

# KD11-K

PDP11/6X FP11E FP RANDOM  
MD-11-DQFPD-A

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

APR 1977  
**digital**  
MADE IN USA

The microfiche card contains a grid of 132 frames of data, arranged in 11 rows and 12 columns. Each frame contains a small table or chart with various numerical and text-based information, typical of a data dump or diagnostic output from a computer system. The data is organized into columns and rows, with some frames containing headers and footers. The overall layout is consistent across the entire card, with each frame providing a snapshot of system data at a specific point in time.

000000

.REPT 0

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DQFPD-A-D
PRODUCT NAME:	PDP-11/6X - FP11-E FLOATING POINT UNIT ADD/SUB/MUL/DIV RANDOM OPERAND EXERCISER
DATE:	MARCH 1977
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	KEN CHAPMAN
REVISED BY:	DONALD 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.

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  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104

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

105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158

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 0-34164(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.

MAINDEC-11-DQFPD-A

PAGE 4

159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214

## 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 TYPEOUTS (WHICH IS AN "ERROR MESSAGE" RESULTING FROM AN ERROR DETECTED IN THE HARDWARE)
SW12=1	010000	INHIBIT STATUS TYPEOUTS (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)
		SET=TEST ONLY UNIT SPECIFIED IN SW00
SW00	000001	SET=SELECT WARM FP, IF SW01=1

215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270

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 PRESENCE/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,

271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316

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

## 6. 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.

317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371

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)
9	001000	FIV - FLOATING OVERFLOW INTERRUPT IF SET AND OVERFLOW, SET FER, STORE ANSWER, EXPONENT WRONG BY -400(8)
8	000400	FIC - FLOATING INTEGER CONVERSION INTERRUPT IF SET AND "STCFI" ERROR, ANSWER <-- ZERO, SET ERROR
7	000200	FD - FLOATING MODE IF CLEAR AND "STCFI" ERROR, ANSWER <-- ZERO 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 FPII-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.



372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426

## 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

427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482

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

-----  
AVERAGE EXECUTION TIME PER PASS

MODEL	SHORTEST PASS	LONGEST PASS
PDP-11/6X	20 SEC	8 MIN:30 SEC
PDP-11/6X W/FP11-E	XX SEC	X MIN:XX SEC

-----  
SEC = SECONDS / MIN = MINUTES

SHORTEST PASS ::= NO ITERATIONS, USING SWR=(004000)

LONGEST PASS ::= 2000(10) ITERATIONS/TEST, USING SWR=(000000)

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

483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538

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

539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593

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

594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649

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:

- SMXCNT - MAXIMUM NUMBER OF ITERATIONS PER TEST (GENERALLY WILL BE 2000(10))
- STSTNM - A COUNTER INDICATING THE NUMBER (1-377(8)) OF THE TEST CURRENTLY BEING EXECUTED
- SLPADR - CONTAINS THE ADDRESS TO WHICH THE SCOPE ROUTINE WILL LOOP, IF THE CURRENT TEST IS BEING LOOPED UPON
- SLPERR - 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 SLPADR, ABOVE.

### 9.3.3 ERROR ROUTINE - SERROR

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

650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705

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 TYPEOUT ROUTINE - STYPERR

THIS ROUTINE (STYPERR 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 TYPEOUT 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 TYPEOUT 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 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.

706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737

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.

.ENDR

738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793

```
.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
.*                   002000      1=BELL ON ERROR
.*          9       001000      LOOP ON ERROR
.*          8       000400      LOOP ON TEST NUMBER IN "SLPTST"
.*          1       000000      0=TEST HFP/WFP ALTERNATELY EACH PASS
.*                   000002      1=TEST ONLY UNIT SPECIFIED IN SW<00>
.*          0       000002      0=SELECT HFP, IF SW<01>=1
.*                   000003      1=SELECT WFP, IF SW<01>=1
.*
.SBTTL BASIC DEFINITIONS
.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
.*MISCELLANEOUS DEFINITIONS
MT= 11                ;;CODE FOR HORIZONTAL TAB
LF= 12                ;;CODE FOR LINE FEED
CR= 15                ;;CODE FOR CARRIAGE RETURN
CRLF= 200            ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776          ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774       ;;STACK LIMIT REGISTER
PIRQ= 177772        ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570        ;;HARDWARE SWITCH REGISTER
DDISP= 177570       ;;HARDWARE DISPLAY REGISTER
.*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0               ;;GENERAL REGISTER
R1= %1               ;;GENERAL REGISTER
R2= %2               ;;GENERAL REGISTER
R3= %3               ;;GENERAL REGISTER
R4= %4               ;;GENERAL REGISTER
R5= %5               ;;GENERAL REGISTER
R6= %6               ;;GENERAL REGISTER
```



794	000007	R7=	%7	:: GENERAL REGISTER
795	000006	SP=	%6	:: STACK POINTER
796	000007	PC=	%7	:: PROGRAM COUNTER
797				
798		:*PRIORITY LEVEL DEFINITIONS		
799	000000	PR0=	0	:: PRIORITY LEVEL 0
800	000040	PR1=	40	:: PRIORITY LEVEL 1
801	000100	PR2=	100	:: PRIORITY LEVEL 2
802	000140	PR3=	140	:: PRIORITY LEVEL 3
803	000200	PR4=	200	:: PRIORITY LEVEL 4
804	000240	PR5=	240	:: PRIORITY LEVEL 5
805	000300	PR6=	300	:: PRIORITY LEVEL 6
806	000340	PR7=	340	:: PRIORITY LEVEL 7
807				
808		:*"SWITCH REGISTER" SWITCH DEFINITIONS		
809	100000	SW15=	100000	
810	040000	SW14=	40000	
811	020000	SW13=	20000	
812	010000	SW12=	10000	
813	004000	SW11=	4000	
814	002000	SW10=	2000	
815	001000	SW09=	1000	
816	000400	SW08=	400	
817	000200	SW07=	200	
818	000100	SW06=	100	
819	000040	SW05=	40	
820	000020	SW04=	20	
821	000010	SW03=	10	
822	000004	SW02=	4	
823	000002	SW01=	2	
824	000001	SW00=	1	
825		.EQUIV	SW09, SW9	
826		.EQUIV	SW08, SW8	
827		.EQUIV	SW07, SW7	
828		.EQUIV	SW06, SW6	
829		.EQUIV	SW05, SW5	
830		.EQUIV	SW04, SW4	
831		.EQUIV	SW03, SW3	
832		.EQUIV	SW02, SW2	
833		.EQUIV	SW01, SW1	
834		.EQUIV	SW00, SW0	
835				
836		:*DATA BIT DEFINITIONS (BIT00 TO BIT15)		
837	100000	BIT15=	100000	
838	040000	BIT14=	40000	
839	020000	BIT13=	20000	
840	010000	BIT12=	10000	
841	004000	BIT11=	4000	
842	002000	BIT10=	2000	
843	001000	BIT09=	1000	
844	000400	BIT08=	400	
845	000200	BIT07=	200	
846	000100	BIT06=	100	
847	000040	BIT05=	40	
848	000020	BIT04=	20	
849	000010	BIT03=	10	

```

850      000004      BIT02= 4
851      000002      BIT01= 2
852      000001      BIT00= 1
853      .EQUIV      BIT09,BIT9
854      .EQUIV      BIT08,BIT8
855      .EQUIV      BIT07,BIT7
856      .EQUIV      BIT06,BIT6
857      .EQUIV      BIT05,BIT5
858      .EQUIV      BIT04,BIT4
859      .EQUIV      BIT03,BIT3
860      .EQUIV      BIT02,BIT2
861      .EQUIV      BIT01,BIT1
862      .EQUIV      BIT00,BIT0
863
864      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
865      000004      ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
866      000010      RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
867      000014      TBITVEC=14         ;; "T" BIT
868      000014      TRTVEC= 14         ;; TRACE TRAP
869      000014      BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
870      000020      IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
871      000024      PWRVEC= 24         ;; POWER FAIL
872      000030      EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
873      000034      TRAPVEC=34        ;; "TRAP" TRAP
874      000060      TKVEC= 60          ;; TTY KEYBOARD VECTOR
875      000064      TPVEC= 64          ;; TTY PRINTER VECTOR
876      000240      PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR
877
878      ;*MED INSTR CODES
879      076600      MED= 076600        ;OPCODE
880
881      000022      RWHAMI= 022         ;READ WHAMI
882
883      000144      RFLAG= 144          ;READ FLAGS
884      000344      WFLAG= 344         ;WRITE FLAGS
885
886      ;*FLOATING POINT INTERRUPT VECTOR
887      000244      FPPVEC= 244
888
889      ;*FLOATING POINT REGISTER DEFINITIONS
890      000000      AC0= %0
891      000001      AC1= %1
892      000002      AC2= %2
893      000003      AC3= %3
894      000004      AC4= %4
895      000005      AC5= %5
896
897
898
899      .SBTTL TRAP CATCHER
900
901      000000      .=0
902      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
903      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
904      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
905      000174      .=174

```

906 000174 000000  
 907 000176 000000  
 908  
 909 000200 000137 002734  
 910  
 911  
 912  
 913  
 914  
 915 000204  
 916 000046 000046  
 917 000046 022160  
 918 000052 000052  
 919 000052 000000  
 920 000204  
 921 001000  
 922  
 923  
 924  
 925  
 926  
 927 001000  
 928 000024 000024  
 929 000024 000200  
 930 000044 000044  
 931 000044 001000  
 932 001000  
 933  
 934  
 935  
 936  
 937 001000  
 938 001000 000000  
 939 001002 001202  
 940 001004 000776  
 941 001006 000001  
 942 001010 000000  
 943 001012 000014  
 944

```
DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0          ;; SOFTWARE SWITCH REGISTER
.SBTTL   STARTING ADDRESS(ES)
JMP      @START ;; JUMP TO STARTING ADDRESS OF PROGRAM

.SBTTL   ACT11 HOOKS

;*****
;HOOKS REQUIRED BY ACT11
$SVPCL=  ;SAVE PC
.=46    ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
$ENDAD  ;
.=52    ;;2)SET LOC.52 TO ZERO
.WORD 0 ;
.=52    ;; RESTORE PC
.=1000 ;
.SBTTL   APT PARAMETER BLOCK

;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
.$X=    ;SAVE CURRENT LOCATION
.=24    ;SET POWER FAIL TO POINT TO START OF PROGRAM
200     ;FOR APT START UP
.=44    ;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR ;POINT TO APT HEADER BLOCK
.=.$X   ;RESET LOCATION COUNTER
;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.

$APTHD:
$SHIBTS: .WORD 0          ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBAADR: .WORD $MAIL     ;; ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:   .WORD 510.      ;; RUN TIM OF LONGEST TEST
$PASTM:  .WORD 1         ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM:  .WORD 0         ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
.WORD   $ETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)
```

945  
946  
947  
948  
949  
950  
951 001100 001100  
952 001100 000000  
953 001100 000000  
954 001102 000000  
955 001104 000000  
956 001106 000000  
957 001110 000000  
958 001112 000000  
959 001114 000000  
960 001116 000000  
961 001120 000000  
962 001122 000001  
963 001124 000000  
964 001126 000000  
965 001130 000000  
966 001132 000000  
967 001134 000000  
968 001136 000000  
969 001140 000000  
970 001142 000  
971 001143 000  
972 001144 000000  
973 001146 177570  
974 001150 177570  
975 001152 177560  
976 001154 177562  
977 001156 177564  
978 001160 177566  
979 001162 000  
980 001163 002  
981 001164 012  
982 001165 000  
983 001166 000000  
984 001170 000000  
985 001172 177607 000377  
986 001176 077  
987 001177 015  
988 001200 000012  
989  
990  
991  
992  
993  
994 001202  
995 001202 000000  
996 001204 000000  
997 001206 000000  
998 001210 000000  
999 001212 000000  
1000 001214 000000

.SBTTL COMMON TAGS

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

```

.=1100
SCMTAG:          .WORD          0          ;; START OF COMMON TAGS
                .WORD          0          ;; CONTAINS THE TEST NUMBER
STSTNM:          .WORD          0          ;; CONTAINS ERROR FLAG
SERFLG:          .WORD          0          ;; CONTAINS SUBTEST ITERATION COUNT
SICNT:           .WORD          0          ;; CONTAINS SCOPE LOOP ADDRESS
SLPADR:          .WORD          0          ;; CONTAINS TEST NUMBER TO LOOP UPON
SLPTST:          .WORD          0          ;; CONTAINS SCOPE RETURN FOR ERRORS
SLPERR:          .WORD          0          ;; CONTAINS TOTAL ERRORS DETECTED
SERTTL:          .WORD          0          ;; CONTAINS ITEM CONTROL BYTE
SITEMB:          .WORD          0          ;; CONTAINS MAX. ERRORS PER TEST
SERMAX:          .WORD          1          ;; CONTAINS PC OF LAST ERROR INSTRUCTION
SERRPC:          .WORD          0          ;; CONTAINS ADDRESS OF 'GOOD' DATA
SGDADR:          .WORD          0          ;; CONTAINS ADDRESS OF 'BAD' DATA
SBDADR:          .WORD          0          ;; CONTAINS 'GOOD' DATA
SGDDAT:          .WORD          0          ;; CONTAINS 'BAD' DATA
SBDDAT:          .WORD          0          ;; RESERVED--NOT TO BE USED
                .WORD          0          ;;
SAUTOB:          .BYTE          0          ;; AUTOMATIC MODE INDICATOR
SINTAG:          .BYTE          0          ;; INTERRUPT MODE INDICATOR
                .WORD          0          ;;
SWR:             .WORD          DSWR      ;; ADDRESS OF SWITCH REGISTER
DISPLAY:         .WORD          DDISP    ;; ADDRESS OF DISPLAY REGISTER
$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"
$STPFLG:        .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:          .WORD          AMSGTY    ;; APT MAILBOX
$MSGTY:         .WORD          AFATAL    ;; MESSAGE TYPE CODE
$FATAL:         .WORD          ATESTN    ;; FATAL ERROR NUMBER
$TESTN:         .WORD          APASS     ;; TEST NUMBER
$PASS:          .WORD          ADEVCT    ;; PASS COUNT
$DEVCT:         .WORD          AUNIT     ;; DEVICE COUNT
$UNIT:          .WORD                   ;; I/O UNIT NUMBER

```

H02

1001 001216 000000  
1002 001220 000000  
1003 001222  
1004 001222 000  
1005 001223 000  
1006 001224 000000  
1007 001226 000000  
1008 001230 000000  
1009  
1010  
1011  
1012  
1013  
1014  
1015 001232  
1016

\$MSGAD: .WORD    AMSGAD  
\$MSGLG: .WORD    AMSGLG  
\$ETABLE:  
\$ENV: .BYTE      AENV  
\$ENVM: .BYTE     AENVM  
\$SWREG: .WORD    ASWREG  
\$USWR: .WORD     AUSWR  
\$CPUOP: .WORD    ACPUOP  
;\*  
;\*  
;\*  
;\*  
;\*  
;\*  
\$ETEND:  
.MEXIT

:: MESSAGE ADDRESS  
:: MESSAGE LENGTH  
:: APT ENVIRONMENT TABLE  
:: ENVIRONMENT BYTE  
:: ENVIRONMENT MODE BITS  
:: APT SWITCH REGISTER  
:: USER SWITCHES  
:: CPU TYPE, OPTIONS  
BITS 15-11=CPU TYPE  
      11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05  
      11/70=06, PDQ=07, Q=10  
BIT 10=REAL TIME CLOCK  
BIT 9=FLOATING POINT PROCESSOR  
BIT 8=MEMORY MANAGEMENT

1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
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

001232

.SBTTL ERROR POINTER TABLE

\$ERRTB:

;\*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
;\*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
;\*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
;\*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).  
;\*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;\* EM ;;POINTS TO THE ERROR MESSAGE  
;\* DH ;;POINTS TO THE DATA HEADER  
;\* DT ;;POINTS TO THE DATA  
;\* DF ;;POINTS TO THE DATA FORMAT  
;\*NOTE: ERROR VECTOR TABLE (\$ERRTB) HAS BEEN MODIFIED,  
;\* SEE \$ERRTYP ROUTINE FOR ITS STRUCTURE

001232	032574	033254	033544	EMV001: .WORD	EMJ,DHA,DTA,0,0,0,0,0,0,0	;ADDF *
001240	000000	000000	000000			
001246	000000	000000	000000			
001254	000000					
001256	032615	033254	033544	EMV002: .WORD	EMK,DHA,DTA,0,0,0,0,0,0,0	;SUBF * FPS ERRORS
001264	000000	000000	000000			
001272	000000	000000	000000			
001300	000000					
001302	032636	033254	033544	EMV003: .WORD	EML,DHA,DTA,0,0,0,0,0,0,0	;MULF *
001310	000000	000000	000000			
001316	000000	000000	000000			
001324	000000					
001326	000000	000000	000000	EMV004: .WORD	0,0,0,0,0,0,0,0,0,0	;(UNUSED)
001334	000000	000000	000000			
001342	000000	000000	000000			
001350	000000					
001352	032345	033270	033564	EMV005: .WORD	EME,DHB,DTD,LOF,HIF,0,0,0,0,0	;ADDF *
001360	033672	033702	000000			
001366	000000	000000	000000			
001374	000000					
001376	032345	033326	033576	EMV006: .WORD	EME,DHC,DTE,LOD,HID,0,0,0,0,0	;ADDD *
001404	033642	033656	000000			
001412	000000	000000	000000			
001420	000000					
001422	032375	033270	033564	EMV007: .WORD	EMF,DHB,DTD,LOF,HIF,0,0,0,0,0	;SUBF *
001430	033672	033702	000000			
001436	000000	000000	000000			
001444	000000					
001446	032375	033326	033576	EMV010: .WORD	EMF,DHC,DTE,LOD,HID,0,0,0,0,0	;SUBD * RESULT ERRORS
001454	033642	033656	000000			
001462	000000	000000	000000			
001470	000000					
001472	032425	033270	033564	EMV011: .WORD	EMG,DHB,DTD,LOF,HIF,0,0,0,0,0	;MULF *
001500	033672	033702	000000			
001506	000000	000000	000000			
001514	000000					
001516	032425	033326	033576	EMV012: .WORD	EMG,DHC,DTE,LOD,HID,0,0,0,0,0	;MULD *
001524	033642	033656	000000			

1073	001532	000000	000000	000000					
1074	001540	000000							
1075	001542	032455	033270	033564	EMV013: .WORD	EMH,DHB,DTD,LOF,HIF,0,0,0,0,0	;DIVF	*	
1076	001550	033672	033702	000000					
1077	001556	000000	000000	000000					
1078	001564	000000							
1079	001566	032455	033326	033576	EMV014: .WORD	EMH,DHC,DTE,LOD,HID,0,0,0,0,0	;DIVD	*	
1080	001574	033642	033656	000000					
1081	001602	000000	000000	000000					
1082	001610	000000							
1083									
1084	001612	032505	033424	033620	EMV015: .WORD	EMI,DHD,DTF,0,0,0,0,0,0,0	;ILLEGAL FPP TRAP		
1085	001620	000000	000000	000000					
1086	001626	000000	000000	000000					
1087	001634	000000							
1088									
1089	001636	032657	033254	033544	EMV016: .WORD	EMM,DHA,DTA,0,0,0,0,0,0,0	;DIVF	*	
1090	001644	000000	000000	000000					
1091	001652	000000	000000	000000					
1092	001660	000000							
1093	001662	032700	033254	033544	EMV017: .WORD	EMN,DHA,DTA,0,0,0,0,0,0,0	;ADD	*	
1094	001670	000000	000000	000000					
1095	001676	000000	000000	000000					
1096	001704	000000							
1097	001706	032721	033254	033544	EMV020: .WORD	EMO,DHA,DTA,0,0,0,0,0,0,0	;SUBD	* FPS ERRORS	
1098	001714	000000	000000	000000					
1099	001722	000000	000000	000000					
1100	001730	000000							
1101	001732	032742	033254	033544	EMV021: .WORD	EMP,DHA,DTA,0,0,0,0,0,0,0	;MULD	*	
1102	001740	000000	000000	000000					
1103	001746	000000	000000	000000					
1104	001754	000000							
1105	001756	032763	033254	033544	EMV022: .WORD	EMQ,DHA,DTA,0,0,0,0,0,0,0	;DIVD	*	
1106	001764	000000	000000	000000					
1107	001772	000000	000000	000000					
1108	002000	000000							
1109									
1110	002002	033004	033503	033552	EMV023: .WORD	EMR,DHE,DTB,0,0,0,0,0,0,0	;ADDF	*	
1111	002010	000000	000000	000000					
1112	002016	000000	000000	000000					
1113	002024	000000							
1114	002026	033031	033503	033552	EMV024: .WORD	EMS,DHE,DTB,0,0,0,0,0,0,0	;SUBF	*	
1115	002034	000000	000000	000000					
1116	002042	000000	000000	000000					
1117	002050	000000							
1118	002052	033056	033503	033552	EMV025: .WORD	EMT,DHE,DTB,0,0,0,0,0,0,0	;MULF	*	
1119	002060	000000	000000	000000					
1120	002066	000000	000000	000000					
1121	002074	000000							
1122	002076	033103	033503	033552	EMV026: .WORD	EMU,DHE,DTB,0,0,0,0,0,0,0	;DIVF	* FEC/FEA ERRORS	
1123	002104	000000	000000	000000					
1124	002112	000000	000000	000000					
1125	002120	000000							
1126	002122	033130	033503	033552	EMV027: .WORD	EMV,DHE,DTB,0,0,0,0,0,0,0	;ADD	*	
1127	002130	000000	000000	000000					
1128	002136	000000	000000	000000					

