPF ANS1,AC3 ; CHECK ANSWERS AGAIN
 CFCC ; COPY FLOATING CONDITION CODES
 BEQ MEND7 ; BRANCH IF OK
 STF AC2,ANS1 ; SAVE FPU ANSWER

H05

2138 013664 104011
2139
2140 013666 005037 002362
2141
2142
2143
2144
2145
2146
2147
2148 013672 000004
2149 013674 012737 014026 002376
2150 013702 012737 004640 002400
2151 013710 005037 002402
2152 013714 005037 002404
2153 013720 005037 002362
2154 013724 005037 002364
2155 013730 005037 002366
2156 013734 004737 023332
2157 013740 002426 002436
2158 013744 004437 023506
2159 013750 023510 002426
2160 013754 023510 002436
2161 013760 025244
2162 013762 023540 002416
2163
2164 013766 013700 002400
2165 013772 170127 040200
2166 013776 172437 002426
2167 014002 172537 002436
2168 014006 172737 002416
2169 014012 170127 004640
2170 014016 012737 014024 001110
2171
2172
2173
2174 014024 172600
2175 014026 171201
2176 014030 170237 002362
2177 014034 023737 002362 002400
2178 014042 001403
2179 014044 174237 022406
2180 014050 104021
2181
2182 014052 173702
2183 014054 170000
2184 014056 001437
2185
2186 014060 174237 002406
2187 014064 062737 000001 002414
2188 014072 005537 002412
2189 014076 005537 002410
2190 014102 005537 002406
2191 014106 173737 002406
2192 014112 170000
2193 014114 001420

ERROR 11 ; FPU AND FORTRAN DISAGREE
MEND7: CLR FPS ; CLEAR FPP FPS BUFFER
;*****
;TEST 30 EXERCISE MULF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
;*****
↑ST30: SCOPE
MOV #MRET10,EXPFEA ; ADDR OF INSTR BEING TESTED
MOV #004640,\$FPS ; SET IE BITS IN FORTRAN ANSWER
CLR \$FEC ; CLR FORTRAN FEC
CLR \$FEA ; CLR FORTRAN FEA
CLR FPS ; CLR FPU FPS BUFFER
CLR FEC ; CLR FPU FEC BUFFER
CLR FEA ; CLR FPU FEA BUFFER
JSR PC,RANDL4 ; GET RANDOM INPUT DATA
;WORD
LONUM,HINUM ;
JSR R4,\$POLSH ; ENTER POLISH MODE
\$PUSH ,LONUM ; PUSH 4 WORDS ON STACK (LONUM)
\$PUSH ,HINUM ; PUSH 4 WORDS ON STACK (HINUM)
\$MUL ; ADDRESS OF FORTRAN MULTIPLY
\$POPX ,ANS2 ; POP 4 WORDS AND EXIT POLISH MODE
MOV \$FPS,R0 ; DISPLAY FLOATING POINT STATUS
LDFPS #040200 ; SET FD OF FPS ONLY, INTERRUPT DISABLE
LDD LONUM,AC0 ; LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ; LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ; LOAD AC3 WITH THE SUM
LDFPS #004640 ; TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
MOV #.+6,\$LPADR ; RESET LOOP ADDRESS
;*****
MRET10: LDD AC0,AC2 ; LOAD AC0 INTO AC2
MULD AC1,AC2 ; MULTIPLY AC1 BY AC2
STFPS FPS ; STORE FLOATING POINT STATUS
CMP FPS,\$FPS ; CHECK FPS
BEQ MTST10 ; BRANCH IF OK
STD AC2,ANS1 ; SAVE FPU ANSWER
ERROR 21 ; FPS ERROR
MTST10: CMPD AC2,AC3 ; COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ; COPY FLOATING CONDITION CODES
BEQ MEND10 ; ANSWERS CHECK
; COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ; SAVE FPU ANSWER
ADD #1,ANS1+6 ; INCREMENT FPU ANSWER
ADC ANS1+4
ADC ANS1+2
ADC ANS1
CMPD ANS1,AC3 ; CHECK ANSWERS AGAIN
CFCC ; COPY FLOATING CONDITION CODES
BEQ MEND10 ; BRANCH IF OK

I05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY1: 27(1006) 04-MAY-77 18:18 PAGE 45 SEQ 0046
 D0FP08.P11 04-MAY-77 17:30 T30 EXERCISE MUL, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

2194	014116	162737	000002	002414	SUB	#2,ANS1+6	;DECREMENT FPU ANSWER
2195	014124	005637	002412		SBC	ANS1+4	
2196	014130	005637	002410		SBC	ANS1+2	
2197	014134	005637	002406		SBC	ANS1	
2198	014140	173737	002406		CMPO	ANS1,AC3	;CHECK ANSWERS AGAIN
2199	014144	170000			CFCC		;COPY FLOATING CONDITION CODES
2200	014146	001403			BEQ	MEND10	;BRANCH IF OK
2201	014150	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
2202	014154	104012			ERROR	12	;FPU AND FORTRAN DISAGREE
2203							
2204	014156	005037	002362		MEND10: CLR	FPS	;CLEAR F1P FPS BUFFER
2205							

J05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 46
 DDFP08.P11 04-MAY-77 17:30 T31

SEQ 0047

EXERCISE DIVF, ALL INTERRUPTS ON, ROUNDING MODE

```

2206 .....
2207 .....
2208 .....
2209 014162 000004
2210 014164 012737 014316 002376
2211 014172 012737 007400 002400
2212 014200 005037 002402
2213 014204 005037 002404
2214 014210 005037 002362
2215 014214 005037 002364
2216 014220 005037 002366
2217 014224 004737 023342
2218 014230 002426 002436
2219 014234 004437 023506
2220 014240 023510 002426
2221 014244 023510 002436
2222 014250 026364
2223 014252 023540 002416
2224
2225 014256 013700 002400
2226 014262 170127 040000
2227 014266 172437 002426
2228 014272 172537 002436
2229 014276 172737 002416
2230 014302 170127 007400
2231 014306 012737 014314 001110
2232
2233 .....
2234 .....
2235 014314 172600
2236 014316 174601
2237 014320 170237 002362
2238 014324 023737 002362 002400
2239 014332 001403
2240 014334 174237 002406
2241 014340 104016
2242
2243 014342 005737 002400
2244 014346 100014
2245 014350 170337 002364
2246
2247 014354 023737 002364 002402
2248 014362 001401
2249 014364 104026
2250
2251 014366 023737 002366 002404
2252 014374 001401
2253 014376 104026
2254
2255 014400 173702
2256 014402 170000
2257 014404 001416
2258
2259 014406 174237 002406
2260 014412 162737 000001 002410
2261 014420 005637 002406
  
```

```

;*****
;TEST 31 EXERCISE DIVF, ALL INTERRUPTS ON, ROUNDING MODE
;*****
↑ST31: SCOPE
MOV #DRET1,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007400,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL2 ;GET RANDOM INPUT DATA
;WORD LONUM,HINUM
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
$DIV ;ADDRESS OF FORTRAN DIVIDE
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;SET INTERRUPT DISABLE
LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007400 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
DRET1: LDF AC1,AC2 ;LOAD AC0 INTO AC2
DIVF AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR1 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 16 ;FPS ERROR

DERR1: TST $FPS ;ERROR BIT SET ?
BPL DTST1 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 15 ;BRANCH IF OK
ERROR 26 ;FEC IS WRONG

15: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ DTST1 ;BRANCH IF OK
ERROR 26 ;WRONG ADDRESS IN FEA

DTST1: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND1 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1
  
```

K05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 47
 DQFPDB.P11 04-MAY-77 17:30 T31

EXERCISE DIVF, ALL INTERRUPTS ON, ROUNDING MODE

SEQ 0048

2262	014424	173737	002406		CMPF	ANS1,AC3	:CHECK ANSWERS AGAIN
2263	014430	170000			CFCC		:COPY FLOATING CONDITION CODES
2264	014432	001403			BEQ	DEND1	:BRANCH IF OK
2265	014434	174237	002406		STF	AC2,ANS1	:SAVE FPU ANSWER
2266	014440	104013			ERROR	13	:FPU AND FORTRAN DISAGREE
2267							
2268	014442	005037	002362		DEND1:	CLR FPS	:CLEAR FPP FPS BUFFER
2269							
2270							
2271							
2272							
2273							
2274							
2275							
2276	014446	000004			TST32:	SCOPE	
2277	014450	012737	014602	002376	MOV	#DRET2,EXPFEA	:ADDR OF INSTR BEING TESTED
2278	014456	012737	007600	002400	MOV	#007600,\$FPS	:SET IE BITS IN FORTRAN ANSWER
2279	014464	005037	002402		CLR	\$FEC	:CLR FORTRAN FEC
2280	014470	005037	002404		CLR	\$FEA	:CLR FORTRAN FEA
2281	014474	005037	002362		CLR	FPS	:CLR FPU FPS BUFFER
2282	014500	005037	002364		CLR	FEC	:CLR FPU FEC BUFFER
2283	014504	005037	002366		CLR	FEA	:CLR FPU FEA BUFFER
2284	014510	004737	023332		JSR	PC,RANDL4	:GET RANDOM INPUT DATA
2285	014514	002426	002436		.WORD	LONUM,HINUM	
2286	014520	004437	023506		JSR	R4,\$POLSH	:ENTER POLISH MODE
2287	014524	023510	002426		\$PUSH	,LONUM	:PUSH 4 WORDS ON STACK (LONUM)
2288	014530	023510	002436		\$PUSH	,HINUM	:PUSH 4 WORDS ON STACK (HINUM)
2289	014534	026364			\$DIV		:ADDRESS OF FORTRAN DIVIDE
2290	014536	023540	002416		\$POPX	,ANS2	:POP 4 WORDS AND EXIT POLISH MODE
2291							
2292	014542	013700	002400		MOV	\$FPS,R0	:DISPLAY FLOATING POINT STATUS
2293	014546	170127	040200		LDFPS	#040200	:SET FID AND FD
2294	014552	172437	002426		LDC	LONUM,AC0	:LOAD AC0 WITH A RANDOM NUMBER
2295	014556	172537	002436		LDD	HINUM,AC1	:LOAD AC1 WITH A RANDOM NUMBER
2296	014562	172737	002416		LDD	ANS2,AC3	:LOAD AC3 WITH THE SUM
2297	014566	170127	007600		LDFPS	#007600	:TURN INTERRUPTS ON
2298	014572	012737	014600	001110	MOV	#,+6,\$LPADR	:RESET LOOP ADDRESS
2299							
2300							
2301							
2302	014600	172600					
2303	014602	174601			DRET2:	LDD AC0,AC2	:LOAD AC0 INTO AC2
2304	014604	170237	002362		DIVD	AC1,AC2	:DIVIDE AC1 INTO AC2
2305	014610	023737	002362	002400	STFPS	FPS	:STORE FLOATING POINT STATUS
2306	014616	001403			CMP	FPS,\$FPS	:CHECK FPS
2307	014620	174237	002406		BEQ	DERR2	:BRANCH IF OK
2308	014624	104022			STD	AC2,ANS1	:SAVE FPU ANSWER
2309					ERROR	22	:FPS ERROR
2310	014626	005737	002400		DERR2:	TST \$FPS	:ERROR BIT SET ?
2311	014632	100014			BPL	DTST2	:NO, DONT GET FEC/FEA
2312	014634	170337	002364		STST	FEC	:YES, CHECK STATUS
2313							
2314	014640	023737	002364	002402	CMP	FEC,\$FEC	:CHECK THE FLOATING EXCEPTION CODES
2315	014646	001401			BEQ	1\$:BRANCH IF OK
2316	014650	104032			ERROR	32	:FEC IS WRONG
2317							

L05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 48
 DDFP08.P11 04-MAY-77 17:30 T32

EXERCISE DIVD, ALL INTERRUPTS ON, ROUNDING MODE

SEQ 0049

2318	014652	023737	002366	002404	18:	CMP	FEA,\$FEA	;CHECK FLOATING PC
2319	014660	001401				BEQ	DTST2	;BRANCH IF OK
2320	014662	104032				ERROR	32	;WRONG ADDRESS IN FEA
2321								
2322	014664	173702			DTST2:	CMPD	AC2,AC3	;COMPARE FPU ANSWER TO FORTRAN ANSWER
2323	014666	170000				CFCC		;COPY FLOATING CONDITION CODES
2324	014670	001422				BEQ	DEND2	;ANSWERS CHECK
2325								;COMPENSATE FOR FORTRAN INACCURACIES.
2326	014672	174237	002406			STD	AC2,ANS1	;SAVE FPU ANSWER
2327	014676	162737	000001	002414		SUB	#1,ANS1+6	;DECREMENT FPU ANSWER
2328	014704	005637	002412			SBC	ANS1+4	
2329	014710	005637	002410			SBC	ANS1+2	
2330	014714	005637	002406			SBC	ANS1	
2331	014720	173737	002406			CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
2332	014724	170000				CFCC		;COPY FLOATING CONDITION CODES
2333	014726	001403				BEQ	DEND2	;BRANCH IF OK
2334	014730	174237	002406			STD	AC2,ANS1	;SAVE FPU ANSWER
2335	014734	104014				ERROR	14	;FPU AND FORTRAN DISAGREE
2336								
2337	014736	005037	002362		DEND2:	CLR	FPS	;CLEAR FPP FPS BUFFER
2338								
2339								
2340								
2341								
2342								
2343								
2344								
2345	014742	000004						
2346	014744	012737	015076	002376				
2347	014752	012737	004400	002400				
2348	014760	005037	002402					
2349	014764	005037	002404					
2350	014770	005037	002362					
2351	014774	005037	002364					
2352	015000	005037	002366					
2353	015004	004737	023342					
2354	015010	002426	002436					
2355	015014	004437	023506					
2356	015020	023510	002426					
2357	015024	023510	002436					
2358	015030	026364						
2359	015032	023540	002416					
2360								
2361	015036	013700	002400					
2362	015042	170127	040000					
2363	015046	172437	002426					
2364	015052	172537	002436					
2365	015056	172737	002416					
2366	015062	170127	004400					
2367	015066	012737	015074	001110				
2368								
2369								
2370								
2371	015074	172600						
2372	015076	174601						
2373	015100	170237	002362					


```

;*****
;TEST 33 EXERCISE DIVD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
;*****
†T33: SCOPE
MOV #DRET3,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #004400,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL2 ;GET RANDOM INPUT DATA
;WORD LONUM,HINUM
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
$DIV ;ADDRESS OF FORTRAN DIVIDE
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;SET FID
LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #004400 ;TURN INTERRUPTS ON, EXCEPT OVERFLOW AND UNDERFLOW
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
DRET3: LDF AC0,AC2 ;LOAD AC0 INTO AC2
DIVF AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS

```

M05

2374	015104	023737	002362	002400		CMP	FPS,\$FPS	;CHECK FPS
2375	015112	001403				BEQ	DERR3	;BRANCH IF OK
2376	015114	174237	002406			STF	AC2,ANS1	;SAVE FPU ANSWER
2377	015120	104016				ERROR	16	;FPS ERROR
2378								
2379	015122	005737	002400		DERR3:	TST	\$FPS	;ERROR BIT SET ?
2380	015126	100014				BPL	DTST3	;NO, DONT GET FEC/FEA
2381	015130	170337	002364			STST	FEC	;YES, CHECK STATUS
2382								
2383	015134	023737	002364	002402		CMP	FEC,\$FEC	;CHECK THE FLOATING EXCEPTION CODES
2384	015142	001401				BEQ	15	;BRANCH IF OK
2385	015144	104026				ERROR	26	;FEC IS WRONG
2386								
2387	015146	023737	002366	002404	15:	CMP	FEA,\$FEA	;CHECK FLOATING PC
2388	015154	001401				BEQ	DTST3	;BRANCH IF OK
2389	015156	104026				ERROR	26	;WRONG ADDRESS IN FEA
2390								
2391	015160	173702			DTST3:	CMPF	AC2,AC3	;COMPARE FPU ANSWER TO FORTRAN ANSWER
2392	015162	170000				CFCC		;COPY FLOATING CONDITION CODES
2393	015164	001416				BEQ	DEND3	;ANSWERS CHECK
2394								;COMPENSATE FOR FORTRAN INACCURACIES.
2395	015166	174237	002406			STF	AC2,ANS1	;SAVE FPU ANSWER
2396	015172	162737	000001	002410		SUB	#1,ANS1+2	;DECREMENT FPU ANSWER
2397	015200	005637	002406			SBC	ANS1	
2398	015204	173737	002406			CMPF	ANS1,AC3	;CHECK ANSWERS AGAIN
2399	015210	170000				CFCC		;COPY FLOATING CONDITION CODES
2400	015212	001403				BEQ	DEND3	;BRANCH IF OK
2401	015214	174237	002406			STF	AC2,ANS1	;SAVE FPU ANSWER
2402	015220	104013				ERROR	13	;FPU AND FORTRAN DISAGREE
2403								
2404	015222	005037	002362		DEND3:	CLR	FPS	;CLEAR FPP FPS BUFFER
2405								
2406								
2407								
2408								
2409								
2410								
2411								
2412	015226	000004						
2413	015230	012737	015362	002376		MOV	#ORET4,EXPFEA	;ADDR OF INSTR BEING TESTED
2414	015236	012737	004600	002400		MOV	#004600,\$FPS	;SET IE BITS IN FORTRAN ANSWER
2415	015244	005037	002402			CLR	\$FEC	;CLR FORTRAN FEC
2416	015250	005037	002404			CLR	\$FEA	;CLR FORTRAN FEA
2417	015254	005037	002362			CLR	FPS	;CLR FPU FPS BUFFER
2418	015260	005037	002364			CLR	FEC	;CLR FPU FEC BUFFER
2419	015264	005037	002366			CLR	FEA	;CLR FPU FEA BUFFER
2420	015270	004737	023332			JSR	PC,RANDL4	;GET RANDOM INPUT DATA
2421	015274	002426	002436			.WORD	LONUM,HINUM	
2422	015300	004437	023506			JSR	R4,\$POLSH	;ENTER POLISH MODE
2423	015304	023510	002426			\$PUSH	,LONUM	;PUSH 4 WORDS ON STACK (LONUM)
2424	015310	023510	002436			\$PUSH	,HINUM	;PUSH 4 WORDS ON STACK (HINUM)
2425	015314	026364				\$DIV		;ADDRESS OF FORTRAN DIVIDE
2426	015316	023540	002416			\$POPX	,ANS2	;POP 4 WORDS AND EXIT POLISH MODE
2427								
2428	015322	013700	002400			MOV	\$FPS,R0	;DISPLAY FLOATING POINT STATUS
2429	015326	170127	040200			LDFPS	#040200	;SET FID AND FD

 *TEST 34 EXERCISE DIVD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE

TST34: SCOPE
 MOV #ORET4,EXPFEA ;ADDR OF INSTR BEING TESTED
 MOV #004600,\$FPS ;SET IE BITS IN FORTRAN ANSWER
 CLR \$FEC ;CLR FORTRAN FEC
 CLR \$FEA ;CLR FORTRAN FEA
 CLR FPS ;CLR FPU FPS BUFFER
 CLR FEC ;CLR FPU FEC BUFFER
 CLR FEA ;CLR FPU FEA BUFFER
 JSR PC,RANDL4 ;GET RANDOM INPUT DATA
 .WORD LONUM,HINUM
 JSR R4,\$POLSH ;ENTER POLISH MODE
 \$PUSH ,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
 \$PUSH ,HINUM ;PUSH 4 WORDS ON STACK (HINUM)
 \$DIV ;ADDRESS OF FORTRAN DIVIDE
 \$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE
 MOV \$FPS,R0 ;DISPLAY FLOATING POINT STATUS
 LDFPS #040200 ;SET FID AND FD

2430	015332	172437	002426		LDD	LONUM,AC0	;LOAD AC0 WITH A RANDOM NUMBER
2431	015336	172537	002436		LDD	HINUM,AC1	;LOAD AC1 WITH A RANDOM NUMBER
2432	015342	172737	002416		LDD	ANS2,AC3	;LOAD AC3 WITH THE SUM
2433	015346	170127	004600		LDFPS	#004600	;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
2434	015352	012737	015360	001110	MOV	#.+6,SLPADR	;RESET LOOP ADDRESS

;*****

2438	015360	172600			LDD	AC0,AC2	;LOAD AC0 INTO AC2
2439	015362	174601			DRET4:	DIVD AC1,AC2	;DIVIDE AC1 INTO AC2
2440	015364	170237	002362		STFPS	FPS	;STORE FLOATING POINT STATUS
2441	015370	023737	002362	002400	CMF	FPS,\$FPS	;CHECK FPS
2442	015376	001403			BEQ	DERR4	;BRANCH IF OK
2443	015400	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
2444	015404	104022			ERROR	22	;FPS ERROR

2446	015406	005737	002400		DERR4:	TST \$FPS	;ERROR BIT SET ?
2447	015412	100014			BPL	DTST4	;NO, DONT GET FEC/FEA
2448	015414	170337	002364		STST	FEC	;YES, CHECK STATUS

2450	015420	023737	002364	002402	CMF	FEC,\$FEC	;CHECK THE FLOATING EXCEPTION CODES
2451	015426	001401			BEQ	1\$;BRANCH IF OK
2452	015430	104032			ERROR	32	;FEC IS WRONG

2454	015432	023737	002366	002404	1\$:	CMF FEA,\$FEA	;CHECK FLOATING PC
2455	015440	001401			BEQ	DTST4	;BRANCH IF OK
2456	015442	104032			ERROR	32	;WRONG ADDRESS IN FEA

2458	015444	173702			DTST4:	CMPO AC2,AC3	;COMPARE FPU ANSWER TO FORTRAN ANSWER
2459	015446	170000			CFCC		;COPY FLOATING CONDITION CODES
2460	015450	001422			BEQ	DEND4	;ANSWERS CHECK

2461	015452	174237	002406		;COMPENSATE FOR FORTRAN INACCURACIES		
2462	015456	162737	000001	002414	STD	AC2,ANS1	;SAVE FPU ANSWER
2463	015464	005637	002412		SUB	#1,ANS1+6	;DECREMENT FPU ANSWER

2464	015464	005637	002412		SBC	ANS1+4	
2465	015470	005637	002410		SBC	ANS1+2	
2466	015474	005637	002406		SBC	ANS1	

2467	015500	173737	002406		CMPO	ANS1,AC3	;CHECK ANSWERS AGAIN
2468	015504	170000			CFCC		;COPY FLOATING CONDITION CODES
2469	015506	001403			BEQ	DEND4	;BRANCH IF OK

2470	015510	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
2471	015514	104014			ERROR	14	;FPU AND FORTRAN DISAGREE
2472							
2473	015516	005037	002362		DEND4:	CLR FPS	;CLEAR FPP FPS BUFFER

;*****
;*TEST 35 EXERCISE DIVD, ALL INTERRUPTS ON, TRUNCATE MODE
;*****

2481	015522	000004			TST35:	SCOPE	
2482	015524	012737	015656	002376	MOV	#DRETS,EXPFEA	;ADDR OF INSTR BEING TESTED
2483	015532	012737	007440	002400	MOV	#007440,\$FPS	;SET IE BITS IN FORTRAN ANSWER
2484	015540	005037	002402		CLR	\$FEC	;CLR FORTRAN FEC
2485	015544	005037	002404		CLR	\$FEA	;CLR FORTRAN FEA

2546	015550	005037	002362		CLR	FPS	: CLR FPU FPS BUFFER
2547	015554	005037	002364		CLR	FEC	: CLR FPU FEC BUFFER
2548	015560	005037	002366		CLR	FEA	: CLR FPU FEA BUFFER
2549	015564	004737	023342		JSR	PC,RANDL2	: GET RANDOM INPUT DATA
2550	015570	002426	002436		.WORD	LONUM,HINUM	
2551	015574	004437	023506		JSR	R4,\$POLSH	: ENTER POLISH MODE
2552	015500	023510	002426		\$PUSH	,LONUM	: PUSH 2 WORDS ON STACK (LONUM)
2553	015604	023510	002436		\$PUSH	,HINUM	: PUSH 2 WORDS ON STACK (HINUM)
2554	015610	026364			\$DIV		: ADDRESS OF FORTRAN DIVIDE
2555	015612	023540	002416		\$POPX	,ANS2	: POP 2 WORDS AND EXIT POLISH MODE
2556							
2557	015616	013700	002400		MOV	\$FPS,R0	: DISPLAY FLOATING POINT STATUS
2558	015622	170127	040000		LDFPS	#040000	: SET FID
2559	015626	172437	002426		LDF	LONUM,AC0	: LOAD AC0 WITH A RANDOM NUMBER
2560	015632	172537	002436		LDF	HINUM,AC1	: LOAD AC1 WITH A RANDOM NUMBER
2561	015636	172737	002416		LDF	ANS2,AC3	: LOAD AC3 WITH THE SUM
2562	015642	170127	007440		LDFPS	#007440	: TURN INTERRUPTS ON
2563	015646	012737	015654	001110	MOV	#.+6,\$LPRDR	: RESET LOOP ADDRESS
2564							
2565							:*****
2566							
2567	015654	172600			LDF	AC0,AC2	: LOAD AC0 INTO AC2
2568	015656	174601		DRETS:	DIVF	AC1,AC2	: DIVIDE AC1 INTO AC2
2569	015660	170237	002362		STFPS	FPS	: STORE FLOATING POINT STATUS
2570	015664	023737	002362	002400	CHP	FPS,\$FPS	: CHECK FPS
2571	015672	001403			BEQ	DERRS	: BRANCH IF OK
2572	015674	174237	002406		STF	AC2,ANS1	: SAVE FPU ANSWER
2573	015700	104016			ERROR	16	: FPS ERROR
2574							
2575	015702	005737	002400		DERRS:	TST	: ERROR BIT SET ?
2576	015706	100014			BPL	DTSTS	: NO, DONT GET FEC/FEA
2577	015710	170337	002364		STST	FEC	: YES, CHECK STATUS
2578							
2579	015714	023737	002364	002402	CHP	FEC,\$FEC	: CHECK THE FLOATING EXCEPTION CODES
2580	015722	001401			BEQ	16	: BRANCH IF OK
2581	015724	104026			ERROR	26	: FEC IS WRONG
2582							
2583	015726	023737	002366	002404	16:	CHP	: CHECK FLOATING PC
2584	015734	001401			BEQ	DTSTS	: BRANCH IF OK
2585	015736	104026			ERROR	26	: WRONG ADDRESS IN FEA
2586							
2587	015740	173702			DTSTS:	CHPF	: COMPARE FPU ANSWER TO FORTRAN ANSWER
2588	015742	170000			CFCC		: COPY FLOATING CONDITION CODES
2589	015744	001427			BEQ	DENDS	: ANSWERS CHECK
2590							: COMPENSATE FOR FORTRAN INACCURACIES.
2591	015746	174237	002406		STF	AC2,ANS1	: SAVE FPU ANSWER
2592	015752	062737	000001	002410	ADD	#1,ANS1+2	: INCREMENT FPU ANSWER
2593	015760	005537	002406		ADC	ANS1	
2594	015764	173737	002406		CHPF	ANS1,AC3	: CHECK ANSWERS AGAIN
2595	015770	170000			CFCC		: COPY FLOATING CONDITION CODES
2596	015772	001414			BEQ	DENDS	: BRANCH IF OK
2597	015774	162737	000002	002410	SUB	#2,ANS1+2	: DECREMENT FPU ANSWER
2598	016002	005637	002406		SBC	ANS1	
2599	016006	173737	002406		CHPF	ANS1,AC3	: CHECK ANSWERS AGAIN
2600	016012	170000			CFCC		: COPY FLOATING CONDITION CODES
2601	016014	001403			BEQ	DENDS	: BRANCH IF OK