1129	002144	000000									
1130	002146	033155	033503	033552	EMV030: .WORD	EMW,DHE,DTB,0,0,0,0,0,0,0		;SUBD	*		
1131	002154	000000	000000	000000							
1132	002162	000000	000000	000000							
1133	002170	000000									
1134	002172	033202	033503	033552	EMV031: .WORD	EMX,DHE,DTB,0,0,0,0,0,0,0		;MULD	*		
1135	002200	000000	000000	000000							
1136	002206	000000	000000	000000							
1137	002214	000000									
1138	002216	033227	033503	033552	EMV032: .WORD	EMY,DHE,DTB,0,0,0,0,0,0,0		;DIVD	*		
1139	002224	000000	000000	000000							
1140	002232	000000	000000	000000							
1141	002240	000000									
1142											
1143	002242	032153	033254	033544	EMV033: .WORD	EMA,DHA,DTA,0,0,0,0,0,0,0		;F-MODE EXERCISER	*	FPS	
1144	002250	000000	000000	000000							
1145	002256	000000	000000	000000							
1146	002264	000000									
1147	002266	032210	033254	033544	EMV034: .WORD	EMB,DHA,DTA,0,0,0,0,0,0,0		;D-MODE EXERCISER	*	ERROR	
1148	002274	000000	000000	000000							
1149	002302	000000	000000	000000							
1150	002310	000000									
1151											
1152	002312	032245	033270	033564	EMV035: .WORD	EMC,DHB,DTD,OP1F,OP2F,OP3F,OP4F,OP5F,OP6F,0		;F-MODE EXERCISER	*	RE	
1153	002320	033712	033722	033732							
1154	002326	033742	033752	033762							
1155	002334	000000									
1156	002336	032305	033326	033576	EMV036: .WORD	EMD,DHC,DTE,OP1D,OP2D,OP3D,OP4D,OP5D,OP6D,0		;D-MODE EXERCISER	*		
1157	002344	033772	034006	034022							
1158	002352	034036	034052	034066							
1159	002360	000000									
1160											
1161											
1162											
1163											
1164											
1165	002362	000000									
1166	002364	000000									
1167	002366	000000									
1168	002370	000000									
1169	002372	000000									
1170	002374	000000									
1171	002376	000000									
1172	002400	000000									
1173	002402	000000									
1174	002404	000000									
1175	002406	000000	000000	000000	ANS1: .WORD	0,0,0,0		;HARDWARE FLOATING POINT ANSWER			
1176	002414	000000									
1177	002416	000000	000000	000000	ANS2: .WORD	0,0,0,0		;SOFTWARE FLOATING POINT ANSWER			
1178	002424	000000									
1179	002426	027005	104552	111730	LONUM: .WORD	027005,104552,111730,000555		;RANDOM EXERCISER OPERAND LONUM			
1180	002434	000555									
1181	002436	102337	166330	007025	HINUM: .WORD	102337,166330,007025,021553		;RANDOM EXERCISER OPERAND HINUM			
1182	002444	021553									
1183	002446	175463	030712	105726	OP1: .WORD	175463,030712,105726,124064					
1184	002454	124064									

.SBTTL PROGRAM DEFINED COMMON TAGS

:\*VARIABLES

FPS: .WORD 0 ;FPS STORED HERE AFTER STFPS  
 FEC: .WORD 0 ;FEC STORED HERE AFTER STST  
 FEA: .WORD 0 ;FEA STORED HERE AFTER STST  
 FPPOPC: .WORD 0 ;OLD PC SAVED HERE AFTER TRAP  
 FPPOPS: .WORD 0 ;OLD PS SAVED HERE AFTER TRAP  
 FPPOSP: .WORD 0 ;SP AFTER TRAP  
 EXPFEA: .WORD 0 ;EXPECTED FEA  
 SFPS: .WORD 0 ;SOFTWARE FPS  
 SFEC: .WORD 0 ;SOFTWARE FEC  
 SFEA: .WORD 0 ;SOFTWARE FEA  
 ANS1: .WORD 0,0,0,0 ;HARDWARE FLOATING POINT ANSWER

;SOFTWARE FLOATING POINT ANSWER



```

1185 002456 156607 002361 070707 OP2: .WORD 156607,002361,070707,061111 ;
1186 002454 061111
1187 002466 003505 134261 062451 OP3: .WORD 003505,134261,062451,052525 ;EXTRA RANDOM OPERANDS
1188 002474 052525
1189 002476 102466 123456 111111 OP4: .WORD 102466,123456,111111,101010 ;
1190 002504 101010
1191 002506 076103 000346 060612 OP5: .WORD 076103,000346,060612,125252 ;
1192 002514 125252
1193 002516 044100 112400 177777 OP6: .WORD 044100,112400,177777,000006 ;
1194 002524 000006
1195
1196

```

```

1197 002526 000000 ;*ADDED COUNTERS FOR TRACING WHERE WE'VE BEEN
1198 002530 000000 ADDC0: .WORD 0 ;TOTAL NUMBER *
1199 002532 000000 ADDC1: .WORD 0 ;A=0, B#0 *
1200 002534 000000 ADDC2: .WORD 0 ;A#0, B=0 *
1201 002536 000000 ADDC3: .WORD 0 ;A=0, B=0 * FOR
1202 002540 000000 ADDC4: .WORD 0 ;NO SHIFT REQ'D * ADD
1203 002542 000000 ADDC5: .WORD 0 ;OVERFLOW * AND
1204 002544 000000 ADDC6: .WORD 0 ;OVERFLOW, ENABLED * SUBTRACT
1205 002546 000000 ADDC7: .WORD 0 ;UNDERFLOW *
1206 ADDC8: .WORD 0 ;UNDERFLOW, ENABLED *
1207 002550 000000 MULC0: .WORD 0 ;TOTAL NUMBER *
1208 002552 000000 MULC1: .WORD 0 ;A=0 AND/OR B=0 *
1209 002554 000000 MULC2: .WORD 0 ;OVERFLOW * FOR
1210 002556 000000 MULC3: .WORD 0 ;OVERFLOW, ENABLED * MULTIPLY
1211 002560 000000 MULC4: .WORD 0 ;UNDERFLOW *
1212 002562 000000 MULC5: .WORD 0 ;UNDERFLOW, ENABLED *
1213
1214 002564 000000 DIVC0: .WORD 0 ;TOTAL NUMBER *
1215 002566 000000 DIVC1: .WORD 0 ;NUM A=0 *
1216 002570 000000 DIVC2: .WORD 0 ;DEN B=0 * FOR
1217 002572 000000 DIVC3: .WORD 0 ;OVERFLOW * DIVIDE
1218 002574 000000 DIVC4: .WORD 0 ;OVERFLOW, ENABLED *
1219 002576 000000 DIVC5: .WORD 0 ;UNDERFLOW *
1220 002600 000000 DIVC6: .WORD 0 ;UNDERFLOW, ENABLED *
1221
1222

```

```

1223 002602 000000 ;*REGISTER CONTENTS, AT ERROR, STORED HERE
1224 002604 000000 EREG0: .WORD 0
1225 002606 000000 EREG1: .WORD 0
1226 002610 000000 EREG2: .WORD 0
1227 002612 000000 EREG3: .WORD 0
1228 002614 000000 EREG4: .WORD 0
1229 002616 000000 EREG5: .WORD 0
1230 002620 000000 EREG6: .WORD 0
1231 EREG7: .WORD 0
1232

```

```

1233 002622 005015 005012 042115 ;*MESSAGES FOR BEGIN PROGRAM/START OF PASS
1234 002630 030455 026461 050504 BGNMES: .ASCII <15><12><12><12>"MD-11-DQFPD-A..."
1235 002636 050106 026504 027101
1236 002644 027056
1237 002646 042120 026520 030461 .ASCIZ "PDP-11/6X F.P.U. ADD/SUB/MUL/DIV EXERCISER"<15><12>
1238 002654 033057 020130 027106
1239 002662 027120 027125 040440
1240 002670 042104 051457 041125

```

M02

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 27  
DQFPDA.P11 09-FEB-77 10:32 PROGRAM DEFINED COMMON TAGS

1241	002676	046457	046125	042057	
1242	002704	053111	042440	042530	
1243	002712	041522	051511	051105	
1244	002720	005015	000		
1245	002723	015	050012	051501	NWPAS1: .ASCIZ <15><12>"PASS #"
1246	002730	020123	000043		

```

1247 .SBTTL START OF PASS ROUTINE
1248
1249 .EVEN ;START ON AN EVEN BOUNDARY
1250
1251 ;*****
1252 .ENABL AMA ; ASSEMBLE ALL RELATIVE REFERENCES AS ABSOLUTE
1253 ;*****
1254
1255 002734 START:
1256 .SBTTL INITIALIZE THE COMMON TAGS
1257 ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
1258 002734 012706 001100 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
1259 002740 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
1260 002742 022706 001146 CMP #SWR,R6 ;;DONE?
1261 002746 001374 BNE -6 ;;LOOP BACK IF NO
1262 002750 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
1263 ;;INITIALIZE A FEW VECTORS
1264 002754 012737 027742 000020 MOV #SSCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1265 002762 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
1266 002770 012737 030220 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1267 002776 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
1268 003004 012737 031700 000034 MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1269 003012 012737 000340 000036 MOV #340,@TRAPVEC+2;LEVEL 7
1270 003020 012737 031746 000024 MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
1271 003026 012737 000340 000026 MOV #340,@PWRVEC+2 ;;LEVEL 7
1272 003034 013737 022114 022106 MOV SENDCT,SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
1273 003042 005037 001166 CLR STIMES ;;INITIALIZE NUMBER OF ITERATIONS
1274 003046 005037 001170 CLR SESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1275 003052 012737 000001 001122 MOV #1,SERMAX ;;ALLOW ONE ERROR PER TEST
1276 003060 012737 003060 001110 MOV #.,SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1277 003066 012737 003066 001114 MOV #.,SLPERR ;;SETUP THE ERROR LOOP ADDRESS
1278 ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
1279 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1280 003074 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
1281 003100 012737 003134 000004 MOV #64$,@ERRVEC ;;SET UP ERROR VECTOR
1282 003106 012737 177570 001146 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
1283 003114 012737 177570 001150 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1284 003122 022777 177777 176016 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
1285 003130 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1286 ;;AND THE HARDWARE SWR IS NOT = -1
1287 003132 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
1288 003134 012716 003142 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
1289 003140 000002 RTI
1290 003142 012737 000176 001146 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
1291 003150 012737 000174 001150 MOV #DISPRG,DISPLAY
1292 003156 012637 000004 66$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
1293
1294 003162 005037 001210 CLR $PASS ;;CLEAR PASS COUNT
1295 003166 132737 000200 001223 BITB #APTSIZE,SENVN ;;TEST USER SIZE UNDER APT
1296 003174 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
1297 003176 012737 001224 001146 MOV #SSWREG,SWR ;;NO,USE APT SWITCH REGISTER
1298 003204 67$:
1299
1300 .SET UP FPP UNEXPECTED TRAP CATCHER - - - - -
1301 003204 012737 023330 000244 MOV #FPPILT,@FPPVEC ;;NEW PC AT FPP TRAP
1302 003212 005037 000246 CLR @FPPVEC+2 ;;NEW PS AT FPP TRAP

```

```

1303
1304
1305 003216 012700 002526
1306 003222 012701 000026
1307 003226 005020
1308 003230 077102
1309
1310 003232 104401 002622
1311
1312
1313
1314
1315 003236 076600 000022
1316 003242 032700 000020
1317 003246 001403
1318 003250 104401 003264
1319 003254 000453
1320 003256 104401 003324
1321 003262 000450
1322
1323 003264 005015 020052 050106 68$: .ASCIZ <15><12>*" FP11-E HFP UNIT PRESENT *"<15><12>
1324 003272 030461 042455 044040
1325 003300 050106 052440 044516
1326 003306 020124 051120 051505
1327 003314 047105 020124 006452
1328 003322 000012
1329 003324 005015 020052 047516 69$: .ASCIZ <15><12>*" NO FP11-E HFP UNIT - ALL TESTS WFP ONLY *"<15><12>
1330 003332 043040 030520 026461
1331 003340 020105 043110 020120
1332 003346 047125 052111 026440
1333 003354 040440 046114 052040
1334 003362 051505 051524 053440
1335 003370 050106 047440 046116
1336 003376 020131 006452 000012
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346 003404 012706 001100 NEWPAS: MOV #STACK,SP ;RESET STACK PTR
1347
1348 003410 032777 010000 175530 BIT #BIT12,JSWR ;INHIBIT STATUS TYPEOUTS ?
1349 003416 001015 BNE SUBPAS ;BR IF YES
1350 003420 023737 022114 022106 CMP $ENDCT,$EOPCT ;TIME FOR A MESSAGE ?
1351 003426 001011 BNE SUBPAS ;NO, NOT YET
1352
1353 003430 104401 002723 TYPE NWPAS1 ;"PASS #"
1354 003434 013746 001210 MOV $PASS,-(SP) ;PASS COUNT INTO ...
1355 003440 005216 INC (SP) ; 1-N RANGE
1356 003442 104403 TYPOS ;TYPE OCTAL
1357 003444 006 000 .BYTE 6,0 ; 6 DIGITS, NO LEADING ZEROS
1358 003446 104401 001177 TYPE ,$CRLF ;END THE LINE

```

```

1359
1360
1361 ;*****
1362 ;NEW SUBPASS ENTERS HERE
1363 ;*****
1364
1365 003452 076600 000022 SUBPAS: MED RWHAMI ;GET WHAMI INTO RO
1366 003456 032700 000020 BIT #BIT04,RO ;1=HFP PRESENT, 0=NO
1367 003462 001430 BEQ 20$ ;IF NO HFP, TEST WARM ONLY
1368
1369 003464 076600 000144 MED ,RFLAG ;GET FLAGS INTO RO
1370
1371 003470 032777 000002 175450 BIT #SW01,2SWR ;SW01: 1=HFP OR WFP TEST ONLY
1372 003476 001413 BEQ 1$ ; 0=ALTERNATE HFP/WFP PER PASS
1373
1374 003500 032777 000001 175440 BIT #SW00,2SWR ;SW00: 1=WFP ONLY
1375 003506 001403 BEQ 2$ ; 0=HFP ONLY
1376 003510 042700 010000 BIC #BIT12,RO ;CLEAR HFP ENABLE FLAG<5> FOR WFP
1377 003514 000402 BR 3$ ;
1378 003516 052700 010000 2$: BIS #BIT12,RO ;SET HFP ENABLE FLAG<5> FOR HFP
1379 003522 076600 000344 3$: MED ,WFLAG ;REWRITE FLAGS
1380
1381 003526 032700 010000 1$: BIT #BIT12,RO ;TEST WHO'S ENABLED: HOT, WARM
1382 003532 001404 BEQ 20$ ;SET APPROPRIATE HEADER:
1383
1384 003534 012737 032136 030464 19$: MOV #ASCHOT,HOTWRM ;"HOT: "
1385 003542 009403 BR 21$ ;
1386 003544 012737 032144 030464 20$: MOV #ASCWRM,HOTWRM ;"WARM: "
1387 003552 005037 001102 21$: CLR $TSTNM ;ALL DONE, RESET TEST NUMBER COUNTER

```

```

1388
1389
1390
1391 003556 000004
1392 003560 012737 003712 002376
1393 003566 012737 007400 002400
1394 003574 005037 002402
1395 003600 005037 002404
1396 003604 005037 002362
1397 003610 005037 002364
1398 003614 005037 002366
1399 003620 004737 023414
1400 003624 002426 002436
1401 003630 004437 023560
1402 003634 023562 002426
1403 003640 023562 002436
1404 003644 023640
1405 003646 023612 002416
1406
1407 003652 013700 002400
1408 003656 170127 040000
1409 003662 172437 002426
1410 003666 172537 002436
1411 003672 172737 002416
1412 003676 170127 007400
1413 003702 012737 003710 001110
1414
1415
1416
1417 003710 172600
1418 003712 172201
1419 003714 170237 002362
1420 003720 023737 002362 002400
1421 003726 001403
1422 003730 174237 002406
1423 003734 104001
1424
1425 003736 005737 002400
1426 003742 100014
1427 003744 170337 002364
1428
1429 003750 023737 002364 002402
1430 003756 001401
1431 003760 104023
1432
1433 003762 023737 002366 002404
1434 003770 001401
1435 003772 104023
1436
1437 003774 173702
1438 003776 170000
1439 004000 001416
1440
1441 004002 174237 002406
1442 004006 162737 000001 002410
1443 004014 005637 002406

```

```

;*****
;TEST 1 EXERCISE ADDF, ALL INTERRUPTS ON, ROUNDING MODE
;*****
↑ST1: SCOPE
MOV #ARET1,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007400,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL2 ;GET RANDOM INPUT DATA
.WORD LONUM,HINUM
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
$ADD ;ADDRESS OF FORTRAN ADD
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

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

;*****
ARET1: LDF AC0,AC2 ;LOAD AC0 INTO AC2
ADDF 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

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

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

```

E03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006)  
DQFPDA.P11 09-FEB-77 10:32 T1

09-FEB-77 10:34 PAGE 32  
EXERCISE ADDF, ALL INTERRUPTS ON, ROUNDING MODE

1444 004020 173737 002406  
1445 004024 170000  
1446 004026 001403  
1447 004030 174237 002406  
1448 004034 104005  
1449  
1450 004036 005037 002362  
1451

CMPF ANS1,AC3  
CFCC  
BEQ AEND1  
STF AC2,ANS1  
ERROR 5  
  
AEND1: CLR FPS

;CHECK ANSWERS AGAIN  
;COPY FLOATING CONDITION CODES  
;BRANCH IF OK  
;SAVE FPU ANSWER  
;FPU AND FORTRAN DISAGREE  
  
;CLR FPU FPS BUFFER

Address	OpCode	Op1	Op2	Op3	Op4	Comment
1452						*****
1453						TEST 2 EXERCISE ADD, ALL INTERRUPTS ON, ROUNDING MODE
1454						*****
1455	004042	000004				↑ST2: SCOPE
1456	004044	012737	004176	002376		MOV #ARET2,EXPFEA ; ADDR OF INSTR BEING TESTED
1457	004052	012737	007600	002400		MOV #007600,\$FPS ; SET IE BITS IN FORTRAN ANSWER
1458	004060	005037	002402			CLR \$FEC ; CLR FORTRAN FEC
1459	004064	005037	002404			CLR \$FEA ; CLR FORTRAN FEA
1460	004070	005037	002362			CLR FPS ; CLR FPU FPS BUFFER
1461	004074	005037	002364			CLR FEC ; CLR FPU FEC BUFFER
1462	004100	005037	002366			CLR FEA ; CLR FPU FEA BUFFER
1463	004104	004737	023404			JSR PC,RANDL4 ; GET RANDOM INPUT DATA
1464	004110	002426	002436			.WORD LONUM,HINUM
1465	004114	004437	023560			JSR R4,\$POLSH ; ENTER POLISH MODE
1466	004120	023562	002426			\$PUSH ,LONUM ; PUSH 4 WORDS ON STACK (LONUM)
1467	004124	023562	002436			\$PUSH ,HINUM ; PUSH 4 WORDS ON STACK (HINUM)
1468	004130	023640				\$ADD ; ADDRESS OF FORTRAN ADD
1469	004132	023612	002416			\$POPX ,ANS2 ; POP 4 WORDS AND EXIT POLISH MODE
1470						
1471	004136	013700	002400			MOV \$FPS,R0 ; DISPLAY FLOATING POINT STATUS
1472	004142	170127	040200			LDFPS #040200 ; LOAD FPS, INTERRUPT DISABLE AND FD
1473	004146	172437	002426			LDD LONUM,AC0 ; LOAD AC0 WITH A RANDOM NUMBER
1474	004152	172537	002436			LDD HINUM,AC1 ; LOAD AC1 WITH A RANDOM NUMBER
1475	004156	172737	002416			LDD ANS2,AC3 ; LOAD AC3 WITH THE SUM
1476	004162	170127	007600			LDFPS #007600 ; TURN INTERRUPTS ON
1477	004166	012737	004174	001110		MOV #.+6,\$LPADR ; RESET LOOP ADDRESS
1478						
1479						*****
1480						
1481	004174	172600				LDD AC0,AC2 ; LOAD AC0 INTO AC2
1482	004176	172201			ARET2:	ADD AC1,AC2 ; ADD AC1 BY AC2
1483	004200	170237	002362			STFPS FPS ; STORE FLOATING POINT STATUS
1484	004204	023737	002362	002400		CMP FPS,\$FPS ; CHECK FPS
1485	004212	001403				BEQ AERR2 ; BRANCH IF OK
1486	004214	174237	002406			STD AC2,ANS1 ; SAVE FPU ANSWER
1487	004220	104017				ERROR 17 ; FPS ERROR
1488						
1489	004222	005737	002400		AERR2:	TST \$FPS ; ERROR BIT SET ?
1490	004226	100014				BPL ATST2 ; NO, DONT GET FEC/FEA
1491	004230	170337	002364			STST FEC ; YES, CHECK STATUS
1492						
1493	004234	023737	002364	002402		CMP FEC,\$FEC ; CHECK THE FLOATING EXCEPTION CODES
1494	004242	001401				BEQ 1\$ ; BRANCH IF OK
1495	004244	104027				ERROR 27 ; FEC IS WRONG
1496						
1497	004246	023737	002366	002404	1\$:	CMP FEA,\$FEA ; CHECK FLOATING PC
1498	004254	001401				BEQ ATST2 ; BRANCH IF OK
1499	004256	104027				ERROR 27 ; WRONG ADDRESS IN FEA
1500						
1501	004260	173702			ATST2:	CMPC AC2,AC3 ; COMPARE FPU ANSWER TO FORTRAN ANSWER
1502	004262	170000				CFCC ; COPY FLOATING CONDITION CODES
1503	004264	001422				BEQ AEND2 ; ANSWERS CHECK
1504						; COMPENSATE FOR FORTRAN INACCURACIES.
1505	004266	174237	002406			STD AC2,ANS1 ; SAVE FPU ANSWER
1506	004272	162737	000001	002414		SUB #1,ANS1+6 ; DECREMENT FPU ANSWER
1507	004300	005637	002412			SBC ANS1+4



G03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 34  
DQFPDA.P11 09-FEB-77 10:32 T2 EXERCISE ADD, ALL INTERRUPTS ON, ROUNDING MODE

1508 004304 005637 002410  
1509 004310 005637 002406  
1510 004314 173737 002406  
1511 004320 170000  
1512 004322 001403  
1513 004324 174237 002406  
1514 004330 104006  
1515  
1516 004332 005037 002362  
1517

SBC ANS1+2  
SBC ANS1  
CMPD ANS1, AC3  
CFCC  
BEQ AEND2  
STD AC2, ANS1  
ERROR 6

AEND2: CLR FPS

;CHECK ANSWERS AGAIN  
;COPY FLOATING CONDITION CODES  
;BRANCH IF OK  
;SAVE FPU ANSWER  
;FPU AND FORTRAN DISAGREE

;CLR FPU FPS BUFFER

# H03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 35  
 DQFPDA.P11 09-FEB-77 10:32 T3 EXERCISE ADDF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE

```

1518
1519
1520
1521 004336 000004
1522 004340 012737 004472 002376
1523 004346 012737 004400 002400
1524 004354 005037 002402
1525 004360 005037 002404
1526 004364 005037 002362
1527 004370 005037 002364
1528 004374 005037 002366
1529 004400 004737 023414
1530 004404 002426 002436
1531 004410 004437 023560
1532 004414 023562 002426
1533 004420 023562 002436
1534 004424 023640
1535 004426 023612 002416
1536
1537 004432 013700 002400
1538 004436 170127 040000
1539 004442 172437 002426
1540 004446 172537 002436
1541 004452 172737 002416
1542 004456 170127 004400
1543 004462 012737 004470 001110
1544
1545
1546
1547 004470 172600
1548 004472 172201
1549 004474 170237 002362 002400
1550 004500 023737 002362 002400
1551 004506 001403
1552 004510 174237 002406
1553 004514 104001
1554
1555 004516 173702
1556 004520 170000
1557 004522 001416
1558
1559 004524 174237 002406
1560 004530 162737 000001 002410
1561 004536 005637 002406
1562 004542 173737 002406
1563 004546 170000
1564 004550 001403
1565 004552 174237 002406
1566 004556 104005
1567
1568 004560 005037 002362
1569
;*****
;TEST 3 EXERCISE ADDF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
;*****
TST3: SCOPE
MOV #ARET3,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)
$ADD ;ADDRESS OF FORTRAN ADD
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

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

;*****
ARET3: LDF AC0,AC2 ;LOAD AC0 INTO AC2
ADDF AC1,AC2 ;ADD AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ ATST3 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 1 ;FPS ERROR

ATST3: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND3 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1
CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND3 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 5 ;FPU AND FORTRAN DISAGREE

AEND3: CLR FPS ;CLR FPU FPS BUFFER
  
```

```

1570
1571
1572
1573 004564 000004
1574 004566 012737 004720 002376
1575 004574 012737 004600 002400
1576 004602 005037 002402
1577 004606 005037 002404
1578 004612 005037 002362
1579 004616 005037 002364
1580 004622 005037 002366
1581 004626 004737 023404
1582 004632 002426 002436
1583 004636 004437 023560
1584 004642 023562 002426
1585 004646 023562 002436
1586 004652 023640
1587 004654 023612 002416
1588
1589 004660 013700 002400
1590 004664 170127 040200
1591 004670 172437 002426
1592 004674 172537 002436
1593 004700 172737 002416
1594 004704 170127 004600
1595 004710 012737 004716 001110
1596
1597
1598
1599 004716 172600
1600 004720 172201
1601 004722 170237 002362 002400
1602 004726 023737 002362
1603 004734 001403
1604 004736 174237 002406
1605 004742 104017
1606
1607 004744 173702
1608 004746 170000
1609 004750 001422
1610
1611 004752 174237 002406
1612 004756 162737 000001 002414
1613 004764 005637 002412
1614 004770 005637 002410
1615 004774 005637 002406
1616 005000 173737 002406
1617 005004 170000
1618 005006 001403
1619 005010 174237 002406
1620 005014 104006
1621
1622 005016 005037 002362
1623
;*****
;TEST 4 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
;*****
†ST4: SCOPE
MOV #ARET4,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)
$ADD ;ADDRESS OF FORTRAN ADD
$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;LOAD FPS, INTERRUPT DISABLE AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #004600 ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
ARET4: LDD AC0,AC2 ;LOAD AC0 INTO AC2
ADD AC1,AC2 ;ADD AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ ATST4 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 17 ;FPS ERROR

ATST4: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND4 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4
SBC ANS1+2
SBC ANS1
CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND4 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 6 ;FPU AND FORTRAN DISAGREE

AEND4: CLR FPS ;CLR FPU FPS BUFFER

```

```

1624
1625
1626
1627 005022 000004
1628 005024 012737 005156 002376
1629 005032 012737 007440 002400
1630 005040 005037 002402
1631 005044 005037 002404
1632 005050 005037 002362
1633 005054 005037 002364
1634 005060 005037 002366
1635 005064 004737 023414
1636 005070 002426 002436
1637 005074 004437 023560
1638 005100 023562 002426
1639 005104 023562 002436
1640 005110 023640
1641 005112 023612 002416
1642
1643 005116 013700 002400
1644 005122 170127 040000
1645 005126 172437 002426
1646 005132 172537 002436
1647 005136 172737 002416
1648 005142 170127 007440
1649 005146 012737 005154 001110
1650
1651
1652
1653 005154 172600
1654 005156 172201
1655 005160 170237 002362
1656 005164 023737 002362 002400
1657 005172 001403
1658 005174 174237 002406
1659 005200 104001
1660
1661 005202 005737 002400
1662 005206 100014
1663 005210 170337 002364
1664
1665 005214 023737 002364 002402
1666 005222 001401
1667 005224 104023
1668
1669 005226 023737 002366 002404
1670 005234 001401
1671 005236 104023
1672
1673 005240 173702
1674 005242 170000
1675 005244 001427
1676
1677 005246 174237 002406
1678 005252 062737 000001 002410
1679 005260 005537 002406

```

```

;*****
;TEST 5 EXERCISE ADDF, ALL INTERRUPTS ON, TRUNCATE MODE
;*****
TSTS: SCOPE
MOV #ARETS,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007440,$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)
$ADD ;ADDRESS OF FORTRAN ADD
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;LOAD FPS, INTERRUPT DISABLE
LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007440 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
ARETS: LDF AC0,AC2 ;LOAD AC0 INTO AC2
ADDF AC1,AC2 ;ADD AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ AERRS ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 1 ;FPS ERROR

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

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 15 ;BRANCH IF OK
ERROR 23 ;FEC IS WRONG

IS: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ ATSTS ;BRANCH IF OK
ERROR 23 ;WRONG ADDRESS IN FEA

ATSTS: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AENDS ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+2 ;INCREMENT FPU ANSWER
ADC ANS1

```



```

1694
1695
1696
1697 005330 003004
1698 005332 012737 005464 002376
1699 005340 012737 007640 002400
1700 005346 005037 002402
1701 005352 005037 002404
1702 005356 005037 002362
1703 005362 005037 002364
1704 005366 005037 002366
1705 005372 004737 023404
1706 005376 002426 002436
1707 005402 004437 023560
1708 005406 023562 002426
1709 005412 023562 002436
1710 005416 023640
1711 005420 023612 002416
1712
1713 005424 013700 002400
1714 005430 170127 040200
1715 005434 172437 002426
1716 005440 172537 002436
1717 005444 172737 002416
1718 005450 170127 007640
1719 005454 012737 005462 001110
1720
1721
1722
1723 005462 172600
1724 005464 172201
1725 005466 170237 002362
1726 005472 023737 002362 002400
1727 005500 001403
1728 005502 174237 002406
1729 005506 104017
1730
1731 005510 005737 002400
1732 005514 100014
1733 005516 170337 002364
1734
1735 005522 023737 002364 002402
1736 005530 001401
1737 005532 104027
1738
1739 005534 023737 002366 002404
1740 005542 001401
1741 005544 104027
1742
1743 005546 173702
1744 005550 170000
1745 005552 001437
1746
1747 005554 174237 002406
1748 005560 062737 000001 002414
1749 005566 005537 002412

```

```

;*****
;TEST 6 EXERCISE ADD, ALL INTERRUPTS ON, TRUNCATE MODE
;*****
TST6: SCOPE
MOV #ARET6,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007640,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL4 ;GET RANDOM INPUT DATA
;WORD LONUM,HINUM
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 4 WORDS ON STACK (HINUM)
$ADD ;ADDRESS OF FORTRAN ADD
$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;LOAD FPS, INTERRUPT DISABLE AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007640 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
ARET6: LDD AC0,AC2 ;LOAD AC0 INTO AC2
ADD AC1,AC2 ;ADD AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ AERR6 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 17 ;FPS ERROR

AERR6: TST $FPS ;ERROR BIT SET ?
BPL ATST6 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 27 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ ATST6 ;BRANCH IF OK
ERROR 27 ;WRONG ADDRESS IN FEA

ATST6: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND6 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
ADC ANS1+4

```

# M03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 40  
 DQFPDA.P11 09-FEB-77 10:32 T6 EXERCISE ADD, ALL INTERRUPTS ON, TRUNCATE MODE

1750	005572	005537	002410		ADC	ANS1+2	
1751	005576	005537	002406		ADC	ANS1	
1752	005602	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
1753	005606	170000			CFCC		;COPY FLOATING CONDITION CODES
1754	005610	001420			BEQ	AEND6	;BRANCH IF OK
1755	005612	162737	000002	002414	SUB	#2,ANS1+6	;DECREMENT FPU ANSWER
1756	005620	005637	002412		SBC	ANS1+4	
1757	005624	005637	002410		SBC	ANS1+2	
1758	005630	005637	002406		SBC	ANS1	
1759	005634	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
1760	005640	170000			CFCC		;COPY FLOATING CONDITION CODES
1761	005642	001403			BEQ	AEND6	;BRANCH IF OK
1762	005644	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
1763	005650	104006			ERROR	6	;FPU AND FORTRAN DISAGREE
1764							
1765	005652	005037	002362		AEND6: CLR	FPS	;CLR FPU FPS BUFFER
1766							
1767							

# N03

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 41  
 DQFPDA.P11 09-FEB-77 10:32 T7 EXERCISE ADDF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

```

1768
1769
1770
1771 005656 000004
1772 005660 012737 006012 002376
1773 005666 012737 004440 002400
1774 005674 005037 002402
1775 005700 005037 002404
1776 005704 005037 002362
1777 005710 005037 002364
1778 005714 005037 002366
1779 005720 004737 023414
1780 005724 002426 002436
1781 005730 004437 023560
1782 005734 023562 002426
1783 005740 023562 002436
1784 005744 023640
1785 005746 023612 002416
1786
1787 005752 013700 002400
1788 005756 170127 040000
1789 005762 172437 002426
1790 005766 172537 002436
1791 005772 172737 002416
1792 005776 170127 004440
1793 006002 012737 006010 001110
1794
1795
1796
1797 006010 172600
1798 006012 172201
1799 006014 170237 002362
1800 006020 023737 002362 002400
1801 006026 001403
1802 006030 174237 002406
1803 006034 104001
1804
1805 006036 173702
1806 006040 170000
1807 006042 001427
1808
1809 006044 174237 002406
1810 006050 062737 000001 002410
1811 006056 005537 002406
1812 006062 173737 002406
1813 006066 170000
1814 006070 001414
1815 006072 162737 000002 002410
1816 006100 005637 002406
1817 006104 173737 002406
1818 006110 170000
1819 006112 001403
1820 006114 174237 002406
1821 006120 104005
1822
1823 006122 005037 002362
  
```

```

;*****
;TEST 7 EXERCISE ADDF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
;*****
†ST7: SCOPE
MOV #ARET7,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)
$ADD ;ADDRESS OF FORTRAN ADD
$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

;*****
ARET7: LDF AC0,AC2 ;LOAD AC0 INTO AC2
ADDF AC1,AC2 ;ADD AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;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 AEND7 ;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 AEND7 ;BRANCH IF OK
SUB #2,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1
CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND7 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 5 ;FPU AND FORTRAN DISAGREE

AEND7: CLR FPS ;CLR FPU FPS BUFFER
  
```



B04

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006)  
DGFPDA.P11 09-FEB-77 10:32 T7

09-FEB-77 10:34 PAGE 42  
EXERCISE ADDF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

1824

```

1825
1826
1827
1828 006126 000004
1829 006130 012737 006262 002376
1830 006136 012737 004640 002400
1831 006144 005037 002402
1832 006150 005037 002404
1833 006154 005037 002362
1834 006160 005037 002364
1835 006164 005037 002366
1836 006170 004737 023404
1837 006174 002426 002436
1838 006200 004437 023560
1839 006204 023562 002426
1840 006210 023562 002436
1841 006214 023640
1842 006216 023612 002416
1843
1844 006222 013700 002400
1845 006226 170127 040200
1846 006232 172437 002426
1847 006236 172537 002436
1848 006242 172737 002416
1849 006246 170127 004640
1850 006252 012737 006260 001110
1851
1852
1853
1854 006260 172600
1855 006262 172201
1856 006264 170237 002362
1857 006270 023737 002362 002400
1858 006276 001403
1859 006300 174237 002406
1860 006304 104017
1861
1862 006306 173702
1863 006310 170000
1864 006312 001437
1865
1866 006314 174237 002406
1867 006320 062737 000001 002414
1868 006326 005537 002412
1869 006332 005537 002410
1870 006336 005537 002406
1871 006342 173737 002406
1872 006346 170000
1873 006350 001420
1874 006352 162737 000002 002414
1875 006360 005637 002412
1876 006364 005637 002410
1877 006370 005637 002406
1878 006374 173737 002406
1879 006400 170000
1880 006402 001403

```

```

;*****
;#TEST 10 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
;*****
†ST10: SCOPE
MOV #ARET10,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #004640,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL4 ;GET RANDOM INPUT DATA
;WORD
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 4 WORDS ON STACK (HINUM)
$ADD ;ADDRESS OF FORTRAN ADD
$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;LOAD FPS, INTERRUPT DISABLE AND FD
LDD LONUM,ACO ;LOAD ACO WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #004640 ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
ARET10: LDD ACO,AC2 ;LOAD ACO INTO AC2
ADD AC1,AC2 ;ADD AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ ATST10 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 17 ;FPS ERROR

ATST10: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND10 ;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 AEND10 ;BRANCH IF OK
SUB #2,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4
SBC ANS1+2
SBC ANS1
CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND10 ;BRANCH IF OK

```

D04

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 44  
DQFPDA.P11 09-FEB-77 10:32 T10 EXERCISE ADD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

1881 006404 174237 002406  
1882 006410 104006  
1883  
1884 006412 005037 002362  
1885

STD AC2,ANS1  
ERROR 6

;SAVE FPU ANSWER  
;FPU AND FORTRAN DISAGREE

AEND10: CLR FPS

;CLR FPU FPS BUFFER

1886  
1887  
1888  
1889 006416 000004  
1890 006420 012737 006552 002376  
1891 006426 012737 007400 002400  
1892 006434 005037 002402  
1893 006440 005037 002404  
1894 006444 005037 002362  
1895 006450 005037 002364  
1896 006454 005037 002366  
1897 006460 004737 023414  
1898 006464 002426 002436  
1899 006470 004437 023560  
1900 006474 023562 002426  
1901 006500 023562 002436  
1902 006504 023634  
1903 006506 023612 002416  
1904  
1905 006512 013700 002400  
1906 006516 170127 040000  
1907 006522 172437 002426  
1908 006526 172537 002436  
1909 006532 172737 002416  
1910 006536 170127 007400  
1911 006542 012737 006550 001110  
1912  
1913  
1914  
1915 006550 172600  
1916 006552 173201  
1917 006554 170237 002362  
1918 006560 023737 002362 002400  
1919 006566 001403  
1920 006570 174237 002406  
1921 006574 104002  
1922  
1923 006576 005737 002400  
1924 006602 100014  
1925 006604 170337 002364  
1926  
1927 006610 023737 002364 002402  
1928 006616 001401  
1929 006620 104024  
1930  
1931 006622 023737 002366 002404 1\$:  
1932 006630 001401  
1933 006632 104024  
1934  
1935 006634 173702  
1936 006636 170000  
1937 006640 001416  
1938  
1939 006642 174237 002406  
1940 006646 162737 000001 002410  
1941 006654 005637 002406

```
*****  
: *TEST 11 EXERCISE SUBF, ALL INTERRUPTS ON, ROUNDING MODE  
: *****  
↑ST11: SCOPE  
MOV #ARET11, 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)  
$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, 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  
  
; *****  
LDF ACO, AC2 ; LOAD ACO INTO AC2  
ARET11: SUBF AC1, AC2 ; SUBTRACT AC1 BY AC2  
STFPS FPS ; STORE FLOATING POINT STATUS  
CMP FPS, $FPS ; CHECK FPS  
BEQ AERR11 ; BRANCH IF OK  
STF AC2, ANS1 ; SAVE FPU ANSWER  
ERROR 2 ; FPS ERROR  
  
AERR11: TST $FPS ; ERROR BIT SET ?  
BPL ATST11 ; NO, DONT GET FEC/FEA  
STST FEC ; YES, CHECK STATUS  
  
CMP FEC, $FEC ; CHECK THE FLOATING EXCEPTION CODES  
BEQ 1$ ; BRANCH IF OK  
ERROR 24 ; FEC IS WRONG  
  
1$: CMP FEA, $FEA ; CHECK FLOATING PC  
BEQ ATST11 ; BRANCH IF OK  
ERROR 24 ; WRONG ADDRESS IN FEA  
  
ATST11: CMPF AC2, AC3 ; COMPARE FPU ANSWER TO FORTRAN ANSWER  
CFCC ; COPY FLOATING CONDITION CODES  
BEQ AEND11 ; ANSWERS CHECK  
 .COMPENSATE FOR FORTRAN INACCURACIES.  
STF AC2, ANS1 ; SAVE FPU ANSWER  
SUB #1, ANS1+2 ; DECREMENT FPU ANSWER  
SBC ANS1
```

# F04

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 46  
DQFPDA.P11 09-FEB-77 10:32 T11 EXERCISE SUBF, ALL INTERRUPTS ON, ROUNDING MODE

1942	006660	173737	002406	CMPF	ANS1,AC3	;CHECK ANSWERS AGAIN
1943	006664	170000		CFCC		;COPY FLOATING CONDITION CODES
1944	006666	001403		BEQ	AEND11	;BRANCH IF OK
1945	006670	174237	002406	STF	AC2,ANS1	;SAVE FPU ANSWER
1946	006674	104007		ERROR	7	;FPU AND FORTRAN DISAGREE
1947						
1948	006676	005037	002362	AEND11: CLR	FPS	;CLR FPU FPS BUFFER
1949						

```

1950          ;*****
1951          ;*TEST 12      EXERCISE SUBD, ALL INTERRUPTS ON, ROUNDING MODE
1952          ;*****
1953 006702  000004          †ST12: SCOPE
1954 006704  012737  007036  002376  MOV    #ARET12,EXPFEA ;ADDR OF INSTR BEING TESTED
1955 006712  012737  007600  002400  MOV    #007600,$FPS  ;SET IE BITS IN FORTRAN ANSWER
1956 006720  005037  002402          CLR    $FEC          ;CLR FORTRAN FEC
1957 006724  005037  002404          CLR    $FEA          ;CLR FORTRAN FEA
1958 006730  005037  002362          CLR    FPS           ;CLR FPU FPS BUFFER
1959 006734  005037  002364          CLR    FEC           ;CLR FPU FEC BUFFER
1960 006740  005037  002366          CLR    FEA           ;CLR FPU FEA BUFFER
1961 006744  004737  023404          JSR    PC,RANDL4     ;GET RANDOM INPUT DATA
1962 006750  002426  002436          .WORD  LONUM,HINUM  ;
1963 006754  004437  023560          JSR    R4,$POLSH    ;ENTER POLISH MODE
1964 006760  023562  002426          $PUSH ,LONUM        ;PUSH 4 WORDS ON STACK (LONUM)
1965 006764  023562  002436          $PUSH ,HINUM        ;PUSH 4 WORDS ON STACK (HINUM)
1966 006770  023634          $$SUB                ;ADDRESS OF FORTRAN SUBTRACT
1967 006772  023612  002416          $POPX ,ANS2         ;POP 4 WORDS AND EXIT POLISH MODE
1968
1969 006776  013700  002400          MOV    $FPS,R0       ;DISPLAY FLOATING POINT STATUS
1970 007002  170127  040200          LDFPS #040200        ;LOAD FPS, INTERRUPT DISABLE AND FD
1971 007006  172437  002426          LDD   LONUM,AC0      ;LOAD AC0 WITH A RANDOM NUMBER
1972 007012  172537  002436          LDD   HINUM,AC1     ;LOAD AC1 WITH A RANDOM NUMBER
1973 007016  172737  002416          LDD   ANS2,AC3       ;LOAD AC3 WITH THE SUM
1974 007022  170127  007600          LDFPS #007600        ;TURN INTERRUPTS ON
1975 007026  012737  007034  001110  MOV    #.+6,$LPADR   ;RESET LOOP ADDRESS
1976
1977          ;*****
1978
1979 007034  172600          LDD   AC0,AC2        ;LOAD AC0 INTO AC2
1980 007036  173201          ARET12: SUBD  AC1,AC2  ;SUBTRACT AC1 BY AC2
1981 007040  170237  002362          STFPS FPS            ;STORE FLOATING POINT STATUS
1982 007044  023737  002362  002400  CMP   FPS,$FPS       ;CHECK FPS
1983 007052  001403          BEQ   AERR12         ;BRANCH IF OK
1984 007054  174237  002406          STD   AC2,ANS1       ;SAVE FPU ANSWER
1985 007060  104020          ERROR 20            ;FPS ERROR
1986
1987 007062  005737  002400          AERR12: TST   $FPS    ;ERROR BIT SET ?
1988 007066  100014          BPL  ATST12          ;NO, DONT GET FEC/FEA
1989 007070  170337  002364          STST  FEC            ;YES, CHECK STATUS
1990
1991 007074  023737  002364  002402          CMP   FEC,$FEC       ;CHECK THE FLOATING EXCEPTION CODES
1992 007102  001401          BEQ   1$             ;BRANCH IF OK
1993 007104  104030          ERROR 30            ;FEC IS WRONG
1994
1995 007106  023737  002366  002404  1$:  CMP   FEA,$FEA      ;CHECK FLOATING PC
1996 007114  001401          BEQ   ATST12         ;BRANCH IF OK
1997 007116  104030          ERROR 30            ;WRONG ADDRESS IN FEA
1998
1999 007120  173702          ATST12: CMPD  AC2,AC3  ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2000 007122  170000          CFCC                ;COPY FLOATING CONDITION CODES
2001 007124  001422          BEQ   AEND12         ;ANSWERS CHECK
2002          ;COMPENSATE FOR FORTRAN INACCURACIES.
2003 007126  174237  002406          STD   AC2,ANS1       ;SAVE FPU ANSWER
2004 007132  162737  000001  002414  SUB   #1,ANS1+6      ;DECREMENT FPU ANSWER
2005 007140  005637  002412          SBC   ANS1+4