016016	174237	002406	STF	AC2,ANS1	;SAVE FPU ANSWER
016022	104013		ERROR	13	;FPU AND FORTRAN DISAGREE
016024	005037	002362	DENDS:	CLR FPS	;CLEAR FPP FPS BUFFER

 ;TEST 36 EXERCISE DIVD, ALL INTERRUPTS ON, TRUNCATE MODE
 ;*****

016030	000004		ST36:	SCOPE	
016032	012737	016164	002376	MOV	#DRET6,EXFCA ;ADDR OF INSTR BEING TESTED
016040	012737	007640	002400	MOV	#007640,\$FPS ;SET IE BITS IN FORTRAN ANSWER
016046	005037	002402		CLR	\$FEC ;CLR FORTRAN FEC
016052	005037	002404		CLR	\$FEA ;CLR FORTRAN FEA
016056	005037	002362		CLR	FPS ;CLR FPU FPS BUFFER
016062	005037	002364		CLR	FEC ;CLR FPU FEC BUFFER
016066	005037	002366		CLR	FEA ;CLR FPU FEA BUFFER
016072	004737	023332		JSR	PC,RANDL4 ;GET RANDOM INPUT DATA
016076	002426	002436		.WORD	LONUM,HINUM ;
016102	004437	023506		JSR	R4,\$POLSH ;ENTER POLISH MODE
016106	023510	002426		\$PUSH	,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
016112	023510	002436		\$PUSH	,HINUM ;PUSH 4 WORDS ON STACK (HINUM)
016116	026364			\$DIV	;ADDRESS OF FORTRAN DIVIDE
016120	023540	002416		\$POPX	,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE
016124	013700	002400		MOV	\$FPS,R0 ;DISPLAY FLOATING POINT STATUS
016130	170127	040200		LDFPS	#040200 ;SET FID AND FD
016134	172437	002426		LDD	LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
016140	172537	002436		LDD	HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
016144	172737	002416		LDD	ANS2,AC3 ;LOAD AC3 WITH THE SUM
016150	170127	007640		LDFPS	#007640 ;TURN INTERRUPTS ON
016154	012737	016162	001110	MOV	#,+6,\$LPADR ;RESET LOOP ADDRESS

016162	172600			LDD	AC0,AC2 ;LOAD AC0 INTO AC2
016164	174601		DRET6:	DIVD	AC1,AC2 ;DIVIDE AC1 INTO AC2
016166	170237	002362		STFPS	FPS ;STORE FLOATING POINT STATUS
016172	023737	002362	002400	CMP	FPS,\$FPS ;CHECK FPS
016200	001403			BEQ	DERR6 ;BRANCH IF OK
016202	174237	002406		STD	AC2,ANS1 ;SAVE FPU ANSWER
016206	104022			ERROR	22 ;FPS ERROR
016210	005737	002400		DERR6:	TST \$FPS ;ERROR BIT SET ?
016214	100014			BPL	DTST6 ;NO, DONT GET FEC/FEA
016216	170337	002364		STST	FEC ;YES, CHECK STATUS
016222	023737	002364	002402	CMP	FEC,\$FEC ;CHECK THE FLOATING EXCEPTION CODES
016230	001401			BEQ	15 ;BRANCH IF OK
016232	104032			ERROR	32 ;FEC IS WRONG
016234	023737	002366	002404	15:	CMP FEA,\$FEA ;CHECK FLOATING PC
016242	001401			BEQ	DTST6 ;BRANCH IF OK
016244	104032			ERROR	32 ;WRONG ADDRESS IN FEA

```

2598
2599 016246 173702 DTST6: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2600 016250 170000 CFCC ;COPY FLOATING CONDITION CODES
2601 016252 001437 BEQ DEND6 ;ANSWERS CHECK
2602 016254 174237 002406 ;COMPENSATE FOR FORTRAN INACCURACIES
2603 016254 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
2604 016260 062737 000001 002414 ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
2605 016266 005537 002412 ROC ANS1+4
2606 016272 005537 002410 ROC ANS1+2
2607 016276 005537 002406 ROC ANS1
2608 016302 173737 002406 CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
2609 016306 170000 CFCC ;COPY FLOATING CONDITION CODES
2610 016310 001420 BEQ DEND6 ;BRANCH IF OK
2611 016312 162737 000002 002414 SUB #2,ANS1+6 ;DECREMENT FPU ANSWER
2612 016320 005637 002412 SBC ANS1+4
2613 016324 005637 002410 SBC ANS1+2
2614 016330 005637 002406 SBC ANS1
2615 016334 173737 002406 CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
2616 016340 170000 CFCC ;COPY FLOATING CONDITION CODES
2617 016342 001403 BEQ DEND6 ;BRANCH IF OK
2618 016344 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
2619 016350 104014 ERROR 14 ;FPU AND FORTRAN DISAGREE
2620
2621 016352 005037 002362 DEND6: CLR FPS ;CLEAR FPP FPS BUFFER
2622
2623
2624
2625
2626
2627 ;*****
2628 ;TEST 37 EXERCISE DIVF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
2629 ;*****
2629 016356 000004 †ST37: SCOPE
2630 016360 012737 016512 002376 MOV #ORET7,EXPFEA ;ADDR OF INSTR BEING TESTED
2631 016366 012737 004440 002400 MOV #004440,$FPS ;SET IE BITS IN FORTRAN ANSWER
2632 016374 005037 002402 CLR $FEC ;CLR FORTRAN FEC
2633 016400 005037 002404 CLR $FEA ;CLR FORTRAN FEA
2634 016404 005037 002362 CLR FPS ;CLR FPU FPS BUFFER
2635 016410 005037 002364 CLR FEC ;CLR FPU FEC BUFFER
2636 016414 005037 002366 CLR FEA ;CLR FPU FEA BUFFER
2637 016420 004737 023342 JSR PC,RANDL2 ;GET RANDOM INPUT DATA
2638 016424 002426 002436 .WORD LONUM,HINUM
2639 016430 004437 023506 JSR R4,$POLSH ;ENTER POLISH MODE
2640 016434 023510 002426 $PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
2641 016440 023510 002436 $PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
2642 016444 026364 $DIV ;ADDRESS OF FORTRAN DIVIDE
2643 016446 023540 002416 $POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE
2644
2645 016452 013700 002400 MOV $FPS,RD ;DISPLAY FLOATING POINT STATUS
2646 016456 170127 040000 LDFPS #040000 ;SET FID
2647 016462 172437 002426 LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
2648 016466 172537 002436 LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
2649 016472 172737 002416 LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
2650 016476 170127 004440 LDFPS #004440 ;TURN INTERRUPTS ON, EXCEPT OVERFLOW AND UNDERFLOW
2651 016502 012737 016510 001110 MOV #.+6,$LPADR ;RESET LOOP ADDRESS
2652
2653 ;*****

```

E06

```

2654
2655 016510 172600
2656 016512 174601 DRET7: LDF AC0,AC2 ;LOAD AC0 INTO AC2
2657 016514 170237 002362 002400 DIVF AC1,AC2 ;DIVIDE AC1 INTO AC2
2658 016520 023737 002362 STEPS FPS ;STORE FLOATING POINT STATUS
2659 016526 001403 CMP FPS,$FPS ;CHECK FPS
2660 016530 174237 002406 BEQ DERR7 ;BRANCH IF OK
2661 016534 104016 STF AC2,ANS1 ;SAVE FPU ANSWER
2662 ERROR 16 ;FPS ERROR
2663 016536 005737 002400 DERR7: TST $FPS ;ERROR BIT SET ?
2664 016542 100014 BPL DTST7 ;NO, DONT GET FEC/FEA
2665 016544 170337 002364 STST FEC ;YES, CHECK STATUS
2666
2667 016550 023737 002364 002402 CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
2668 016556 001401 BEQ 16 ;BRANCH IF OK
2669 016560 104026 ERROR 26 ;FEC IS WRONG
2670
2671 016562 023737 002366 002404 16: CMP FEA,$FEA ;CHECK FLOATING PC
2672 016570 001401 BEQ DTST7 ;BRANCH IF OK
2673 016572 104026 ERROR 26 ;WRONG ADDRESS IN FEA
2674
2675 016574 173702 DTST7: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2676 016576 170000 CFCC ;COPY FLOATING CONDITION CODES
2677 016600 001427 BEQ DEND7 ;ANSWERS CHECK
2678 ;COMPENSATE FOR FORTRAN INACCURACIES.
2679 016602 174237 002406 STF AC2,ANS1 ;SAVE FPU ANSWER
2680 016606 062737 000001 002410 ADD #1,ANS1+2 ;INCREMENT FPU ANSWER
2681 016614 005537 002406 ADC ANS1
2682 016620 173737 002406 CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
2683 016624 170000 CFCC ;COPY FLOATING CONDITION CODES
2684 016626 001414 BEQ DEND7 ;BRANCH IF OK
2685 016630 162737 000002 002410 SUB #2,ANS1+2 ;DECREMENT FPU ANSWER
2686 016636 005637 002406 SBC ANS1
2687 016642 173737 002406 CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
2688 016646 170000 CFCC ;COPY FLOATING CONDITION CODES
2689 016650 001403 BEQ DEND7 ;BRANCH IF OK
2690 016652 174237 002406 STF AC2,ANS1 ;SAVE FPU ANSWER
2691 016656 104013 ERROR 13 ;FPU AND FORTRAN DISAGREE
2692
2693 016660 005037 002362 DEND7: CLR FPS ;CLEAR FPP FPS BUFFER
2694
2695
2696
2697
2698
2699
2700
2701 016664 000004
2702 016666 012737 017020 002376
2703 016674 012737 004640 002400
2704 016702 005037 002402
2705 016706 005037 002404
2706 016712 005037 002362
2707 016716 005037 002364
2708 016722 005037 002366
2709 016726 004737 023332

;*****
;TEST 40 EXERCISE DIVD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
;*****
†ST40: SCOPE
MOV #DRET10,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #004640,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL4 ;GET RANDOM INPUT DATA

```

F06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 55 SEQ 0056
 D8FP08.P11 04-MAY-77 17:30 T40 EXERCISE DIVD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

2710	016732	002426	002436		WORD	LONUM,HINUM	:
2711	016736	004437	023506		JSR	R4,\$POLSH	:ENTER POLISH MODE
2712	016742	023510	002426		\$PUSH	,LONUM	:PUSH 4 WORDS ON STACK (LONUM)
2713	016746	023510	002436		\$PUSH	,HINUM	:PUSH 4 WORDS ON STACK (HINUM)
2714	016752	026364			\$DIV		:ADDRESS OF FORTRAN DIVIDE
2715	016754	023540	002416		\$POPX	,ANS2	:POP 4 WORDS AND EXIT POLISH MODE
2716							
2717	016760	013700	002400		MOV	\$FPS,RO	:DISPLAY FLOATING POINT STATUS
2718	016764	170127	040200		LDFPS	#040200	:SET FID AND FD
2719	016770	172437	002426		LDD	LONUM,AC0	:LOAD AC0 WITH A RANDOM NUMBER
2720	016774	172537	002436		LDD	HINUM,AC1	:LOAD AC1 WITH A RANDOM NUMBER
2721	017000	172737	002416		LDD	ANS2,AC3	:LOAD AC3 WITH THE SUM
2722	017004	170127	004640		LDFPS	#004640	:TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
2723	017010	012737	017016	001110	MOV	#.+6,\$LPROR	:RESET LOOP ADDRESS
2724							
2725							:*****
2726							
2727	017016	172600			LDD	AC0,AC2	:LOAD AC0 INTO AC2
2728	017020	174601			DRET10:	DIVD AC1,AC2	:DIVIDE AC1 INTO AC2
2729	017022	170237	002362		STFPS	FPS	:STORE FLOATING POINT STATUS
2730	017026	023737	002362	002400	CHP	FPS,\$FPS	:CHECK FPS
2731	017034	001403			BEQ	DERR10	:BRANCH IF OK
2732	017036	174237	002406		STD	AC2,ANS1	:SAVE FPU ANSWER
2733	017042	104022			ERROR	22	:FPS ERROR
2734							
2735	017044	005737	002400		DERR10:	TST \$FPS	:ERROR BIT SET ?
2736	017050	100014			BPL	DTST10	:NO, DONT GET FEC/FEA
2737	017052	170337	002364		STST	FEC	:YES, CHECK STATUS
2738							
2739	017056	023737	002364	002402	CHP	FEC,\$FEC	:CHECK THE FLOATING EXCEPTION CODES
2740	017064	001401			BEQ	1\$:BRANCH IF OK
2741	017066	104032			ERROR	32	:FEC IS WRONG
2742							
2743	017070	023737	002366	002404	1\$:	CHP FEA,\$FEA	:CHECK FLOATING PC
2744	017076	001443			BEQ	DEND10	:BRANCH IF OK
2745	017100	104032			ERROR	32	:WRONG ADDRESS IN FEA
2746							
2747	017102	173702			DTST10:	CHPD AC2,AC3	:COMPARE FPU ANSWER TO FORTRAN ANSWER
2748	017104	170000			CFCC		:COPY FLOATING CONDITION CODES
2749	017106	001437			BEQ	DEND10	:ANSWERS CHECK
2750							:COMPENSATE FOR FORTRAN INACCURACIES.
2751	017110	174237	002406		STD	AC2,ANS1	:SAVE FPU ANSWER
2752	017114	062737	000001	002414	ADD	#1,ANS1+6	:INCREMENT FPU ANSWER
2753	017122	005537	002412		ADC	ANS1+4	
2754	017126	005537	002410		ADC	ANS1+2	
2755	017132	005537	002406		ADC	ANS1	
2756	017136	173737	002406		CHPD	ANS1,AC3	:CHECK ANSWERS AGAIN
2757	017142	170000			CFCC		:COPY FLOATING CONDITION CODES
2758	017144	001420			BEQ	DEND10	:BRANCH IF OK
2759	017146	162737	000002	002414	SUB	#2,ANS1+6	:DECREMENT FPU ANSWER
2760	017154	005637	002412		SBC	ANS1+4	
2761	017160	005637	002410		SBC	ANS1+2	
2762	017164	005637	002406		SBC	ANS1	
2763	017170	173737	002406		CHPD	ANS1,AC3	:CHECK ANSWERS AGAIN
2764	017174	170000			CFCC		:COPY FLOATING CONDITION CODES
2765	017176	001403			BEQ	DEND10	:BRANCH IF OK


```

2766 017200 174237 002406          STD   AC2,ANS1      ;SAVE FPU ANSWER
2767 017204 104014                    ERROR  14           ;FPU AND FORTRAN DISAGREE
2768
2769 017206 005037 002362          DEND10. CLR   FPS      ;CLEAR FPP FPS BUFFER
2770
2771
2772
2773
2774
2775
2776
2777 017212 000004                    ;*****
2778 017214 012737 017346 002376    MOV   #DRET11,EXPFEA ;ADDR OF INSTR BEING TESTED
2779 017222 012737 047400 002400    MOV   #047400,$FPS   ;SET IE BITS IN FORTRAN ANSWER
2780 017230 005037 002402            CLR   $FEC           ;CLR FORTRAN FEC
2781 017234 005037 002404            CLR   $FEA           ;CLR FORTRAN FEA
2782 017240 005037 002362            CLR   FPS            ;CLR FPU FPS BUFFER
2783 017244 005037 002364            CLR   FEC            ;CLR FPU FEC BUFFER
2784 017250 005037 002366            CLR   FEA           ;CLR FPU FEA BUFFER
2785 017254 004737 023342            JSR   PC,RANDL2      ;GET RANDOM INPUT DATA
2786 017260 002426 002436            .WORD LONUM,HINUM
2787 017264 004437 023506            JSR   R4,$POLSH     ;ENTER POLISH MODE
2788 017270 023510 002426            $PUSH ,LONUM        ;PUSH 2 WORDS ON STACK (LONUM)
2789 017274 023510 002436            $PUSH ,HINUM        ;PUSH 2 WORDS ON STACK (HINUM)
2790 017300 026364                    $DIV                    ;ADDRESS OF FORTRAN DIVIDE
2791 017302 023540 002416            $POPX ,ANS2         ;POP 2 WORDS AND EXIT POLISH MODE
2792
2793 017306 013700 002400            MOV   $FPS,R0        ;DISPLAY FLOATING POINT STATUS
2794 017312 170127 040000            LDFPS #040000        ;SET FID
2795 017316 172437 002426            LDF   LONUM,AC0     ;LOAD AC0 WITH A RANDOM NUMBER
2796 017322 172537 002436            LDF   HINUM,AC1     ;LOAD AC1 WITH A RANDOM NUMBER
2797 017326 172737 002416            LDF   ANS2,AC3      ;LOAD AC3 WITH THE SUM
2798 017332 170127 047400            LDFPS #047400        ;SET INTERRUPT DISABLE AND INTERRUPT BITS
2799 017336 012737 017344 001110    MOV   #.+6,$LPAOR   ;RESET LOOP ADDRESS
2800
2801
2802
2803 017344 172600                    ;*****
2804 017346 174601                    DRET11: LDF   AC0,AC2   ;LOAD AC0 INTO AC2
2805 017350 170237 002362            DIVF  AC1,AC2       ;DIVIDE AC1 INTO AC2
2806 017354 023737 002362 002400    STFPS FPS           ;STORE FLOATING POINT STATUS
2807 017362 001403                    CMP   FPS,$FPS      ;CHECK FPS
2808 017364 174237 002406            BEQ   DERR11        ;BRANCH IF OK
2809 017370 104016                    STF   AC2,ANS1      ;SAVE FPU ANSWER
2810
2811
2812
2813
2814
2815 017404 023737 002364 002402    ERROR  16           ;FPS ERROR
2816 017412 001401                    DERR11: TST   $FPS    ;ERROR BIT SET ?
2817 017414 104026                    BPL   DTST11        ;NO, DONT GET FEC/FEA
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000

```

```

2822
2823 017431 173702 DTST11: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2824 017432 170000 CFCC ;COPY FLOATING CONDITION CODES
2825 017434 001416 BEQ DEND11 ;ANSWERS CHECK
2826 ; COMPENSATE FOR FORTRAN INACCURACIES.
2827 017436 174237 002406 STF AC2,ANS1 ;SAVE FPU ANSWER
2828 017442 162737 000001 002410 SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
2829 017450 005637 002406 SBC ANS1
2830 017454 173737 002406 CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
2831 017460 170000 CFCC ;COPY FLOATING CONDITION CODES
2832 017462 001403 BEQ DEND11 ;BRANCH IF OK
2833 017464 174237 002406 STF AC2,ANS1 ;SAVE FPU ANSWER
2834 017470 104013 ERROR 13 ;FPU AND FORTRAN DISAGREE
2835
2836 017472 005037 002362 DEND11: CLR FPS ;CLEAR FPP FPS BUFFER
2837
2838
2839
2840
2841 ;*****
2842 ;*TEST 42 EXERCISE DIVD, INTERRUPT DISABLE SET, ROUNDING MODE
2843 ;*****
2843 017476 000004 †ST42: SCOPE
2844 017500 012737 017632 002376 MOV #DRET12,EXPFEA ;ADDR OF INSTR BEING TESTED
2845 017506 012737 047600 002400 MOV #047600,$FPS ;SET FID AND IE BITS IN FORTRAN ANSWER
2846 017514 005037 002402 CLR $FEC ;CLR FORTRAN FEC
2847 017520 005037 002404 CLR $FEA ;CLR FORTRAN FEA
2848 017524 005037 002362 CLR FPS ;CLR FPU FPS BUFFER
2849 017530 005037 002364 CLR FEC ;CLR FPU FEC BUFFER
2850 017534 005037 002366 CLR FEA ;CLR FPU FEA BUFFER
2851 017540 004737 023332 JSR PC,RANDL4 ;GET RANDOM INPUT DATA
2852 017544 002426 002436 .WORD LONUM,HINUM
2853 017550 004437 023506 JSR R4,SPOLSH ;ENTER POLISH MODE
2854 017554 023510 002426 $PUSH ,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
2855 017560 023510 002436 $PUSH ,HINUM ;PUSH 4 WORDS ON STACK (HINUM)
2856 017564 026364 $DIV ;ADDRESS OF FORTRAN DIVIDE
2857 017566 023540 002416 $POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE
2858
2859 017572 013700 002400 MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
2860 017576 170127 040200 LDFPS #040200 ;SET FID AND FD
2861 017602 172437 002426 LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
2862 017606 172537 002436 LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
2863 017612 172737 002416 LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
2864 017616 170127 047600 LDFPS #047600 ;SET INTERRUPT DISABLE AND INTERRUPT BITS
2865 017622 012737 017630 001110 MOV #.+6,$LPADR ;RESET LOOP ADDRESS
2866
2867 ;*****
2868
2869 017630 172600 DRET12: LDD AC0,AC2 ;LOAD AC0 INTO AC2
2870 017632 174601 DIVD AC1,AC2 ;DIVIDE AC1 INTO AC2
2871 017634 170237 002362 STFPS FPS ;STORE FLOATING POINT STATUS
2872 017640 023737 002362 002400 CMP FPS,$FPS ;CHECK FPS
2873 017646 001403 BEQ DERR12 ;BRANCH IF OK
2874 017650 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
2875 017654 104022 ERROR 22 ;FPS ERROR
2876
2877 017656 005737 002400 DERR12: TST $FPS ;ERROR BIT SET ?

```

```

2878 017662 100014          BPL   DTST12          ;NO, DONT GET FEC/FEA
2879 017664 170337 002364  STST  FEC             ;YES, CHECK STATUS
2880
2881 017670 023737 002364 002402  CMP   FEC,$FEC       ;CHECK THE FLOATING EXCEPTION CODES
2882 017676 001401          BEQ   15              ;BRANCH IF OK
2883 017700 104032          ERROR 32             ;FEC IS WRONG
2884
2885 017702 023737 002366 002404 15:  CMP   FEA,$FEA       ;CHECK FLOATING PC
2886 017710 001401          BEQ   DTST12         ;BRANCH IF OK
2887 017712 104032          ERROR 32             ;WRONG ADDRESS IN FEA
2888
2889 017714 173702          DTST12: CMPD  AC2,AC3     ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2890 017716 170000          CFCC                    ;COPY FLOATING CONDITION CODES
2891 017720 001422          BEQ   DEND12         ;ANSWERS CHECK
2892          ;COMPENSATE FOR FORTRAN INACCURACIES.
2893 017722 174237 002406          STD   AC2,ANS1       ;SAVE FPU ANSWER
2894 017726 162737 000001 002414  SUB   #1,ANS1+6      ;DECREMENT FPU ANSWER
2895 017734 005637 002412          SBC   ANS1+4
2896 017740 005637 002410          SBC   ANS1+2
2897 017744 005637 002406          SBC   ANS1
2898 017750 173737 002406          CMPD  ANS1,AC3       ;CHECK ANSWERS AGAIN
2899 017754 170000          CFCC                    ;COPY FLOATING CONDITION CODES
2900 017756 001403          BEQ   DEND12         ;BRANCH IF OK
2901 017760 174237 002406          STD   AC2,ANS1       ;SAVE FPU ANSWER
2902 017764 104014          ERROR 14             ;FPU AND FORTRAN DISAGREE
2903
2904 017766 005037 002362          DEND12: CLR  FPS      ;CLEAR FPP FPS BUFFER
2905
2906
2907
2908
2909
2910
2911
2912 017772 000004          ;*****
2913 017774 012737 020126 002376  ;*TEST 43 EXERCISE DIVF, INTERRUPT DISABLE SET, TRUNCATE MODE
2914 020002 012737 047440 002400  ;*****
2915 020010 005037 002402          ;*****
2916 020014 005037 002404          ;*****
2917 020020 005037 002362          ;*****
2918 020024 005037 002364          ;*****
2919 020030 005037 002366          ;*****
2920 020034 004737 023342          ;*****
2921 020040 002426 002436          ;*****
2922 020044 004437 023506          ;*****
2923 020050 023510 002426          ;*****
2924 020054 023510 002436          ;*****
2925 020060 026364          ;*****
2926 020062 023540 002416          ;*****
2927
2928 020066 013700 002400          ;*****
2929 020072 170127 040000          ;*****
2930 020076 172437 002426          ;*****
2931 020102 172537 002436          ;*****
2932 020106 172737 002416          ;*****
2933 020112 170127 047440          ;*****

```

J06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 59
 DQFP08.P11 04-MAY-77 17:30 T43

EXERCISE DIVF, INTERRUPT DISABLE SET, TRUNCATE MODE

SEQ 0060

```

2934 020116 012737 020124 001110      MOV      #.+6,$LPAOR      ;RESET LOOP ADDRESS
;*****
2935
2936
2937
2938 020124 172600                      LDF      AC0,AC2          ;LOAD AC0 INTO AC2
2939 020126 174601      DRET13: DIVF     AC1,AC2    ;DIVIDE AC1 INTO AC2
2940 020130 170237 002362      STFPS    FPS             ;STORE FLOATING POINT STATUS
2941 020134 023737 002362 002400      CMP      FPS,$FPS        ;CHECK FPS
2942 020142 001403      BEQ      DEND13         ;BRANCH IF OK
2943 020144 174237 002406      STF      AC2,ANS1       ;SAVE FPU ANSWER
2944 020150 104016      ERROR    16             ;FPS ERROR
2945
2946 020152 005737 002400      DERR13: TST      $FPS     ;ERROR BIT SET ?
2947 020156 100014      BPL      DTST13        ;NO, DONT GET FEC/FEA
2948 020160 170337 002364      STST     FEC           ;YES, CHECK STATUS
2949
2950 020164 023737 002364 002402      CMP      FEC,$FEC      ;CHECK THE FLOATING EXCEPTION CODES
2951 020172 001401      BEQ      1$            ;BRANCH IF OK
2952 020174 104026      ERROR    26           ;FEC IS WRONG
2953
2954 020176 023737 002366 002404 1$:  CMP      FEA,$FEA     ;CHECK FLOATING PC
2955 020204 001401      BEQ      DTST13        ;BRANCH IF OK
2956 020206 104026      ERROR    26           ;WRONG ADDRESS IN FEA
2957
2958 020210 173702      DTST13: CMPF     AC2,AC3    ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2959 020212 170000      CFCC                    ;COPY FLOATING CONDITION CODES
2960 020214 001427      BEQ      DEND13        ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
2961
2962 020216 174237 002406      STF      AC2,ANS1       ;SAVE FPU ANSWER
2963 020222 062737 000001 002410      ADD      #1,ANS1+2     ;INCREMENT FPU ANSWER
2964 020230 005537 002406      ADC      ANS1
2965 020234 173737 002406      CMPF     ANS1,AC3      ;CHECK ANSWERS AGAIN
2966 020240 170000      CFCC                    ;COPY FLOATING CONDITION CODES
2967 020242 001414      BEQ      DEND13        ;BRANCH IF OK
2968 020244 162737 000002 002410      SUB      #2,ANS1+2     ;DECREMENT FPU ANSWER
2969 020252 005637 002406      SBC      ANS1
2970 020256 173737 002406      CMPF     ANS1,AC3      ;CHECK ANSWERS AGAIN
2971 020262 170000      CFCC                    ;COPY FLOATING CONDITION CODES
2972 020264 001403      BEQ      DEND13        ;BRANCH IF OK
2973 020266 174237 002406      STF      AC2,ANS1       ;SAVE FPU ANSWER
2974 020272 104013      ERROR    13           ;FPU AND FORTRAN DISAGREE
2975
2976 020274 005037 002362      DEND13: CLR      FPS     ;CLEAR FPP FPS BUFFER
2977
2978
2979
2980
2981 ;*****
2981 ;TEST 44      EXERCISE DIVD, INTERRUPT DISABLE SET, TRUNCATE MODE
2982 ;*****
2983 020300 000004      †ST44:  SCOPE
2984 020302 012737 020434 002376      MOV      #DRET14,EXPFEA ;ADDR OF INSTR BEING TESTED
2985 020310 012737 047640 002400      MOV      #047640,$FPS   ;SET FID AND IE BITS IN FORTRAN ANSWER
2986 020316 005037 002402      CLR      $FEC          ;CLR FORTRAN FEC
2987 020322 005037 002404      CLR      $FEA          ;CLR FORTRAN FEA
2988 020326 005037 002362      CLR      FPS           ;CLR FPU FPS BUFFER
2989 020332 005037 002364      CLR      FEC           ;CLR FPU FEC BUFFER
  
```

K06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 60
 DQFP08.P11 04-MAY-77 17:30 T44

EXERCISE DIVD, INTERRUPT DISABLE SET, TRUNCATE MODE

SEQ 0061

2990	020336	005037	002366		CLR	FEA	: CLR FPU FEA BUFFER
2991	020342	004737	023332		JSR	PC,RANDL4	: GET RANDOM INPUT DATA
2992	020346	002426	002426		.WORD	LONUM,HINUM	
2993	020352	004437	023506		JSR	R4,SPOLSH	: ENTER POLISH MODE
2994	020356	023510	002426		\$PUSH	,LONUM	: PUSH 4 WORDS ON STACK (LONUM)
2995	020362	023510	002436		\$PUSH	,HINUM	: PUSH 4 WORDS ON STACK (HINUM)
2996	020366	026364			\$OIV		: ADDRESS OF FORTRAN DIVIDE
2997	020370	023540	002416		\$POPX	,ANS2	: POP 4 WORDS AND EXIT POLISH MODE
2998							
2999	020374	013700	002400		MOV	\$FPS,RO	: DISPLAY FLOATING POINT STATUS
3000	020400	170127	040200		LDFPS	#040200	: SET FID AND FD
3001	020404	172437	002426		LDD	LONUM,AC0	: LOAD AC0 WITH A RANDOM NUMBER
3002	020410	172537	002436		LDD	HINUM,AC1	: LOAD AC1 WITH A RANDOM NUMBER
3003	020414	172737	002416		LDD	ANS2,AC3	: LOAD AC3 WITH THE SUM
3004	020420	170127	047640		LDFPS	#047640	: SET INTERRUPT DISABLE AND INTERRUPT BITS
3005	020424	012737	020432	001110	MOV	#.+6,\$LPAOR	: RESET LOOP ADDRESS
3006							
3007							:*****
3008							
3009	020432	172600			LDD	AC0,AC2	: LOAD AC0 INTO AC2
3010	020434	174601			DRET14:	DIVD AC1,AC2	: DIVIDE AC1 INTO AC2
3011	020436	170237	002362		STFPS	FPS	: STORE FLOATING POINT STATUS
3012	020442	023737	002362	002400	CMP	FPS,\$FPS	: CHECK FPS
3013	020450	001403			BEQ	DERR14	: BRANCH IF OK
3014	020452	174237	002406		STD	AC2,ANS1	: SAVE FPU ANSWER
3015	020456	104022			ERROR	22	: FPS ERROR
3016							
3017	020460	005737	002400		DERR14:	TST \$FPS	: ERROR BIT SET ?
3018	020464	100014			BPL	DTST14	: NO, DONT GET FEC/FEA
3019	020466	170337	002364		STST	FEC	: YES, CHECK STATUS
3020							
3021	020472	023737	002364	002402	CMP	FEC,\$FEC	: CHECK THE FLOATING EXCEPTION CODES
3022	020500	001401			BEQ	18	: BRANCH IF OK
3023	020502	104032			ERROR	32	: FEC IS WRONG
3024							
3025	020504	023737	002366	002404	18:	CMP FEA,\$FEA	: CHECK FLOATING PC
3026	020512	001401			BEQ	DTST14	: BRANCH IF OK
3027	020514	104032			ERROR	32	: WRONG ADDRESS IN FEA
3028							
3029	020516	173702			DTST14:	CMPD AC2,AC3	: COMPARE FPU ANSWER TO FORTRAN ANSWER
3030	020520	170000			CFCC		: COPY FLOATING CONDITION CODES
3031	020522	001437			BEQ	DEND14	: ANSWERS CHECK
3032							: COMPENSATE FOR FORTRAN INACCURACIES.
3033	020524	174237	002406		STD	AC2,ANS1	: SAVE FPU ANSWER
3034	020530	062737	000001	002414	ADD	#1,ANS1+6	: INCREMENT FPU ANSWER
3035	020536	005537	002412		ADC	ANS1+4	
3036	020542	005537	002410		ADC	ANS1+2	
3037	020546	005537	002406		ADC	ANS1	
3038	020552	173737	002406		CMPD	ANS1,AC3	: CHECK ANSWERS AGAIN
3039	020556	170000			CFCC		: COPY FLOATING CONDITION CODES
3040	020560	001420			BEQ	DEND14	: BRANCH IF OK
3041	020562	162737	000002	002414	SUB	#2,ANS1+6	: DECREMENT FPU ANSWER
3042	020570	005637	002412		SBC	ANS1+4	
3043	020574	005637	002410		SBC	ANS1+2	
3044	020600	005637	002406		SBC	ANS1	
3045	020604	173737	002406		CMPD	ANS1,AC3	: CHECK ANSWERS AGAIN

L06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006)
DQFP08.P11 04-MAY-77 17:30

04-MAY-77 18:18 PAGE 61
EXERCISE DIVD, INTERRUPT DISABLE SET, TRUNCATE MODE

SEQ 0062

3046 020610 170000
3047 020612 001403
3048 020614 174237 002406
3049 020620 104014
3050
3051 020622 005027 002362

CFCC
BEQ DEND14
STD AC2,ANS1
ERROR 14

DEND14: CLR FPS

;COPY FLOATING CONDITION CODES
;BRANCH IF OK
;SAVE FPU ANSWER
;FPU AND FORTRAN DISAGREE
;CLEAR FPP FPS BUFFER

M06

```

3052      ;*****
3053      ;*TEST 45      A00F, SUBF, MULF, DIVF EXERCISER
3054      ;*****
3055      020626 000004      ST45: SCOPE
3056      ;*UNDERFLOW, OVERFLOW INTERRUPTS OFF; TRUNCATE MODE
3057
3058      020630 012737 000440 002400      MOV      #440, $FPS      ;SOFTWARE STATUS
3059      020636 005037 002402      CLR      $FEC          ;CLEAR STATUS
3060      020642 005037 002404      CLR      $FEA
3061      020646 005037 002362      CLR      FPS
3062      020652 005037 002364      CLR      FEC
3063      020656 005037 002366      CLR      FEA
3064
3065      020662 004737 023342      JSR      PC, RANDL2    ;GET 6 FLOAT RANDOM NUMBERS
3066      020666 002446 002476      .WORD   OP1, OP4
3067      020672 004737 023342      JSR      PC, RANDL2
3068      020676 002456 002506      .WORD   OP2, OP5
3069      020702 004737 023342      JSR      PC, RANDL2    15:
3070      020706 002466 002516      .WORD   OP3, OP6
3071      020712 032737 077600 002516      BIT      #077600, OP6  ;LET'S NEVER DIVIDE BY ZERO
3072      020720 001770      BEQ      15           ;EXPO OF OP6 IS ZERO, SO GET ANOTHER
3073
3074      020722 004437 023506      JSR      R4, $POLSH    ;ENTER POLISH MODE TO CALCULATE:
3075      020726 023510 002446      $PUSH   ,OP1
3076      020732 023510 002456      $PUSH   ,OP2
3077      020736 023562      $$SUB
3078      020740 023510 002466      $PUSH   ,OP3
3079      020744 023510 002476      $PUSH   ,OP4
3080      020750 023510 002516      $PUSH   ,OP6
3081      020754 026364      $DIV
3082      020756 023566      $ADD
3083      020760 025244      $MUL
3084      020762 023510 002506      $PUSH   ,OP5
3085      020766 025244      $MUL
3086      020770 023540 002416      $POPX   ,ANS2
3087
3088      020774 170127 040000      LDFPS   #040000      ;NO CHECKS
3089      021000 172437 002416      LDF     ANS2, AC0     ;GET SOFTWARE ANSWER
3090      021004 013700 002400      MOV     $FPS, R0      ;DISPLAY $FPS
3091      021010 012737 021016 001110      MOV     #.+6, $LPADR  ;RESET LOOP ADDRESS
3092
3093      ;*****
3094
3095      021016 170127 000440      LDFPS   #000440      ;INITIAL FPS
3096      021022 172537 002446      LDF     OP1, AC1     ;AC1 <- OP1
3097      021026 173137 002456      SUBF   OP2, AC1     ;AC1 <- OP1-OP2
3098      021032 172637 002476      LDF     OP4, AC2     ;AC2 <- OP4
3099      021036 174637 002516      DIVF   OP6, AC2     ;AC2 <- OP4/OP6
3100      021042 172237 002466      A00F   OP3, AC2     ;AC2 <- OP3+OP4/OP6
3101      021046 171102      MULF   AC2, AC1     ;AC1 <- (OP1-OP2)*(OP3+OP4/OP6)
3102      021050 171137 002506      MULF   OP5, AC1     ;AC1 <- (OP1-OP2)*(OP3+OP4/OP6)*OP5
3103
3104      021054 170237 002362      STFPS   FPS          ;STORE STATUS AFTERWARD
3105      021060 023737 002362 002400      CMP     FPS, $FPS    ;CHECK STATUS
3106      021066 001403      BEQ     ETS1         ;BRANCH IF OK
3107      021070 174137 002406      STF     AC1, ANS1    ;SAVE FPU ANSWER
  
```

```

3108 021074 104033          ERROR 33          ;FPS ERROR
3109
3110 021076 173401          ETST1: CMPF AC1,ACD ;ANSWER OK ? (FPU:SOFTWARE)
3111 021100 170000          CFCC              ;COPY CC-S
3112 021102 001436          BEQ EEND1         ;ANSWER CHECKS
3113
3114          ;COMPENSATE IN LOB FOR INACCURACIES
3115 021104 174137 002406  STF AC1,ANS1     ;FPU ANSWER
3116
3117 021110 163737 002420 002410  SUB ANS2+2,ANS1+2 ;GET (SOFT-ANS) - (FPU-ANS)
3118 021116 005637 002406          SBC ANS1+0        ;
3119 021122 163737 002416 002406  SUB ANS2+0,ANS1+0 ;
3120 021130 100011          BPL 10$          ;ALWAYS MAKE +
3121 021132 005137 002410          COM ANS1+2        ;
3122 021136 005137 002406          COM ANS1+0        ;
3123 021142 062737 000001 002410  ADD #1,ANS1+2    ;
3124 021150 005537 002406          ADC ANS1+0        ;
3125
3126 021154 005737 002406          10$: TST ANS1+0    ;
3127 021160 001004          BNE EERR1        ;IF NONZERO IN 16 HOB, SIGN/EXP/FRAC DIFFERS
3128
3129 021162 023727 002410 000006  CMP ANS1+2,#6    ;ALLOW +/- 6 IN LSB OF FRAC
3130 021170 003403          BLE EEND1        ;BR IF OK
3131
3132 021172 174137 002406          EERR1: STF AC1,ANS1 ;FPU ANSWER
3133 021176 104035          ERROR 35         ;ANSWERS DON'T CHECK
3134
3135 021200 005037 002362          EEND1: CLR FPS   ;CLEAR BUFFER
3136
3137
3138
3139
3140          ;*****
3141          ;*TEST 46      ADD, SUB, MUL, DIV EXERCISER
3142          ;*****
3143 021204 000004          †ST46: SCOPE
3144          ;*UNDERFLOW, OVERFLOW INTERRUPTS OFF; ROUND MODE
3145
3146 021206 012737 000600 002400  MOV #600,$FPS    ;SOFTWARE STATUS
3147 021214 005037 002402          CLR $FEC        ;CLEAR STATUS
3148 021220 005037 002404          CLR $FEA
3149 021224 005037 002362          CLR FPS
3150 021230 005037 002364          CLR FEC
3151 021234 005037 002366          CLR FEA
3152
3153 021240 004737 023332          JSR PC,RANDL4   ;GET 6 DOUBLE FLOAT RANDOM NUMBERS
3154 021244 002446 002476          .WORD OP1,OP4
3155 021250 004737 023332          JSR PC,RANDL4
3156 021254 002466 002456          .WORD OP3,OP2
3157 021260 004737 023332          JSR PC,RANDL4
3158 021264 002506 002516          .WORD OP5,OP6
3159 021270 032737 077600 002506  BIT #077600,OP5 ;LET'S NEVER DIVIDE BY ZERO
3160 021276 001770          BEQ 1$          ;OP5 IS ZERO, TRY AGAIN
3161 021300 032737 077600 002516  BIT #077600,OP6 ;
3162 021306 001764          BEQ 1$          ;OP6 IS ZERO, TRY AGAIN
3163

```


3164	021310	004437	023506	JSR	R4 \$POLSH	: ENTER POLISH MODE TO CALCULATE:
3165	021314	023510	002446	\$PUSH	,OP1	
3166	021320	023510	002466	\$PUSH	,OP3	
3167	021324	023566		\$A00		ANS2 = I-----I * I-----I
3168	021326	023510	002506	\$PUSH	,OP5	\ OP5 / \ OP6 /
3169	021332	026364		\$DIV		
3170	021334	023510	002456	\$PUSH	,OP2	
3171	021340	023510	002476	\$PUSH	,OP4	
3172	021344	023562		\$SUB		
3173	021346	023510	002516	\$PUSH	,OP6	
3174	021352	026364		\$DIV		
3175	021354	025244		\$MUL		
3176	021356	023540	002416	\$POPX	,ANS2	
3177						
3178	021362	170127	040200	LDFPS	#040200	: NO CHECKS
3179	021366	172437	002416	LDO	ANS2,ACO	: GET SOFTWARE ANSWER
3180	021372	013700	002400	MOV	\$FPS,RO	: DISPLAY \$FPS
3181	021376	012737	021404	MOV	#.+6,\$LPRDR	: RESET LOOP ADDRESS
3182						
3183						:*****
3184						
3185	021404	170127	000600	LDFPS	#000600	: INITIAL FPS
3186	021410	172537	002446	LDO	OP1,AC1	: AC1 (- OP1
3187	021414	172137	002466	A000	OP3,AC1	: AC1 (- OP1+OP3
3188	021420	174537	002506	DIV0	OP5,AC1	: AC1 (- (OP1+OP3)/OP5
3189	021424	172637	002456	LDO	OP2,AC2	: AC2 (- OP2
3190	021430	173237	002476	SUB0	OP4,AC2	: AC2 (- OP2-OP4
3191	021434	174637	002516	DIV0	OP6,AC2	: AC2 (- (OP2-OP4)/OP6
3192	021440	171102		MULD	AC2,AC1	: AC1 (- (OP1+OP3)/OP5*(OP2-OP4)/OP6
3193						
3194	021442	170237	002362	STFPS	FPS	: STORE STATUS AFTERWARD
3195	021446	023737	002362	002400	FPS,\$FPS	: CHECK STATUS
3196	021454	001403		BEQ	ETS12	: BRANCH IF OK
3197	021456	174137	002406	STD	AC1,ANS1	: SAVE FPU ANSWER
3198	021462	104034		ERROR	34	: FPS ERROR
3199						
3200	021464	173401		ETST2:	CMPO AC1,ACO	: ANSWER OK ? (FPU:SOFTWARE)
3201	021466	170000			CFCC	: COPY CC-S
3202	021470	001474			BEQ EEND2	: ANSWER CHECKS
3203						
3204						: COMPENSATE IN LOB FOR INACCURACIES
3205	021472	174137	002406	STD	AC1,ANS1	: FPU ANSWER
3206						
3207	021476	163737	002424	002414	IB ANS2+6,ANS1+6	: GET (SOFT-ANS) - (FPU-ANS)
3208	021504	005637	002412		JC ANS1+4	
3209	021510	005637	002410		SBC ANS1+2	
3210	021514	005637	002406		SBC ANS1+0	
3211	021520	163737	002422	002412	SUB ANS2+4,ANS1+4	
3212	021526	005637	002410		SBC ANS1+2	
3213	021532	005637	002406		SBC ANS1+0	
3214	021536	163737	002420	002410	SUB ANS2+2,ANS1+2	
3215	021544	005637	002406		SBC ANS1+0	
3216	021550	163737	002416	002406	SUB ANS2+0,ANS1+0	
3217	021556	100021			BPL 10\$: ALWAYS MAKE +
3218	021560	005137	002414		COM ANS1+6	
3219	021564	005137	002412		COM ANS1+4	

```

3220 021570 005137 002410      COM      ANS1+2      ;
3221 021574 005137 002406      COM      ANS1+0      ;
3222 021600 062737 000001 002414  ADD      #1,ANS1+6  ;
3223 021606 005537 002412      ADC      ANS1+4      ;
3224 021612 005537 002410      ADC      ANS1+2      ;
3225 021616 005537 002406      ADC      ANS1+0      ;
3226
3227 021622 005737 002406      10$:  TST      ANS1+0      ;
3228 021626 001012      BNE      EERR2      ; IF NONZERO IN 16 HOB, SIGN/EXP/FRAC DIFFERS
3229 021630 005737 002410      TST      ANS1+2      ;
3230 021634 001007      BNE      EERR2      ; IF NONZERO IN 16 H-MOB, FRAC-B DIFFERS
3231 021636 005737 002412      TST      ANS1+4      ;
3232 021642 001004      BNE      EERR2      ; IF NONZERO IN 16 L-MOB, FRAC-C DIFFERS
3233
3234 021644 023727 002414 000005  CMP      ANS1+6,#5  ; ALLOW +/- 5 IN LSB OF FRAC
3235 021652 003403      BLE      EEND2      ; BR IF OK
3236
3237 021654 174137 002406      EERR2: STD      AC1,ANS1 ; FPU ANSWER
3238 021660 104036      ERROR   36          ; ANSWERS DON'T CHECK
3239
3240 021662 005037 002362      EEND2: CLR      FPS      ; CLEAR BUFFER

```

3241
3242
3243
3244 021666 000004
3245 021670 005037 001166
3246 021674 005037 001104
3247 021700 005037 001102
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259 021704 076600 000022
3260 021710 032700 000020
3261 021714 001423
3262
3263 021716 032777 000002 157220
3264 021724 001017
3265
3266 021726 012701 010000
3267 021732 076600 000144
3268 021736 030100
3269 021740 001402
3270 021742 040100
3271 021744 000401
3272 021746 050100
3273 021750 076600 000344
3274
3275 021754 030100
3276 021756 001002
3277 021760 000137 003516
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291 021764
3292 021764 005037 001104
3293 021770 005037 001102
3294 021774 005037 001166
3295
3296 022000 005327

:SBTTL SUB PASS END CONTROL

SCOPE ;CHECK FOR TEST ITERATIONS HERE
CLR \$TIMES ;DONT ITERATE THIS "TEST"
CLR \$ERFLG ;NO ERRORS HERE
CLR \$STNM ;ZAP TEST ## WHEN DONE WITH A PASS
;IF TEST ONLY EITHER HFP OR WFP, ENTER "EOP" ROUTINE DIRECTLY

IF IN ALTERNATE HFP/WFP MODE,
COMPLEMENT FLAG<5> HFP ENABLE BIT,
ENTER EOP ROUTINE ONLY IF ABOUT TO TEST HFP NEXT,
TESTING SEQUENCE IS: PASS#1 HFP SUB-PASS
PASS#1 HFP SUB-PASS
PASS#2 HFP SUB-PASS
...

MED \$R0,WHAMI ;GET WHAMI INTO R0
BIT \$BIT04,R0 ;1=HFP PRESENT, 0=NONE
BEQ \$EOP ;EXIT IF NONE
BIT \$SW01,\$SWR ;1=HFP OR WFP TEST ONLY
BNE \$EOP ;0=ALTERNATE HFP AND WFP TESTS
MOV \$R12,R1 ;HFP PRESENT, AND IN ALTERNATE MODE;
MED \$RFLAG ;SO READ FLAGS
BIT \$R1,R0 ;COMPLEMENT FLAG<5>=BIT12=HFP ENABLE FLAG
BEQ \$S ;
BIC \$R1,R0 ;CLEAR BIT 12
BR \$S ;
1\$: BIS \$R1,R0 ;SET BIT 12
2\$: MED \$RFLAG ;REWRITE FLAGS
BIT \$R1,R0 ;HFP OR WFP NEXT ?
BNE \$EOP ;IF HFP AGAIN, START NEW PASS
JMP \$SUBPAS ;IF WFP, NEXT SUBPASS

:SBTTL END OF PASS ROUTINE (MODIFIED SYSMAC)

*INCREMENT THE PASS NUMBER (\$PASS)
*LOOP FOR 256 SUBPASSES TO MAKE A PASS
*IF SW<10>=0, DING BELL ON PASS END
*IF SW<12>=0, TYPE STATISTICS ON PASS END
*IF THERE'S A MONITOR, GO TO IT
* ELSE JUMP TO NEWPAS

\$EOP:
CLR \$ERFLG ;ZERO ERROR FLAG
CLR \$STNM ;ZERO TEST NUMBER
CLR \$TIMES ;ZERO NUMBER OF ITERATIONS
DEC (PC)+ ;SUBPASS LOOP ?