```

H04

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 48  
DQFPDA.P11 09-FEB-77 10:32 T12 EXERCISE SUBD, ALL INTERRUPTS ON, ROUNDING MODE

2006	007144	005637	002410	SBC	ANS1+2	
2007	007150	005637	002406	SBC	ANS1	
2008	007154	173737	002406	CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
2009	007160	170000		CFCC		;COPY FLOATING CONDITION CODES
2010	007162	001403		BEQ	AEND12	;BRANCH IF OK
2011	007164	174237	002406	STD	AC2,ANS1	;SAVE FPU ANSWER
2012	007170	104010		ERROR	10	;FPU AND FORTRAN DISAGREE
2013						
2014	007172	005037	002362	AEND12: CLR	FPS	;CLR FPU FPS BUFFER
2015						

```

2016
2017
2018
2019 007176 000004
2020 007200 012737 007332 002376
2021 007206 012737 004400 002400
2022 007214 005037 002402
2023 007220 005037 002404
2024 007224 005037 002362
2025 007230 005037 002364
2026 007234 005037 002366
2027 007240 004737 023414
2028 007244 002426 002436
2029 007250 004437 023560
2030 007254 023562 002426
2031 007260 023562 002436
2032 007264 023634
2033 007266 023612 002416
2034
2035 007272 013700 002400
2036 007276 170127 040000
2037 007302 172437 002426
2038 007306 172537 002436
2039 007312 172737 002416
2040 007316 170127 004400
2041 007322 012737 007330 001110
2042
2043
2044
2045 007330 172600
2046 007332 173201
2047 007334 170237 002362 002400
2048 007340 023737 002362
2049 007346 001403
2050 007350 174237 002406
2051 007354 104002
2052
2053 007356 173702
2054 007360 170000
2055 007362 001416
2056
2057 007364 174237 002406
2058 007370 162737 000001 002410
2059 007376 005637 002406
2060 007402 173737 002406
2061 007406 170000
2062 007410 001403
2063 007412 174237 002406
2064 007416 104007
2065
2066 007420 005037 002362
2067

```

```

;*****
;TEST 13 EXERCISE SUBF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
;*****
†ST13: SCOPE
MOV #ARET13,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)
$SUB ;ADDRESS OF FORTRAN SUBTRACT
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

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

;*****
ARET13: 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 ATST13 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 2 ;FPS ERROR

ATST13: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND13 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1
CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND13 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 7 ;FPU AND FORTRAN DISAGREE

AEND13: CLR FPS ;CLR FPU FPS BUFFER

```



```

2068
2069
2070
2071 007424 000004
2072 007426 012737 007560 002376
2073 007434 012737 004600 002400
2074 007442 005037 002402
2075 007446 005037 002404
2076 007452 005037 002362
2077 007456 005037 002364
2078 007462 005037 002366
2079 007466 004737 023404
2080 007472 002426 002436
2081 007476 004437 023560
2082 007502 023562 002426
2083 007506 023562 002436
2084 007512 023634
2085 007514 023612 002416
2086
2087 007520 013700 002400
2088 007524 170127 040200
2089 007530 172437 002426
2090 007534 172537 002436
2091 007540 172737 002416
2092 007544 170127 004600
2093 007550 012737 007556 001110
2094
2095
2096
2097 007556 172600
2098 007560 173201
2099 007562 170237 002362
2100 007566 023737 002362 002400
2101 007574 001403
2102 007576 174237 002406
2103 007602 104020
2104
2105 007604 173702
2106 007606 170000
2107 007610 001422
2108
2109 007612 174237 002406
2110 007616 162737 000001 002414
2111 007624 005637 002412
2112 007630 005637 002410
2113 007634 005637 002406
2114 007640 173737 002406
2115 007644 170000
2116 007646 001403
2117 007650 174237 002406
2118 007654 104010
2119
2120 007656 005037 002362
2121

```

```

;*****
;TEST 14 EXERCISE SUBD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
;*****
†ST14: SCOPE
MOV #ARET14,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)
$SUB ;ADDRESS OF FORTRAN SUBTRACT
$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;LOAD FPS, INTERRUPT DISABLE AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #004600 ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
ARET14: LDD AC0,AC2 ;LOAD AC0 INTO AC2
SUBD AC1,AC2 ;SUBTRACT AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ ATST14 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 20 ;FPS ERROR

ATST14: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND14 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4
SBC ANS1+2
SBC ANS1
CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND14 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 10 ;FPU AND FORTRAN DISAGREE

AEND14: CLR FPS ;CLR FPU FPS BUFFER

```

K04

```

2122
2123
2124
2125 007662 000004
2126 007664 012737 010016 002376
2127 007672 012737 007440 002400
2128 007700 005037 002402
2129 007704 005037 002404
2130 007710 005037 002362
2131 007714 005037 002364
2132 007720 005037 002366
2133 007724 004737 023414
2134 007730 002426 002436
2135 007734 004437 023560
2136 007740 023562 002426
2137 007744 023562 002436
2138 007750 023634
2139 007752 023612 002416
2140
2141 007756 013700 002400
2142 007762 170127 040000
2143 007766 172437 002426
2144 007772 172537 002436
2145 007776 172737 002416
2146 010002 170127 007440
2147 010006 012737 010014 001110
2148
2149
2150
2151 010014 172600
2152 010016 173201
2153 010020 170237 002362 002400
2154 010024 023737 002362
2155 010032 001403
2156 010034 174237 002406
2157 010040 104002
2158
2159 010042 005737 002400
2160 010046 100014
2161 010050 170337 002364
2162
2163 010054 023737 002364 002402
2164 010062 001401
2165 010064 104024
2166
2167 010066 023737 002366 002404 1$:
2168 010074 001401
2169 010076 104024
2170
2171 010100 173702
2172 010102 170000
2173 010104 001427
2174
2175 010106 174237 002406
2176 010112 062737 000001 002410
2177 010120 005537 002406

```

```

*****
;TEST 15 EXERCISE SUBF, ALL INTERRUPTS ON, TRUNCATE MODE
*****
TST15: SCOPE
MOV #ARET15,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007440,$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,SPOLSH ;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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;LOAD FPS, INTERRUPT DISABLE
LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007440 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
ARET15: 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 AERR15 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 2 ;FPS ERROR

AERR15: TST $FPS ;ERROR BIT SET ?
BPL ATST15 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 24 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ ATST15 ;BRANCH IF OK
ERROR 24 ;WRONG ADDRESS IN FEA

ATST15: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND15 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+2 ;INCREMENT FPU ANSWER
ADC ANS1

```



```

2191
2192
2193
2194 010170 000004
2195 010172 012737 010324 002376
2196 010200 012737 007640 002400
2197 010206 005037 002402
2198 010212 005037 002404
2199 010216 005037 002362
2200 010222 005037 002364
2201 010226 005037 002366
2202 010232 004737 023404
2203 010236 002426 002436
2204 010242 004437 023560
2205 010246 023562 002426
2206 010252 023562 002436
2207 010256 023634
2208 010260 023612 002416
2209
2210 010264 013700 002400
2211 010270 170127 040200
2212 010274 172437 002426
2213 010300 172537 002436
2214 010304 172737 002416
2215 010310 170127 007640
2216 010314 012737 010322 001110
2217
2218
2219
2220 010322 172600
2221 010324 173201
2222 010326 170237 002362
2223 010332 023737 002362 002400
2224 010340 001403
2225 010342 174237 002406
2226 010346 104020
2227
2228 010350 005737 002400
2229 010354 100014
2230 010356 170337 002364
2231
2232 010362 023737 002364 002402
2233 010370 001401
2234 010372 104030
2235
2236 010374 023737 002366 002404
2237 010402 001401
2238 010404 104030
2239
2240 010406 173702
2241 010410 170000
2242 010412 001437
2243
2244 010414 174237 002406
2245 010420 062737 000001 002414
2246 010426 005537 002412

```

```

;*****
;#TEST 16 EXERCISE SUBD, ALL INTERRUPTS ON, TRUNCATE MODE
;*****
TST16: SCOPE
MOV #ARET16,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007640,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL4 ;GET RANDOM INPUT DATA
;WORD LONUM,HINUM
JSR R4,$POLSH ;ENTER POLISH MODE
$PUSH ,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
$PUSH ,HINUM ;PUSH 4 WORDS ON STACK (HINUM)
$SUB ;ADDRESS OF FORTRAN SUBTRACT
$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;LOAD FPS, INTERRUPT DISABLE AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007640 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
ARET16: LDD AC0,AC2 ;LOAD AC0 INTO AC2
SUBD AC1,AC2 ;SUBTRACT AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ AERR16 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 20 ;FPS ERROR

AERR16: TST $FPS ;ERROR BIT SET ?
BPL ATST16 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 15 ;BRANCH IF OK
ERROR 30 ;FEC IS WRONG

15: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ ATST16 ;BRANCH IF OK
ERROR 30 ;WRONG ADDRESS IN FEA

ATST16: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AEND16 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
ADC ANS1+4

```

2247	010432	005537	002410		ADC	ANS1+2	
2248	010436	005537	002406		ADC	ANS1	
2249	010442	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
2250	010446	170000			CFCC		;COPY FLOATING CONDITION CODES
2251	010450	001420			BEQ	AEND16	;BRANCH IF OK
2252	010452	162737	000002	002414	SUB	#2,ANS1+6	;DECREMENT FPU ANSWER
2253	010460	005637	002412		SBC	ANS1+4	
2254	010464	005637	002410		SBC	ANS1+2	
2255	010470	005637	002406		SBC	ANS1	
2256	010474	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
2257	010500	170000			CFCC		;COPY FLOATING CONDITION CODES
2258	010502	001403			BEQ	AEND16	;BRANCH IF OK
2259	010504	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
2260	010510	104010			ERROR	10	;FPU AND FORTRAN DISAGREE
2261							
2262	010512	005037	002362		AEND16: CLR	FPS	;CLR FPU FPS BUFFER
2263							

```

2264
2265
2266
2267 010516 000004
2268 010520 012737 010652 002376
2269 010526 012737 004440 002400
2270 010534 005037 002402
2271 010540 005037 002404
2272 010544 005037 002362
2273 010550 005037 002364
2274 010554 005037 002366
2275 010560 004737 023414
2276 010564 002426 002436
2277 010570 004437 023560
2278 010574 023562 002426
2279 010600 023562 002436
2280 010604 023634
2281 010606 023612 002416
2282
2283 010612 013700 002400
2284 010616 170127 040000
2285 010622 172437 002426
2286 010626 172537 002436
2287 010632 172737 002416
2288 010636 170127 004440
2289 010642 012737 010650 001110
2290
2291
2292
2293 010650 172600
2294 010652 173201
2295 010654 170237 002362 002400
2296 010660 023737 002362
2297 010666 001403
2298 010670 174237 002406
2299 010674 104002
2300
2301 010676 173702
2302 010700 170000
2303 010702 001427
2304
2305 010704 174237 002406
2306 010710 062737 000001 002410
2307 010716 005537 002406
2308 010722 173737 002406
2309 010726 170000
2310 010730 001414
2311 010732 162737 000002 002410
2312 010740 005637 002406
2313 010744 173737 002406
2314 010750 170000
2315 010752 001403
2316 010754 174237 002406
2317 010760 104007
2318
2319 010762 005037 002362

```

```

*****
;TEST 17 EXERCISE SUBF, 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
ADD #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
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 7 ;FPU AND FORTRAN DISAGREE

AEND17: CLR FPS ;CLR FPU FPS BUFFER

```

C05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 56  
DQFPDA.P11 09-FEB-77 10:32 T17 EXERCISE SUBF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

2320

```

2321
2322
2323
2324 010766 000004
2325 010770 012737 011122 002376
2326 010776 012737 004640 002400
2327 011004 005037 002402
2328 011010 005037 002404
2329 011014 005037 002362
2330 011020 005037 002364
2331 011024 005037 002366
2332 011030 004737 023404
2333 011034 002426 002436
2334 011040 004437 023560
2335 011044 023562 002426
2336 011050 023562 002436
2337 011054 023634
2338 011056 023612 002416
2339
2340 011062 013700 002400
2341 011066 170127 040200
2342 011072 172437 002426
2343 011076 172537 002436
2344 011102 172737 002416
2345 011106 170127 004640
2346 011112 012737 011120 001110
2347
2348
2349
2350 011120 172600
2351 011122 173201
2352 011124 170237 002362
2353 011130 023737 002362 002400
2354 011136 001403
2355 011140 174237 002406
2356 011144 104020
2357
2358 011146 173702
2359 011150 170000
2360 011152 001437
2361
2362 011154 174237 002406
2363 011160 062737 000001 002414
2364 011166 005537 002412
2365 011172 005537 002410
2366 011176 005537 002406
2367 011202 173737 002406
2368 011206 170000
2369 011210 001420
2370 011212 162737 000002 002414
2371 011220 005637 002412
2372 011224 005637 002410
2373 011230 005637 002406
2374 011234 173737 002406
2375 011240 170000
2376 011242 001403

```

```

:*****
:TEST 20 EXERCISE SUBD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
:*****
TST20: SCOPE
MOV #ARET20,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)
$SUB ;ADDRESS OF FORTRAN SUBTRACT
$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;LOAD FPS, INTERRUPT DISABLE AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #004640 ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

:*****
ARET20: LDD AC0,AC2 ;LOAD AC0 INTO AC2
SUBD AC1,AC2 ;SUBTRACT AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ ATST20 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 20 ;FPS ERROR

ATST20: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ AREND20 ;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 AREND20 ;BRANCH IF OK
SUB #2,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4
SBC ANS1+2
SBC ANS1
CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ AREND20 ;BRANCH IF OK

```



E05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 58  
DQFPDA.P11 09-FEB-77 10:32 T20 EXERCISE SUBD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

2377 011244 174237 002406  
2378 011250 104010  
2379  
2380 011252 005037 002362  
2381

STD AC2,ANSI ;SAVE FPU ANSWER  
ERROR 10 ;FPU AND FORTRAN DISAGREE  
AEND20: CLR FPS ;CLR FPU FPS BUFFER

# F05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 59  
 DQFPDA.P11 09-FEB-77 10:32 T21 EXERCISE MULF, ALL INTERRUPTS ON, ROUNDING MODE

```

2382 ;*****
2383 ;*TEST 21 EXERCISE MULF, ALL INTERRUPTS ON, ROUNDING MODE
2384 ;*****
2385 011256 000004          ST21: SCOPE
2386 011260 012737 011412 002376 MOV #MRET1,EXPFEA ;ADDR OF INSTR BEING TESTED
2387 011266 012737 007400 002400 MOV #007400,$FPS ;SET IE BITS IN FORTRAN ANSWER
2388 011274 005037 002402 CLR $FEC ;CLR FORTRAN FEC
2389 011300 005037 002404 CLR $FEA ;CLR FORTRAN FEA
2390 011304 005037 002362 CLR FPS ;CLR FPU FPS BUFFER
2391 011310 005037 002364 CLR FEC ;CLR FPU FEC BUFFER
2392 011314 005037 002366 CLR FEA ;CLR FPU FEA BUFFER
2393 011320 004737 023414 JSR PC,RANDL2 ;GET RANDOM INPUT DATA
2394 011324 002426 002436 .WORD LONUM,HINUM
2395 011330 004437 023560 JSR R4,$POLSH ;ENTER POLISH MODE
2396 011334 023562 002426 $PUSH ,LONUM ;PUSH 2 WORDS ON STACK (LONUM)
2397 011340 023562 002436 $PUSH ,HINUM ;PUSH 2 WORDS ON STACK (HINUM)
2398 011344 025314 $MUL ;ADDRESS OF FORTRAN MULTIPLY
2399 011346 023612 002416 $POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE
2400
2401 011352 013700 002400 MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
2402 011356 170127 040000 LDFPS #040000 ;CLEAR THE FPS, INTERRUPT DISABLE
2403 011362 172437 002426 LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
2404 011366 172537 002436 LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
2405 011372 172737 002416 LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
2406 011376 170127 007400 LDFPS #007400 ;TURN INTERRUPTS ON
2407 011402 012737 011410 001110 MOV #.+6,$LPADR ;RESET LOOP ADDRESS
2408
2409 ;*****
2410
2411 011410 172600          MRET1: LDF AC0,AC2 ;LOAD AC0 INTO AC2
2412 011412 171201          MULF AC1,AC2 ;MULTIPLY AC1 BY AC2
2413 011414 170237 002362          STFPS FPS ;STORE FLOATING POINT STATUS
2414 011420 023737 002362 002400          CMP FPS,$FPS ;CHECK FPS
2415 011426 001403          BEQ MERR1 ;BRANCH IF OK
2416 011430 174237 002406          STF AC2,ANS1 ;SAVE FPU ANSWER
2417 011434 104003          ERROR 3 ;FPS ERROR
2418
2419 011436 005737 002400          MERR1: TST $FPS ;ERROR BIT SET ?
2420 011442 100014          BPL MTST1 ;NO, DONT GET FEC/FEA
2421 011444 170337 002364          STST FEC ;YES, CHECK STATUS
2422
2423 011450 023737 002364 002402          CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
2424 011456 001401          BEQ 1$ ;BRANCH IF OK
2425 011460 104025          ERROR 25 ;FEC IS WRONG
2426
2427 011462 023737 002366 002404 1$: CMP FEA,$FEA ;CHECK FLOATING PC
2428 011470 001401          BEQ MTST1 ;BRANCH IF OK
2429 011472 104025          ERROR 25 ;WRONG ADDRESS IN FEA
2430
2431 011474 173702          MTST1: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2432 011476 170000          CFCC ;COPY FLOATING CONDITION CODES
2433 011500 001416          BEQ MEND1 ;ANSWERS CHECK
2434 ;COMPENSATE FOR FORTRAN INACCURACIES.
2435 011502 174237 002406          STF AC2,ANS1 ;SAVE FPU ANSWER
2436 011506 162737 000001 002410          SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
2437 011514 005637 002406          SBC ANS1
  
```

G05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 60  
DQFPDA.P11 09-FEB-77 10:32 T21 EXERCISE MULF, ALL INTERRUPTS ON, ROUNDING MODE

2438	011520	173737	002406	CMPF	ANS1,AC3	;CHECK ANSWERS AGAIN
2439	011524	170000		CFCC		;COPY FLOATING CONDITION CODES
2440	011526	001403		BEQ	MEND1	;BRANCH IF OK
2441	011530	174237	002406	STF	AC2,ANS1	;SAVE FPU ANSWER
2442	011534	104011		ERROR	11	;FPU AND FORTRAN DISAGREE
2443						
2444	011536	005037	002362	MEND1:	CLR FPS	;CLEAR FPP FPS BUFFER
2445						

# H05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 61  
 DQFPDA.P11 09-FEB-77 10:32 T22 EXERCISE MULD, ALL INTERRUPTS ON, ROUNDING MODE

```

2446
2447
2448
2449 011542 000004
2450 011544 012737 011676 002376
2451 011552 012737 007600 002400
2452 011560 005037 002402
2453 011564 005037 002404
2454 011570 005037 002362
2455 011574 005037 002364
2456 011600 005037 002366
2457 011604 004737 023404
2458 011610 002426 002436
2459 011614 004437 023560
2460 011620 023562 002426
2461 011624 023562 002436
2462 011630 025314
2463 011632 023612 002416
2464
2465 011636 013700 002400
2466 011642 170127 040200
2467 011646 172437 002426
2468 011652 172537 002436
2469 011656 172737 002416
2470 011662 170127 007600
2471 011666 012737 011674 001110
2472
2473
2474
2475 011674 172600
2476 011676 171201
2477 011700 170237 002362
2478 011704 023737 002362 002400
2479 011712 001403
2480 011714 174237 002406
2481 011720 104021
2482
2483 011722 005737 002400
2484 011726 100014
2485 011730 170337 002364
2486
2487 011734 023737 002364 002402
2488 011742 001401
2489 011744 104031
2490
2491 011746 023737 002366 002404
2492 011754 001401
2493 011756 104031
2494
2495 011760 173702
2496 011762 170000
2497 011764 001422
2498
2499 011766 174237 002406
2500 011772 162737 000001 002414
2501 012000 005637 002412
  
```

```

;*****
;TEST 22 EXERCISE MULD, ALL INTERRUPTS ON, ROUNDING MODE
;*****
↑ST22: SCOPE
MOV #MRET2,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007600,$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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;SET FD OF FPS ONLY, INTERRUPT DISABLE
LDD LONUM,ACO ;LOAD ACO WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007600 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
MRET2: LDD ACO,AC2 ;LOAD ACO INTO AC2
MULD AC1,AC2 ;MULTIPLY AC1 BY AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ MERR2 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 21 ;FPS ERROR

MERR2: TST $FPS ;ERROR BIT SET ?
BPL MTST2 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 31 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ MTST2 ;BRANCH IF OK
ERROR 31 ;WRONG ADDRESS IN FEA

MTST2: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ MEND2 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4
  