3297	022002	000400			10S:	.WORD	256.	:	USE 256. SUBPASSES PER LOOP
3298	022004	003402				BLE	11S	:	NO, GO BUMP PASS COUNTER
3299	022006	000137	003516			JMP	2#SUBPAS	:	NEXT SUB PASS
3300	022012	012737			11S:	MOV	(PC)+,2(PC)+	:	RESTORE COUNTER
3301	022014	000400				.WORD	256.	:	
3302	022016	022002				.WORD	10S	:	
3303								:	
3304	022020	012737	001210			INC	\$PASS	:	INCREMENT PASS COUNT
3305	022024	04737	100000	001210		BIC	#100000,\$PASS	:	BUT NEVER LET IN GO NEGATIVE
3306	022032	005327				DEC	(PC)+	:	PASS LOOP ?
3307	022034	000001			SEOPCT:	.WORD	1	:	FALL THRU
3308	022036	003027				BIT	\$DOAGN	:	YES
3309	022040	012737				MOV	(PC)+,2(PC)+	:	RESTORE COUNTER
3310	022042	000001			SENOCT:	.WORD	1	:	
3311	022044	022034				.WORD	SEOPCT	:	
3312								:	
3313	022046	032777	002000	157070		BIT	#SW10,2SWR	:	BELL ON PASS END ?
3314	022054	001002				BNE	1S	:	NO
3315	022056	104401	001172			TYPE	,\$BELL	:	YES
3316								:	
3317	022062	032777	010000	157054	1S:	BIT	#SW12,2SWR	:	INHIBIT MESSAGE ?
3318	022070	001002				BNE	\$GET42	:	YES
3319	022072	004737	022122			JSR	PC,STATS	:	TYPE STATISTICS
3320								:	
3321	022076	013700	000042		\$GET42:	MOV	2#42,R0	:	GET MONITOR ADDRESS
3322	022102	001405				BEQ	\$DOAGN	:	NO MONITOR
3323	022104	000005				RESET		:	CLEAR WORLD
3324	022106	004710			SENDAD:	JSR	PC,(R0)	:	GO TO MONITOR
3325	022110	000240				NOP		:	
3326	022112	000240				NOP		:	RESERVED FOR ACT11
3327	022114	000240				NOP		:	
3328								:	
3329	022116	000137	003450		\$DOAGN:	JMP	2#NEWPAS	:	RETURN

3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385

022122 010046
022124 012700 022230
022130 001404
022132 104401 022317
022136 004737 022202
022142 012700 022254
022146 001404
022150 104401 022664
022154 004737 022202
022160 012700 022272
022164 001404
022166 104401 023055
022172 004737 022202
022176 012600
022200 000207
022202 013046
022204 104403
022206 006
022207 000
022210 005710
022212 001403
022214 104401 022312
022220 000770
022222 104401 022314
022226 000207
022230 002526 002540 002542
000036 000544 002546 002530
000044 002532 002534 002536
000052 000000
020054 002550 002554 002556
022262 002560 002562 002552
022270 000000
022272 002564 002572 002574
022300 002576 002600 002566
022306 002570 000000
022312 000011
022314 005015 000

```

.SBTTL STATISTICS TYPEOUT SUBROUTINE
;#THIS ROUTINE TYPES OUT A NICELY FORMATTED REPORT
;#CONTAINING STATISTICS ON THE NUMBER OF OPERANDS
;#USED IN DIFFERENT SELECT CASES FOR THE $ADD, $SUB,
;#$MUL, AND $DIV ROUTINES. THE DATA VALUES ARE TAKEN
;#FROM THE COUNTERS ADDC?, MULC?, AND DIVC?.
;#
;#CALLED BY: JSR PC,STATS
;#
STATS: MOV RO,-(SP) ;SAVE REGISTERS
;#
MOV #SVEC1,RO ;FIRST DATA LINE ADDR VECTOR
1$ BEQ 1$ ;NONE IF ZERO
TYPE SHDR1 ;HEADER
JSR PC,10$ ;DATA LINE
1$: MOV #SVEC2,RO ;SECOND DATA LINE ADDR VECTOR
BEQ 2$ ;NONE IF ZERO
TYPE SHDR2 ;HEADER
JSR PC,10$ ;DATA LINE
2$: MOV #SVEC3,RO ;THIRD DATA LINE ADDR VECTOR
BEQ 3$ ;NONE IF ZERO
TYPE SHDR3 ;HEADER
JSR PC,10$ ;DATA LINE
3$: MOV (SP)+,RO ;RESTORE REGISTERS
RTS PC ;EXIT
;#INTERNAL SUBR FOR DATA LINE TYPEOUT
10$: MOV 2(RO)+,-(SP) ;MOVE NUMBER ON STACK, BUMP TO NEXT
TYPOS ;TYPE OCTAL
.BYTE 6 ;MAX 6 DIGITS
.BYTE 0 ;SUPPRESS LEADING ZEROS
TST (RO) ;BUMPED TO LAST VECTOR?
BEQ 11$ ;YES, ONTO NEXT LINE
TYPE SHT ;TYPE A <HT>
BR 10$ ;CONTINUE WITH THIS LINE
11$: TYPE SCRLF ;TYPE A <CR><LF>
RTS PC ;DONE
;#DATA VECTORS:
SVEC1: .WORD ADDC0,ADDC5,ADDC6,ADDC7,ADDC8,ADDC1,ADDC2,ADDC3,ADDC4,0
SVEC2: .WORD MULC0,MULC2,MULC3,MULC4,MULC5,MULC1,0
SVEC3: .WORD DIVC0,DIVC3,DIVC4,DIVC5,DIVC6,DIVC1,DIVC2,0
;#HEADERS, ETC:
SHT: .ASCIZ (1) ;<HT>
SCRLF: .ASCIZ (15)(12) ;<CR><LF>
    
```

3386	022317	015	005012
3387	022318	025052	020052
3388	022319	022101	051511
3389	022320	051503	025040
3390	022344	005015	012
3391	022347	050	047516
3392	022354	026440	040440
3393	022362	047040	046525
3394	022370	051522	040440
3395	022376	052440	051516
3396	022404	042516	020104
3397	022412	045224	020114
3398	022420	042524	042507
3399	022430	006451	07012
3400	022433	022101	027504
3401	022440	020102	047111
3402	022446	052522	052103
3403	022454	051516	006472
3404	022461	124	052117
3405	022466	026411	026455
3406	022474	051105	046106
3407	022502	026455	004455
3408	022510	047125	042504
3409	022516	047514	026527
3410	022524	047411	020120
3411	022532	004460	050117
3412	022540	030043	047411
3413	022546	036501	004460
3414	022554	047040	006517
3415	022561	116	046525
3416	022566	004522	047524
3417	022574	020114	027527
3418	022602	041101	042514
3419	022610	052117	046101
3420	022616	042457	040516
3421	022624	004505	050117
3422	022632	030043	047411
3423	022640	036502	004460
3424	022646	041040	030075
3425	022654	044123	043111
3426	022662	000012	
3427	022664	005015	052515
3428	022672	050111	054514
3429	022700	051516	051124
3430	022706	044514	047117
3431	022714	005015	
3432	022716	047524	040524
3433	022724	026455	047455
3434	022732	043122	047514
3435	022740	026455	026411
3436	022746	042116	051105
3437	022754	053517	026455
3438	022762	036501	020060
3439	022770	027504	051117

```
SHDR1: .ASCII (15)(12)(12)
        .ASCII "*** STATISTICS ***"(15)(12)(12)
```

```
.ASCII "(NOTE - ALL NUMBERS ARE UNSIGNED OCTAL INTEGERS)"(15)(12)(12)
```

```
.ASCII "ADD/SUB INSTRUCTIONS:"(15)(12)
```

```
.ASCII "TOTAL ---OVERFLOW--- --UNDERFLOW--- OP A=0 OP A=0 OP A=0 NO"(15)(12)
```

```
.ASCIZ "NUMBER TOTAL W/ENABLE TOTAL W/ENABLE OP B=0 OP B=C OP B=0 SHIFT"(15)(12)
```

```
SHDR2: .ASCII (15)(12)"MULTIPLY INSTRUCTIONS:"(15)(12)
```

```
.ASCII "TOTAL ---OVERFLOW--- --UNDERFLOW--- A=0 AND/OR"(15)(12)
```

H07

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 70
 DGFPOB.P11 04-MAY-77 17:30 STATISTICS TYPEOUT SUBROUTINE

SEQ 0071

3440	022776	052516	041115	051105	.ASCIZ "NUMBER TOTAL W/ENABLE TOTAL W/ENABLE	B=0" (15) (12)
3441	023004	052011	052117	046101		
3442	023012	052740	042457	040516		
3443	023020	047105	004505	047524		
3444	023028	040524	020114	027527		
3445	023034	047105	041101	042514		
3446	023042	020011	020040	041040		
3447	023050	030075	005015	000		
3448	023055	015	042012	053111	SHOR3: .ASCII (15) (12) "DIVIDE INSTRUCTIONS:" (15) (12)	
3449	023063	042111	020105	047111		
3450	023070	052123	052522	052103		
3451	023076	047511	051516	006472		
3452	023104	012				
3453	023105	124	052117	046101	.ASCII "TOTAL ---OVERFLOW--- --UNDERFLOW---	NUMER DENOM" (15) (12)
3454	023112	026411	026455	053117		
3455	023120	051105	046106	053517		
3456	023126	026455	004455	026455		
3457	023134	047123	042504	043122		
3458	023142	047514	026527	026455		
3459	023150	020011	052516	042515		
3460	023156	004522	042040	047105		
3461	023164	046517	005015			
3462	023170	052516	041115	051105	.ASCIZ "NUMBER TOTAL W/ENABLE TOTAL W/ENABLE	A=0 B=0" (15) (12)
3463	023176	052011	052117	046101		
3464	023204	053440	042457	040516		
3465	023212	046102	004505	047524		
3466	023220	040524	020114	027527		
3467	023226	047105	041101	042514		
3468	023234	020011	020040	036501		
3469	023242	004460	020040	041040		
3470	023250	030075	005015	000		
3471	023256				EVEN ;BACK TO AN EVEN BOUNDARY	

```

3472 .SBTTL FPP TRAP CATCHER
3473
3474 023256 012637 002370 FPPILT: MOV (SP)+, FPPOPC ; POP OLD PC FOR DISPLAY
3475 023262 012637 002372 MOV (SP)+, FPPOPS ; POP OLD PS FOR DISPLAY
3476 023266 170237 002362 STFPS FPS ; GET FPS
3477 023272 170337 002364 STST FEC ; GET FEC/FEA
3478 023276 005737 002362 TST FPS ; TEST ERROR BIT
3479 023302 100005 BPL 1$ ; OFF - NO ERROR BIT SET, BUT TRAPPED
3480
3481 023304 032737 040000 002362 BIT #040000, FPS ; ON - IT SHOULD BE ON A TRAP
3482 023312 001001 BNE 1$ ; TEST INTERRUPT ENABLE BIT
3483
3484 023314 000401 BR 2$ ; ON - INTR DISABLED, BUT TRAPPED
3485 023316 104015 1$: ERROR 1$ ; OFF - ABLE TO INTR, SO IGNORE IT,
3486 023320 013746 002372 2$: MOV FPPOPS, -(SP) ; AND SKIP THE ERROR
3487 023324 013746 002370 MOV FPPOPC, -(SP) ; SIGNAL UNEXPECTED FPP TRAP
3488 023330 000002 RTI ; PUSH PSW
3489 ; PUSH PC
; CONTINUE, RECOVER AT LAST TRAP ONLY
  
```


3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544

023332 010546
023334 012705 000004
023340 000403
023342 010546
023344 012705 000002
023350 010446
023352 010346
023354 010246
023356 010146
023360 010046
023362 005046
023364 016600 000016
023370 012003
023372 012004
023374 010066 000016
023400 011300
023402 011401
023404 012702 000007
023410 005016
023412 006300
023414 006101
023416 006116
023420 077204
023422 061316
023424 005501
023426 061401
023430 005516
023432 062700 001057
023436 005501
023440 005516
023442 062701 047401
023446 005516
023450 062716 000006
023454 061600
023456 005501
023460 010023
023462 010124
023464 077531
023466 005726
023470 012600
023472 012601
023474 012602
023476 012603
023500 012604
J23502 012605
023504 000207

.SBTTL RANDOM NUMBER GENERATOR

```

: *CALLED BY JSR PC,RANDL4 - FOR DOUBLE FLOAT NUMBERS
: *          .WORD N1,N2 - IN LOCATIONS N1 AND N2
: *
: *          JSR PC,RANDL2 - FOR SINGLE FLOAT NUMBERS
: *          .WORD N1,N2 - IN LOCATIONS N1 AND N2
: *
RANDL4: MOV R5,-(SP) :SAVE R5
        MOV #4,R5 :4 WORDS AT EACH
        BR RAND
RANDL2: MOV R5,-(SP) :SAVE R5
        MOV #2,R5 :2 WORDS AT EACH
RAND:   MOV R4,-(SP) :SAVE REGISTERS
        MOV R3,-(SP)
        MOV R2,-(SP)
        MOV R1,-(SP)
        MOV R0,-(SP)
        CLR -(SP) :EXTRA REGISTER
        MOV 16(SP),R0 :GET PC FOR RETURN
        MOV (R0)+,R3 :FIRST NUMBER DEST PTR
        MOV (R0)+,R4 :SECOND NUMBER DEST PTR
        MOV R0,16(SP) :STORE NEW RETURN ADDRESS
        MOV (R3),R0 :R0 INITIAL NUMBER
        MOV (R4),R1 :R1 INITIAL NUMBER
1$:     MOV #7,R2 :SHIFT COUNT
        CLR (SP) :CLEAR LOB
2$:     ASL R0 :SHIFT R0 LEFT
        ROL R1 :AND ROTATE CARRY INTO R1
        ROL (SP) :AND ROTATE CARRY INTO EXT
        SOB R2,2$: :7 SHIFTS
        ADD # TO MAKE # 129
        ADC R1 :PROPOGATE CARRY
        ADD (R4),R1 :ADD # TO MAKE # 129
        ADC (SP) :PROPOGATE CARRY
        ADD #001057,R0 :ADD LOW CONSTANT
        ADC R1 :PROPOGATE CARRY
        ADC (SP) :PROPOGATE CARRY
        ADD #047401,R1 :ADD HIGH CONSTANT
        ADC (SP) :PROPOGATE CARRY
        ADD #000006,(SP) :ADD HIGHEST CONSTANT
        ADD (SP),R0 :REPRIME R0 WITH HIGHEST DIGIT
        ADC R1 :PROPOGATE CARRY
        MOV R0,(R3)+ :SAVE R0
        MOV R1,(R4)+ :SAVE R1
        SOB R5,1$: :LOOP FOR REQ'D NUMBER OF WORDS
        TST (SP)+ :POP TEMP REG
        MOV (SP)+,R0
        MOV (SP)+,R1
        MOV (SP)+,R2
        MOV (SP)+,R3
        MOV (SP)+,R4
        MOV (SP)+,R5
        RTS PC :RETURN
  
```

3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600

.SBTTL POLISH EXPRESSION ROUTINES

```

:*"POLISH EXPRESSION" ALGEBRA IS A STACK ORIENTED PROCEDURE FOR
:*THE EVALUATION OF ALGEBRAIC EXPRESSIONS. BY THIS, WE MEAN
:*THAT THE STACK (IN OUR CASE, WE WILL BE USING THE STANDARD
:*SYSTEM STACK USING R6) IS USED FOR THE STORAGE, ON A LAST IN-
:*FIRST OUT BASIS, OF ALL OPERANDS, AND RESULTS. ALL THE
:*ARITHMETIC ROUTINES (SPECIFICALLY, OUR $ADD, $SUB, $MUL, AND
:*$DIV ROUTINES) EXPECT THEIR TWO OPERANDS TO BE THE TOP TWO
:*ELEMENTS ON THE STACK, AND THEY LEAVE THEIR RESULT AS THE
:*TOP ELEMENT ON THE STACK, REMOVING (POPPING) THE INITIAL
:*OPERANDS IN THE PROCESS. OTHER ROUTINES ARE PRESENT FOR
:*ADDING/REMOVING ELEMENTS TO/FROM THE STACK - $POPX, TO TAKE
:*THE TOP ELEMENT OFF, AND $PUSH, TO PUT A NEW ELEMENT ON THE
:*TOP. IT IS IMPORTANT TO NOTE THAT OPERATORS WILL AT MOST
:*REFERENCE THE TOP TWO ELEMENTS ON THE STACK; THE OTHERS ARE
:*INACCESSIBLE UNTIL OUTER ELEMENTS ARE OPERATED UPON. FOR
:*EXAMPLE, THE EXPRESSION:

```

$$E = (A + B) * (C - D)$$

```

:*COULD BE EVALUATED BY THE POLISH EXPRESSION:

```

```

:*      $PUSH  A      ;OPERAND A ONTO STACK
:*      $PUSH  B      ;OPERAND B ONTO STACK
:*      $ADD   ;FORM A+B, SAVE FOR LATER
:*      $PUSH  C      ;OPERAND C ONTO STACK
:*      $PUSH  D      ;OPERAND D ONTO STACK
:*      $SUB   ;FORM C-D ON TOP
:*      ;NOTE - THE TOP TWO OPERANDS ARE NOW:
:*              (A+B) AND (C-D)
:*      $MUL   ;FORM (A+B) * (C-D) ON TOP
:*      $POPX  E      ;POP RESULT FROM STACK INTO E

```

```

:*NOTE THAT OTHER POLISH EXPRESSIONS ARE POSSIBLE FOR COMPUTING
:*THIS EXAMPLE, IN GENERAL THERE IS MORE THAN ONE WAY TO
:*CALCULATE A GIVEN EXPRESSION.

```

```

:*THIS ROUTINE ENTERS US INTO POLISH MODE
$POLSH: JMP      2(R4)+      ;ENTER POLISH MODE

```

```

:*PUSH OPERAND ON STACK FROM LOCATION SPECIFIED
:*IN CONTENTS OF NEXT WORD AFTER $PUSH CALL
:*2/4 WORDS DEPENDING UPON F/D MODE
$PUSH: MOV      (R4)+,R0      ;GET PTR TO SOURCE
        TSTB   $FPS
        BPL    IS
        MOV    6(R0),-(SP)    ;FUSH DOUBLE ON STACK
        MOV    4(R0),-(SP)
IS:     MOV    2(R0),-(SP)    ;PUSH FLOAT ON STACK
        MOV    (R0),-(SP)
        JMP    2(R4)+

```

```

:*POP OPERAND FROM STACK INTO LOCATION SPECIFIED
:*IN CONTENTS OF NEXT WORD AFTER $POPX CALL

```

```

023506 000134
023510 012400
023512 105737 002400
023516 100004
023520 016046 000006
023524 016046 000004
023530 016046 000002
023534 011046
023536 000134

```

3601			
3602	023540	012400	
3603	023542	012620	
3604	023544	012620	
3605	023546	105737	002400
3606	023552	100002	
3607	023554	012620	
3608	023556	012620	
3609	023560	000204	

```

:*2/4 WORDS DEPENDING UPON F/D MODE
$POPX:  MOV    (R4)+,R0      ;GET PTR TO DESTINATION
        MOV    (SP)+,(R0)+  ;POP FLOAT FROM STACK
        MOV    (SP)+,(R0)+  ;
        TSTB  $FPS          ;
        BPL   1$           ;
        MOV    (SP)+,(R0)+  ;POP DOUBLE FROM STACK
        MOV    (SP)+,(R0)+  ;
1$:     RTS    R4           ;EXIT POLISH MODE

```

3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665

.SBTTL FLOATING POINT SOFTWARE ROUTINES

*FLOATING POINT SOFTWARE ADD/SUBTRACT ROUTINES

* THIS ROUTINE TAKES THE TOP TWO ELEMENTS ON THE STACK
* (LENGTH OF 2 OR 4 WORDS DEPENDING UPON WHETHER THE
* F/D MODE BIT7 IN SFPS IS 0 OR 1 RESPECTIVELY) AND
* REPLACES THEM WITH THEIR SUM/DIFFERENCE.

EXAMPLE:	SPUSH	SPUSH
	ADDR(OPERAND A)	ADDR(OPERAND A)
	SPUSH	SPUSH
	ADDR(OPERAND B)	ADDR(OPERAND B)
	SADD	SSUB
	SPOPX	SPOPX
	ADDR(RESULT)	ADDR(RESULT)
	RESULT=A+B	RESULT=A-B

* NOTE ROUTINE IS CALLED THROUGH POLISH MODE OPERATORS.

* ALL OPERATIONS ARE CARRIED OUT IN DOUBLE (LENGTH=4)
* PRECISION MODE. SINGLE PRECISION OPERANDS ARE
* EXTENDED (VIA \$CONV SUBROUTINE) WITH LOW-ORDER
* ZEROS, AND LATER TRUNCATED BACK TO SINGLE PRECISION
* FROM THE DOUBLE PRECISION RESULT.

* STATUS BITS:

* THE N, Z, V, AND C BITS OF SFPS ARE SET AS FOLLOWS:
* N = 1 IF RESULT NEGATIVE (IE, BIT15 = 1),
* ELSE N = 0
* Z = 1 IF RESULT ZERO (IE, EXPONENT = 000(8)),
* ELSE Z = 0
* V = 1 IF ARITHMETIC OVERFLOW OCCURRED,
* ELSE V = 0
* C = 0 ALWAYS

* ERROR CONDITIONS:

* IF AN ARITHMETIC OVERFLOW CONDITION OCCURS, THE RESULT
* WILL BE SET TO ZERO IF THE OVERFLOW ENABLE BIT OF SFPS
* (BIT09) IS ZERO. IF THIS BIT IS SET, THEN THE RESULT
* WILL BE CORRECT, EXCEPT THE EXPONENT WILL BE UNDERBIASED
* BY 400(8). ALSO, THE SFPS ERROR BIT (BIT15) WILL BE SET,
* SFEC WILL BE SET TO 10(8), AND SFEA WILL BE LOADED WITH
* THE VALUE IN LOCATION "EXPFEA". IN EITHER INSTANCE,
* THE V-BIT (BIT01) WILL BE SET.
* IF AN ARITHMETIC UNDERFLOW CONDITION OCCURS, THE RESULT
* WILL BE SET TO ZERO IF THE UNDERFLOW ENABLE BIT OF SFPS
* (BIT10) IS ZERO. IF THIS BIT IS SET, THEN THE RESULT
* WILL BE CORRECT, EXCEPT THE EXPONENT WILL BE OVERBIASED
* BY 400(8). ALSO, THE SFPS ERROR BIT (BIT15) WILL BE SET,
* SFEC WILL BE SET TO 12(8), AND SFEA WILL BE LOADED WITH
* THE VALUE IN LOCATION "EXPFEA".

* ALL REGISTERS ARE PRESERVED UPON ENTRY, AND RESTORED UPON
* EXIT. THE ROUTINES ARE RE-ENTRANT.

```

: * ADAPTED FROM POP-11 FORTRAN SOFTWARE
: * BY DONALD NORTH, SEPTEMBER, 1975.
: *
3666
3667
3668
3669
3670 023562 062716 100000 $SUB: ADD #100000, (SP) ; CHANGE SIGN OF TOP ITEM
3671 023566 005237 002526 $ADD: INC ADDC0 ; CTR: TOTAL NUMBER OF ADD/SUB
3672 023572 042737 000017 002400 BIC #17, $FPS ; CLEAR STATUS BITS N Z V C
3673 023600 105737 002400 TSTB $FPS ; TEST MODE
3674 023604 100402 BMI 27$ ; D-MODE
3675 023606 004737 027622 JSR PC, $CONV ; D-MODE: CONVERT 2 F OPDS TO D
3676 023612 010046 27$: MOV R0, -(SP)
3677 023614 010146 MOV R1, -(SP)
3678 023616 010246 MOV R2, -(SP) ; SAVE ALL REGISTERS
3679 023620 010346 MOV R3, -(SP)
3680 023622 010446 MOV R4, -(SP)
3681 023624 010546 MOV R5, -(SP)
3682 023626 005046 CLR -(SP) ; CLEAR SIGNS
3683 023630 005004 CLR R4 ; CLEAR EXPONENTS
3684 023632 005005 CLR R5
3685 023634 006366 000024 ASL 24(SP) ; SHIFT OUT SIGN OF TOP ITEM
3686 023640 006166 000022 ROL 22(SP)
3687 023644 006166 000020 ROL 20(SP)
3688 023650 006166 000016 ROL 16(SP) ; SHIFT A1
3689 023654 156604 000017 BISB 17(SP), R4 ; GET E1
3690 023660 001011 BNE 31$ ; JUMP IF NON ZERO
3691 023662 005726 TST (SP)+ ; FLUSH SIGNS
3692 023664 032766 077600 000024 BIT #077600, 24(SP) ; B=0, A=0 TOO ?
3693 023672 001456 BEQ 32$ ; YES, BOTH ZERO
3694 023674 005237 002532 INC ADDC2 ; NO, CTR: B=0, A#0
3695 023700 000137 024712 JMP 3$ ; DONE
3696 023704 106116 31$: ROLB (SP) ; GET S1
3697 023706 006366 000034 ASL 34(SP) ; SHIFT OUT SIGN OF 2ND ITEM
3698 023712 006166 000032 ROL 32(SP)
3699 023716 006166 000030 ROL 30(SP)
3700 023722 006166 000026 ROL 26(SP) ; SHIFT A2
3701 023726 156605 000027 BISB 27(SP), R5 ; GET E2
3702 023732 001042 BNE 2$ ; JUMP IF NON ZERO
3703 023734 106016 RORB (SP) ; RECONSTRUCT A1
3704 023736 006066 000016 ROR 16(SP)
3705 023742 006066 000020 ROR 20(SP)
3706 023746 006066 000022 ROR 22(SP)
3707 023752 006066 000024 ROR 24(SP)
3708 023756 016666 000016 000026 MOV 16(SP), 26(SP) ; FIRST ARG TO TOP OF STACK
3709 023764 016666 000020 000030 MOV 20(SP), 30(SP)
3710 023772 016666 000022 000032 MOV 22(SP), 32(SP)
3711 024000 016666 000024 000034 MOV 24(SP), 34(SP)
3712 024006 005726 TST (SP)+ ; FLUSH SIGNS
3713 024010 032766 077600 000024 BIT #077600, 24(SP) ; A=0, B=0 TOO ?
3714 024016 001404 BEQ 32$ ; YES, BOTH ZERO
3715 024020 005237 002530 INC ADDC1 ; NO, CTR: A=0, B#0
3716 024024 000137 024712 JMP 3$ ; DONE
3717 024030 005237 002534 32$: INC ADDC3 ; CTR: A=0, B=0
3718 024034 000137 024672 JMP 29$ ; DONE, TRUE ZERO RESULT
3719 024040 106166 000001 2$: ROLB 1(SP) ; GET S2
3720 024044 112766 000001 000027 MOVB #1, 27(SP) ; INSERT NORMAL BIT
3721 024052 112766 000001 000017 MOVB #1, 17(SP)
  
```

3722	024060	160405			SUB	R4, R5	RS=E2-E1, R4=E1
3723	024062	003011			BGT	45,	JUMP IF E2>E1
3724	024064	016600	000026		MOV	20(SP), R0	R0=R2
3725	024070	016601	000030		MOV	20(SP), R1	R1=R2
3726	024074	016602	000032		MOV	22(SP), R2	R2=C2
3727	024100	016603	000034		MOV	24(SP), R3	R3=C2
3728	024104	000427			BR	55,	GO CHECK SIGNS
3729	024106	060504		45:	ADD	R5, R7	RS=E2-E1, R4=E2, E2>E1
3730	024110	016600	000016		MOV	16(SP), R0	R0=R1
3731	024114	016601	000020		MOV	20(SP), R1	R1=B1
3732	024120	016602	000022		MOV	22(SP), R2	R2=C1
3733	024124	016603	000024		MOV	24(SP), R3	R3=D1
3734	024130	016666	000016	000016	MOV	16(SP), R0	
3735	024136	016666	000030	000020	MOV	20(SP), R1	
3736	024144	016666	000032	000022	MOV	22(SP), R2	
3737	024150	016666	000034	000024	MOV	24(SP), R3	
3738	024160	000316			SHR	(SP)	EXCHANGE SIGNS
3739	024166	005405			NEG	R5	E1-E2
3740	024164	126616	000001		CMPS	1(SP), (SP)	COMPARE SIGNS
3741	024170	001412		55:	BEQ	55,	SAME, GO CHECK EXPONENT
3742	024172	005403			NEG	R3	NEGATE OPERAND
3743	024174	005502			ROC	R2	
3744	024176	005501			ROC	R1	
3745	024200	005500			ROC	R0	
3746	024202	005402			NEG	R2	
3747	024204	005501			ROC	R1	
3748	024206	005500			ROC	R0	
3749	024210	005401			NEG	R1	
3750	024212	005500			ROC	R0	
3751	024214	005400			NEG	R0	
3752	024216	005705		65:	TST	R5	CHECK EXPONENTS
3753	024220	001466			BEQ	75,	JUMP IF E1=E2
3754	024222	022705	177707		CMPS	#-57., R5	ANY POINT IN SHIFTING?
3755	024226	003413			BLE	85,	YES
3756	024230	016600	000016		MOV	16(SP), R0	NO, ANSWER IS OPERAND
3757	024234	016601	000020		MOV	20(SP), R1	WITH LARGER EXPONENT
3758	024240	016602	000022		MOV	22(SP), R2	
3759	024244	016603	000024		MOV	24(SP), R3	
3760	024250	005237	002536		INC	R0C4	CTR: NO SHIFT
3761	024254	000501			BR	95,	
3762	024256	022705	177770		CMPS	#-8., R5	CHECK # OF BITS TO SHIFT
3763	024262	003437		85:	BLE	105,	JUMP IF LESS THAN 1/2 WORD
3764	024264	005700			TST	R0	
3765	024266	006746			SXT	-(SP)	EXTEND SIGN
3766	024270	022705	177760		CMPS	#-16., R5	
3767	024274	002411		125:	BLT	115,	JUMP IF LESS THAN 1 WORD
3768	024276	010203			MOV	R2, R3	SHIFT A WORD AT A TIME
3769	024300	010102			MOV	R1, R2	
3770	024332	010001			MOV	R0, R1	
3771	024304	011600			MOV	(SP), R0	USE EXTENSION
3772	024306	062705	000020		ADD	#16., R5	ADJUST EXPONENT
3773	024312	001366			BNE	125,	TRY AGAIN
3774	024314	005726			TST	(SP)+	POP EXTENSION
3775	024316	000427			BR	75,	SHIFT DONE
3776	024320	022705	177775		CMPS	#-3, R5	JUMP IF LESS THAN 4 TO SHIFT
3777	024324	003415			BLE	135,	

3778	024326	010416		MOV	R4, (SP)	: SAVE EXP & SHIFT COUNT	
3779	024330	010546		MOV	R5, -(SP)		
3780	024332	010104		MOV	R1, R4	: SAVE R1	
3781	024334	073005		ASHC	R5, R5	: SHIFT HIGH ORDER	
3782	024336	010205		MOV	R2, R5	: SAVE R2	
3783	024340	073416		ASHC	(SP), R4	: SHIFT IT	
3784	024342	010204		MOV	R2, R4		
3785	024344	010502		MOV	R5, R2	: R2 DONE	
3786	024346	010305		MOV	R3, R5	: SET UP LOW ORDER	
3787	024350	073426		ASHC	(SP)+, R4	: DO LOW ORDER	
3788	024352	010503		MOV	R5, R3		
3789	024354	012604		MOV	(SP)+, R4	: RESTORE EXP TO R4	
3790	024356	000407		BR	7\$		
3791	024360	005726	13\$:	TST	(SP)+	: POP EXTENSION	
3792	024362	006200	10\$:	RSR	R0	: SHIFT RIGHT	
3793	024364	006001		ROR	R1		
3794	024366	006002		ROR	R2		
3795	024370	006003		ROR	R3		
3796	024372	005205		INC	R5	: COUNT LOOP	
3797	024374	002772		BLT	10\$		
3798	024376	065603	000024	7\$:	ADD	24(SP), R3	: FORM SUM
3799	024402	005502		RDC	R2		
3800	024404	005501		RDC	R1		
3801	024406	005500		RDC	R0		
3802	024410	066602	000022	ADD	22(SP), R2		
3803	024414	005501		RDC	R1		
3804	024416	005500		RDC	R0		
3805	024420	005501	000020	ADD	20(SP), R1		
3806	024424	005500		RDC	R0		
3807	024426	066600	000016	ADD	16(SP), R0		
3808	024432	126616	000001	CMPB	1(SP), (SP)	: CHECK FOR UNEQUAL SIGNS	
3809	024436	001162		BNE	14\$: CLEAN UP SUBTRACT	
3810	024440	030027	001000	BIT	R0, #1000		
3811	024444	001405		BEQ	9\$: JUMP IF NO NORMAL BIT OVERFLOW	
3812	024446	005200		RSR	R0		
3813	024450	006001		ROR	R1		
3814	024452	006002		ROR	R2		
3815	024454	006003		ROR	R3		
3816	024456	005204		INC	R4	: INCREASE EXP	
3817	024460	000304	9\$:	SWAB	R4	: MOVE EXP LEFT	
3818	024462	001425		BEQ	16\$: JUMP IF NO OVERFLOW	
3819	024464	105004		CLRB	R4	: CLEAR OVERFLOWED BITS	
3820	024466	052737	000002 002400	BIS	#02, \$FPS	: SET V BIT ON OVERFLOW	
3821	024474	005237	002540	INC	ADOC5	: CTR: OVERFLOW	
3822	024500	032737	001000 002400	BIT	#001000, \$FPS	: OVERFLOW ENABLED ?	
3823	024506	001464		BEQ	34\$: NO, ZERO RESULT	
3824	024510	005237	002542	INC	ADOC6	: CTR: OVERFLOW, ENABLED	
3825	024514	052737	100000 002400	BIS	#100000, \$FPS	: YES, SET ERROR BIT	
3826	024522	012737	000010 002402	MOV	#10, \$FEC	: SET \$FEC	
3827	024530	013737	002376 002404	MOV	EXP\$EA, \$FEA	: SET \$FEA	
3828	024536	150004	16\$:	BISB	R0, R4	: INSERT HIGH ORDER FRACTION	
3829	024540	006026		ROR	(SP)+	: INSERT SIGN	
3830	024542	006004		ROR	R4		
3831	024544	006001		ROR	R1		
3832	024546	006002		ROR	R2		
3833	024550	006003		ROR	R3		

3834	024552	005503				ADC	R3		
3835	024554	005502				ADC	R2		
3836	024556	005501				ADC	R1		
3837	024558	005504				ADC	R4		
3838	024560	103401				BCS	20\$		OVERFLOW ON ROUND ?
3839	024562	122024				BVC	30\$		
3840	024566	052737	000002	002400	20\$:	BIS	#02, \$FPS		YES - SET V BIT
3841	024574	005237	002540			INC	R00C5		CTR: OVERFLOW
3842	024576	032737	001000	002400		BIT	#001000, \$FPS		OVERFLOW ENABLED ?
3843	024578	001431				BEQ	29\$		NO, ZERO RESULT
3844	024580	005237	002542			INC	R00C6		CTR: OVERFLOW, ENABLED
3845	024582	052737	100000	002400		BIS	#100000, \$FPS		SET ERROR BIT
3846	024584	012737	000010	002402		MOV	#10, \$FEC		SET \$FEC
3847	024586	013737	002376	002404		MOV	EXPFEA, \$FEA		SET \$FEA
3848	024588	010466	000024		30\$:	MOV	R4, 24(SP)		STORE EXP AND SIGN
3849	024590	010166	000026			MOV	R1, 26(SP)		INSERT LOW ORDER FRACTION
3850	024592	010266	000030			MOV	R2, 30(SP)		
3851	024594	010366	000032			MOV	R3, 32(SP)		
3852	024596	000415				BR	3\$		
3853	024598	005726			34\$:	TST	(SP)+		FLUSH SIGN
3854	024600	000403				BR	29\$		AND ZERO RESULT
3855	024602	005237	002544		33\$:	INC	R00C7		CTR: UNDERFLOW
3856	024604	005726				TST	(SP)+		FLUSH SIGN
3857	024606	005066	000024		29\$:	CLR	24(SP)		ZERO RESULT
3858	024608	005066	000026			CLR	26(SP)		
3859	024702	005066	000030			CLR	30(SP)		
3860	024706	005066	000032			CLR	32(SP)		
3861	024712	032766	077600	000024	3\$:	BIT	#077600, 24(SP)		SET Z BIT IF EXPONENT ZERO
3862	024720	001003				BNE	17\$		
3863	024722	052737	000004	002400		BIS	#04, \$FPS		
3864	024730	005766	000024		17\$:	TST	24(SP)		SET N BIT IF RESULT NEGATIVE
3865	024734	100003				BPL	18\$		
3866	024736	052737	000010	002400		BIS	#10, \$FPS		
3867	024744	012605			18\$:	MOV	(SP)+, R5		
3868	024746	012604				MOV	(SP)+, R4		
3869	024750	012603				MOV	(SP)+, R3		RESTORE REGISTERS
3870	024752	012602				MOV	(SP)+, R2		
3871	024754	012601				MOV	(SP)+, R1		
3872	024756	012600				MOV	(SP)+, R0		
3873	024760	062706	000010			ADD	#8, SP		POP SECOND ARGUMENT
3874	024764	105737	002400			TSTB	\$FPS		FOR D MODE ?
3875	024770	100404				BMI	26\$		D MODE
3876	024772	012666	000002			MOV	(SP)+, 2(SP)		F MODE - CONVERT
3877	024776	012666	000002			MOV	(SP)+, 2(SP)		
3878	025002	000134			26\$:	JMP	@(R4)+		DONE
3879									
3880									
3881	025004	005700			14\$:	TST	R0		CHECK HIGH ORDER FRACTION RESULT
3882	025006	003014				BGT	22\$		IF + SIGN OK
3883	025010	001453				BEQ	23\$		CHECK FOR ZERO RESULT
3884	025012	005403				NEG	R3		ABS VALUE
3885	025014	005502				ADC	R2		
3886	025016	005501				ADC	R1		
3887	025020	005500				ADC	R0		
3888	025022	005402				NEG	R2		
3889	025024	005501				ADC	R1		

3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001

*FLOATING POINT SOFTWARE MULTIPLY ROUTINE

* THIS ROUTINE TAKES THE TOP TWO ELEMENTS ON THE STACK
* (LENGTH OF 2 OR 4 WORDS DEPENDING UPON WHETHER THE
* F/D MODE BIT7 IN SFPS IS 0 OR 1 RESPECTIVELY) AND
* REPLACES THEM WITH THEIR PRODUCT.

* EXAMPLE: SPUSH
* AOPR(OPERAND A)
* SPUSH
* BOPR(OPERAND B)
* SPUL
* SPOPX
* AOPR(RESULT)
* RESULT=A*B

* NOTE ROUTINE IS CALLED THROUGH POLISH MODE OPERATORS.

* ALL OPERATIONS ARE CARRIED OUT IN DOUBLE (LENGTH=4)
* PRECISION MODE. SINGLE PRECISION OPERANDS ARE
* EXTENDED (VIA SCONV SUBROUTINE) WITH LOW-ORDER
* ZEROS, AND LATER TRUNCATED BACK TO SINGLE PRECISION
* FROM THE DOUBLE PRECISION RESULT.

* STATUS BITS:

* THE N, Z, V, AND C BITS OF SFPS ARE SET AS FOLLOWS:
* N = 1 IF RESULT NEGATIVE (IE, BIT15 = 1),
* ELSE N = 0
* Z = 1 IF RESULT ZERO (IE, EXPONENT = 000(8)),
* ELSE Z = 0
* V = 1 IF ARITHMETIC OVERFLOW OCCURRED,
* ELSE V = 0
* C = 0 ALWAYS

* ERROR CONDITIONS:

* IF AN ARITHMETIC OVERFLOW CONDITION OCCURS, THE RESULT
* WILL BE SET TO ZERO IF THE OVERFLOW ENABLE BIT OF SFPS
* (BIT09) IS ZERO. IF THIS BIT IS SET, THEN THE RESULT
* WILL BE CORRECT, EXCEPT THE EXPONENT WILL BE UNDERBIASED
* BY 400(8). ALSO, THE SFPS ERROR BIT (BIT15) WILL BE SET,
* SFEC WILL BE SET TO 10(8), AND SFEA WILL BE LOADED WITH
* THE VALUE IN LOCATION "EXOPFEA". IN EITHER INSTANCE,
* THE V-BIT (BIT01) WILL BE SET.
* IF AN ARITHMETIC UNDERFLOW CONDITION OCCURS, THE RESULT
* WILL BE SET TO ZERO IF THE UNDERFLOW ENABLE BIT OF SFPS
* (BIT10) IS ZERO. IF THIS BIT IS SET, THEN THE RESULT
* WILL BE CORRECT, EXCEPT THE EXPONENT WILL BE OVERBIASED
* BY 400(8). ALSO, THE SFPS ERROR BIT (BIT15) WILL BE SET,
* SFEC WILL BE SET TO 12(8), AND SFEA WILL BE LOADED WITH
* THE VALUE IN LOCATION "EXOPFEA".

* ALL REGISTERS ARE PRESERVED UPON ENTRY, AND RESTORED UPON
* EXIT. THE ROUTINES ARE RE-ENTRANT.

* ADAPTED FROM PDP-11 FORTRAN SOFTWARE
* BY DONALD NORTH, SEPTEMBER, 1975.

4058	025532	001003				BNE	78		
4059	025534	005766	000040			TST	40(SP)		
4060	025540	001402				BEQ	88		
4061	025542	004737	026222		78:	JSR	PC,328		
4062	025546	016604	000032		88:	MOV	32(SP),R4		GET HIGH ORDER BITS
4063	025548	012716	000007			MOV	87(SP)		SEVEN OF THEM
4064	025556	004737	026222			JSR	PC,328		
4065	025562	004737	026226			JSR	PC,338		NORMAL BIT
4066	025566	005726				TST	(SP)+		FLUSH ITERATION COUNT
4067	025570	062604				ADD	(SP)+,R4		ADD EXP
4068	025572	006303				ASL	R3		SHIFT OUT NORMAL BIT
4069	025574	006102				ROL	R2		
4070	025576	006101				ROL	R1		
4071	025580	006100				ROL	R0		
4072	025602	103405				BCS	98		NORMAL BIT FOUND
4073	025604	006303				ASL	R3		
4074	025606	006102				ROL	R2		
4075	025610	006101				ROL	R1		
4076	025612	006100				ROL	R0		HAVE IT
4077	025614	005304				DEC	R4		ADJUST EXP
4078	025616	162704	000200		98:	SUB	#200,R4		REMOVE BIAS FROM EXP
4079	025622	003022				BGT	108		BR IF NO UNDERFLOW
4080	025624	005237	002560			INC	MULC4		CTR: UNDERFLOW
4081	025630	032737	002000	002400		BIT	#002000,\$FPS		UNDERFLOW - IS IT ENABLED ?
4082	025636	001517				BEQ	128		NO, MAKE ZERO RESULT
4083	025640	005237	002562			INC	MULC5		CTR: UNDERFLOW, ENABLED
4084	025644	052737	100000	002400		BIS	#100000,\$FPS		SET ERROR BIT
4085	025652	012737	000012	002402		MOV	#12,\$FEC		SET \$FEC
4086	025660	013737	002376	002404		MOV	EXPFEA,\$FEA		SET \$FEA
4087	025666	000427				BR	118		CONTINUE
4088	025670	022704	000377		108:	CMR	#377,R4		CHECK FOR OVERFLOW
4089	025674	002024				BGE	118		BR IF NO OVERFLOW
4090	025676	052737	000002	002400		BIS	#02,\$FPS		SET V BIT ON OVERFLOW
4091	025704	005237	002554			INC	MULC2		CTR: OVERFLOW
4092	025710	032737	001000	002400		BIT	#001000,\$FPS		OVERFLOW ENABLED ?
4093	025716	001467				BEQ	128		NO, ZERO RESULT
4094	025720	005237	002556			INC	MULC3		CTR: OVERFLOW, ENABLED
4095	025724	052737	100000	002400		BIS	#100000,\$FPS		SET ERROR BIT
4096	025732	012737	000010	002402		MOV	#10,\$FEC		SET \$FEC
4097	025740	013737	002376	002404		MOV	EXPFEA,\$FEA		SET \$FEA
4098	025746	105003			118:	CLRB	R3		
4099	025750	150203				BISB	R2,R3		SHIFT FRACTION RIGHT
4100	025752	000303				SWAB	R3		
4101	025754	105002				CLRB	R2		
4102	025756	150102				BISB	R1,R2		
4103	025760	000302				SWAB	R2		
4104	025762	105001				CLRB	R1		
4105	025764	150001				BISB	R0,R1		
4106	025766	000301				SWAB	R1		
4107	025770	105000				CLRB	R0		
4108	025772	150400				BISB	R4,R0		
4109	025774	000300				SWAB	R0		
4110	025776	006026				ROR	(SP)+		GET PRODUCT SIGN
4111	026000	006000				ROR	R0		PUT IN RESULT
4112	026002	006001				ROR	R1		
4113	026004	006002				ROR	R2		

4114	026006	005003				ROR	R3		
4115	026010	005503				ADC	R3		
4116	026012	005502				ADC	R2		ROUND RESULT
4117	026014	005501				ADC	R1		
4118	026016	005500				ADC	R0		
4119	026020	103401				BCS	16S		OVERFLOW ON ROUND ?
4120	026022	102032				BVC	13S		
4121	026024	052737	000002	002400	16S:	BIS	#02,\$FPS		YES - SET V BIT
4122	026032	005237	002554			INC	MULC2		CTR: OVERFLOW
4123	026036	032737	001000	002400		BIT	#001000,\$FPS		OVERFLOW ENABLED ?
4124	026044	001415				BEQ	3S		NO, ZERO RESULT
4125	026046	005237	002556			INC	MULC3		CTR: OVERFLOW, ENABLED
4126	026052	052737	100000	002400		BIS	#100000,\$FPS		SET ERROR BIT
4127	026060	012737	000010	002402		MOV	#10,\$FEC		SET \$FEC
4128	026066	013737	002376	002404		MOV	EXP\$EA,\$FEA		SET \$FEA
4129	026074	000405				BR	13S		CONTINUE
4130	026076	005726			12S:	TST	(SP)+		FLUSH SIGN
4131	026100	005000			3S:	CLR	R0		CLEAR RESULT
4132	026102	005001				CLR	R1		
4133	026104	005002				CLR	R2		
4134	026106	005003				CLR	R3		
4135	026110	010066	000024		13S:	MOV	R0,24(SP)		
4136	026114	010166	000026			MOV	R1,26(SP)		STUFF RESULT
4137	026120	010266	000030			MOV	R2,30(SP)		
4138	026124	010366	000032			MOV	R3,32(SP)		
4139	026130	032766	077600	000024		BIT	#077600,24(SP)		SET Z BIT IF EXPONENT ZERO
4140	026136	001003				BNE	17S		
4141	026140	052737	000004	002400		BIS	#04,\$FPS		
4142	026146	005766	000024		17S:	TST	24(SP)		SET N BIT IF RESULT NEGATIVE
4143	026152	100003				BPL	18S		
4144	026154	052737	000010	002400		BIS	#10,\$FPS		
4145	026162	012605			18S:	MOV	(SP)+,R5		
4146	026164	012604				MOV	(SP)+,R4		
4147	026166	012603				MOV	(SP)+,R3		RESTORE REGISTERS
4148	026170	012602				MOV	(SP)+,R2		
4149	026172	012601				MOV	(SP)+,R1		
4150	026174	012600				MOV	(SP)+,R0		
4151	026176	062706	000010			ADD	#8,\$SP		CLEAR SECOND OPERAND OFF STACK
4152	026202	105737	002400			TSTB	\$FPS		F OR D MODE
4153	026206	100404				BMI	14S		D-MODE
4154	026210	012666	000002			MOV	(SP)+,2(SP)		F MODE - CONVERT
4155	026214	012666	000002			MOV	(SP)+,2(SP)		
4156	026220	000134			14S:	JMP	@(R4)+		RETURN
4157									
4158	026222	006204			32S:	ASR	R4		TEST NEXT MULTIPLIER BIT
4159	026224	103022				BCC	34S		JUMP IF ZERO
4160	026226	066603	000032		33S:	ADD	32(SP),R3		ADD IN MULTIPLICAND
4161	026232	005502				ADC	R2		
4162	026234	005501				ADC	R1		
4163	026236	005500				ADC	R0		
4164	026240	005505				ADC	R5		SAVE OVERFLOW
4165	026242	066602	000030			ADD	30(SP),R2		
4166	026246	005501				ADC	R1		
4167	026250	005500				ADC	R0		
4168	026252	005505				ADC	R5		
4169	026254	066601	000026			ADD	26(SP),R1		

```

4170 026260 005500      ROC      R0
4171 026262 005505      ROC      R0
4172 026264 066600 000024  ROC      24(SP),R0
4173 026270 005505      ROC      R5
4174 026272 006205      34$:  ASR      R5      RECOVER OVERFLOW IF ANY
4175 026274 006000      ROR      R0      SHIFT PRODUCT
4176 026276 006001      ROR      R1
4177 026300 006002      ROR      R2
4178 026302 006003      ROR      R3
4179 026304 005366 000002  DEC      2(SP)    COUNT LOOP
4180 026310 003344      BGT      32$     AGAIN
4181 026312 000207      RTS      PC     RETURN
4182 026314 005366 000002  31$:  DEC      2(SP)    ONLY 15 BITS THIS PASS
4183 026320 006204 30$:  ASR      R4     TEST NEXT MULTIPLIER BIT
4184 026322 103007      BCC      35$     JUMP IF ZERO
4185 026324 066501 000026  ADD      26(SP),R1  USE ONLY HIGH ORDER MULTIPLICAND
4186 026330 006500      ROC      R0
4187 026332 005505      ROC      R5
4188 026334 066600 000024  ADD      24(SP),R0
4189 026340 005505      ROC      R5
4190 026342 006205      35$:  ASR      R5     RECOVER ANY OVERFLOW
4191 026344 006000      ROR      R0
4192 026346 006001      ROR      R1
4193 026350 006002      ROR      R2
4194 026352 006003      ROR      R3
4195 026354 005366 000002  DEC      2(SP)    COUNT LOOP
4196 026360 003357      BGT      30$
4197 026362 000207      RTS      PC     RETURN
4198      ;*END OF MUL

```

4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254

*FLOATING POINT SOFTWARE DIVIDE ROUTINE

* THIS ROUTINE TAKES THE TOP TWO ELEMENTS ON THE STACK
* (LENGTH OF 2 OR 4 WORDS DEPENDING UPON WHETHER THE
* F/D MODE BIT7 IN SFPS IS 0 OR 1 RESPECTIVELY) AND
* REPLACES THEM WITH THEIR QUOTIENT.

* EXAMPLE: SPUSH
* ADDR(OPERAND A)
* SPUSH
* ADDR(OPERAND B)
* SDIV
* SPOPX
* ADDR(RESULT)
* RESULT=A/B

* NOTE ROUTINE IS CALLED THROUGH POLISH MODE OPERATORS.

* ALL OPERATIONS ARE CARRIED OUT IN DOUBLE (LENGTH=4)
* PRECISION MODE. SINGLE PRECISION OPERANDS ARE
* EXTENDED (VIA \$CONV SUBROUTINE) WITH LOW-ORDER
* ZEROS, AND LATER TRUNCATED BACK TO SINGLE PRECISION
* FROM THE DOUBLE PRECISION RESULT.

* STATUS BITS:

* THE N, Z, V, AND C BITS OF SFPS ARE SET AS FOLLOWS:
* N = 1 IF RESULT NEGATIVE (IE, BIT15 = 1),
* ELSE N = 0
* Z = 1 IF RESULT ZERO (IE, EXPONENT = 000(8)),
* ELSE Z = 0
* V = 1 IF ARITHMETIC OVERFLOW OCCURRED,
* ELSE V = 0
* C = 0 ALWAYS

* ERROR CONDITIONS:

* IF AN ARITHMETIC OVERFLOW CONDITION OCCURS, THE RESULT
* WILL BE SET TO ZERO IF THE OVERFLOW ENABLE BIT OF SFPS
* (BIT09) IS ZERO. IF THIS BIT IS SET, THEN THE RESULT
* WILL BE CORRECT, EXCEPT THE EXPONENT WILL BE UNDERBIASED
* BY 400(8). ALSO, THE SFPS ERROR BIT (BIT15) WILL BE SET,
* \$FEC WILL BE SET TO 10(8), AND \$FEA WILL BE LOADED WITH
* THE VALUE IN LOCATION "EXPFEA". IN EITHER INSTANCE,
* THE V-BIT (BIT01) WILL BE SET.
* IF AN ARITHMETIC UNDERFLOW CONDITION OCCURS, THE RESULT
* WILL BE SET TO ZERO IF THE UNDERFLOW ENABLE BIT OF SFPS
* (BIT10) IS ZERO. IF THIS BIT IS SET, THEN THE RESULT
* WILL BE CORRECT, EXCEPT THE EXPONENT WILL BE OVERBIASED
* BY 400(8). ALSO, THE SFPS ERROR BIT (BIT15) WILL BE SET,
* \$FEC WILL BE SET TO 12(8), AND \$FEA WILL BE LOADED WITH
* THE VALUE IN LOCATION "EXPFEA".
* IF DIVISION BY ZERO IS ATTEMPTED (EG, EXPONENT OF
* DENOMINATOR OPERAND IS ZERO), THE RESULT LEFT ON THE
* STACK WILL BE THE NUMERATOR, WITH THE CONDITION CODES SET
* ACCORDINGLY. THE SFPS ERROR BIT WILL BE SET, \$FEC WILL BE
* SET TO 4(8), AND \$FEA WILL BE SET TO THE VALUE CONTAINED
* IN THE LOCATION "EXPFEA".