```

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 62  
DQFPDA.P11 09-FEB-77 10:32 T22 EXERCISE MULD, ALL INTERRUPTS ON, ROUNDING MODE

2502 012004 005637 002410  
2503 012010 005637 002406  
2504 012014 173737 002406  
2505 012020 170000  
2506 012022 001403  
2507 012024 174237 002406  
2508 012030 104012  
2509  
2510 012032 005037 002362  
2511

SBC ANS1+2  
SBC ANS1  
CMPD ANS1,AC3  
CFCC  
BEQ MEND2  
STD AC2,ANS1  
ERROR 12

MEND2: CLR FPS

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

```

2512
2513
2514
2515 012036 000004
2516 012040 012737 012172 002376
2517 012046 012737 004400 002400
2518 012054 005037 002402
2519 012060 005037 002404
2520 012064 005037 002362
2521 012070 005037 002364
2522 012074 005037 002366
2523 012100 004737 023414
2524 012104 002426 002436
2525 012110 004437 023560
2526 012114 023562 002426
2527 012120 023562 002436
2528 012124 025314
2529 012126 023612 002416
2530
2531 012132 013700 002400
2532 012136 170127 040000
2533 012142 172437 002426
2534 012146 172537 002436
2535 012152 172737 002416
2536 012156 170127 004400
2537 012162 012737 012170 001110
2538
2539
2540
2541 012170 172600
2542 012172 171201
2543 012174 170237 002362
2544 012200 023737 002362 002400
2545 012206 001403
2546 012210 174237 002406
2547 012214 104003
2548
2549 012216 173702
2550 012220 170000
2551 012222 001416
2552
2553 012224 174237 002406
2554 012230 162737 000001 002410
2555 012236 005637 002406
2556 012242 173737 002406
2557 012246 170000
2558 012250 001403
2559 012252 174237 002406
2560 012256 104011
2561
2562 012260 005037 002362
2563

```

```

;*****
;TEST 23 EXERCISE MULF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
;*****
†ST23: SCOPE
MOV #MRET3,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)
$MUL ;ADDRESS OF FORTRAN MULTIPLY
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

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

;*****
MRET3: 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 MTST3 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 3 ;FPS ERROR

MTST3: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ MEND3 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1

CMPF ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ MEND3 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 11 ;FPU AND FORTRAN DISAGREE

MEND3: CLR FPS ;CLEAR FPP FPS BUFFER

```

# K05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 64  
 DQFPDA.P11 09-FEB-77 10:32 T24 EXERCISE MULD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE

```

2564
2565
2566
2567 012264 000004
2568 012266 012737 012420 002376
2569 012274 012737 004600 002400
2570 012302 005037 002402
2571 012306 005037 002404
2572 012312 005037 002362
2573 012316 005037 002364
2574 012322 005037 002366
2575 012326 004737 023404
2576 012332 002426 002436
2577 012336 004437 023560
2578 012342 023562 002426
2579 012346 023562 002436
2580 012352 025314
2581 012354 023612 002416
2582
2583 012360 013700 002400
2584 012364 170127 040200
2585 012370 172437 002426
2586 012374 172537 002436
2587 012400 172737 002416
2588 012404 170127 004600
2589 012410 012737 012416 001110
2590
2591
2592
2593 012416 172600
2594 012420 171201
2595 012422 170237 002362
2596 012426 023737 002362 002400
2597 012434 001403
2598 012436 174237 002406
2599 012442 104021
2600
2601 012444 173702
2602 012446 170000
2603 012450 001422
2604
2605 012452 174237 002406
2606 012456 162737 000001 002414
2607 012464 005637 002412
2608 012470 005637 002410
2609 012474 005637 002406
2610 012500 173737 002406
2611 012504 170000
2612 012506 001403
2613 012510 174237 002406
2614 012514 104012
2615
2616 012516 005037 002362

;*****
;TEST 24 EXERCISE MULD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
;*****
†ST24: SCOPE
MOV #MRET4,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)
$MUL ;ADDRESS OF FORTRAN MULTIPLY
$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;SET FD OF FPS ONLY, INTERRUPT DISABLE
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #004600 ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
MRET4: 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 MTST4 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 21 ;FPS ERROR

MTST4: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ MEND4 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4
SBC ANS1+2
SBC ANS1
CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
CFCC ;COPY FLOATING CONDITION CODES
BEQ MEND4 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 12 ;FPU AND FORTRAN DISAGREE

MEND4: CLR FPS ;CLEAR FPP FPS BUFFER
  
```

```

2617
2618
2619
2620 012522 000004
2621 012524 012737 012656 002376
2622 012532 012737 007440 002400
2623 012540 005037 002402
2624 012544 005037 002404
2625 012550 005037 002362
2626 012554 005037 002364
2627 012560 005037 002366
2628 012564 004737 023414
2629 012570 002426 002436
2630 012574 004437 023560
2631 012600 023562 002426
2632 012604 023562 002436
2633 012610 025314
2634 012612 023612 002416
2635
2636 012616 013700 002400
2637 012622 170127 040000
2638 012626 172437 002426
2639 012632 172537 002436
2640 012636 172737 002416
2641 012642 170127 007440
2642 012646 012737 012654 001110
2643
2644
2645
2646 012654 172600
2647 012656 171201
2648 012660 170237 002362
2649 012664 023737 002362 002400
2650 012672 001403
2651 012674 174237 002406
2652 012700 104003
2653
2654 012702 005737 002400
2655 012706 100014
2656 012710 170337 002364
2657
2658 012714 023737 002364 002402
2659 012722 001401
2660 012724 104025
2661
2662 012726 023737 002366 002404
2663 012734 001401
2664 012736 104025
2665
2666 012740 173702
2667 012742 170000
2668 012744 001427
2669
2670 012746 174237 002406
2671 012752 062737 000001 002410
2672 012760 005537 002406

```

```

;*****
;TEST 25 EXERCISE MULF, ALL INTERRUPTS ON, TRUNCATE MODE
;*****
TST25: SCOPE
MOV #MRETS,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007440,$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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;CLEAR THE FPS, INTERRUPT DISABLE
LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007440 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
MRETS: 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 MERRS ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 3 ;FPS ERROR

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

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

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ MTSTS ;BRANCH IF OK
ERROR 25 ;WRONG ADDRESS IN FEA

MTSTS: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ MENDS ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+2 ;INCREMENT FPU ANSWER
ADC ANS1

```



M05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 66  
 DQFPDA.P11 09-FEB-77 10:32 T25 EXERCISE MULF, ALL INTERRUPTS ON, TRUNCATE MODE

2673	012764	173737	002406		CMPF	ANS1,AC3		;CHECK ANSWERS AGAIN
2674	012770	170000			CFCC			;COPY FLOATING CONDITION CODES
2675	012772	001414			BEQ	MENDS		;BRANCH IF OK
2676	012774	162737	000002	002410	SUB	#2,ANS1+2		;DECREMENT FPU ANSWER
2677	013002	005637	002406		SBC	ANS1		
2678	013006	173737	002406		CMPF	ANS1,AC3		;CHECK ANSWERS AGAIN
2679	013012	170000			CFCC			;COPY FLOATING CONDITION CODES
2680	013014	001403			BEQ	MENDS		;BRANCH IF OK
2681	013016	174237	002406		STF	AC2,ANS1		;SAVE FPU ANSWER
2682	013022	104011			ERROR	11		;FPU AND FORTRAN DISAGREE
2683								
2684	013024	005037	002362		MENDS:	CLR	FPS	;CLEAR FPP FPS BUFFER
2685								

# N05

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 67  
 DQFPDA.P11 09-FEB-77 10:32 T26 EXERCISE MULD, ALL INTERRUPTS ON, TRUNCATE MODE

```

2686
2687
2688
2689 013030 000004
2690 013032 012737 013164 002376
2691 013040 012737 007640 002400
2692 013046 005037 002402
2693 013052 005037 002404
2694 013056 005037 002362
2695 013062 005037 002364
2696 013066 005037 002366
2697 013072 004737 023404
2698 013076 002426 002436
2699 013102 004437 023560
2700 013106 023562 002426
2701 013112 023562 002436
2702 013116 025314
2703 013120 023612 002416
2704
2705 013124 013700 002400
2706 013130 170127 040200
2707 013134 172437 002426
2708 013140 172537 002436
2709 013144 172737 002416
2710 013150 170127 007640
2711 013154 012737 013162 001110
2712
2713
2714
2715 013162 172600
2716 013164 171201
2717 013166 170237 002362 002400
2718 013172 023737 002362
2719 013200 001403
2720 013202 174237 002406
2721 013206 104021
2722
2723 013210 005737 002400
2724 013214 100014
2725 013216 170337 002364
2726
2727 013222 023737 002364 002402
2728 013230 001401
2729 013232 104031
2730
2731 013234 023737 002366 002404 15:
2732 013242 001401
2733 013244 104031
2734
2735 013246 173702
2736 013250 170000
2737 013252 001437
2738
2739 013254 174237 002406
2740 013260 062737 000001 002414
2741 013266 005537 002412

```

```

;*****
;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 $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL4 ;GET RANDOM INPUT DATA
;WORD
LONUM,HINUM
JSR R4,$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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;SET FD OF FPS ONLY, INTERRUPT DISABLE
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007640 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
MRET6: 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 MERR6 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 21 ;FPS ERROR

MERR6: TST $FPS ;ERROR BIT SET ?
BPL MTST6 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 15 ;BRANCH IF OK
ERROR 31 ;FEC IS WRONG

15: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ MTST6 ;BRANCH IF OK
ERROR 31 ;WRONG ADDRESS IN FEA

MTST6: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ MEND6 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
ADC ANS1+4

```

B06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 68  
DQFPDA.P11 09-FEB-77 10:32 T26 EXERCISE MULD, ALL INTERRUPTS ON, TRUNCATE MODE

2742	013272	005537	002410		ADC	ANS1+2	
2743	013276	005537	002406		ADC	ANS1	
2744	013302	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
2745	013306	170000			CFCC		;COPY FLOATING CONDITION CODES
2746	013310	001420			BEQ	MEND6	;BRANCH IF OK
2747	013312	162737	000002	002414	SUB	#2,ANS1+6	;DECREMENT FPU ANSWER
2748	013320	005637	002412		SBC	ANS1+4	
2749	013324	005637	002410		SBC	ANS1+2	
2750	013330	005637	002406		SBC	ANS1	
2751	013334	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
2752	013340	170000			CFCC		;COPY FLOATING CONDITION CODES
2753	013342	001403			BEQ	MEND6	;BRANCH IF OK
2754	013344	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
2755	013350	104012			ERROR	12	;FPU AND FORTRAN DISAGREE
2756							
2757	013352	005037	002362	MEND6:	CLR	FPS	;CLEAR FPP FPS BUFFER
2758							

```

2759
2760
2761
2762 013356 000004
2763 013360 012737 013512 002376
2764 013366 012737 004440 002400
2765 013374 005037 002402
2766 013400 005037 002404
2767 013404 005037 002362
2768 013410 005037 002364
2769 013414 005037 002366
2770 013420 004737 023414
2771 013424 002426 002436
2772 013430 004437 023560
2773 013434 023562 002426
2774 013440 023562 002436
2775 013444 025314
2776 013446 023612 002416
2777
2778 013452 013700 002400
2779 013456 170127 040000
2780 013462 172437 002426
2781 013466 172537 002436
2782 013472 172737 002416
2783 013476 170127 004440
2784 013502 012737 013510 001110
2785
2786
2787
2788 013510 172600
2789 013512 171201
2790 013514 170237 002362
2791 013520 023737 002362 002400
2792 013526 001403
2793 013530 174237 002406
2794 013534 104003
2795
2796 013536 173702
2797 013540 170000
2798 013542 001427
2799
2800 013544 174237 002406
2801 013550 062737 000001 002410
2802 013556 005537 002406
2803 013562 173737 002406
2804 013566 170000
2805 013570 001414
2806 013572 162737 000002 002410
2807 013600 005637 002406
2808 013604 173737 002406
2809 013610 170000
2810 013612 001403
2811 013614 174237 002406
2812 013620 104011
2813
2814 013622 005037 002362

```

```

;*****
;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 $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL2 ;GET RANDOM INPUT DATA
;WORD
JSR R4,$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
ERROR 11 ;FPU AND FORTRAN DISAGREE

MEND7: CLR FPS ;CLEAR FPP FPS BUFFER

```

006

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 70  
DQFPDA.P11 09-FEB-77 10:32 T27 EXERCISE MULF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

2815

E06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 71  
 DQFPDA.P11 09-FEB-77 10:32 T30 EXERCISE MULD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

```

2816 ;*****
2817 ;*TEST 30 EXERCISE MULD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
2818 ;*****
2819 013626 000004 TST30: SCOPE
2820 013630 012737 013762 002376 MOV #MRET10,EXPFEA ;ADDR OF INSTR BEING TESTED
2821 013636 012737 004640 002400 MOV #004640,$FPS ;SET IE BITS IN FORTRAN ANSWER
2822 013644 005037 002402 CLR $FEC ;CLR FORTRAN FEC
2823 013650 005037 002404 CLR $FEA ;CLR FORTRAN FEA
2824 013654 005037 002362 CLR FPS ;CLR FPU FPS BUFFER
2825 013660 005037 002364 CLR FEC ;CLR FPU FEC BUFFER
2826 013664 005037 002366 CLR FEA ;CLR FPU FEA BUFFER
2827 013670 004737 023404 JSR PC,RANDL4 ;GET RANDOM INPUT DATA
2828 013674 002426 002436 .WORD LONUM,HINUM
2829 013700 004437 023560 JSR R4,$POLSH ;ENTER POLISH MODE
2830 013704 023562 002426 $PUSH ,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
2831 013710 023562 002436 $PUSH ,HINUM ;PUSH 4 WORDS ON STACK (HINUM)
2832 013714 025314 $MUL ;ADDRESS OF FORTRAN MULTIPLY
2833 013716 023612 002416 $POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE
2834
2835 013722 013700 002400 MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
2836 013726 170127 040200 LDFPS #040200 ;SET FD OF FPS ONLY, INTERRUPT DISABLE
2837 013732 172437 002426 LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
2838 013736 172537 002436 LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
2839 013742 172737 002416 LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
2840 013746 170127 004640 LDFPS #004640 ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
2841 013752 012737 013760 001110 MOV #.+6,$LPADR ;RESET LOOP ADDRESS
2842
2843 ;*****
2844
2845 013760 172600 MRET10: LDD AC0,AC2 ;LOAD AC0 INTO AC2
2846 013762 171201 MULD AC1,AC2 ;MULTIPLY AC1 BY AC2
2847 013764 170237 002362 STFPS FPS ;STORE FLOATING POINT STATUS
2848 013770 023737 002362 002400 CMP FPS,$FPS ;CHECK FPS
2849 013776 001403 BEQ MTST10 ;BRANCH IF OK
2850 014000 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
2851 014004 104021 ERROR 21 ;FPS ERROR
2852
2853 014006 173702 MTST10: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
2854 014010 170000 CFCC ;COPY FLOATING CONDITION CODES
2855 014012 001437 BEQ MEND10 ;ANSWERS CHECK
2856 ;COMPENSATE FOR FORTRAN INACCURACIES.
2857 014014 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
2858 014020 062737 000001 002414 ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
2859 014026 005537 002412 ADC ANS1+4
2860 014032 005537 002410 ADC ANS1+2
2861 014036 005537 002406 ADC ANS1
2862 014042 173737 002406 CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
2863 014046 170000 CFCC ;COPY FLOATING CONDITION CODES
2864 014050 001420 BEQ MEND10 ;BRANCH IF OK
2865 014052 162737 000002 002414 SUB #2,ANS1+6 ;DECREMENT FPU ANSWER
2866 014060 005637 002412 SBC ANS1+4
2867 014064 005637 002410 SBC ANS1+2
2868 014070 005637 002406 SBC ANS1
2869 014074 173737 002406 CMPD ANS1,AC3 ;CHECK ANSWERS AGAIN
2870 014100 170000 CFCC ;COPY FLOATING CONDITION CODES
2871 014102 001403 BEQ MEND10 ;BRANCH IF OK
  
```

F06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 72  
DQFPDA.P11 09-FEB-77 10:32 T30 EXERCISE MULD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

2872 014104 174237 002406  
2873 014110 104012  
2874  
2875 014112 005037 002362  
2876

STD AC2,ANS1 ;SAVE FPU ANSWER  
ERROR 12 ;FPU AND FORTRAN DISAGREE  
MEND10: CLR FPS ;CLEAR FPP FPS BUFFER

```

2877
2878
2879
2880 014116 000004
2881 014120 012737 014252 002376
2882 014126 012737 007400 002400
2883 014134 005037 002402
2884 014140 005037 002404
2885 014144 005037 002362
2886 014150 005037 002364
2887 014154 005037 002366
2888 014160 004737 023414
2889 014164 002426 002436
2890 014170 004437 023560
2891 014174 023562 002426
2892 014200 023562 002436
2893 014204 026434
2894 014206 023612 002416
2895
2896 014212 013700 002400
2897 014216 170127 040000
2898 014222 172437 002426
2899 014226 172537 002436
2900 014232 172737 002416
2901 014236 170127 007400
2902 014242 012737 014250 001110
2903
2904
2905
2906 014250 172600
2907 014252 174601
2908 014254 170237 002362
2909 014260 023737 002362 002400
2910 014266 001403
2911 014270 174237 002406
2912 014274 104016
2913
2914 014276 005737 002400
2915 014302 100014
2916 014304 170337 002364
2917
2918 014310 023737 002364 002402
2919 014316 001401
2920 014320 104026
2921
2922 014322 023737 002366 002404 1$:
2923 014330 001401
2924 014332 104026
2925
2926 014334 173702
2927 014336 170000
2928 014340 001416
2929
2930 014342 174237 002406
2931 014346 162737 000001 002410
2932 014354 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 AC0,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,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 26 ;FEC IS WRONG

1$: CMP FEA,$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

```



H06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 74  
DQFPDA.P11 09-FEB-77 10:32 T31 EXERCISE DIVF, ALL INTERRUPTS ON, ROUNDING MODE

2933	014360	173737	002406	CMPF	ANS1,AC3	;CHECK ANSWERS AGAIN
2934	014364	170000		CFCC		;COPY FLOATING CONDITION CODES
2935	014366	001403		BEQ	DEND1	;BRANCH IF OK
2936	014370	174237	002406	STF	AC2,ANS1	;SAVE FPU ANSWER
2937	014374	104013		ERROR	13	;FPU AND FORTRAN DISAGREE
2938						
2939	014376	005037	002362	DEND1:	CLR FPS	;CLEAR FPP FPS BUFFER
2940						

```

2941
2942
2943
2944 014402 000004
2945 014404 012737 014536 002376
2946 014412 012737 007600 002400
2947 014420 005037 002402
2948 014424 005037 002404
2949 014430 005037 002362
2950 014434 005037 002364
2951 014440 005037 002366
2952 014444 004737 023404
2953 014450 002426 002436
2954 014454 004437 023560
2955 014460 023562 002426
2956 014464 023562 002436
2957 014470 026434
2958 014472 023612 002416
2959
2960 014476 013700 002400
2961 014502 170127 040200
2962 014506 172437 002426
2963 014512 172537 002436
2964 014516 172737 002416
2965 014522 170127 007600
2966 014526 012737 014534 001110
2967
2968
2969
2970 014534 172600
2971 014536 174601
2972 014540 170237 002362
2973 014544 023737 002362 002400
2974 014552 001403
2975 014554 174237 002406
2976 014560 104022
2977
2978 014562 005737 002400
2979 014566 100014
2980 014570 170337 002364
2981
2982 014574 023737 002364 002402
2983 014602 001401
2984 014604 104032
2985
2986 014606 023737 002366 002404
2987 014614 001401
2988 014616 104032
2989
2990 014620 173702
2991 014622 170000
2992 014624 001422
2993
2994 014626 174237 002406
2995 014632 162737 000001 002414
2996 014640 005637 002412

```

```

;*****
;TEST 32 EXERCISE DIVD, ALL INTERRUPTS ON, ROUNDING MODE
;*****
↑ST32: SCOPE
MOV #DRET2,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007600,$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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;SET FID AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007600 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
DRET2: LDD AC0,AC2 ;LOAD AC0 INTO AC2
DIVD AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR2 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 22 ;FPS ERROR

DERR2: TST $FPS ;ERROR BIT SET ?
BPL DTST2 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 32 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ DTST2 ;BRANCH IF OK
ERROR 32 ;WRONG ADDRESS IN FEA

DTST2: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND2 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4

```

J06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 76  
DQFPDA.P11 09-FEB-77 10:32 T32 EXERCISE DIVD, ALL INTERRUPTS ON, ROUNDING MODE

2997 014644 005637 002410  
2998 014650 005637 002406  
2999 014654 173737 002406  
3000 014660 170000  
3001 014662 001403  
3002 014664 174237 002406  
3003 014670 104014  
3004  
3005 014672 005037 002362  
3006

SBC ANS1+2  
SBC ANS1  
CMPD ANS1,AC3  
CFCC  
BEQ DEND2  
STD AC2,ANS1  
ERROR 14  
DEND2: CLR FPS

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

K06

```

3007
3008
3009
3010 014676 000004
3011 014700 012737 015032 002376
3012 014706 012737 004400 002400
3013 014714 005037 002402
3014 014720 005037 002404
3015 014724 005037 002362
3016 014730 005037 002364
3017 014734 005037 002366
3018 014740 004737 023414
3019 014744 002426 002436
3020 014750 004437 023560
3021 014754 023562 002426
3022 014760 023562 002436
3023 014764 026434
3024 014766 023612 002416
3025
3026 014772 013700 002400
3027 014776 170127 040000
3028 015002 172437 002426
3029 015006 172537 002436
3030 015012 172737 002416
3031 015016 170127 004400
3032 015022 012737 015030 001110
3033
3034
3035
3036 015030 172600
3037 015032 174601
3038 015034 170237 002362
3039 015040 023737 002362 002400
3040 015046 001403
3041 015050 174237 002406
3042 015054 104016
3043
3044 015056 005737 002400
3045 015062 100014
3046 015064 170337 002364
3047
3048 015070 023737 002364 002402
3049 015076 001401
3050 015100 104026
3051
3052 015102 023737 002366 002404
3053 015110 001401
3054 015112 104026
3055
3056 015114 173702
3057 015116 170000
3058 015120 001416
3059
3060 015122 174237 002406
3061 015126 162737 000001 002410
3062 015134 005637 002406

```

```

;*****
;TEST 33 EXERCISE DIVF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
;*****
TST33: 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
CMP FPS,$FPS ;CHECK FPS
BEQ DERR3 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 16 ;FPS ERROR

DERR3: TST $FPS ;ERROR BIT SET ?
BPL DTST3 ;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 DTST3 ;BRANCH IF OK
ERROR 26 ;WRONG ADDRESS IN FEA

DTST3: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND3 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1

```

L06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 78  
DQFPA.P11 09-FEB-77 10:32 T33 EXERCISE DIVF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE

3063	015140	173737	002406	CMPF	ANS1,AC3	;CHECK ANSWERS AGAIN
3064	015144	170000		CFCC		;COPY FLOATING CONDITION CODES
3065	015146	001403		BEQ	DEND3	;BRANCH IF OK
3066	015150	174237	002406	STF	AC2,ANS1	;SAVE FPU ANSWER
3067	015154	104013		ERROR	13	;FPU AND FORTRAN DISAGREE
3068						
3069	015156	005037	002362	DEND3:	CLR FPS	;CLEAR FPP FPS BUFFER
3070						

M06

Address	Op1	Op2	Op3	Op4	Instruction	Comment
3071						
3072						
3073						
3074	015162	000004				
3075	015164	012737	015316	002376		
3076	015172	012737	004600	002400		
3077	015200	005037	002402			
3078	015204	005037	002404			
3079	015210	005037	002362			
3080	015214	005037	002364			
3081	015220	005037	002366			
3082	015224	004737	023404			
3083	015230	002426	002436			
3084	015234	004437	023560			
3085	015240	023562	002426			
3086	015244	023562	002436			
3087	015250	026434				
3088	015252	023612	002416			
3089						
3090	015256	013700	002400			
3091	015262	170127	040200			
3092	015266	172437	002426			
3093	015272	172537	002436			
3094	015276	172737	002416			
3095	015302	170127	004600			
3096	015306	012737	015314	001110		
3097						
3098						
3099						
3100	015314	172600				
3101	015316	174601				
3102	015320	170237	002362			
3103	015324	023737	002362	002400		
3104	015332	001403				
3105	015334	174237	002406			
3106	015340	104022				
3107						
3108	015342	005737	002400			
3109	015346	100014				
3110	015350	170337	002364			
3111						
3112	015354	023737	002364	002402		
3113	015362	001401				
3114	015364	104032				
3115						
3116	015366	023737	002366	002404		
3117	015374	001401				
3118	015376	104032				
3119						
3120	015400	173702				
3121	015402	170000				
3122	015404	001422				
3123						
3124	015406	174237	002406			
3125	015412	162737	000001	002414		
3126	015420	005637	002412			

```

*****
;TEST 34 EXERCISE DIVD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE
*****
↑ST34: SCOPE
MOV #DRET4,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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;SET FID AND FD
LDD LONUM,ACO ;LOAD ACO WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #004600 ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

*****
DRET4: LDD ACO,AC2 ;LOAD ACO INTO AC2
DIVD AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR4 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 22 ;FPS ERROR

DERR4: TST $FPS ;ERROR BIT SET ?
BPL DTST4 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 15 ;BRANCH IF OK
ERROR 32 ;FEC IS WRONG

15: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ DTST4 ;BRANCH IF OK
ERROR 32 ;WRONG ADDRESS IN FEA

DTST4: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND4 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4

```

N06

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006)  
DQFPDA.P11 09-FEB-77 10:32 T34

09-FEB-77 10:34 PAGE 80  
EXERCISE DIVD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, ROUNDING MODE

3127 015424 005637 002410  
3128 015430 005637 002406  
3129 015434 173737 002406  
3130 015440 170000  
3131 015442 001403  
3132 015444 174237 002406  
3133 015450 104014  
3134  
3135 015452 005037 002362  
3136

SBC ANS1+2  
SBC ANS1  
CMPD ANS1,AC3  
CFCC  
BEQ DEND4  
STD AC2,ANS1  
ERROR 14

;CHECK ANSWERS AGAIN  
;COPY FLOATING CONDITION CODES  
;BRANCH IF OK  
;SAVE FPU ANSWER  
;FPU AND FORTRAN DISAGREE

DEND4: CLR FPS

;CLEAR FPP FPS BUFFER

```

3137
3138
3139
3140 015456 000004
3141 015460 012737 015612 002376
3142 015466 012737 007440 002400
3143 015474 005037 002402
3144 015500 005037 002404
3145 015504 005037 002362
3146 015510 005037 002364
3147 015514 005037 002366
3148 015520 004737 023414
3149 015524 002426 002436
3150 015530 004437 023560
3151 015534 023562 002426
3152 015540 023562 002436
3153 015544 026434
3154 015546 023612 002416
3155
3156 015552 013700 002400
3157 015556 170127 040000
3158 015562 172437 002426
3159 015566 172537 002436
3160 015572 172737 002416
3161 015576 170127 007440
3162 015602 012737 015610 001110
3163
3164
3165
3166 015610 172600
3167 015612 174601
3168 015614 170237 002362
3169 015620 023737 002362 002400
3170 015626 001403
3171 015630 174237 002406
3172 015634 104016
3173
3174 015636 005737 002400
3175 015642 100014
3176 015644 170337 002364
3177
3178 015650 023737 002364 002402
3179 015656 001401
3180 015660 104026
3181
3182 015662 023737 002366 002404
3183 015670 001401
3184 015672 104026
3185
3186 015674 173702
3187 015676 170000
3188 015700 001427
3189
3190 015702 174237 002406
3191 015706 062737 000001 002410
3192 015714 005537 002406

```

```

;*****
;TEST 35 EXERCISE DIVF, ALL INTERRUPTS ON, TRUNCATE MODE
;*****
↑T35: SCOPE
MOV #DRETS,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007440,$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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;SET FID
LDF LONUM,ACO ;LOAD ACO WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007440 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
DRETS: LDF ACO,AC2 ;LOAD ACO INTO AC2
DIVF AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR5 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 16 ;FPS ERROR

DERR5: TST $FPS ;ERROR BIT SET ?
BPL DTST5 ;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 DTST5 ;BRANCH IF OK
ERROR 26 ;WRONG ADDRESS IN FEA

DTST5: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND5 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+2 ;INCREMENT FPU ANSWER
ADC ANS1

```



C07

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 82  
DQFPDA.P11 09-FEB-77 10:32 T35 EXERCISE DIVF, ALL INTERRUPTS ON, TRUNCATE MODE

3193	015720	173737	002406		CMPF	ANS1,AC3		;CHECK ANSWERS AGAIN
3194	015724	170000			CFCC			;COPY FLOATING CONDITION CODES
3195	015726	001414			BEQ	DEND5		;BRANCH IF OK
3196	015730	162737	000002	002410	SUB	#2,ANS1+2		;DECREMENT FPU ANSWER
3197	015736	005637	002406		SBC	ANS1		
3198	015742	173737	002406		CMPF	ANS1,AC3		;CHECK ANSWERS AGAIN
3199	015746	170000			CFCC			;COPY FLOATING CONDITION CODES
3200	015750	001403			BEQ	DEND5		;BRANCH IF OK
3201	015752	174237	002406		STF	AC2,ANS1		;SAVE FPU ANSWER
3202	015756	104013			ERROR	13		;FPU AND FORTRAN DISAGREE
3203								
3204	015760	005037	002362		DEND5:	CLR	FPS	;CLEAR FPP FPS BUFFER
3205								

```

3206
3207
3208
3209 015764 000004
3210 015766 012737 016120 002376
3211 015774 012737 007640 002400
3212 016002 005037 002402
3213 016006 005037 002404
3214 016012 005037 002362
3215 016016 005037 002364
3216 016022 005037 002366
3217 016026 004737 023404
3218 016032 002426 002436
3219 016036 004437 023560
3220 016042 023562 002426
3221 016046 023562 002436
3222 016052 026434
3223 016054 023612 002416
3224
3225 016060 013700 002400
3226 016064 170127 040200
3227 016070 172437 002426
3228 016074 172537 002436
3229 016100 172737 002416
3230 016104 170127 007640
3231 016110 012737 016116 001110
3232
3233
3234
3235 016116 172600
3236 016120 174601
3237 016122 170237 002362
3238 016126 023737 002362 002400
3239 016134 001403
3240 016136 174237 002406
3241 016142 104022
3242
3243 016144 005737 002400
3244 016150 100014
3245 016152 170337 002364
3246
3247 016156 023737 002364 002402
3248 016164 001401
3249 016166 104032
3250
3251 016170 023737 002366 002404
3252 016176 001401
3253 016200 104032
3254
3255 016202 173702
3256 016204 170000
3257 016206 001437
3258
3259 016210 174237 002406
3260 016214 062737 000001 002414
3261 016222 005537 002412

```

```

;*****
;TEST 36 EXERCISE DIVD, ALL INTERRUPTS ON, TRUNCATE MODE
;*****
†ST36: SCOPE
MOV #DRET6,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #007640,$FPS ;SET IE BITS IN FORTRAN ANSWER
CLR $FEC ;CLR FORTRAN FEC
CLR $FEA ;CLR FORTRAN FEA
CLR FPS ;CLR FPU FPS BUFFER
CLR FEC ;CLR FPU FEC BUFFER
CLR FEA ;CLR FPU FEA BUFFER
JSR PC,RANDL4 ;GET RANDOM INPUT DATA
.WORD LONUM,HINUM
JSR R4,$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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;SET FID AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #007640 ;TURN INTERRUPTS ON
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
DRET6: LDD AC0,AC2 ;LOAD AC0 INTO AC2
DIVD AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR6 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 22 ;FPS ERROR

DERR6: TST $FPS ;ERROR BIT SET ?
BPL DTST6 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 32 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ DTST6 ;BRANCH IF OK
ERROR 32 ;WRONG ADDRESS IN FEA

DTST6: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND6 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
ADC ANS1+4

```

E07

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 84  
DQFPDA.P11 09-FEB-77 10:32 T36 EXERCISE DIVD, ALL INTERRUPTS ON, TRUNCATE MODE

3262	016226	005537	002410		ADC	ANS1+2	
3263	016232	005537	002406		ADC	ANS1	
3264	016236	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
3265	016242	170000			CFCC		;COPY FLOATING CONDITION CODES
3266	016244	001420			BEQ	DEND6	;BRANCH IF OK
3267	016246	162737	000002	002414	SUB	#2,ANS1+6	;DECREMENT FPU ANSWER
3268	016254	005637	002412		SBC	ANS1+4	
3269	016260	005637	002410		SBC	ANS1+2	
3270	016264	005637	002406		SBC	ANS1	
3271	016270	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
3272	016274	170000			CFCC		;COPY FLOATING CONDITION CODES
3273	016276	001403			BEQ	DEND6	;BRANCH IF OK
3274	016300	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
3275	016304	104014			ERROR	14	;FPU AND FORTRAN DISAGREE
3276							
3277	016306	005037	002362	DEND6:	CLR	FPS	;CLEAR FPP FPS BUFFER
3278							

# F07

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 85  
 DQFPDA.P11 09-FEB-77 10:32 T37 EXERCISE DIVF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

```

3279
3280
3281
3282 016312 000004
3283 016314 012737 016446 002376
3284 016322 012737 004440 002400
3285 016330 005037 002402
3286 016334 005037 002404
3287 016340 005037 002362
3288 016344 005037 002364
3289 016350 005037 002366
3290 016354 004737 023414
3291 016360 002426 002436
3292 016364 004437 023560
3293 016370 023562 002426
3294 016374 023562 002436
3295 016400 026434
3296 016402 023612 002416
3297
3298 016406 013700 002400
3299 016412 170127 040000
3300 016416 172437 002426
3301 016422 172537 002436
3302 016426 172737 002416
3303 016432 170127 004440
3304 016436 012737 016444 001110
3305
3306
3307
3308 016444 172600
3309 016446 174601
3310 016450 170237 002362 002400
3311 016454 023737 002362
3312 016462 001403
3313 016464 174237 002406
3314 016470 104016
3315
3316 016472 005737 002400
3317 016476 100014
3318 016500 170337 002364
3319
3320 016504 023737 002364 002402
3321 016512 001401
3322 016514 104026
3323
3324 016516 023737 002366 002404
3325 016524 001401
3326 016526 104026
3327
3328 016530 173702
3329 016532 170000
3330 016534 001427
3331
3332 016536 174237 002406
3333 016542 062737 000001 002410
3334 016550 005537 002406

```

```

;*****
;TEST 37 EXERCISE DIVF, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
;*****
†ST37: SCOPE
MOV #DRET7,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)
$DIV ;ADDRESS OF FORTRAN DIVIDE
$POPX ,ANS2 ;POP 2 WORDS AND EXIT POLISH MODE

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

;*****
DRET7: LDF AC0,AC2 ;LOAD AC0 INTO AC2
DIVF AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR7 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 16 ;FPS ERROR

DERR7: TST $FPS ;ERROR BIT SET ?
BPL DTST7 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 26 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ DTST7 ;BRANCH IF OK
ERROR 26 ;WRONG ADDRESS IN FEA

DTST7: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND7 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+2 ;INCREMENT FPU ANSWER
ADC ANS1

```



# H07

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 87  
 DQFPDA.P11 09-FEB-77 10:32 T40 EXERCISE DIVD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE

```

3348 ;*****
3349 ;*TEST 40 EXERCISE DIVD, OVERFLOW AND UNDERFLOW INTERRUPTS OFF, TRUNCATE MODE
3350 ;*****
3351 016620 000004 TST40: SCOPE
3352 016622 012737 016754 002376 MOV #DRET10,EXPFEA ;ADDR OF INSTR BEING TESTED
3353 016630 012737 004640 002400 MOV #004640,$FPS ;SET IE BITS IN FORTRAN ANSWER
3354 016636 005037 002402 CLR $FEC ;CLR FORTRAN FEC
3355 016642 005037 002404 CLR $FEA ;CLR FORTRAN FEA
3356 016646 005037 002362 CLR FPS ;CLR FPU FPS BUFFER
3357 016652 005037 002364 CLR FEC ;CLR FPU FEC BUFFER
3358 016656 005037 002366 CLR FEA ;CLR FPU FEA BUFFER
3359 016662 004737 023404 JSR PC,RANDL4 ;GET RANDOM INPUT DATA
3360 016666 002426 002436 .WORD LONUM,HINUM
3361 016672 004437 023560 JSR R4,$POLSH ;ENTER POLISH MODE
3362 016676 023562 002426 $PUSH ,LONUM ;PUSH 4 WORDS ON STACK (LONUM)
3363 016702 023562 002436 $PUSH ,HINUM ;PUSH 4 WORDS ON STACK (HINUM)
3364 016706 026434 $DIV ;ADDRESS OF FORTRAN DIVIDE
3365 016710 023612 002416 $POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE
3366
3367 016714 013700 002400 MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
3368 016720 170127 040200 LDFPS #040200 ;SET FID AND FD
3369 016724 172437 002426 LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
3370 016730 172537 002436 LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
3371 016734 172737 002416 LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
3372 016740 170127 004640 LDFPS #004640 ;TURN INTERRUPTS ON, EXCEPT OVER AND UNDERFLOW
3373 016744 012737 016752 001110 MOV #.+6,$LPADR ;RESET LOOP ADDRESS
3374
3375 ;*****
3376
3377 016752 172600 DRET10: LDD AC0,AC2 ;LOAD AC0 INTO AC2
3378 016754 174601 DIVD AC1,AC2 ;DIVIDE AC1 INTO AC2
3379 016756 170237 002362 STFPS FPS ;STORE FLOATING POINT STATUS
3380 016762 023737 002362 002400 CMP FPS,$FPS ;CHECK FPS
3381 016770 001403 BEQ DERR10 ;BRANCH IF OK
3382 016772 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
3383 016776 104022 ERROR 22 ;FPS ERROR
3384
3385 017000 005737 002400 DERR10: TST $FPS ;ERROR BIT SET ?
3386 017004 100014 BPL DTST10 ;NO, DONT GET FEC/FEA
3387 017006 170337 002364 STST FEC ;YES, CHECK STATUS
3388
3389 017012 023737 002364 002402 CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
3390 017020 001401 BEQ 1$ ;BRANCH IF OK
3391 017022 104032 ERROR 32 ;FEC IS WRONG
3392
3393 017024 023737 002366 002404 1$: CMP FEA,$FEA ;CHECK FLOATING PC
3394 017032 001443 BEQ DEND10 ;BRANCH IF OK
3395 017034 104032 ERROR 32 ;WRONG ADDRESS IN FEA
3396
3397 017036 173702 DTST10: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
3398 017040 170000 CFCC ;COPY FLOATING CONDITION CODES
3399 017042 001437 BEQ DEND10 ;ANSWERS CHECK
3400 ;COMPENSATE FOR FORTRAN INACCURACIES.
3401 017044 174237 002406 STD AC2,ANS1 ;SAVE FPU ANSWER
3402 017050 062737 000001 002414 ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
3403 017056 005537 002412 ADC ANS1+4

```

3404	017062	005537	002410		ADC	ANS1+2	
3405	017066	005537	002406		ADC	ANS1	
3406	017072	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
3407	017076	170000			CFCC		;COPY FLOATING CONDITION CODES
3408	017100	001420			BEQ	DEND10	;BRANCH IF OK
3409	017102	162737	000002	002414	SUB	#2,ANS1+6	;DECREMENT FPU ANSWER
3410	017110	005637	002412		SBC	ANS1+4	
3411	017114	005637	002410		SBC	ANS1+2	
3412	017120	005637	002406		SBC	ANS1	
3413	017124	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
3414	017130	170000			CFCC		;COPY FLOATING CONDITION CODES
3415	017132	001403			BEQ	DEND10	;BRANCH IF OK
3416	017134	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
3417	017140	104014			ERROR	14	;FPU AND FORTRAN DISAGREE
3418							
3419	017142	005037	002362		DEND10: CLR	FPS	;CLEAR FPP FPS BUFFER
3420							

```

3421
3422
3423
3424 017146 000004
3425 017150 012737 017302 002376
3426 017156 012737 047400 002400
3427 017164 005037 002402
3428 017170 005037 002404
3429 017174 005037 002362
3430 017200 005037 002364
3431 017204 005037 002366
3432 017210 004737 023414
3433 017214 002426 002436
3434 017220 004437 023560
3435 017224 023562 002426
3436 017230 023562 002436
3437 017234 026434
3438 017236 023612 002416
3439
3440 017242 013700 002400
3441 017246 170127 040000
3442 017252 172437 002426
3443 017256 172537 002436
3444 017262 172737 002416
3445 017266 170127 047400
3446 017272 012737 017300 001110
3447
3448
3449
3450 017300 172600
3451 017302 174601
3452 017304 170237 002362
3453 017310 023737 002362 002400
3454 017316 001403
3455 017320 174237 002406
3456 017324 104016
3457
3458 017326 005737 002400
3459 017332 100014
3460 017334 170337 002364
3461
3462 017340 023737 002364 002402
3463 017346 001401
3464 017350 104026
3465
3466 017352 023737 002366 002404
3467 017360 001401
3468 017362 104026
3469
3470 017364 173702
3471 017366 170000
3472 017370 001416
3473
3474 017372 174237 002406
3475 017376 162737 000001 002410
3476 017404 005637 002406
;*****
;TEST 41 EXERCISE DIVF, INTERRUPT DISABLE SET, ROUNDING MODE
;*****
TST41: SCOPE
MOV #DRET11,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #047400,$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,SPOLSH ;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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;SET FID
LDF LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #047400 ;SET INTERRUPT DISABLE AND INTERRUPT BITS
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
DRET11: LDF AC0,AC2 ;LOAD AC0 INTO AC2
DIVF AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR11 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 16 ;FPS ERROR

DERR11: TST $FPS ;ERROR BIT SET ?
BPL DTST11 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 26 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ DTST11 ;BRANCH IF OK
ERROR 26 ;WRONG ADDRESS IN FEA

DTST11: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND11 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+2 ;DECREMENT FPU ANSWER
SBC ANS1
    
```



K07

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 90  
DQFPDA.P11 09-FEB-77 10:32 T41 EXERCISE DIVF, INTERRUPT DISABLE SET, ROUNDING MODE