M08

4311	026616	000366	000020		SWAB	20(SP)	: LEFT JUSTIFY DENOM
4312	026622	000261			SEC		: INSERT NORMAL BIT
4313	026624	006066	000020		ROR	20(SP)	
4314	026630	116666	000023	000020	MOVB	23(SP), 20(SP)	
4315	026636	116666	000022	000023	MOVB	22(SP), 23(SP)	
4316	026644	116666	000025	000022	MOVB	25(SP), 22(SP)	
4317	026652	116666	000024	000025	MOVB	24(SP), 25(SP)	
4318	026660	116666	000027	000024	MOVB	27(SP), 24(SP)	
4319	026666	116666	000026	000027	MOVB	26(SP), 27(SP)	
4320	026674	105066	000026		CLRB	26(SP)	
4321	026700	005066	000030		CLR	30(SP)	: CLEAR QUOTIENT
4322	026704	005066	000032		CLR	32(SP)	
4323	026710	005066	000034		CLR	34(SP)	
4324	026714	020066	000020		CMP	R0, 20(SP)	: COMPARE HIGH NUM + DENOM
4325	026720	101020			BHI	3\$: JUMP IF DENOM LOW
4326	026722	103424			BLO	4\$: JUMP IF DENOM HI
4327	026724	020166	000022		CMP	R1, 22(SP)	: COMPARE LOW ORDER PARTS
4328	026730	101014			BHI	3\$	
4329	026732	103420			BLO	4\$	
4330	026734	020266	000024		CMP	R2, 24(SP)	
4331	026740	101010			BHI	3\$	
4332	026742	103414			BLO	4\$	
4333	026744	020366	000026		CMP	R3, 26(SP)	
4334	026750	101004			BHI	3\$	
4335	026752	001010			BNE	4\$	
4336	026754	005216			INC	(SP)	: BUMP EXP
4337	026756	005004			CLR	R4	
4338	026760	000443			BR	5\$	
4339							
4340	026762	006000		3\$:	ROR	R0	: HALVE DENOM (C=0)
4341	026764	006001			ROR	R1	: TO INSURE N<D
4342	026766	006002			ROR	R2	
4343	026770	006003			ROR	R3	
4344	026772	005216			INC	(SP)	: COMPENSATE EXP
4345	026774	012705	000011	4\$:	MOV	#9, R5	: FIRST NINE QUOTIENT BITS
4346	027000	004737	027464		JSR	PC, 30\$	
4347	027004	110466	000030		MOVB	R4, 30(SP)	: SAVE ALL HIGH ORDER Q FRACTION
4348	027010	005705			TST	R5	: DONE?
4349	027012	001025			BNE	10\$: YES - REST OF NUMBER IS 0
4350	027014	012705	000020		MOV	#16, R5	: 16 MORE BITS
4351	027020	004737	027464		JSR	PC, 30\$	
4352	027024	010466	000032		MOV	R4, 32(SP)	
4353	027030	005705			TST	R5	
4354	027032	001015			BNE	10\$	
4355	027034	012705	000020		MOV	#16, R5	
4356	027040	004737	027464		JSR	PC, 30\$	
4357	027044	010466	000034		MOV	R4, 34(SP)	
4358	027050	005705			TST	R5	
4359	027052	001005			BNE	10\$	
4360	027054	012705	000020		MOV	#16, R5	
4361	027060	004737	027464		JSR	PC, 30\$	
4362	027064	000401			BR	5\$	
4363	027066	005004		10\$:	CLR	R4	: CLEAR LOWEST ORDER QUOTIENT
4364	027070	012605		5\$:	MOV	(SP)+, R5	: PUSH UP EXPONENT
4365	027072	062705	000200		ADD	#200, R5	: INSERT BIAS
4366	027076	003022			BGT	7\$: BR IF NO UNDERFLOW

N08

4367	027100	005237	002576			INC	DIVC5	:CTR: UNDERFLOW
4368	027104	032737	002000	002400		BIT	#002000,\$FPS	:UNDERFLOW - IS IT ENABLED ?
4369	027112	001516				BEQ	15\$:NO, MAKE ZERO RESULT
4370	027114	005237	002600			INC	DIVC6	:CTR: UNDERFLOW, ENABLED
4371	027120	052737	100000	002400		BIS	#100000,\$FPS	:SET ERROR BIT
4372	027126	012737	000012	002402		MOV	#12,\$FEC	:SET \$FEC
4373	027134	013737	002376	002404		MOV	EXPFEA,\$FEA	:SET \$FEA
4374	027142	000427				BR	11\$:CONTINUE
4375	027144	022705	000377		7\$:	CMP	#377,\$R5	:CHECK FOR OVERFLOW
4376	027150	002024				BGE	11\$:BB IF NO OVERFLOW
4377	027152	052737	000002	002400		BIS	#02,\$FPS	:SET V BIT ON OVERFLOW
4378	027160	005237	002572			INC	DIVC3	:CTR: OVERFLOW
4379	027164	032737	001000	002400		BIT	#001000,\$FPS	:OVERFLOW ENABLED ?
4380	027172	001466				BEQ	15\$:NO, ZERO RESULT
4381	027174	005237	002574			INC	DIVC4	:CTR: OVERFLOW, ENABLED
4382	027200	052737	100000	002400		BIS	#100000,\$FPS	:SET ERROR BIT
4383	027206	012737	000010	002402		MOV	#10,\$FEC	:SET \$FEC
4384	027214	013737	002376	002404		MOV	EXPFEA,\$FEA	:SET \$FEA
4385	027222	110566	000027		11\$:	MOVB	\$R5,27(\$P)	:PUT EXPIN RESULT
4386	027236	006026				ROR	(\$P)+	:INSERT SIGN
4387	027230	006066	000024			ROR	24(\$P)	
4388	027234	006066	000026			ROR	26(\$P)	
4389	027240	006066	000030			ROR	30(\$P)	
4390	027244	006004				ROR	\$R4	
4391	027246	005504				ADC	\$R4	:ROUND
4392	027250	005566	000030			ADC	30(\$P)	
4393	027254	005566	000026			ADC	26(\$P)	
4394	027260	005566	000024			ADC	24(\$P)	
4395	027264	010466	000032			MOV	\$R4,32(\$P)	:INSERT LOW ORDER FRACTION
4396	027270	103401				BCC	16\$:OVERFLOW ON ROUND ?
4397	027272	102037				BVC	9\$	
4398	027274	052737	000002	002400	16\$:	BIS	#02,\$FPS	:YES - SET V BIT
4399	027302	005237	002572			INC	DIVC3	:CTR: OVERFLOW
4400	027306	032737	001000	002400		BIT	#001000,\$FPS	:OVERFLOW ENABLED ?
4401	027314	001416				BEQ	19\$:NO, ZERO RESULT
4402	027316	005237	002574			INC	DIVC4	:CTR: OVERFLOW, ENABLED
4403	027322	052737	100000	002400		BIS	#100000,\$FPS	:SET ERROR BIT
4404	027330	012737	000010	002402		MOV	#10,\$FEC	:SET \$FEC
4405	027336	013737	002376	002404		MOV	EXPFEA,\$FEA	:SET \$FEA
4406	027344	000412				BR	9\$:CONTINUE
4407	027346	005726			1\$:	TST	(\$P)+	:FLUSH EXP
4408	027350	005726			15\$:	TST	(\$P)+	:FLUSH SIGN
4409	027352	005066	000024		19\$:	CLR	24(\$P)	:CLEAR RESULT
4410	027356	005066	000026			CLR	26(\$P)	
4411	027362	005066	000030			CLR	30(\$P)	
4412	027366	005066	000032			CLR	32(\$P)	
4413	027372	032766	077600	000024	9\$:	BIT	#077600,24(\$P)	:SET Z BIT IF EXPONENT ZERO
4414	027400	001003				BNE	17\$	
4415	027402	052737	000004	002400		BIS	#04,\$FPS	
4416	027410	005766	000024		17\$:	TST	24(\$P)	:SET N BIT IF RESULT NEGATIVE
4417	027414	100003				BPL	18\$	
4418	027416	052737	000010	002400		BIS	#10,\$FPS	
4419	027424	012605			18\$:	MOV	(\$P)+,\$R5	
4420	027426	012604				MOV	(\$P)+,\$R4	
4421	027430	012603				MOV	(\$P)+,\$R3	
4422	027432	012602				MOV	(\$P)+,\$R2	:RESTORE REGISTERS

F423	027434	012601		MOV	(SP)+,R1	
F424	027436	012600		MOV	(SP)+,R0	
F425	027440	062706	000010	ADD	#8,SP	: FLUSH FIRST ARG
F426	027444	105737	002400	TSTB	\$FPS	: F OR D MODE
F427	027450	100404		BMI	13\$: D MODE
F428	027452	012666	000002	MOV	(SP)+,2(SP)	: F MODE - CONVERT
F429	027456	012666	000002	MOV	(SP)+,2(SP)	
F430	027462	000134		JMP	2(R4)+	: RETURN
F431	027464	006304		ASL	R4	: SHIFT QUOTIENT
F432	027466	006303		ASL	R3	
F433	027470	006102		ROL	R2	
F434	027472	006101		ROL	R1	
F435	027474	006100		ROL	R0	
F436	027478	103420		BCS	31\$: GUARANTEED TO GO
F437	027500	026600	000022	CMP	22(SP),R0	: COMPARE HIGH DIVISOR AND DIVIDEND
F438	027504	101034		BHI	32\$: DIVISOR BIGGER
F439	027506	103414		BLO	31\$	
F440	027510	026601	000024	CMP	24(SP),R1	: CHECK LOW ORDERS
F441	027514	101030		BHI	32\$	
F442	027516	103410		BLO	31\$	
F443	027520	026602	000026	CMP	26(SP),R2	
F444	027524	101024		BHI	32\$	
F445	027526	103404		BLO	31\$	
F446	027530	026603	000030	CMP	30(SP),R3	
F447	027534	101020		BHI	32\$	
F448	027536	001422		BEQ	33\$: NUMER=DENOM
F449	027540	166603	000030	SUB	30(SP),R3	: N=N-D
F450	027544	005602		SBC	R2	
F451	027546	005601		SBC	R1	
F452	027550	005600		SBC	R0	
F453	027552	166602	000026	SUB	26(SP),R2	
F454	027556	005601		SBC	R1	
F455	027560	005600		SBC	R0	
F456	027562	166601	000024	SUB	24(SP),R1	
F457	027566	005600		SBC	R0	
F458	027570	166600	000022	SUB	22(SP),R0	
F459	027574	005204		INC	R4	: INSERT QUOTIENT BIT
F460	027576	005305		DEC	R5	: COUNT LOOP
F461	027600	003331		BGT	30\$	
F462	027602	000207		RTS	PC	: RETURN
F463	027604	005204		INC	R4	: INSERT LAST 1 BIT IN QUOTIENT
F464	027606	000401		BR	34\$	
F465	027610	006304		ASL	R4	: FINISH OUT QUOTIENT WITH ZEROS
F466	027612	005305		DEC	R5	
F467	027614	003375		BGT	35\$	
F468	027616	005205		INC	R5	: FLAG NO MORE NUMER
F469	027620	000207		RTS	PC	: RETURN
F470						
F471						

; *END OF DIV

4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492

027622 005046
027624 016646 000006
027630 016646 000006
027634 016646 000006
027640 016666 000020 000014
027646 016666 000016 000012
027654 005066 000020
027660 005066 000016
027664 005066 000010
027670 000207

\$CONV:

```

*CONVERT TOP 2 OPERANDS ON STACK FROM F MODE
* TO D MODE
*
* THIS ROUTINE TAKES THE TOP TWO 2-WORD (SINGLE
* PRECISION) FLOATING POINT NUMBERS ON THE
* STACK AND CONVERTS THEM BOTH TO 4-WORD
* (DOUBLE PRECISION) FORMAT BY APPENDING
* TWO WORDS OF ZEROS AS THE LOW ORDER BIT
* EXTENSION.
*
CLR -(SP) ; CLEAR WORD 3 OF A
MOV 6(SP), -(SP) ; MOVE WORD 2 OF A
MOV 6(SP), -(SP) ; MOVE WORD 1 OF A
MOV 6(SP), -(SP) ; MOVE RETURN ADDR TO TOP
MOV 20(SP), 14(SP) ; MOVE WORD 2 OF B
MOV 16(SP), 12(SP) ; MOVE WORD 1 OF B
CLR 20(SP) ; CLEAR WORD 4 OF B
CLR 16(SP) ; CLEAR WORD 3 OF B
CLR 10(SP) ; CLEAR WORD 4 OF A
RTS PC

```

4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548

.SBTTL SCOPE HANDLER ROUTINE

*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER(\$STNM) INTO THE DISPLAY REG.(DISPLAY<15:0>)
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*SW11=1 INHIBIT ITERATIONS
*SW09=1 LOOP ON ERROR
*SW08=1 LOOP ON TEST IN "SLPTST"
*CALL SCOPE ;;SCOPE=IOT

\$SCOPE:
64\$:
1\$: BIT #BIT14,\$SWR ;;LOOP ON PRESENT TEST?
BNE \$OVER ;;YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
\$XTSTR: BR 6\$
;;IF RUNNING ON THE "XOR" TESTER CHANGE
;;THIS INSTRUCTION TO A "NOP" (NOP=240)
;;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV @ERRVEC,-(SP) ;;SET FOR TIMEOUT
MOV \$S,\$ERRVEC ;;TIME OUT ON XOR?
TST @177060
MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
BR \$SVLAD ;;GO TO THE NEXT TEST
5\$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
BR 7\$;;LOOP ON THE PRESENT TEST
6\$:;*****END OF CODE FOR THE XOR TESTER*****
BIT #BIT08,\$SWR ;;LOOP ON SPEC. TEST?
BEQ 2\$;;BR IF NO
CMP \$LPTST,\$STNM ;;ON THE RIGHT TEST?
BEQ \$OVER ;;BR IF YES
2\$: TST \$ERFLG ;;HAS AN ERROR OCCURRED?
BEQ 3\$;;BR IF NO
CMP \$ERMAX,\$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 3\$;;BR IF NO
BIT #BIT09,\$SWR ;;LOOP ON ERROR?
BEQ 4\$;;BR IF NO
7\$: MOV \$LPERR,\$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
BR \$OVER
4\$: CLR \$ERFLG ;;ZERO THE ERROR FLAG
CLR \$TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 1\$;;ESCAPE TO THE NEXT TEST
3\$: BIT #BIT11,\$SWR ;;INHIBIT ITERATIONS?
BNE 1\$;;BR IF YES
TST \$PASS ;;IF FIRST PASS OF PROGRAM
BEQ 1\$;;INHIBIT ITERATIONS
INC \$ICNT ;;INCREMENT ITERATION COUNT
CMP \$TIMES,\$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
BGE \$OVER ;;BR IF MORE ITERATION REQUIRED
1\$: MOV \$I,\$ICNT ;;REINITIALIZE THE ITERATION COUNTER
MOV \$MXCNT,\$TIMES ;;SET NUMBER OF ITERATIONS TO DO
\$SVLAD: INC \$STNM ;;COUNT TEST NUMBERS
MOV \$STNM,\$TESTN ;;SET TEST NUMBER IN APT MAILBOX
MOV (SP),\$LPADR ;;SAVE SCOPE LOOP ADDRESS

027672
027672
027672 032777 040000 151244
027700 001114
027702 000416
027704 013746 000004
027710 012737 027730 000004
027716 005737 177060
027722 012637 000004
027726 000463
027730 022626
027732 012637 000004
027736 000423
027740
027740 032777 000400 151176
027746 001404
027750 023737 001150 001102
027756 001465
027760 005737 001104
027764 001421
027766 023737 001120 001104
027774 101015
027776 032777 001000 151140
030004 001404
030006 013737 001112 001110
030014 000446
030016 005037 001104
030022 005037 001166
030026 000415
030030 032777 004000 151106
030036 001011
030040 005737 001210
030044 001406
030046 005237 001106
030052 023737 001166 001106
030060 002024
030062 012737 000001 001106
030070 013737 030146 001166
030076 005237 001102
030102 013737 001102 001206
030110 011637 001110

4549	030114	011637	001112		MOV	(SP), SLPERR	:: SAVE ERROR LOOP ADDRESS
4550	030120	005037	001170		CLR	SESCAPE	:: CLEAR THE ESCAPE FROM ERROR ADDRESS
4551	030124	012737	000001	001120	MOV	#1, SERMAX	:: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4552	030132	013777	001102	151006	SOVER: MOV	\$STNM, \$DISPLAY	:: DISPLAY TEST NUMBER
4553	030140	013716	001110		MOV	SLPADR, (SP)	:: FUDGE RETURN ADDRESS
4554	030144	000002			RTI		:: FIXES PS
4555	030146	000400			SMXCNT: 400		:: MAX. NUMBER OF ITERATIONS

.SBTTL ERROR HANDLER ROUTINE

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO STYPERA ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1 HALT ON ERROR
*SW13=1 INHIBIT ERROR TYPEOUTS
*SW10=1 BELL ON ERROR
*SW09=1 LOOP ON ERROR
*CALL
* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

```

SEERROR:

```

MOV R0, EREG0      ; DISPLAY R0
MOV R1, EREG1      ; R1
MOV R2, EREG2      ; R2
MOV R3, EREG3      ; R3
MOV R4, EREG4      ; R4
MOV R5, EREG5      ; R5
MOV R6, EREG6      ; GET R6(SP) BEFORE TRAP
ADD R4, EREG6
MOV (SP), EREG7    ; PC -> ERROR CALL INSTR
INC SERFLG         ; SET THE ERROR FLAG
BEQ 78             ; DON'T LET THE FLAG GO TO ZERO
MOV $STMM, @DISPLAY ; DISPLAY TEST NUMBER
BIT @BIT10, @SWR   ; BELL ON ERROR?
BEQ 18             ; NO - SKIP
TYPE $BELL         ; RING BELL
INC $ERTL          ; COUNT THE NUMBER OF ERRORS
MOV (SP), $ERRPC   ; GET ADDRESS OF ERROR INSTRUCTION
SUB R2, $PC        ; STRIP AND SAVE THE ERROR ITEM CODE
MOV @PC, @ITEMB    ; SKIP TYPEOUT IF SET
BIT @BIT13, @SWR   ; SKIP TYPEOUTS
BNE 208           ; GO TO USER ERROR ROUTINE
JSR PC, $TYPERA
TYPE $CALF

208: CMPB @PTENV, $ENV ; RUNNING IN APT MODE
BNE 28            ; NO SKIP APT ERROR REPORT
MOV @ITEMB, @218  ; SET ITEM NUMBER AS ERROR NUMBER
JSR PC, @ATY4    ; REPORT FATAL ERROR TO APT

218: .BYTE 0
.BYTE 0

228: BR 228       ; APT ERROR LOOP
28: TST @SWR
BPL 38
HALT

38: BIT @BIT09, @SWR ; LOOP ON ERROR SWITCH SET?
BEQ 48
MOV @STYPERA, (SP) ; FUDGE RETURN FOR LOOPING
TST $ESCAPE        ; CHECK FOR AN ESCAPE ADDRESS
BEQ 58
MOV $ESCAPE, (SP) ; FUDGE RETURN ADDRESS FOR ESCAPE

58:

```

```

4560 030150 010037 002602
4561 030150 010137 002604
4562 030154 010237 002606
4563 030160 010337 002610
4564 030164 010437 002612
4565 030170 010537 002614
4566 030174 010637 002616
4567 030200 010637 002616
4568 030204 010637 002616
4569 030204 062737 000004 002616
4570 030212 011637 002620
4571 030216 005237 001104
4572 030220 001775
4573 030224 013777 001102 150714
4574 030228 022777 002000 150704
4575 030232 001402
4576 030236 104401 001172
4577 030240 005237 001114
4578 030244 011637 001122
4579 030248 162737 000002 001122
4580 030252 117737 150632 001116
4581 030256 032777 020000 150644
4582 030260 001004
4583 030264 004737 030412
4584 030268 104401 001177
4585 030312 122737 000001 001222
4586 030320 001707
4587 030324 113737 001116 030334
4588 030330 004737 031152
4589 030334 000
4590 030338 000
4591 030342 000777
4592 030346 005777 150600
4593 030350 100001
4594 030354 000000
4595 030358 032777 001000 150566
4596 030362 001402
4597 030366 013716 001112
4598 030370 005737 001170
4599 030374 001402
4600 030378 013716 001170
4601 030382

```

G09

FPL ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 95
D8FP08.P11 04-MAY-77 17:30 ERROR HANDLER ROUTINE

SEQ 0096

4612	030376	022737	022106	000042	CMP	SENDAD,2042	::ACT-11 AUTO-ACCEPT?
4613	030404	001001			BNE	68	::BRANCH IF NO
4614	030406	000000			HALT		::YES
4615	030410			68:			
4616	030410	000002		648:	RTI		:RETURN


```

4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641 030412
4642 030412 104401
4643 030414 001177
4644 030416 010046
4645 030420 010146
4646 030422 005000
4647 030424 153700 001116
4648 030430 001004
4649
4650 030432 013746 001122
4651 030436 104402
4652 030440 000454
4653 030442 005300
4654 030444 006300
4655 030446 006300
4656 030450 010001
4657 030452 006300
4658 030454 006300
4659 030456 060100
4660 030460 062700 001232
4661 030464 012037 030474
4662 030470 001404
4663 030472 104401
4664 030474 000000
4665 030476 104401 001177
4666 030502 104401 030632
4667 030506 012037 030516
4668 030512 001402
4669 030514 104401
4670 030516 000000
4671 030520 104401 001177
4672 030524 017746 000074

```

;;*****

.SBTTL ERROR MESSAGE TIMEOUT ROUTINE (MODIFIED SYSPAC)

```

*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE",
*(SERRTB) THE ERROR MESSAGE, DATA HEADER, AND DATA VALUES TO PRINT.

```

```

*   FORMAT:      W1:   PTR TO ASCIZ ERROR MESSAGE, 0 IF NONE
*                W2:   PTR TO ASCIZ DATA HEADER, 0 IF NONE
*                W3:   PTR TO DATA VALUES ADDR LIST, 0 IF NONE
*                W4-W9: PTR TO OPERAND VALUES ADDR LIST, 0 IF NONE
*                W10:  ALWAYS 0

```

```

DATA VALUES LIST FORMAT:
A VARIABLE LENGTH LIST OF POINTERS TO
WORDS TO PRINT AS 6 OCTAL DIGITS. LIST
MUST BE TERMINATED BY A ZERO WORD.

```

```

OPERAND VALUES LIST FORMAT:
FIRST WORD IS ADDRESS OF ASCIZ MESSAGE TO
PRINT AT START OF LINE; REST OF LIST IS
IN SAME FORMAT AS DATA VALUES LAST

```

\$TYPERR:

```

HOTWARM: .WORD $SCRLF ; START WITH MESSAGE PREFIX, HOT OR WARM
          MOV RO,-(SP) ; PTR TO "HOT" OR "WARM"
          MOV RI,-(SP) ; SAVE RO
          CLR RO ; SAVE RI
          BISB 2,$ITEMB,RO ; PICKUP ITEM INDEX
          BNE 1$ ; IF ITEM NUMBER FROM ERROR 0,
                   ; JUST TYPE PC OF ERROR
                   ; GET ERROR PC FOR TYPEOUT
                   ; TYPE OCTAL, ALL DIGITS
                   ; EXIT
                   ; ADJUST ERROR # FOR TABLE INDEX
                   ; OF 20. BYTES/ENTRY
1$: DEC RO
   ASL RO
   ASL RO
   MOV RO,RI
   ASL RO
   ASL RO
   ADD RI,RO
   ADD #SERRTB,RO ; FORM TABLE PTR
   MOV (RO)+,2$ ; PICKUP "ERROR MESSAGE" PTR
   BEQ 3$ ; SKIP TYPEOUT IF NULL
   TYPE "ERROR MESSAGE" ; TYPE "ERROR MESSAGE"
   "ERROR MESSAGE" PTR HERE
2$: .WORD 0 ; CR & LF
   TYPE , $SCRLF ; "TEST # ERR PC" HEADER
3$: TYPE , 11$ ; PICKUP "DATA HEADER" PTR
   MOV (RO)+,4$ ; SKIP TYPEOUT IF NULL
   BEQ 5$ ; TYPE "DATA HEADER"
   TYPE "DATA HEADER" ; "DATA HEADER" PTR HERE
4$: .WORD 0 ; CR & LF
5$: TYPE , $SCRLF ; ($TESTN)
   MOV 28$,-(SP)

```

ERROR MESSAGE TYPEOUT ROUTINE (MODIFIED SYSMAC)

4673	030530	104402			TYP0C			:OCTAL W/ LEADING ZEROS
4674	030532	104401	030630		TYPE	10\$		<HT>
4675	030536	017746	000064		MOV	29\$, -(SP)		(SERRPC)
4676	030542	104402			TYP0C			:OCTAL W/ LEADING ZEROS
4677	030544	104401	030630		TYPE	10\$		<HT>
4678	030550	012001			MOV	(R0)+, R1		PICKUP "DATA TABLE" PTR
4679	030552	001407			BEQ	7\$		EXIT IF NULL
4680	030554	013146		6\$:	MOV	2(R1)+, -(SP)		SAVE FOR TYPEOUT
4681	030556	104402			TYP0C			TYPE OCTAL, ALL DIGITS
4682	030560	005711		14\$:	TST	(R1)		ANOTHER NUMBER ?
4683	030562	001403			BEQ	7\$		NO - EXIT
4684	030564	104401	030630		TYPE	10\$		TAB BETWEEN ELEMENTS
4685	030570	000771			BR	6\$		LOOP ON DATA TABLE VECTOR
4686	030572	104401	001177		TYPE	\$CRLF		END THE LINE
4687	030576	012001		7\$:	MOV	(R0)+, R1		GET OPERAND LIST PTR
4688	030600	001406			BEQ	15\$		ALL DONE
4689	030602	012137	030612		MOV	(R1)+, 12\$		OPERAND TITLE
4690	030606	001402			BEQ	13\$		SKIP IF ZERO
4691	030610	104401			TYPE			TYPE IT
4692	030612	000000		12\$:	.WORD	0		FROM HERE
4693	030614	000761		13\$:	BR	14\$		GO FINISH DATA LIST
4694	030616	012601		15\$:	MOV	(SP)+, R1		RESTORE R1
4695	030620	012600			MOV	(SP)+, R0		RESTORE R0
4696	030622	000207			RTS	PC		RETURN
4697	030624	001206		8\$:	.WORD	\$TESTN		
4698	030626	001122		9\$:	.WORD	\$ERRPC		
4699	030630	000011		10\$:	.ASCIZ	<11>		<HT>
4700	030632	042524	052123 021440	11\$:	.ASCIZ	"TEST # ERR PC		
4701	030640	042411	051122 050040					
4702	030646	004503	000					
4703		030652			.EVEN			

4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720

.SBTTL TYPE ROUTINE

```

*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
*          TYPE          ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
*          TYPE
*          MESADR
*

```

4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759

```

030652 105737 001165
030656 100002
030660 000000
030662 000430
030664 010046
030666 017600 000002
030672 122737 000001 001222
030700 001011
030702 132737 000100 001223
030710 001405
030712 010037 030722
030716 004737 031142
030722 000000
030724 132737 000040 001223
030732 001003
030734 112046
030736 001005
030740 005726
030742 012600
030744 062716 000002
030750 000002
030752 122716 000011
030756 001430
030760 122716 000200
030764 001006
030766 005726
030770 104401
030772 001177
030774 105037 031130
031000 000755
031002 004737 031064
031006 123726 001164
031012 001350
031014 013746 001162
031020 105366 000001
031024 002770
031026 004737 031064
031032 105337 031130

```

```

STYPE:  TSTB      $TPFLG
         BPL      1$
         HALT
         BR       3$
1$:      MOV      RO, -(SP)
         MOV      22(SP), RO
         CMPB    #APTENV, $ENV
         BNE     62$
         BITB    #APTSPool, $ENVM
         BEQ     62$
         MOV      RO, 61$
         JSR     PC, $ATY3
61$:     .WORD    0
62$:     BITB    #APTCsup, $ENVM
         BNE     60$
2$:      MOVB    (RO)+, -(SP)
         BNE     4$
         TST    (SP)+
60$:     MOV      (SP)+, RO
3$:      ADD     #2, (SP)
         RTI
4$:      CMPB    #HT, (SP)
         BEQ     8$
         CMPB    #CRLF, (SP)
         BNE     5$
         TST    (SP)+
         TYPE
         $CRLF
         CLRB   $CHARCNT
         BR     2$
5$:      JSR     PC, $TYPEC
6$:      CMPB    $FILLC, (SP)+
         BNE     2$
         MOV     $NULL, -(SP)
7$:      DECB   1(SP)
         BLT    6$
         JSR     PC, $TYPEC
         DECB   $CHARCNT

```

```

;; IS THERE A TERMINAL?
;; BR IF YES
;; HALT HERE IF NO TERMINAL
;; LEAVE
;; SAVE RO
;; GET ADDRESS OF ASCIZ STRING
;; RUNNING IN APT MODE
;; NO GO CHECK FOR APT CONSOLE
;; SPOOL MESSAGE TO APT
;; NO GO CHECK FOR CONSOLE
;; SETUP MESSAGE ADDRESS FOR APT
;; SPOOL MESSAGE TO APT
;; MESSAGE ADDRESS
;; APT CONSOLE SUPPRESSED
;; YES, SKIP TYPE OUT
;; PUSH CHARACTER TO BE TYPED ONTO STACK
;; BR IF IT ISN'T THE TERMINATOR
;; IF TERMINATOR POP IT OFF THE STACK
;; RESTORE RO
;; ADJUST RETURN PC
;; RETURN
;; BRANCH IF <HT>
;; BRANCH IF NOT <CRLF>
;; POP <CR><LF> EQUIV
;; TYPE A CR AND LF
;; CLEAR CHARACTER COUNT
;; GET NEXT CHARACTER
;; GO TYPE THIS CHARACTER
;; IS IT TIME FOR FILLER CHARS.?
;; IF NO GO GET NEXT CHAR.
;; GET # OF FILLER CHARS. NEEDED
;; AND THE NULL CHAR.
;; DOES A NULL NEED TO BE TYPED?
;; BR IF NO--GO POP THE NULL OFF OF STACK
;; GO TYPE A NULL
;; DO NOT COUNT AS A COUNT

```

```

4760 031036 000770          BR      7$          ;;LOOP
4761
4762          ;HORIZONTAL TAB PROCESSOR
4763
4764 031040 112716 000040      8$:      MOVB      #' (SP)          ;;REPLACE TAB WITH SPACE
4765 031044 004737 031064      9$:      JSR      PC,$TYPEC          ;;TYPE A SPACE
4766 031050 132737 000007 031130      BITB      #7,$CHARCNT          ;;BRANCH IF NOT AT
4767 031056 001372          BNE      9$          ;;TAB STOP
4768 031060 005726          TST      (SP)+          ;;POP SPACE OFF STACK
4769 031062 000724          BR      2$          ;;GET NEXT CHARACTER
4770 031064 105777 150066      $TYPEC: TSTB      #STPS          ;;WAIT UNTIL PRINTER IS READY
4771 031070 100375          BPL      $TYPEC
4772 031072 116677 000002 150060      MOVB      2(SP),#STPB          ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4773 031100 122766 000015 000002      CMPB      #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
4774 031106 001003          BNE      1$          ;;BRANCH IF NO
4775 031110 105037 031130          CLRB      $CHARCNT          ;;YES--CLEAR CHARACTER COUNT
4776 031114 000406          BR      $TYPEX          ;;EXIT
4777 031116 122766 000012 000002 1$:      CMPB      #LF,2(SP)          ;;IS CHARACTER A LINE FEED?
4778 031124 001402          BEQ      $TYPEX          ;;BRANCH IF YES
4779 031126 105227          INCB      (PC)+          ;;COUNT THE CHARACTER
4780 031130 000000          $CHARCNT: .WORD 0          ;;CHARACTER COUNT STORAGE
4781 031132 000207          $TYPEX: RTS      PC
4782
  
```

```

4783      .SBTTL  APT COMMUNICATIONS ROUTINE
4784
4785      ;*****
4786 031134 112737 000001 031400 SATY1:  MOVB  #1,$FFLG      ;; TO REPORT FATAL ERROR
4787 031142 112737 000001 031376 SATY3:  MOVB  #1,$MFLG      ;; TO TYPE A MESSAGE
4788 031150 000403          SATYC          BR
4789 031152 112737 000001 031400 SATY4:  MOVB  #1,$FFLG      ;; TO ONLY REPORT FATAL ERROR
4790 031160          SATYC          MOV      RO,-(SP)      ;; PUSH RO ON STACK
4791 031160 010046          MOV      R1,-(SP)      ;; PUSH R1 ON STACK
4792 031162 010146          TSTB   $MFLG      ;; SHOULD TYPE A MESSAGE?
4793 031164 105737 031376          BEQ    5$          ;; IF NOT: BR
4794 031170 001450          CMPB   #APTENV,$ENV      ;; OPERATING UNDER APT?
4795 031172 122737 000001 001222          BNE   3$          ;; IF NOT: BR
4796 031200 001031          BITB   #APTSPOOL,$ENVM  ;; SHOULD SPOOL MESSAGES?
4797 031202 132737 000100 001223          BEQ    3$          ;; IF NOT: BR
4798 031210 001425          MOV      #4(SP),RO      ;; GET MESSAGE ADDR.
4799 031212 017600 000004          ADD      #2,4(SP)      ;; BUMP RETURN ADDR.
4800 031216 062766 000002 000004          TST   $MSGTYPE      ;; SEE IF DONE W/ LAST XMISSION?
4801 031224 005737 001202          BNE   1$          ;; IF NOT: WAIT
4802 031230 001375          MOV      RO,$MSGAD      ;; PUT ADDR IN MAILBOX
4803 031232 010037 001216          TSTB   (RO)+          ;; FIND END OF MESSAGE
4804 031236 105720          BNE   2$
4805 031240 001376          SUB      $MSGAD,RO      ;; SUB START OF MESSAGE
4806 031242 163700 001216          ASR      RO          ;; GET MESSAGE LNTH IN WORDS
4807 031246 006200          MOV      RO,$MSGLEN      ;; PUT LENGTH IN MAILBOX
4808 031250 010037 001220          MOV      #4,$MSGTYPE    ;; TELL APT TO TAKE MSG.
4809 031254 012737 000004 001202          BR     5$
4810 031262 000413          MOV      #4(SP),4$      ;; PUT MSG ADDR IN JSR LINKAGE
4811 031264 017637 000004 031310 3$:  ADD      #2,4(SP)      ;; BUMP RETURN ADDRESS
4812 031272 062766 000002 000004          MOV      177776,-(SP)  ;; PUSH 177776 ON STACK
4813 031300 013746 177776          JSR     PC,$TYPE      ;; CALL TYPE MACRO
4814 031304 004737 030652          .WORD  0
4815 031310 000000          4$:
4816 031312          5$:
4817 031312 105737 031400          10$:  TSTB   $FFLG          ;; SHOULD REPORT FATAL ERROR?
4818 031316 001416          BEQ    12$          ;; IF NOT: BR
4819 031320 005737 001222          TST   $ENV          ;; RUNNING UNDER APT?
4820 031324 001413          BEQ    12$          ;; IF NOT: BR
4821 031326 005737 001202          11$:  TST   $MSGTYPE      ;; FINISHED LAST MESSAGE?
4822 031332 001375          BNE   11$          ;; IF NOT: WAIT
4823 031334 017637 000004 001204          MOV      #4(SP),$FATAL  ;; GET ERROR #
4824 031342 062766 000002 000004          ADD      #2,4(SP)      ;; BUMP RETURN ADDR.
4825 031350 005237 001202          INC      $MSGTYPE      ;; TELL APT TO TAKE ERROR
4826 031354 105037 031400          12$:  CLRB   $FFLG          ;; CLEAR FATAL FLAG
4827 031360 105037 031377          CLRB   $LFLG          ;; CLEAR LOG FLAG
4828 031364 105037 031376          CLRB   $MFLG          ;; CLEAR MESSAGE FLAG
4829 031370 012601          MOV      (SP)+,R1      ;; POP STACK INTO R1
4830 031372 012600          MOV      (SP)+,RO      ;; POP STACK INTO RO
4831 031374 000207          RTS     PC          ;; RETURN
4832 031376 000          $MFLG: .BYTE 0      ;; MESSG. FLAG
4833 031377 000          $LFLG: .BYTE 0      ;; LOG FLAG
4834 031400 000          $FFLG: .BYTE 0      ;; FATAL FLAG
4835          .EVEN
4836          APTSIZE=200
4837          APTENV=001
4838          APTSPOOL=100

```

M09

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 101
D9FP08.P11 04-MAY-77 17:30 APT COMMUNICATIONS ROUTINE

SEQ 0102

4839

000040

APTC SUP=040

4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895

031402 017646 000000
 031406 116637 000001 031625
 031414 112637 031627
 031420 062716 000002
 031424 000406
 031426 112737 000001 031625
 031434 112737 000006 031627
 031442 112737 000005 031624
 031450 010346
 031452 010446
 031454 010546
 031456 113704 031627
 031462 005404
 031464 062704 000006
 031470 110437 031626
 031474 113704 031625
 031500 016605 000012
 031504 005003
 031506 006105
 031510 000404
 031512 006105
 031514 006105
 031516 006105
 031520 010503
 031522 006103
 031524 105337 031626
 031530 100016
 031532 042703 177770
 031536 001002
 031540 005704
 031542 001403

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
: THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
: OCTAL (ASCII) NUMBER AND TYPE IT.
: $TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
: CALL:
:   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
:   TYPOS   ;; CALL FOR TYPEOUT
:   .BYTE  N               ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
:   .BYTE  M               ;; M=1 OR 0
:                               ;; 1=TYPE LEADING ZEROS
:                               ;; 0=SUPPRESS LEADING ZEROS
: $STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
: $TYPOS OR $TYPOC
: CALL:
:   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
:   TYPON   ;; CALL FOR TYPEOUT
: $STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
: CALL:
:   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
:   TYPOC   ;; CALL FOR TYPEOUT
: $TYPOS: MOV     2(SP),-(SP) ;; PICKUP THE MODE
:         MOV     1(SP),SOFILL ;; LOAD ZERO FILL SWITCH
:         MOV     (SP)+,SOMODE+1 ;; NUMBER OF DIGITS TO TYPE
:         ADD     #2,(SP) ;; ADJUST RETURN ADDRESS
:         BR     $STYPON
: $TYPOC: MOV     #1,SOFILL ;; SET THE ZERO FILL SWITCH
:         MOV     #6,SOMODE+1 ;; SET FOR SIX(6) DIGITS
: $STYPON: MOV     #5,SOCNT ;; SET THE ITERATION COUNT
:         MOV     R3,-(SP) ;; SAVE R3
:         MOV     R4,-(SP) ;; SAVE R4
:         MOV     R5,-(SP) ;; SAVE R5
:         MOV     SOMODE+1,R4 ;; GET THE NUMBER OF DIGITS TO TYPE
:         NEG     R4
:         ADD     #6,R4 ;; SUBTRACT IT FOR MAX. ALLOWED
:         MOV     R4,SOMODE ;; SAVE IT FOR USE
:         MOV     SOFILL,R4 ;; GET THE ZERO FILL SWITCH
:         MOV     12(SP),R5 ;; PICKUP THE INPUT NUMBER
:         CLR     R3 ;; CLEAR THE OUTPUT WORD
: 1$: ROL     R5 ;; ROTATE MSB INTO "C"
:     BR     3$ ;; GO DO MSB
: 2$: ROL     R5 ;; FORM THIS DIGIT
:     ROL     R5
:     ROL     R5
:     MOV     R5,R3
: 3$: ROL     R3 ;; GET LSB OF THIS DIGIT
:     DECB    SOMODE ;; TYPE THIS DIGIT?
:     BPL     7$ ;; BR IF NO
:     BIC     #177770,R3 ;; GET RID OF JUNK
:     BNE     4$ ;; TEST FOR 0
:     TST     R4 ;; SUPPRESS THIS 0?
:     BEQ     5$ ;; BR IF YES
    
```

B10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 103
 DQFP08.P11 04-MAY-77 17:30 BINARY TO OCTAL (ASCII) AND TYPE

SEQ 0104

4896	031544	005204		45:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
4897	031546	052703	000060		BIS	0'0,R3	:: MAKE THIS DIGIT ASCII
4898	031552	052703	000040	55:	BIS	0' R3	:: MAKE ASCII IF NOT ALREADY
4899	031556	110337	031622		MOVB	R3,B5	:: SAVE FOR TYPING
4900	031562	104401	031622		TYPE	B5	:: GO TYPE THIS DIGIT
4901	031566	105337	031624	75:	DECB	\$OCNT	:: COUNT BY 1
4902	031572	003347			BGT	25	:: BR IF MORE TO DO
4903	031574	002402			BLT	65	:: BR IF DONE
4904	031576	005204			INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
4905	031600	000744			BR	25	:: GO DO THE LAST DIGIT
4906	031602	012605		65:	MOV	(SP)+,R5	:: RESTORE R5
4907	031604	012604			MOV	(SP)+,R4	:: RESTORE R4
4908	031606	012603			MOV	(SP)+,R3	:: RESTORE R3
4909	031610	016666	000002 000004		MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
4910	031616	012616			MOV	(SP)+,(SP)	
4911	031620	000002			RTI		:: RETURN
4912	031622	000		85:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
4913	031623	000			.BYTE	0	:: TERMINATOR FOR TYPE ROUTINE
4914	031624	000			.BYTE	0	:: OCTAL DIGIT COUNTER
4915	031625	000			.BYTE	0	:: ZERO FILL SWITCH
4916	031626	000000			.WORD	0	:: NUMBER OF DIGITS TO TYPE

4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953

.SBTTL TRAP DECODER

; THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
; AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
; OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
; GO TO THAT ROUTINE.

\$TRAP: MOV RO, -(SP) ; SAVE RO
MOV 2(SP), RO ; GET TRAP ADDRESS
TST -(RO) ; BACKUP BY 2
MOVB (RO), RO ; GET RIGHT BYTE OF TRAP
ASL RO ; POSITION FOR INDEXING
MOV \$TRPAD(RO), RO ; INDEX TO TABLE
RTS RO ; GO TO ROUTINE

;; THIS IS USE TO HANDLE THE "GETPRI" MACRO

\$TRAP2: MOV (SP), -(SP) ; MOVE THE PC DOWN
MOV 4(SP), 2(SP) ; MOVE THE PSW DOWN
RTI ; RESTORE THE PSW

.SBTTL TRAP TABLE

; THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
; BY THE "TRAP" INSTRUCTION.

ROUTINE

\$TRPAD: .WORD \$TRAP2 TRAP+1(104401) TTY TYPEOUT ROUTINE
\$TYPE ; CALL=TYPE TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOC ; CALL=TYPOC TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPOS ; CALL=TYPOS TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPON ; CALL=TYPON

```

4954 .SBTTL POWER DOWN AND UP ROUTINES
4955
4956
4957 *****
4958 031676 012737 032050 000024 $PWRDN: MOV $SILLUP,2#PWRVEC ;; SET FOR FAST UP
4959 031704 012737 000340 000026 MOV $340,2#PWRVEC+2 ;; Prio:7
4960 031712 010046 MOV R0,-(SP) ;; PUSH R0 ON STACK
4961 031714 010146 MOV R1,-(SP) ;; PUSH R1 ON STACK
4962 031716 010246 MOV R2,-(SP) ;; PUSH R2 ON STACK
4963 031720 010346 MOV R3,-(SP) ;; PUSH R3 ON STACK
4964 031722 010446 MOV R4,-(SP) ;; PUSH R4 ON STACK
4965 031724 010546 MOV R5,-(SP) ;; PUSH R5 ON STACK
4966 031726 017746 147212 MOV @SWR,-(SP) ;; PUSH @SWR ON STACK
4967 031732 010637 032054 MOV SP,$SAVR6 ;; SAVE SP
4968 031736 012737 031750 000024 MOV $PWRUP,2#PWRVEC ;; SET UP VECTOR
4969 031744 000000 HALT
4970 031746 000776 BR -2 ;; HANG UP
4971
4972 *****
4973 POWER UP ROUTINE
4974 031750 012737 032050 000024 $PWRUP: MOV $SILLUP,2#PWRVEC ;; SET FOR FAST DOWN
4975 031756 013706 032054 MOV $SAVR6,SP ;; GET SP
4976 031762 005037 032054 CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
4977 031766 005237 032054 15: INC $SAVR6 ;; WAIT FOR THE INC
4978 031772 001375 BNE 15 OF WORD
4979 031774 011600 MOV (SP),R0 ;; GET SAVED SWR OFF STACK
4980 031776 076600 000226 MED 226 ;; RESTORE SWR CONTENTS
4981 032002 012677 147136 MOV (SP)+,@SWR ;; POP STACK INTO @SWR
4982 032006 012605 MOV (SP)+,R5 ;; POP STACK INTO R5
4983 032010 012604 MOV (SP)+,R4 ;; POP STACK INTO R4
4984 032012 012603 MOV (SP)+,R3 ;; POP STACK INTO R3
4985 032014 012602 MOV (SP)+,R2 ;; POP STACK INTO R2
4986 032016 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
4987 032020 012600 MOV (SP)+,R0 ;; POP STACK INTO R0
4988 032022 012737 031676 000024 MOV $PWRDN,2#PWRVEC ;; SET UP THE POWER DOWN VECTOR
4989 032030 012737 000340 000026 MOV $340,2#PWRVEC+2 ;; Prio:7
4990 032036 104401 TYPE $POWER ;; REPORT THE POWER FAILURE
4991 032040 032056 $PWRMG: .WORD $POWER ;; POWER FAIL MESSAGE POINTER
4992 032042 012716 MOV (PC)+,(SP) ;; RESTART AT START
4993 032044 003000 $PWRAD: .WORD START ;; RESTART ADDRESS
4994 032046 000002 RTI
4995 032050 000000 $SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
4996 032052 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
4997 032054 000000 $SAVR6: 0 ;; PUT THE SP HERE
4998 032056 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
4999 032064 000122 .EVEN
5000

```

E10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11
DAPPOB.P11 04-MAY-77 17:30

27(1006) 04-MAY-77 18:18 PAGE 106
ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC

SEQ 0107

```
5001 .SBTTL ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC
5002
5003 ;MESSAGE PREFIX
5004 032066 047510 035124 000040 ASCHOT: .ASCIZ "HOT: "
5005 032074 040527 046522 020072 ASCHRM: .ASCIZ "WARM: "
5006 032102 000
5007
5008 ;ERROR MESSAGES HERE
5009 032103 106 046455 042117 EMA: .ASCIZ "F-MODE EXERCISER - FPS ERROR"
5010 032110 020105 054105 051105
5011 032116 044503 042523 020122
5012 032124 020055 050106 C-0123
5013 032132 051105 047522 000122
5014 032140 026504 047515 042504 EMB: .ASCIZ "D-MODE EXERCISER - FPS ERROR"
5015 032146 042440 042530 041522
5016 032154 051511 051105 026440
5017 032162 043040 051520 042440
5018 032170 051122 051117 000
5019 032175 106 046455 042117 EMC: .ASCIZ "F-MODE EXERCISER - RESULT ERROR"
5020 032202 020105 054105 051105
5021 032210 044503 042523 020122
5022 032216 020055 042522 052523
5023 032224 052114 042440 051122
5024 032232 051117 000
5025 032235 104 046455 042117 EMD: .ASCIZ "D-MODE EXERCISER - RESULT ERROR"
5026 032242 020105 054105 051105
5027 032250 044503 042523 020122
5028 032256 020055 042522 052523
5029 032264 052114 042440 051122
5030 032272 051117 000
5031 032275 101 042104 043050 EME: .ASCIZ "ADD(F/D) - RESULT ERROR"
5032 032302 042057 020051 020055
5033 032310 042522 052523 052114
5034 032316 042440 051122 051117
5035 032324 000
5036 032325 123 041125 043050 EMF: .ASCIZ "SUB(F/D) - RESULT ERROR"
5037 032332 042057 020051 020055
5038 032340 042522 052523 052114
5039 032346 042440 051122 051117
5040 032354 000
5041 032355 115 046125 043050 EMG: .ASCIZ "MUL(F/D) - RESULT ERROR"
5042 032362 042057 020051 020055
5043 032370 042522 052523 052114
5044 032376 042440 051122 051117
5045 032404 000
5046 032405 104 053111 043050 EMH: .ASCIZ "DIV(F/D) - RESULT ERROR"
5047 032412 042057 020051 020055
5048 032420 042522 052523 052114
5049 032426 042440 051122 051117
5050 032434 000
5051 032435 125 042516 050130 EMI: .ASCIZ "UNEXPECTED FLOATING POINT TRAP, IGNORED AND CONTINUING"
5052 032442 041505 042524 020104
5053 032450 046106 040517 044524
5054 032456 043516 050040 044517
5055 032464 052116 052040 040522
5056 032472 026120 044440 047107
```

F10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11
 D8FP08.P11 04-MAY-77 17:30

27(1006) 04-MAY-77 18:18 PAGE 107
 ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC

SEQ 0108

5057	032500	051117	042105	040440	
5058	032506	042116	041440	047117	
5059	032514	044524	052516	047111	
5060	032522	000107			
5061	032524	042101	043104	026440	EMJ: .ASCIZ "A00F - FPS ERROR"
5062	032530	043040	051520	042440	
5063	032540	051123	051117	000	
5064	032546	123	041125	020106	EMK: .ASCIZ "SUBF - FPS ERROR"
5065	032550	020055	050106	020123	
5066	032560	051105	047522	000122	
5067	032566	052515	043114	026440	EML: .ASCIZ "MULF - FPS ERROR"
5068	032574	043040	051520	042440	
5069	032602	051122	051117	000	
5070	032607	104	053111	020106	EMM: .ASCIZ "DIVF - FPS ERROR"
5071	032614	020055	050106	020123	
5072	032622	051105	047522	000122	
5073	032630	042101	042104	026440	EMN: .ASCIZ "A000 - FPS ERROR"
5074	032636	043040	051520	042440	
5075	032644	051122	051117	000	
5076	032651	123	041125	020104	EMO: .ASCIZ "SUB0 - FPS ERROR"
5077	032656	020055	050106	020123	
5078	032664	051105	047522	000122	
5079	032672	052515	042114	026440	EMP: .ASCIZ "MULD - FPS ERROR"
5080	032700	043040	051520	042440	
5081	032706	051122	051117	000	
5082	032713	104	053111	020104	EMQ: .ASCIZ "DIV0 - FPS ERROR"
5083	032720	020055	050106	020123	
5084	032726	051105	047522	000122	
5085	032734	042101	043104	026440	EMR: .ASCIZ "A00F - FEC/FEA ERROR"
5086	032742	043040	041505	043057	
5087	032750	040505	042440	051122	
5088	032756	051117	000		
5089	032761	123	041125	020106	EMS: .ASCIZ "SUBF - FEC/FEA ERROR"
5090	032766	020055	042506	027503	
5091	032774	042506	020101	051105	
5092	033002	047522	000122		
5093	033006	052515	043114	026440	EMT: .ASCIZ "MULF - FEC/FEA ERROR"
5094	033014	043040	041505	043057	
5095	033022	040505	042440	051122	
5096	033030	051117	000		
5097	033033	104	053111	020106	EMU: .ASCIZ "DIVF - FEC/FEA ERROR"
5098	033040	020055	042506	027503	
5099	033046	042506	020101	051105	
5100	033054	047522	000122		
5101	033060	042101	042104	026440	EMV: .ASCIZ "A000 - FEC/FEA ERROR"
5102	033066	043040	041505	043057	
5103	033074	040505	042440	051122	
5104	033100	051117	000		
5105	033106	123	041125	020104	EMW: .ASCIZ "SUB0 - FEC/FEA ERROR"
5106	033112	020055	042506	027503	
5107	033120	042506	020101	051105	
5108	033126	047522	000122		
5109	033132	052515	042114	026440	EMX: .ASCIZ "MULD - FEC/FEA ERROR"
5110	033140	043040	041505	043057	
5111	033146	040505	042440	051122	
5112	033154	051117	000		

S113 033157 104 053111 020104
S114 033164 020055 042506 027503
S115 033172 042506 020101 051105
S116 033200 047522 000122
S117
S118
S119
S120 033204 054110 027520 004504
S121 033212 041505 042503 000104
S122 033220 026455 042503 050130
S123 033228 041505 042503 026504
S124 033234 026455 026455 026455
S125 033242 026455 026455 026455
S126 033250 026455 026455 026455
S127 033258 026455 026455 026455
S128 033264 026455 026455 026455
S129 033272 026455 026455 026455
S130 033280 026455 026455 026455
S131 033288 026455 026455 026455
S132 033314 026455 026455 026455
S133 033322 026455 026455 026455
S134 033330 042503 042503 053111
S135 033336 042105 026455 026455
S136 033344 026455 026455 026455
S137 033352 000055 000055 000055
S138 033354 046117 020104 041520
S139 033362 047411 042114 050040
S140 033370 004523 043040 051520
S141 033376 020011 042506 004503
S142 033404 043040 040505 020011
S143 033412 043040 051520 020011
S144 033420 043040 041505 020011
S145 033426 043040 040505 000
S146 033433 105 050130 042047
S147 033440 043055 041505 051055
S148 033446 053103 042047 042411
S149 033454 050130 042047 043055
S150 033462 040505 051055 053103
S151 033470 042047 000
S152
S153
S154
S155
S156 033474 002400 002362 000000
S157 033502 002402 002364 002404
S158 033510 002366 000000
S159 033514 002416 002420 002406
S160 033522 002410 000000
S161 033528 002416 002420 002422
S162 033534 002424
S163 033536 002406 002410 002412
S164 033544 002414 000000
S165 033550 002370 002372 002362
S166 033556 002364 002366 002400
S167 033564 002402 002404 000000
S168