3477	017410	173737	002406	CMPF	ANS1,AC3	;CHECK ANSWERS AGAIN
3478	017414	170000		CFCC		;COPY FLOATING CONDITION CODES
3479	017416	001403		BEG	DEND11	;BRANCH IF OK
3480	017420	174237	002406	STF	AC2,ANS1	;SAVE FPU ANSWER
3481	017424	104013		ERROR	13	;FPU AND FORTRAN DISAGREE
3482						
3483	017426	005037	002362	DEND11: CLR	FPS	;CLEAR FPP FPS BUFFER

```

3484
3485
3486
3487 017432 000004
3488 017434 012737 017566 002376
3489 017442 012737 047600 002400
3490 017450 005037 002402
3491 017454 005037 002404
3492 017460 005037 002362
3493 017464 005037 002364
3494 017470 005037 002366
3495 017474 004737 023404
3496 017500 002426 002436
3497 017504 004437 023560
3498 017510 023562 002426
3499 017514 023562 002436
3500 017520 026434
3501 017522 023612 002416
3502
3503 017526 013700 002400
3504 017532 170127 040200
3505 017536 172437 002426
3506 017542 172537 002436
3507 017546 172737 002416
3508 017552 170127 047600
3509 017556 012737 017564 001110
3510
3511
3512
3513 017564 172600
3514 017566 174601
3515 017570 170237 002362
3516 017574 023737 002362 002400
3517 017602 001403
3518 017604 174237 002406
3519 017610 104022
3520
3521 017612 005737 002400
3522 017616 100014
3523 017620 170337 002364
3524
3525 017624 023737 002364 002402
3526 017632 001401
3527 017634 104032
3528
3529 017636 023737 002366 002404 15:
3530 017644 001401
3531 017646 104032
3532
3533 017650 173702
3534 017652 170000
3535 017654 001422
3536
3537 017656 174237 002406
3538 017662 162737 000001 002414
3539 017670 005637 002412

```

```

;*****
;TEST 42 EXERCISE DIVD, INTERRUPT DISABLE SET, ROUNDING MODE
;*****
†ST42: SCOPE
MOV #DRET12,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #047600,$FPS ;SET FID AND 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)
$DIV ;ADDRESS OF FORTRAN DIVIDE
$POPX ,ANS2 ;POP 4 WORDS AND EXIT POLISH MODE

MOV $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;SET FID AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #047600 ;SET INTERRUPT DISABLE AND INTERRUPT BITS
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
DRET12: LDD AC0,AC2 ;LOAD AC0 INTO AC2
DIVD AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR12 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 22 ;FPS ERROR

DERR12: TST $FPS ;ERROR BIT SET ?
BPL DTST12 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 15 ;BRANCH IF OK
ERROR 32 ;FEC IS WRONG

15: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ DTST12 ;BRANCH IF OK
ERROR 32 ;WRONG ADDRESS IN FEA

DTST12: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND12 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
SUB #1,ANS1+6 ;DECREMENT FPU ANSWER
SBC ANS1+4

```

M07

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 92  
DQFPDA.P11 09-FEB-77 10:32 T42 EXERCISE DIVD, INTERRUPT DISABLE SET, ROUNDING MODE

3540	017674	005637	002410	SBC	ANS1+2	
3541	017700	005637	002406	SBC	ANS1	
3542	017704	173737	002406	CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
3543	017710	170000		CFCC		;COPY FLOATING CONDITION CODES
3544	017712	001403		BEQ	DEND12	;BRANCH IF OK
3545	017714	174237	002406	STD	AC2,ANS1	;SAVE FPU ANSWER
3546	017720	104014		ERROR	14	;FPU AND FORTRAN DISAGREE
3547						
3548	017722	005037	002362	DEND12: CLR	FPS	;CLEAR FPP FPS BUFFER
3549						

```

3550
3551
3552
3553 017726 000004
3554 017730 012737 020062 002376
3555 017736 012737 047440 002400
3556 017744 005037 002402
3557 017750 005037 002404
3558 017754 005037 002362
3559 017760 005037 002364
3560 017764 005037 002366
3561 017770 004737 023414
3562 017774 002426 002436
3563 020000 004437 023560
3564 020004 023562 002426
3565 020010 023562 002436
3566 020014 026434
3567 020016 023612 002416
3568
3569 020022 013700 002400
3570 020026 170127 040000
3571 020032 172437 002426
3572 020036 172537 002436
3573 020042 172737 002416
3574 020046 170127 047440
3575 020052 012737 020060 001110
3576
3577
3578
3579 020060 172600
3580 020062 174601
3581 020064 170237 002362 002400
3582 020070 023737 002362
3583 020076 001403
3584 020100 174237 002406
3585 020104 104016
3586
3587 020106 005737 002400
3588 020112 100014
3589 020114 170337 002364
3590
3591 020120 023737 002364 002402
3592 020126 001401
3593 020130 104026
3594
3595 020132 023737 002366 002404
3596 020140 001401
3597 020142 104026
3598
3599 020144 173702
3600 020146 170000
3601 020150 001427
3602
3603 020152 174237 002406
3604 020156 062737 000001 002410
3605 020164 005537 002406

```

```

*****
*TEST 43 EXERCISE DIVF, INTERRUPT DISABLE SET, TRUNCATE MODE
*****
TST43: SCOPE
MOV #DRET13,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #047440,$FPS ;SET FID AND 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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040000 ;SET FID
LDF LONUM,ACO ;LOAD ACO WITH A RANDOM NUMBER
LDF HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDF ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #047440 ;SET INTERRUPT DISABLE AND INTERRUPT BITS
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

*****
DRET13: LDF ACO,AC2 ;LOAD ACO INTO AC2
DIVF AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR13 ;BRANCH IF OK
STF AC2,ANS1 ;SAVE FPU ANSWER
ERROR 16 ;FPS ERROR

DERR13: TST $FPS ;ERROR BIT SET ?
BPL DTST13 ;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 DTST13 ;BRANCH IF OK
ERROR 26 ;WRONG ADDRESS IN FEA

DTST13: CMPF AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND13 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STF AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+2 ;INCREMENT FPU ANSWER
ADC ANS1

```

3606	020170	173737	002406		CMPF	ANS1,AC3		;CHECK ANSWERS AGAIN
3607	020174	170000			CFCC			;COPY FLOATING CONDITION CODES
3608	020176	001414			BEQ	DEND13		;BRANCH IF OK
3609	020200	162737	000002	002410	SUB	#2,ANS1+2		;DECREMENT FPU ANSWER
3610	020206	005637	002406		SBC	ANS1		
3611	020212	173737	002406		CMPF	ANS1,AC3		;CHECK ANSWERS AGAIN
3612	020216	170000			CFCC			;COPY FLOATING CONDITION CODES
3613	020220	001403			BEQ	DEND13		;BRANCH IF OK
3614	020222	174237	002406		STF	AC2,ANS1		;SAVE FPU ANSWER
3615	020226	104013			ERROR	13		;FPU AND FORTRAN DISAGREE
3616								
3617	020230	005037	002362		DEND13: CLR	FPS		;CLEAR FPP FPS BUFFER

```

3618
3619
3620
3621 020234 000004
3622 020236 012737 020370 002376
3623 020244 012737 047640 002400
3624 020252 005037 002402
3625 020256 005037 002404
3626 020262 005037 002362
3627 020266 005037 002364
3628 020272 005037 002366
3629 020276 004737 023404
3630 020302 002426 002436
3631 020306 004437 023560
3632 020312 023562 002426
3633 020316 023562 002436
3634 020322 026434
3635 020324 023612 002416
3636
3637 020330 013700 002400
3638 020334 170127 040200
3639 020340 172437 002426
3640 020344 172537 002436
3641 020350 172737 002416
3642 020354 170127 047640
3643 020360 012737 020366 001110
3644
3645
3646
3647 020366 172600
3648 020370 174601
3649 020372 170237 002362 002400
3650 020376 023737 002362
3651 020404 001403
3652 020406 174237 002406
3653 020412 104022
3654
3655 020414 005737 002400
3656 020420 100014
3657 020422 170337 002364
3658
3659 020426 023737 002364 002402
3660 020434 001401
3661 020436 104032
3662
3663 020440 023737 002366 002404
3664 020446 001401
3665 020450 104032
3666
3667 020452 173702
3668 020454 170000
3669 020456 001437
3670
3671 020460 174237 002406
3672 020464 062737 000001 002414
3673 020472 005537 002412

;*****
;TEST 44 EXERCISE DIVD, INTERRUPT DISABLE SET, TRUNCATE MODE
;*****
†ST44: SCOPE
MOV #DRET14,EXPFEA ;ADDR OF INSTR BEING TESTED
MOV #047640,$FPS ;SET FID AND 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 $FPS,R0 ;DISPLAY FLOATING POINT STATUS
LDFPS #040200 ;SET FID AND FD
LDD LONUM,AC0 ;LOAD AC0 WITH A RANDOM NUMBER
LDD HINUM,AC1 ;LOAD AC1 WITH A RANDOM NUMBER
LDD ANS2,AC3 ;LOAD AC3 WITH THE SUM
LDFPS #047640 ;SET INTERRUPT DISABLE AND INTERRUPT BITS
MOV #.+6,$LPADR ;RESET LOOP ADDRESS

;*****
DRET14: LDD AC0,AC2 ;LOAD AC0 INTO AC2
DIVD AC1,AC2 ;DIVIDE AC1 INTO AC2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP FPS,$FPS ;CHECK FPS
BEQ DERR14 ;BRANCH IF OK
STD AC2,ANS1 ;SAVE FPU ANSWER
ERROR 22 ;FPS ERROR

DERR14: TST $FPS ;ERROR BIT SET ?
BPL DTST14 ;NO, DONT GET FEC/FEA
STST FEC ;YES, CHECK STATUS

CMP FEC,$FEC ;CHECK THE FLOATING EXCEPTION CODES
BEQ 1$ ;BRANCH IF OK
ERROR 32 ;FEC IS WRONG

1$: CMP FEA,$FEA ;CHECK FLOATING PC
BEQ DTST14 ;BRANCH IF OK
ERROR 32 ;WRONG ADDRESS IN FEA

DTST14: CMPD AC2,AC3 ;COMPARE FPU ANSWER TO FORTRAN ANSWER
CFCC ;COPY FLOATING CONDITION CODES
BEQ DEND14 ;ANSWERS CHECK
;COMPENSATE FOR FORTRAN INACCURACIES.
STD AC2,ANS1 ;SAVE FPU ANSWER
ADD #1,ANS1+6 ;INCREMENT FPU ANSWER
ADC ANS1+4
    
```

3674	020476	005537	002410		ADC	ANS1+2	
3675	020502	005537	002406		ADC	ANS1	
3676	020506	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
3677	020512	170000			CFCC		;COPY FLOATING CONDITION CODES
3678	020514	001420			BEQ	DEND14	;BRANCH IF OK
3679	020516	162737	000002	002414	SUB	#2,ANS1+6	;DECREMENT FPU ANSWER
3680	020524	005637	002412		SBC	ANS1+4	
3681	020530	005637	002410		SBC	ANS1+2	
3682	020534	005637	002406		SBC	ANS1	
3683	020540	173737	002406		CMPD	ANS1,AC3	;CHECK ANSWERS AGAIN
3684	020544	170000			CFCC		;COPY FLOATING CONDITION CODES
3685	020546	001403			BEQ	DEND14	;BRANCH IF OK
3686	020550	174237	002406		STD	AC2,ANS1	;SAVE FPU ANSWER
3687	020554	104014			ERROR	14	;FPU AND FORTRAN DISAGREE
3688							
3689	020556	005037	002362		DEND14: CLR	FPS	;CLEAR FPP FPS BUFFER

```

3690
3691
3692
3693 020562 000004
3694
3695
3696 020564 012737 000440 002400 MOV #440,$FPS ;SOFTWARE STATUS
3697 020572 005037 002402 CLR $FEC ;CLEAR STATUS
3698 020576 005037 002404 CLR $FEA
3699 020602 005037 002362 CLR FPS
3700 020606 005037 002364 CLR FEC
3701 020612 005037 002366 CLR FEA
3702
3703 020616 004737 023414 JSR PC,RANDL2 ;GET 6 FLOAT RANDOM NUMBERS
3704 020622 002446 002476 .WORD OP1,OP4
3705 020626 004737 023414 JSR PC,RANDL2
3706 020632 002456 002506 .WORD OP2,OP5
3707 020636 004737 023414 JSR PC,RANDL2
3708 020642 002466 002516 .WORD OP3,OP6
3709 020646 032737 077600 002516 BIT #077600,OP6 ;LET'S NEVER DIVIDE BY ZERO
3710 020654 001770 BEQ IS ;EXPO OF OP6 IS ZERO, SO GET ANOTHER
3711
3712 020656 004437 023560 JSR R4,$POLSH ;ENTER POLISH MODE TO CALCULATE:
3713 020662 023562 002446 $PUSH ,OP1
3714 020666 023562 002456 $PUSH ,OP2
3715 020672 023634 $SUB ;ANS2 = (OP1-OP2) * (OP3+OP4/OP6) * OP5
3716 020674 023562 002466 $PUSH ,OP3
3717 020700 023562 002476 $PUSH ,OP4
3718 020704 023562 002516 $PUSH ,OP6
3719 020710 026434 $DIV
3720 020712 023640 $ADD
3721 020714 025314 $MUL
3722 020716 023562 002506 $PUSH ,OP5
3723 020722 025314 $MUL
3724 020724 023612 002416 $POPX ,ANS2
3725
3726 020730 170127 040000 LDFPS #040000 ;NO CHECKS
3727 020734 172437 002416 LDF ANS2,AC0 ;GET SOFTWARE ANSWER
3728 020740 013700 002400 MOV $FPS,RO ;DISPLAY $FPS
3729 020744 012737 020752 001110 MOV #.+6,$LPADR ;RESET LOOP ADDRESS
3730
3731
3732
3733 020752 170127 000440 LDFPS #000440 ;INITIAL FPS
3734 020756 172537 002446 LDF OP1,AC1 ;AC1 <- OP1
3735 020762 173137 002456 SUBF OP2,AC1 ;AC1 <- OP1-OP2
3736 020766 172637 002476 LDF OP4,AC2 ;AC2 <- OP4
3737 020772 174637 002516 DIVF OP6,AC2 ;AC2 <- OP4/OP6
3738 020776 172237 002466 ADDF OP3,AC2 ;AC2 <- OP3+OP4/OP6
3739 021002 171102 MULF AC2,AC1 ;AC1 <- (OP1-OP2)*(OP3+OP4/OP6)
3740 021004 171137 002506 MULF OP5,AC1 ;AC1 <- (OP1-OP2)*(OP3+OP4/OP6)*OP5
3741
3742 021010 170237 002362 STFPS FPS ;STORE STATUS AFTERWARD
3743 021014 023737 002362 002400 CMP FPS,$FPS ;CHECK STATUS
3744 021022 001403 BEQ ETST1 ;BRANCH IF OK
3745 021024 174137 002406 STF AC1,ANS1 ;SAVE FPU ANSWER

```



# F08

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 98  
 DQFPDA.P11 09-FEB-77 10:32 T45 ADDF, SUBF, MULF, DIVF EXERCISER

```

3746 021030 104033          ERROR 33          ;FPS ERROR
3747
3748 021032 173401          ETST1: CMPF  AC1,ACO          ;ANSWER OK ? (FPU:SOFTWARE)
3749 021034 170000          CFCC
3750 021036 001473          BEQ  EEND1          ;ANSWER CHECKS
3751
3752          ;COMPENSATE IN LOB FOR INACCURACIES
3753 021040 174137 002406          STF  AC1,ANS1          ;FPU ANSWER
3754 021044 062737 000002 002410          ADD  #2,ANS1+2          ;INC +2, KEEP CARRY
3755 021052 005537 002406          ADC  ANS1
3756 021056 173437 002406          CMPF ANS1,ACO          ;CHECK AGAIN
3757 021062 170000          CFCC          ;COPY CC-S
3758 021064 001460          BEQ  EEND1          ;ANSWER CHECKS
3759
3760 021066 162737 000001 002410          SUB  #1,ANS1+2          ;INC +1, KEEP CARRY
3761 021074 005537 002406          ADC  ANS1
3762 021100 173437 002406          CMPF ANS1,ACO          ;CHECK AGAIN
3763 021104 170000          CFCC          ;COPY CC-S
3764 021106 001447          BEQ  EEND1          ;ANSWER CHECKS
3765
3766 021110 162737 000002 002410          SUB  #2,ANS1+2          ;DEC -1, KEEP CARRY
3767 021116 005637 002406          SBC  ANS1
3768 021122 173437 002406          CMPF ANS1,ACO          ;CHECK AGAIN
3769 021126 170000          CFCC          ;COPY CC-S
3770 021130 001436          BEQ  EEND1          ;ANSWER CHECKS
3771
3772 021132 162737 000001 002410          SUB  #1,ANS1+2          ;DEC -2, KEEP CARRY
3773 021140 005637 002406          SBC  ANS1
3774 021144 173437 002406          CMPF ANS1,ACO          ;CHECK AGAIN
3775 021150 170000          CFCC          ;COPY CC-S
3776 021152 001425          BEQ  EEND1          ;ANSWER CHECKS
3777
3778 021154 162737 000001 002410          SUB  #1,ANS1+2          ;DEC -3, KEEP CARRY
3779 021162 005637 002406          SBC  ANS1
3780 021166 173437 002406          CMPF ANS1,ACO          ;CHECK AGAIN
3781 021172 170000          CFCC          ;COPY CC-S
3782 021174 001414          BEQ  EEND1          ;ANSWER CHECKS
3783
3784 021176 162737 000001 002410          SUB  #1,ANS1+2          ;DEC -4, KEEP CARRY
3785 021204 005637 002406          SBC  ANS1
3786 021210 173437 002406          CMPF ANS1,ACO          ;CHECK AGAIN
3787 021214 170000          CFCC          ;COPY CC-S
3788 021216 001403          BEQ  EEND1          ;ANSWER CHECKS
3789
3790 021220 174137 002406          STF  AC1,ANS1          ;FPU ANSWER
3791 021224 104035          ERROR 35          ;ANSWERS DON'T CHECK
3792
3793 021226 005037 002362          EEND1: CLR  FPS          ;CLEAR BUFFER
3794
  
```

```

3795 ;*****
3796 ;*TEST 46 ADD, SUBD, MUL, DIVD EXERCISER
3797 ;*****
3798 021232 000004 TST46: SCOPE
3799 ;*UNDERFLOW, OVERFLOW INTERRUPTS OFF; ROUND MODE
3800
3801 021234 012737 000600 002400 MOV #600,$FPS ;SOFTWARE STATUS
3802 021242 005037 002402 CLR $FEC ;CLEAR STATUS
3803 021246 005037 002404 CLR $FEA
3804 021252 005037 002362 CLR FPS
3805 021256 005037 002364 CLR FEC
3806 021262 005037 002366 CLR FEA
3807
3808 021266 004737 023404 JSR PC,RANDL4 ;GET 6 DOUBLE FLOAT RANDOM NUMBERS
3809 021272 002446 002476 .WORD OP1,OP4
3810 021276 004737 023404 JSR PC,RANDL4
3811 021302 002466 002456 .WORD OP3,OP2
3812 021306 004737 023404 JSR PC,RANDL4
3813 021312 002506 002516 .WORD OP5,OP6
3814 021316 032737 077600 002506 BIT #077600,OP5 ;LET'S NEVER DIVIDE BY ZERO
3815 021324 001770 BEQ 1$ ;OP5 IS ZERO, TRY AGAIN
3816 021326 032737 077600 002516 BIT #077600,OP6
3817 021334 001764 BEQ 1$ ;OP6 IS ZERO, TRY AGAIN
3818
3819 021336 004437 023560 JSR R4,$POLSH ;ENTER POLISH MODE TO CALCULATE:
3820 021342 023562 002446 $PUSH ,OP1
3821 021346 023562 002466 $PUSH ,OP3
3822 021352 023640 $ADD
3823 021354 023562 002506 $PUSH ,OP5
3824 021360 026434 $DIV
3825 021362 023562 002456 $PUSH ,OP2
3826 021366 023562 002476 $PUSH ,OP4
3827 021372 023634 $SUB
3828 021374 023562 002516 $PUSH ,OP6
3829 021400 026434 $DIV
3830 021402 025314 $MUL
3831 021404 023612 002416 $POPX ,ANS2
3832
3833 021410 170127 040200 LDFPS #040200 ;NO CHECKS
3834 021414 172437 002416 LDD ANS2,ACO ;GET SOFTWARE ANSWER
3835 021420 013700 002400 MOV $FPS,RO ;DISPLAY $FPS
3836 021424 012737 021432 001110 MOV #.+6,$LPADR ;RESET LOOP ADDRESS
3837
3838 ;*****
3839
3840 021432 170127 000600 LDFPS #000600 ;INITIAL FPS
3841 021436 172537 002446 LDD OP1,AC1 ;AC1 <- OP1
3842 021442 172137 002466 ADD OP3,AC1 ;AC1 <- OP1+OP3
3843 021446 174537 002506 DIVD OP5,AC1 ;AC1 <- (OP1+OP3)/OP5
3844 021452 172637 002456 LDD OP2,AC2 ;AC2 <- OP2
3845 021456 173237 002476 SUBD OP4,AC2 ;AC2 <- OP2-OP4
3846 021462 174637 002516 DIVD OP6,AC2 ;AC2 <- (OP2-OP4)/OP6
3847 021466 171102 MUL AC2,AC1 ;AC1 <- (OP1+OP3)/OP5*(OP2-OP4)/OP6
3848
3849 021470 170237 002362 STFPS FPS ;STORE STATUS AFTERWARD
3850 021474 023737 002362 002400 CMP FPS,$FPS ;CHECK STATUS