EMY: .ASCIZ "DIVD - FEC/FEA ERROR"

:DATA HEADERS HERE
DMA: .ASCIZ "EXP'D RCV'D"

DMB: .ASCIZ "----EXPECTED---- --RECEIVED--"

DMC: .ASCIZ "-----EXPECTED----- --RECEIVED-----"

DMD: .ASCIZ "OLD PC OLD PS FPS FEC FEA \$FPS \$FEC \$FEA"

DME: .ASCIZ "EXP'D-FEC-RCV'D EXP'D-FEA-RCV'D"

:DATA VECTORS HERE

DTA: .EVEN
.WORD \$FPS, FPS, 0
DTB: .WORD \$FEC, FEC, \$FEA, FEA, 0

DTD: .WORD ANS2, ANS2+2, ANS1, ANS1+2, 0

DTE: .WORD ANS2, ANS2+2, ANS2+4, ANS2+6

.WORD ANS1, ANS1+2, ANS1+4, ANS1+6, 0

DTF: .WORD FPPOPC, FPPOPS, FPS, FEC, FEA, \$FPS, \$FEC, \$FEA, 0

H10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 109
 DDFP08.P11 04-MAY-77 17:30 ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC

SEQ 0110

```

5169 ; OPERAND VECTORS HERE
5170 .EVEN
5171 033572 034032 002426 002430 L00: .WORD XLO,LONUM,LONUM+2,LONUM+4,LONUM+6,0
5172 033600 002432 002434 000000
5173 033606 034042 002436 002440 H10: .WORD XHI,?INUM,HINUM+2,HINUM+4,HINUM+6,0
5174 033614 002442 002444 000000
5175 033622 034032 002426 002430 L0F: .WORD XLO,LONUM,LONUM+2,0
5176 033630 000000
5177 033632 034042 002436 002440 H1F: .WORD XHI,HINUM,HINUM+2,0
5178 033640 000000
5179 033642 034052 002446 002450 OP1F: .WORD XOP1,OP1,OP1+2,0
5180 033650 000000
5181 033652 034060 002456 002460 OP2F: .WORD XOP2,OP2,OP2+2,0
5182 033660 000000
5183 033662 034066 002466 002470 OP3F: .WORD XOP3,OP3,OP3+2,0
5184 033670 000000
5185 033672 034074 002476 002500 OP4F: .WORD XOP4,OP4,OP4+2,0
5186 033700 000000
5187 033702 034110 002506 002510 OP5F: .WORD XOP5,OP5,OP5+2,0
5188 033710 000000
5189 033712 034102 002516 002520 OP6F: .WORD XOP6,OP6,OP6+2,0
5190 033720 000000
5191 033722 034052 002446 002450 OP10: .WORD XOP1,OP1,OP1+2,OP1+4,OP1+6,0
5192 033730 002452 002454 000000
5193 033736 034060 002456 002460 OP20: .WORD XOP2,OP2,OP2+2,OP2+4,OP2+6,0
5194 033744 002462 002464 000000
5195 033752 034066 002466 002470 OP30: .WORD XOP3,OP3,OP3+2,OP3+4,OP3+6,0
5196 033760 002472 002474 000000
5197 033766 034074 002476 002500 OP40: .WORD XOP4,OP4,OP4+2,OP4+4,OP4+6,0
5198 033774 002502 002504 000000
5199 034002 034110 002506 002510 OP50: .WORD XOP5,OP5,OP5+2,OP5+4,OP5+6,0
5200 034010 002512 002514 000000
5201 034016 034102 002516 002520 OP60: .WORD XOP6,OP6,OP6+2,OP6+4,OP6+6,0
5202 034024 002522 002524 000000
  
```

```

5205 ; OPERAND TITLES
5206 034032 047514 052516 035115 XLO: .ASCIZ "LONUM:"(11)
5207 034040 000011
5208 034042 044510 052516 035115 XHI: .ASCIZ "HINUM:"(11)
5209 034050 000011
5210 034052 050117 035061 000011 XOP1: .ASCIZ "OP1:"(11)
5211 034060 050117 035062 000011 XOP2: .ASCIZ "OP2:"(11)
5212 034066 050117 035063 000011 XOP3: .ASCIZ "OP3:"(11)
5213 034074 050117 035064 000011 XOP4: .ASCIZ "OP4:"(11)
5214 034102 050117 035066 000011 XOP6: .ASCIZ "OP6:"(11)
5215 034110 050117 035065 000011 XOP5: .ASCIZ "OP5:"(11)
5216 ; THE END
5217 000001 .END
  
```

RBASE =	000000	258			
ACOM1 =	000000	258			
ACOM2 =	000000	258			
ACPUOP =	000000	258	273		
A00C0	002526	463#	570	3372	3671#
A00C1	002530	463#	3372	3715#	
A00C2	002532	464#	3372	3694#	
A00C3	002534	465#	3372	3717#	
A00C4	002536	466#	3372	3760#	
A00C5	002540	467#	3372	3821#	3841#
A00C6	002542	468#	3372	3824#	3844#
A00C7	002544	469#	3372	3855#	3901#
A00C8	002546	470#	3372	3902#	
A00M0 =	000000	258			
A00M1 =	000000	258			
A00M10 =	000000	258			
A00M11 =	000000	258			
A00M12 =	000000	258			
A00M13 =	000000	258			
A00M14 =	000000	258			
A00M15 =	000000	258			
A00M2 =	000000	258			
A00M3 =	000000	258			
A00M4 =	000000	258			
A00M5 =	000000	258			
A00M6 =	000000	258			
A00M7 =	000000	258			
A00M8 =	000000	258			
A00M9 =	000000	258			
ADEVCT =	000000	258	264		
ADEVH =	000000	258			
AEND1	004102	706	713	717#	
AEND10	006456	1151	1160	1167	1171#
AEND11	006742	1224	1231	1235#	
AEND12	007236	1291	1300	1304#	
AEND13	007464	1348	1355	1359#	
AEND14	007722	1403	1412	1416#	
AEND15	010230	1472	1479	1484	1488#
AEND16	010556	1544	1553	1560	1564#
AEND17	011026	1608	1615	1620	1624#
AEND2	004376	772	781	785#	
AEND20	011316	1668	1677	1684	1688#
AEND3	004624	829	836	840#	
AEND4	005062	884	893	897#	
AEND5	005370	953	960	965	969#
AEND6	005716	1026	1035	1042	1046#
AEND7	006166	1091	1098	1103	1107#
AENV =	000000	258	269		
AENVH =	000000	258	270		
AERR1	004002	688	692#		
AERR11	006642	1206	1210#		
AERR12	007126	1273	1277#		
AERR15	010106	1454	1458#		
AERR16	010414	1526	1530#		
AERR2	004266	754	758#		
AERR5	005246	935	939#		

APTSIZ=	000200	560	4836#				
APTSPO=	000100	4729	4797	4838#			
ARET1	003756	659	685#				
ARET10	006326	1116	1142#				
ARET11	006616	1177	1203#				
ARET12	007102	1244	1270#				
ARET13	007376	1313	1339#				
ARET14	007624	1368	1394#				
ARET15	010062	1425	1451#				
ARET16	010370	1497	1523#				
ARET17	010716	1573	1599#				
ARET2	004242	725	751#				
ARET20	011166	1633	1659#				
ARET3	004536	794	820#				
ARET4	004764	849	875#				
ARET5	005222	906	932#				
ARET6	005530	979	1005#				
ARET7	006056	1056	1082#				
ASCHOT	032066	651	5004#				
ASCHRM	032074	653	5005#				
ASWREG=	000000	258	271				
ATESTN=	000000	258	262				
ATST1	004040	693	701	704#			
ATST10	006352	1145	1149#				
ATST11	006700	1211	1219	1222#			
ATST12	007164	1278	1286	1289#			
ATST13	007422	1342	1346#				
ATST14	007650	1397	1401#				
ATST15	010144	1459	1467	1470#			
ATST16	010452	1531	1539	1542#			
ATST17	010742	1602	1606#				
ATST2	004324	759	767	770#			
ATST20	011212	1662	1666#				
ATST3	004562	823	827#				
ATST4	005010	878	882#				
ATST5	005304	940	948	951#			
ATST6	005612	1013	1021	1024#			
ATST7	006102	1085	1089#				
AUNIT =	000000	258	265				
AUSMR =	000000	258	272				
AVECT1=	000000	258					
AVECT2=	000000	258					
BGNMES	002622	498#	575				
BIT0 =	000001	125#					
BIT00 =	000001	115#	125				
BIT01 =	000002	114#	124				
BIT02 =	000004	113#	123				
BIT03 =	000010	112#	122				
BIT04 =	000020	111#	121	581	633	3260	
BIT05 =	000040	110#	120				
BIT06 =	000100	109#	119				
BIT07 =	000200	108#	118				
BIT08 =	000400	107#	117	4522			
BIT09 =	001000	106#	116	4530	4605		
BIT1 =	000002	124#					
BIT10 =	002000	105#	4583				

EMV001	001232	298#												
EMV002	001256	302#												
EMV003	001302	306#												
EMV004	001326	311#												
EMV005	001352	316#												
EMV006	001376	320#												
EMV007	001422	324#												
EMV010	001446	328#												
EMV011	001472	332#												
EMV012	001516	336#												
EMV013	001542	340#												
EMV014	001566	344#												
EMV015	001612	349#												
EMV016	001636	354#												
EMV017	001662	358#												
EMV020	001706	362#												
EMV021	001732	366#												
EMV022	001756	370#												
EMV023	002002	375#												
EMV024	002026	379#												
EMV025	002052	383#												
EMV026	002076	387#												
EMV027	002122	391#												
EMV030	002146	395#												
EMV031	002172	399#												
EMV032	002216	403#												
EMV033	002242	408#												
EMV034	002266	412#												
EMV035	002312	417#												
EMV036	002336	421#												
EMW	033105	395	5105#											
EMX	033132	399	5109#											
EMY	033157	403	5113#											
EREG0	002602	488#	4571#											
EREG1	002604	489#	4572#											
EREG2	002606	490#	4573#											
EREG3	002610	491#	4574#											
EREG4	002612	492#	4575#											
EREG5	002614	493#	4576#											
EREG6	002616	494#	4577#	4578#										
EREG7	002620	495#	4579#											
ERRVEC=	000004	128#	545	546#	557#	4513	4514*	4516*	4519*					
ETST1	021076	3106	3110#											
ETST2	021464	3196	3200#											
EXPFEA	002376	436#	659#	725#	794#	849#	906#	979#	1056#	1116#	1177#	1244#	1313#	1368#
		1425#	1497#	1573#	1633#	1694#	1761#	1830#	1885#	1941#	2013#	2089#	2149#	2210#
		2277#	2346#	2413#	2482#	2554#	2630#	2702#	2778#	2844#	2913#	2984#	3827	3847
		3905	4086	4097	4128	4279	4373	4384	4405					
FEA	002366	432#	665#	700	731#	766	800#	855#	912#	947	985#	1020	1062#	1122#
		1183#	1218	1250#	1285	1319#	1374#	1431#	1466	1503#	1538	1579#	1639#	1700#
		1735	1767#	1802	1836#	1891#	1947#	1982	2019#	2054	2095#	2155#	2216#	2251
		2283#	2318	2352#	2387	2419#	2454	2488#	2523	2560#	2595	2636#	2671	2708#
		2743	2784#	2819	2850#	2885	2919#	2954	2990#	3025	3063#	3151#	5157	5165
FEC	002364	431#	664#	694#	696	730#	760#	762	799#	854#	911#	941#	943	984#
		1014#	1016	1061#	1121#	1182#	1212#	1214	1249#	1279#	1281	1318#	1373#	1430#
		1460#	1462	1502#	1532#	1534	1578#	1638#	1699#	1729#	1731	1766#	1796#	1798

G11

		1117#	1131	1144	1178#	1192	1205	1210	1245#	1259	1272	1277	1314#	1328
		1341	1369#	1383	1396	1426#	1440	1453	1458#	1498#	1512	1525	1530	1574#
		1588	1601	1624#	1648	1661	1695#	1709	1722	1727	1762#	1776	1789	1794
		1831#	1845	1858	1886#	1900	1913	1942#	1956	1969	1974	2014#	2028	2041
		2046	2090#	2104	2117	2150#	2164	2177	2211#	2225	2238	2243	2278#	2292
		2305	2310	2347#	2361	2374	2379	2414#	2428	2441	2446	2483#	2497	2510
		2515	2555#	2569	2582	2587	2631#	2645	2658	2663	2703#	2717	2730	2735
		2779#	2793	2806	2811	2845#	2859	2872	2877	2914#	2928	2941	2946	2985#
		2999	3012	3017	3058#	3090	3105	3146#	3180	3195	3591	3605	3672#	3673
		3820#	3822	3825#	3840#	3842	3845#	3863#	3866#	3874	3899	3903#	4005#	4006
		4081	4084#	4090#	4092	4095#	4121#	4123	4126#	4141#	4144#	4152	4264#	4265
		4277#	4368	4371#	4377#	4379	4382#	4398#	4400	4403#	4415#	4418#	4426	5156
		5165												
SGOADR	001124	227#												
SGOAT	001130	229#												
SGEY42	022076	3318	3321#											
SGTSMR=	***** U	4953												
SHO =	000000	12												
SHIBTS	001000	201#												
SICNT	001106	220#	4541#	4542	4544#	4555								
SILLUP	032050	4958	4974	4995#										
SINTAG	001141	234#												
SITEMB	001116	224#	4589#	4597	4617	4647								
SLF	001200	253#	4617	4783										
SLFLG	031377	4827#	4833#											
SLPADR	001110	221#	541#	680#	746#	815#	870#	927#	1000#	1077#	1137#	1198#	1265#	1334#
		1389#	1446#	1518#	1594#	1654#	1715#	1782#	1851#	1906#	1962#	2034#	2110#	2170#
		2231#	2298#	2367#	2434#	2503#	2575#	2651#	2723#	2799#	2865#	2934#	3005#	3091#
		3181#	4532#	4548#	4553	4555								
SLPERR	001112	222#	542#	4532	4549#	4555	4607							
SLPTST	001150	239#	4524											
SMAIL	001202	202	206	259#	559	4547	4595	4727						
SMBADR	001002	202#												
SMFLG	031376	4787#	4793	4828#	4832#									
SMSCAD	001216	266#	4803#	4806										
SMGLG	001220	267#	4808#											
SMSTY	001202	260#	4801	4809#	4821	4825#								
SMUL	025244	1706	1773	1842	1897	1953	2025	2101	2161	3083	3085	3175	4004#	
SMXCNT	030146	4545	4555#											
SMULL	001162	244#	4754	4783										
SMWTST=	000001	655#	721#	790#	845#	902#	975#	1052#	1112#	1173#	1240#	1309#	1364#	1421#
		1493#	1569#	1629#	1690#	1757#	1826#	1881#	1937#	2009#	2085#	2145#	2206#	2273#
		2342#	2409#	2478#	2550#	2626#	2698#	2774#	2840#	2909#	2980#	3052#	3140#	
SOCNT	031624	4872#	4901#	4914#										
SOMODE	031626	4867#	4871#	4876	4879#	4890#	4916#							
SOVER	030132	4509	4525	4533	4543	4552#								
SPASS	001210	263#	559#	619	3304#	3305#	4539	4556						
SPASTM	001006	204#												
SPOLSH	023506	668	734	803	858	915	988	1065	1125	1186	1253	1322	1377	1434
		1506	1582	1642	1703	1770	1839	1894	1950	2022	2098	2158	2219	2286
		2355	2422	2491	2563	2639	2711	2787	2853	2922	2993	3074	3164	3585#
SPOPX	023540	672	738	807	862	919	992	1069	1129	1190	1257	1326	1381	1438
		1510	1586	1646	1707	1774	1843	1898	1954	2026	2102	2162	2223	2290
		2359	2426	2495	2567	2643	2715	2791	2857	2926	2997	3086	3176	3602#
SPOWER	032056	4991	4998#											
SPUSH	023510	669	670	735	736	804	805	859	860	916	917	989	990	1066

		1067	1126	1127	1187	1188	1254	1255	1323	1324	1378	1379	1435	1436
		1507	1508	1583	1584	1643	1644	1704	1705	1771	1772	1840	1841	1895
		1896	1951	1952	2023	2024	2099	2100	2159	2160	2220	2221	2287	2288
		2356	2357	2423	2424	2492	2493	2564	2565	2640	2641	2712	2713	2788
		2789	2854	2855	2923	2924	2994	2995	3075	3076	3078	3079	3080	3084
		3165	3166	3168	3170	3171	3173	3590#						
\$PWAD	032044	4993#												
\$PWADN	031676	535	4958#	4988										
\$PWAPG	032040	4991#												
\$PWRLP	031750	4968	4974#											
\$QUES	001176	251#	4617	4783										
\$ROCHR=	*****	4954												
\$RODEC=	*****	4954												
\$ROLIN=	*****	4954												
\$ROOCT=	*****	4954												
\$R2A =	*****	4954												
\$SAVRE=	*****	4954												
\$SAVRE	032054	4967*	4975	4976*	4977*	4997#								
\$SCOPE	027672	529	4506#											
\$SETUP=	000037	514#	528	529	531	533	535	537	538	539	541	4507	4571	4605
		4612												
\$STUP =	177777	514#												
\$SUB	023562	1189	1256	1325	1380	1437	1509	1585	1645	3077	3172	3670#		
\$SVLAD	030076	4517	4546#											
\$SVPC =	000204	178#	183											
\$SMR =	167400	1#	12	248	249	250	538	539	541	542	659	725	794	849
		906	979	1056	1116	1177	1244	1313	1368	1425	1497	1573	1633	1694
		1761	1830	1885	1941	2013	2089	2149	2210	2277	2346	2413	2482	2554
		2630	2702	2778	2844	2913	2984	3056	3144	4498	4499	4500	4501	4502
		4508	4520	4522	4523	4526	4527	4528	4535	4536	4537	4549	4552	4555
		4562	4563	4564	4565	4566	4583	4590	4602	4605	4617	4994		
\$SMREG	001224	271#	562											
\$SMR#K=	000000	4502	4524											
\$TESTN	001206	262#	4547*	4697										
\$TIMES	001166	248#	538*	3245*	3294*	4535*	4542	4545*	4555					
\$TKB	001154	241#												
\$TKS	001152	240#												
\$TN =	000047	1#	12	655	659#	721	725#	790	794#	845	849#	902	906#	975
		979#	1052	1056#	1112	1116#	1173	1177#	1240	1244#	1309	1313#	1364	1368#
		1421	1425#	1493	1497#	1569	1573#	1629	1633#	1690	1694#	1757	1761#	1826
		1830#	1881	1885#	1937	1941#	2009	2013#	2085	2089#	2145	2149#	2206	2210#
		2273	2277#	2342	2346#	2409	2413#	2478	2482#	2550	2554#	2626	2630#	2698
		2702#	2774	2778#	2840	2844#	2909	2913#	2980	2984#	3052	3056#	3140	3144#
\$TPB	001160	243#	4772*	4783										
\$TPFLG	001165	247#	4721	4783										
\$TPS	001156	242#	4770	4783										
\$TRAP	031630	533	4925#											
\$TRAP2	031652	4936#	4947											
\$TRP =	000005	4940#	4949#	4950#	4951#	4952#								
\$TRPAD	031664	4930	4947#											
\$STH	001004	203#												
\$STHM	001102	218#	3247*	3293*	4524	4546*	4547	4552	4556	4582	4617			
\$STYBN=	*****	4952												
\$STYDS=	*****	4952												
\$STYPE	030652	4721#	4814	4940	4948									
\$TYPEC	031064	4751	4758	4765	4770#	4771								

U
U
U
U
U
U
U

U
U

CHPFLT	18														
COMMEN	1408														
COMM00	18														
COMM01	18														
COMM02	18														
COMM03	18														
COMM04	18														
COMM05	18														
COMM06	18														
COMM07	18														
COMM1	18														
COMM10	18														
COMM11	18														
COMM12	18														
COMM13	18														
COMM14	18														
COMM15	18														
COMM16	18														
COMM17	18														
COMM2	18														
COMM20	18														
COMM21	18														
COMM22	18														
COMM23	18														
COMM24	18														
COMM25	18														
COMM26	18														
COMM27	18														
COMM3	18														
COMM30	18														
COMM31	18														
COMM32	18														
COMM33	18														
COMM34	18														
COMM35	18														
COMM36	18														
COMM37	18														
COMM4	18														
COMM40	18														
COMM41	18														
COMM42	18														
COMM43	18														
COMM44	18														
COMM45	18														
COMM46	18														
COMM47	18														
ENDCOM	1408														
ERRCMP	18														
ERRLUR	18	4616													
ERROR	348	690	698	702	715	756	764	768	783	825	838	880	895	937	945
	949	967	1010	1018	1022	1044	1087	1105	1147	1169	1208	1216	1220	1233	1275
	1283	1287	1302	1344	1357	1399	1414	1456	1464	1468	1486	1528	1536	1540	1562
	1604	1622	1664	1686	1725	1735	1737	1750	1792	1800	1804	1819	1861	1874	1916
	1931	1972	1980	1984	2002	2044	2052	2056	2078	2120	2138	2180	2202	2241	2249
	2253	2266	2308	2316	2320	2335	2377	2385	2389	2402	2444	2452	2456	2471	2513
	2521	2525	2543	2585	2593	2597	2619	2661	2669	2673	2691	2733	2741	2745	2767

	2809	2817	2821	2834	2875	2883	2887	2902	2944	2952	2956	2974	3015	3023	3027
ESCAPE	140#														
FCOM0	1#														
FCOM1	1#														
FCOM2	1#														
FCOM3	1#														
FCOM4	1#														
FPRGTO	1#														
FPRGT1	1#														
FPSFEC	1#														
FPSTST	1#														
GENCOM	1#														
GENTS1	1#														
GENTS2	1#														
GENTS3	1#														
GENTS4	1#														
GETPRI	140#														
GETSWR	140#														
GTSTD	1#														
GTSTF	1#														
HTSTD	1#														
HTSTF	1#														
MOVDIS	1#														
MULT	140#														
NEWTST	140#	655	721	790	845	902	975	1052	1112	1173	1240	1309	1364	1421	1493
	1569	1629	1690	1757	1826	1881	1937	2009	2085	2145	2206	2273	2342	2409	2478
	2550	2626	2698	2774	2840	2909	2980	3052	3140						
POP	140#	4829	4830	4981	4982										
PUSH	140#	4790	4792	4813	4960	4966									
REPORT	140#														
SBTST1	1#														
SBTST2	1#														
SCOM0	1#														
SCOM1	1#														
SCOM2	1#														
SCOM3	1#														
SCOM4	1#														
SCOM5	1#														
SCOM6	1#														
SCOM7	1#														
SCOPE	35#	658	724	793	848	905	978	1055	1115	1176	1243	1312	1367	1424	1496
	1572	1632	1693	1760	1829	1884	1940	2012	2088	2148	2209	2276	2345	2412	2481
	2553	2629	2701	2777	2843	2912	2983	3055	3143	3244					
SCPLUR	1#	4507													
SEADAT	1#														
SETPRI	140#														
SETREG	1#	4571													
SETTRA	4940#	4949	4950	4951											
SETUP	1#	140#	520												
SKIP	140#														
SLASH	140#	577	604												
SPACE	140#														
STARS	140#	176	187	189	196	210	254	257	516	518	607	609	626	628	655
	657	721	723	790	792	845	847	902	904	975	977	1052	1054	1112	1114
	1173	1175	1240	1242	1309	1311	1364	1366	1421	1423	1493	1495	1569	1571	1629

	1631	1690	1692	1757	1759	1826	1828	1881	1883	1937	1939	2009	2011	2085	2087
	2145	2147	2206	2208	2273	2275	2342	2344	2409	2411	2478	2480	2550	2552	2626
	2628	2698	2700	2774	2776	2840	2842	2909	2911	2980	2982	3052	3054	3093	3140
	3142	3183	3241	3280	4495	4558	4617	4706	4785	4842	4919	4956	4972		
STATUS	18														
SWRSU	1408	5438													
TAD001	18														
TAD002	18														
TAD0F1	18														
TAD0F2	18														
TAD0R1	18														
TAD0R2	18														
TRMTRP	49408														
TYPBIN	1408														
TYPDEC	1408														
TYPNAM	1408														
TYPNUM	1408														
TYPOCS	1408														
TYPOCT	1408														
TYPTXT	1408														
UPCODE	18	4979													
SSCHRE	2088														
SSCHTH	2088														
SSESCA	1408														
SSNEWT	1408	655	721	790	845	902	975	1052	1112	1173	1240	1309	1364	1421	1493
	1569	1629	1690	1757	1826	1881	1937	2009	2085	2145	2206	2273	2342	2409	2478
	2550	2626	2698	2774	2840	2909	2980	3052	3140						
SSSET	49408	4949	4950	4951											
SSSETH	5598														
SSSKIP	1408														
.EQUAT	18	30													
.HEADE	18	2													
.SBPAS	18	3241													
.SETUP	18	514													
.STPAS	18	577													
.SACT1	18	174													
.SAPT8	18	2558													
.SAPTH	18	185													
.SAPTY	18	4783													
.SCATC	18	162													
.SCHTA	18	208													
.SEOP	18														
.SERRO	18	4556													
.SPOWE	18	4954													
.SSCOP	18	4493													
.STRAP	18	4917													
.STYER	18														
.STYPE	18	4704													
.STYPO	18	4840													

. ABS. 034116 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

M11

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 04-MAY-77 18:18 PAGE 129
DQFPDB.P11 04-MAY-77 17:30 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0128

DQFPDB.BIN,DQFPDB.LST/CRF/SOL/P/DOC/CPU:70/EX/EN:WRP/NL:TTM=DQFPDB.MAC,DQFPDB.P11
RUN-TIME: 19 16 1 SECONDS
RUN-TIME RATIO: 102/37=2.7
CORE USED: 25K (50 PAGES)

DOCUMENT PAGES: 128
WRAP-AROUND: 0%

USER SYMBOLS: 606
MACRO NAMES: 141
UNDF SYMBOLS: 14
DISK BLOCKS READ: 1238
DISK BLKS WRITTEN: 629
KILO CORE SECONDS: 1477