```

$$ANS2 = \frac{OP1+OP3}{OP5} * \frac{OP2-OP4}{OP6}$$

# H08

3851	021502	001403		BEG	ETST2		: BRANCH IF OK
3852	021504	174137	002406	STD	AC1,ANS1		: SAVE FPU ANSWER
3853	021510	104034		ERROR	34		: FPS ERROR
3854							
3855	021512	173401		ETST2:	CMPD	AC1,ACO	: ANSWER OK ? (FPU:SOFTWARE)
3856	021514	170000		CFCC			: COPY CC-S
3857	021516	001523		BEG	EEND2		: ANSWER CHECKS
3858							
3859							: COMPENSATE IN LOB FOR INACCURACIES
3860	021520	174137	002406	STD	AC1,ANS1		: FPU ANSWER
3861	021524	062737	000002	ADD	#2,ANS1+6	002414	: INC +2, KEEP CARRY
3862	021532	005537	002412	ADC	ANS1+4		:
3863	021536	005537	002410	ADC	ANS1+2		:
3864	021542	005537	002406	ADC	ANS1		:
3865	021546	173437	002406	CMPD	ANS1,ACO		: CHECK AGAIN
3866	021552	170000		CFCC			: COPY CC-S
3867	021554	001504		BEG	EEND2		: ANSWER CHECKS
3868							
3869	021556	162737	000001	SUB	#1,ANS1+6	002414	: INC +1, KEEP CARRY
3870	021564	005537	002412	ADC	ANS1+4		:
3871	021570	005537	002410	ADC	ANS1+2		:
3872	021574	005537	002406	ADC	ANS1		:
3873	021600	173437	002406	CMPD	ANS1,ACO		: CHECK AGAIN
3874	021604	170000		CFCC			: COPY CC-S
3875	021606	001467		BEG	EEND2		: ANSWER CHECKS
3876							
3877	021610	162737	000002	SUB	#2,ANS1+6	002414	: DEC -1, KEEP CARRY
3878	021616	005637	002412	SBC	ANS1+4		:
3879	021622	005637	002410	SBC	ANS1+2		:
3880	021626	005637	002406	SBC	ANS1		:
3881	021632	173437	002406	CMPD	ANS1,ACO		: CHECK AGAIN
3882	021636	170000		CFCC			: COPY CC-S
3883	021640	001452		BEG	EEND2		: ANSWER CHECKS
3884							
3885	021642	162737	000001	SUB	#1,ANS1+6	002414	: DEC -2, KEEP CARRY
3886	021650	005637	002412	SBC	ANS1+4		:
3887	021654	005637	002410	SBC	ANS1+2		:
3888	021660	005637	002406	SBC	ANS1		:
3889	021664	173437	002406	CMPD	ANS1,ACO		: CHECK AGAIN
3890	021670	170000		CFCC			: COPY CC-S
3891	021672	001435		BEG	EEND2		: ANSWER CHECKS
3892							
3893	021674	162737	000001	SUB	#1,ANS1+6	002414	: DEC -3, KEEP CARRY
3894	021702	005637	002412	SBC	ANS1+4		:
3895	021706	005637	002410	SBC	ANS1+2		:
3896	021712	005637	002406	SBC	ANS1		:
3897	021716	173437	002406	CMPD	ANS1,ACO		: CHECK AGAIN
3898	021722	170000		CFCC			: COPY CC-S
3899	021724	001420		BEG	EEND2		: ANSWER CHECKS
3900							
3901	021726	162737	000001	SUB	#1,ANS1+6	002414	: DEC -4, KEEP CARRY
3902	021734	005637	002412	SBC	ANS1+4		:
3903	021740	005637	002410	SBC	ANS1+2		:
3904	021744	005637	002406	SBC	ANS1		:
3905	021750	173437	002406	CMPD	ANS1,ACO		: CHECK AGAIN
3906	021754	170000		CFCC			: COPY CC-S

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 101  
DQFPDA.P11 09-FEB-77 10:32 T46 ADD, SUBD, MULD, DIVD EXERCISER

3907	021756	001403		BEQ	EEND2	;ANSWER CHECKS
3908						
3909	021760	174137	002406	STD	AC1,ANS1	;FPU ANSWER
3910	021764	104036		ERROR	36	;ANSWERS DON'T CHECK
3911						
3912	021766	005037	002362	EEND2:	CLR FPS	;CLEAR BUFFER
3913						

J08

```

3914                                     ;:*****
3915                                     ;.SBTTL SUB PASS END CONTROL
3916
3917 021772 000004                       SCOPE                ;CHECK FOR TEST ITERATIONS HERE
3918
3919                                     ;IF TEST ONLY EITHER HFP OR WFP, ENTER "EOP" ROUTINE DIRECTLY
3920
3921                                     ; IF IN ALTERNATE HFP/WFP MODE,
3922                                     ; COMPLEMENT FLAG<5>, HFP ENABLE BIT,
3923                                     ; ENTER EOP ROUTINE ONLY IF ABOUT TO TEST HFP NEXT,
3924                                     ; TESTING SEQUENCE IS:  PASS#1 HFP SUB-PASS
3925                                     ;                   PASS#1 WFP SUB-PASS
3926                                     ;                   PASS#2 HFP SUB-PASS
3927                                     ;                   ...
3928
3929 021774 076600 000022                 MED      ,RWHAMI          ;GET WHAMI INTO RO
3930 022000 032700 000020                 BIT      #BIT04,RO      ;1=HFP PRESENT, 0=NONE
3931 022004 001423                       BEQ      $EOP           ;EXIT IF NONE
3932
3933 022006 032777 000002 157132         BIT      #SW01,JSWR     ;1=HFP OR WFP TEST ONLY
3934 022014 001017                       BNE      $EOP           ;0=ALTERNATE HFP AND WFP TESTS
3935
3936 022016 012701 010000                 MOV      #BIT12,R1      ;HFP PRESENT, AND IN ALTERNATE MODE;
3937 022022 076600 000144                 MED      ,RFLAG         ;SO READ FLAGS,
3938 022026 030100                       BIT      R1,RO          ;COMPLEMENT FLAG<5>=BIT12=HFP ENABLE FLAG
3939 022030 001402                       BEQ      1$             ;
3940 022032 040100                       BIC      R1,RO          ;CLEAR BIT 12
3941 022034 000401                       BR       2$             ;
3942 022036 050100 1$: BIS      R1,RO          ;SET BIT 12
3943 022040 076600 000344 2$: MED      ,WFLAG         ;REWRITE FLAGS
3944
3945 022044 030100                       BIT      R1,RO          ;HFP OR WFP NEXT ?
3946 022046 001002                       BNE      $EOP           ;IF HFP AGAIN, START NEW PASS
3947 022050 000137 003452                       JMP      J#SUBPAS       ;IF WFP, NEXT SUBPASS
3948
3949
3950                                     ;:*****
3951                                     ;.SBTTL END OF PASS ROUTINE (MODIFIED SYSMAC)
3952
3953                                     ;*INCREMENT THE PASS NUMBER ($PASS)
3954                                     ;*IF SW<10>=0, DING BELL ON PASS END
3955                                     ;*IF SW<12>=0, TYPE STATISTICS ON PASS END
3956                                     ;*IF THERE'S A MONITOR, GO TO IT
3957                                     ;* ELSE JUMP TO NEWPAS
3958
3959
3960 022054 000004                       $EOP:  SCOPE
3961 022056 005037 001104                 CLR      $ERFLG        ;ZERO ERROR FLAG
3962 022062 005037 001102                 CLR      $STNM         ;ZERO TEST NUMBER
3963 022066 005037 001166                 CLR      $TIMES        ;ZERO NUMBER OF ITERATIONS
3964 022072 005237 001210                 INC      $PASS         ;INCREMENT PASS COUNT,
3965 022076 042737 100000 001210         BIC      #100000,$PASS ; BUT NEVER LET IN GO NEGATIVE
3966 022104 005327                       DEC      (PC)+         ;PASS LOOP ?
3967 022106 000400                       $EOPCT: .WORD 400      ;
3968 022110 003027                       BGT     $DOAGN         ;YES
3969 022112 012737                       MOV     (PC)+,J(PC)+  ;RESTORE COUNTER
  
```

K08

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 103  
 DQFPDA.P11 09-FEB-77 10:32 END OF PASS ROUTINE (MODIFIED SYSMAC)

3970	022114	000400			SENDCT:	.WORD	400	:
3971	022116	022106				\$EOPCT		:
3972	022120	032777	002000	157020		BIT	#SW10,@SWR	: BELL ON PASS END ?
3973	022126	001002				BNE	1\$	: NO
3974	022130	104401	001172			TYPE	, \$BELL	: YES
3975	022134	032777	010000	157004	1\$:	BIT	#SW12,@SWR	: INHIBIT MESSAGE ?
3976	022142	001002				BNE	\$GET42	: YES
3977	022144	004737	022174			JSR	PC, STATS	: TYPE STATISTICS
3978	022150	013700	000042		\$GET42:	MOV	@42, R0	: GET MONITOR ADDRESS
3979	022154	001405				BEG	\$DOAGN	: NO MONITOR
3980	022156	000005				RESET		: CLEAR WORLD
3981								
3982	022160	004710			SENDAD:	JSR	PC, (R0)	: GO TO MONITOR
3983	022162	000240				NOP		:
3984	022164	000240				NOP		: RESERVED FOR ACT11
3985	022166	000240				NOP		:
3986								
3987	022170	000137	003404		\$DOAGN:	JMP	@NEWPAS	: RETURN

.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
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 <11> ;<HT>
SCRLF: .ASCIZ <15><12> ;<CR><LF>

```

```

3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998 022174 010046
3999
4000 022176 012700 022302
4001 022202 001404
4002 022204 104401 022371
4003 022210 004737 022254
4004 022214 012700 022326
4005 022220 001404
4006 022222 104401 022736
4007 022226 004737 022254
4008 022232 012700 022344
4009 022236 001404
4010 022240 104401 023127
4011 022244 004737 022254
4012
4013 022250 012600
4014 022252 000207
4015
4016
4017 022254 013046
4018 022256 104403
4019 022260 006
4020 022261 000
4021 022262 005710
4022 022264 001403
4023 022266 104401 022364
4024 022272 000770
4025 022274 104401 022366
4026 022300 000207
4027
4028
4029
4030 022302 002526 002540 002542
4031 022310 002544 002546 002530
4032 022316 002532 002534 002536
4033 022324 000000
4034 022326 002550 002554 002556
4035 022334 002560 002562 002552
4036 022342 000000
4037 022344 002564 002572 002574
4038 022352 002576 002600 002566
4039 022360 002570 000000
4040
4041
4042 022364 000011
4043 022366 005015 000

```

M08

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 105  
 DQFPDA.P11 09-FEB-77 10:32 STATISTICS TYPEOUT SUBROUTINE

4044	022371	015	005012	
4045	022374	025052	020052	052123
4046	022402	052101	051511	044524
4047	022410	051503	025040	025052
4048	022416	005015	012	
4049	022421	050	047516	042524
4050	022426	026440	040440	046114
4051	022434	047040	046525	042502
4052	022442	051522	040440	042522
4053	022450	052440	051516	043511
4054	022456	042516	020104	041517
4055	022464	040524	020114	047111
4056	022472	042524	042507	051522
4057	022500	006451	005012	
4058	022504	042101	027504	052523
4059	022512	020102	047111	052123
4060	022520	052522	052103	047511
4061	022526	051516	006472	012
4062	022533	124	052117	046101
4063	022540	026411	026455	053117
4064	022546	051105	046106	053517
4065	022554	026455	004455	026455
4066	022562	047125	042504	043122
4067	022570	047514	026527	026455
4068	022576	047411	020120	036501
4069	022604	004460	050117	040440
4070	022612	030043	047411	020120
4071	022620	036501	004460	020040
4072	022626	047040	006517	012
4073	022633	116	046525	042502
4074	022640	004522	047524	040524
4075	022646	020114	027527	047105
4076	022654	041101	042514	052011
4077	022662	052117	046101	053440
4078	022670	042457	040516	046102
4079	022676	004505	050117	041040
4080	022704	030043	047411	020120
4081	022712	036502	004460	050117
4082	022720	041040	030075	020011
4083	022726	044123	043111	006524
4084	022734	000012		
4085	022736	005015	052515	052114
4086	022744	050111	054514	044440
4087	022752	051516	051124	041525
4088	022760	044524	047117	035123
4089	022766	005015		
4090	022770	047524	040524	004514
4091	022776	026455	047455	042526
4092	023004	043122	047514	026527
4093	023012	026455	026411	052455
4094	023020	042116	051105	046106
4095	023026	053517	026455	004455
4096	023034	036501	020060	047101
4097	023042	027504	051117	005015
4098	023050	052516	041115	051105
4099	023056	052011	052117	046101

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"<1

.ASCIZ "NUMBER TOTAL W/ENABLE TOTAL W/ENABLE OP B#0 OP B=0 OP B=0 SHIFT"<

SHDR2: .ASCII <15><12>"MULTIPLY INSTRUCTIONS:"<15><12>

.ASCII "TOTAL ---OVERFLOW--- --UNDERFLOW--- A=0 AND/OR"<15><12>

.ASCIZ "NUMBER TOTAL W/ENABLE TOTAL W/ENABLE B=0"<15><12>



4100	023064	053440	042457	040516
4101	023072	046102	004505	047524
4102	023100	040524	020114	027527
4103	023106	047105	041101	042514
4104	023114	020011	020040	041040
4105	023122	030075	005015	000
4106	023127	015	042012	053111
4107	023134	042111	020105	047111
4108	023142	052123	052522	052103
4109	023150	047511	051516	006472
4110	023156	012		
4111	023157	124	052117	046101
4112	023164	026411	026455	053117
4113	023172	051105	046106	053517
4114	023200	026455	004455	026455
4115	023206	047125	042504	043122
4116	023214	047514	026527	026455
4117	023222	020011	052516	042515
4118	023230	004522	042040	047105
4119	023236	046517	005015	
4120	023242	052516	041115	051105
4121	023250	052011	052117	046101
4122	023256	053440	042457	040516
4123	023264	046102	004505	047524
4124	023272	040524	020114	027527
4125	023300	047105	041101	042514
4126	023306	020011	020040	036501
4127	023314	004460	020040	041040
4128	023322	030075	005015	000
4129		023330		

SHDR3: .ASCII <15><12>"DIVIDE INSTRUCTIONS:"<15><12>

.ASCII "TOTAL ---OVERFLOW--- --UNDERFLOW--- NUMER DENOM"<15><12>

.ASCIZ "NUMBER TOTAL W/ENABLE TOTAL W/ENABLE A=0 B=0"<15><12>

.EVEN ;BACK TO AN EVEN BOUNDARY

```

4130          .SBTTL FPP TRAP CATCHER
4131
4132 023330 012637 002370 FPPILT: MOV (SP)+, FPPOPC ;POP OLD PC FOR DISPLAY
4133 023334 012637 002372      MOV (SP)+, FPPOPS ;POP OLD PS FOR DISPLAY
4134 023340 170237 002362      STFPS FPS ;GET FPS
4135 023344 170337 002364      STST FEC ;GET FEC/FEA
4136 023350 005737 002362      TST FPS ;TEST ERROR BIT
4137 023354 100005      BPL 1$ ;OFF - NO ERROR BIT SET, BUT TRAPPED
4138
4139 023356 032737 040000 002362      BIT #040000, FPS ;ON - IT SHOULD BE ON A TRAP
4140 023364 001001      BNE 1$ ;TEST INTERRUPT ENABLE BIT
4141
4142 023366 000401      BR 2$ ;ON - INTR DISABLED, BUT TRAPPED
4143 023370 104015      1$: ERROR 1$ ;OFF - ABLE TO INTR, SO IGNORE IT,
4144 023372 013746 002372      2$: MOV FPPOPS, -(SP) ;AND SKIP THE ERROR
4145 023376 013746 002370      MOV FPPOPC, -(SP) ;SIGNAL UNEXPECTED FPP TRAP
4146 023402 000002      RTI ;PUSH PSW
4147 ;PUSH PC
;CONTINUE, RECOVER AT LAST TRAP ONLY
  
```

```

4148
4149
4150
4151
4152
4153
4154
4155
4156 023404 010546
4157 023406 012705 000004
4158 023412 000403
4159 023414 010546
4160 023416 012705 000002
4161 023422 010446
4162 023424 010346
4163 023426 010246
4164 023430 010146
4165 023432 010046
4166 023434 005046
4167 023436 016600 000016
4168 023442 012003
4169 023444 012004
4170 023446 010066 000016
4171 023452 011300
4172 023454 011401
4173 023456 012702 000007
4174 023462 005016
4175 023464 006300
4176 023466 006101
4177 023470 006116
4178 023472 077204
4179 023474 061316
4180 023476 005501
4181 023500 061401
4182 023502 005516
4183 023504 062700 001057
4184 023510 005501
4185 023512 005516
4186 023514 062701 047401
4187 023520 005516
4188 023522 062716 000006
4189 023526 061600
4190 023530 005501
4191 023532 010023
4192 023534 010124
4193 023536 077531
4194 023540 005726
4195 023542 012600
4196 023544 012601
4197 023546 012602
4198 023550 012603
4199 023552 012604
4200 023554 012605
4201 023556 000207
4202

.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 (R3),(SP) ;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 ;RESTORE REGISTERS
MOV (SP)+,R1
MOV (SP)+,R2
MOV (SP)+,R3
MOV (SP)+,R4
MOV (SP)+,R5
RTS PC ;RETURN

```

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  
4255  
4256  
4257  
4258

.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 $FPS
        BPL 1$
        MOV 6(R0),-(SP) ;PUSH DOUBLE ON STACK
        MOV 4(R0),-(SP)
1$: 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

```

```

023560 000134
023562 012400
023564 105737 002400
023570 100004
023572 016046 000006
023576 016046 000004
023602 016046 000002
023606 011046
023610 000134

```

4259			
4260	023612	012400	
4261	023614	012620	
4262	023616	012620	
4263	023620	1J5737	002400
4264	023624	100002	
4265	023626	012620	
4266	023630	012620	
4267	023632	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

```

4268  
4269  
4270  
4271  
4272  
4273  
4274  
4275  
4276  
4277  
4278  
4279  
4280  
4281  
4282  
4283  
4284  
4285  
4286  
4287  
4288  
4289  
4290  
4291  
4292  
4293  
4294  
4295  
4296  
4297  
4298  
4299  
4300  
4301  
4302  
4303  
4304  
4305  
4306  
4307  
4308  
4309  
4310  
4311  
4312  
4313  
4314  
4315  
4316  
4317  
4318  
4319  
4320  
4321  
4322  
4323

## .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 \$FPS IS 0 OR 1 RESPECTIVELY) AND  
:\* REPLACES THEM WITH THEIR SUM/DIFFERENCE.

```

:* EXAMPLE:      $PUSH                      $PUSH
:*              ADDR(OPERAND A)            ADDR(OPERAND A)
:*              $PUSH                      $PUSH
:*              ADDR(OPERAND B)            ADDR(OPERAND B)
:*              $ADD                        $SUB
:*              $POPX                      $POPX
:*              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 \$FPS 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 \$FPS  
(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 \$FPS 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 \$FPS  
(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 \$FPS ERROR BIT (BIT15) WILL BE SET,  
\$FEC WILL BE SET TO 12(8), AND \$FEA WILL BE LOADED WITH  
THE VALUE IN LOCATION "EXPFEA".

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

```

4324                                     ;* ADAPTED FROM PDP-11 FORTRAN SOFTWARE
4325                                     ;* BY DONALD NORTH, SEPTEMBER, 1975.
4326                                     ;*
4327
4328 023634 062716 100000  $SUB:  ADD      #100000,(SP)  ;CHANGE SIGN OF TOP ITEM
4329 023640 005237 002526  $ADD:  INC      ADDC0    ;CTR:  TOTAL NUMBER OF ADD/SUB
4330 023644 042737 000017 002400  BIC      #17,$FPS  ;CLEAR STATUS BITS N Z V C
4331 023652 105737 002400  TSTB    $FPS      ;TEST MODE
4332 023656 100402  BMI      27$     ;D-MODE
4333 023660 004737 027672  JSR     PC,$CONV ;D-MODE: CONVERT 2 F OPDS TO D
4334 023664 010046 27$:  MOV     R0,-(SP)
4335 023666 010146  MOV     R1,-(SP)
4336 023670 010246  MOV     R2,-(SP)
4337 023672 010346  MOV     R3,-(SP)
4338 023674 010446  MOV     R4,-(SP)
4339 023676 010546  MOV     R5,-(SP)
4340 023700 005046  CLR     -(SP)   ;CLEAR SIGNS
4341 023702 005004  CLR     R4      ;CLEAR EXPONENTS
4342 023704 005005  CLR     R5
4343 023706 006366 000024  ASL     24(SP)  ;SHIFT OUT SIGN OF TOP ITEM
4344 023712 006166 000022  ROL     22(SP)
4345 023716 006166 000020  ROL     20(SP)
4346 023722 006166 000016  ROL     16(SP)
4347 023726 156604 000017  BISB   17(SP),R4 ;SHIFT A1
4348 023732 001011  BNE     31$     ;GET E1
4349 023734 005726  TST     (SP)+   ;JUMP IF NON ZERO
4350 023736 032766 077600 000024  BIT     #077600,24(SP) ;FLUSH SIGNS
4351 023744 001456  BEQ     32$     ;B=0, A=0 TOO ?
4352 023746 005237 002532  INC     ADDC2   ;YES, BOTH ZERO
4353 023752 000137 024762  JMP     3$      ;NO, CTR:  B=0,A#0
4354 023756 106116 31$:  ROLB   (SP)    ;DONE
4355 023760 006366 000034  ASL     34(SP)  ;GET S1
4356 023764 006166 000032  ROL     32(SP)  ;SHIFT OUT SIGN OF 2ND ITEM
4357 023770 006166 000030  ROL     30(SP)
4358 023774 006166 000026  ROL     26(SP)  ;SHIFT A2
4359 024000 156605 000027  BISB   27(SP),R5 ;GET E2
4360 024004 001042  BNE     2$      ;JUMP IF NON ZERO
4361 024006 106016  RORB   (SP)    ;RECONSTRUCT A1
4362 024010 006066 000016  ROR     16(SP)
4363 024014 006066 000020  ROR     20(SP)
4364 024020 006066 000022  ROR     22(SP)
4365 024024 006066 000024  ROR     24(SP)
4366 024030 016666 000016 000026  MOV     16(SP),26(SP) ;FIRST ARG TO TOP OF STACK
4367 024036 016666 000020 000030  MOV     20(SP),30(SP)
4368 024044 016666 000022 000032  MOV     22(SP),32(SP)
4369 024052 016666 000024 000034  MOV     24(SP),34(SP)
4370 024060 005726  TST     (SP)+   ;FLUSH SIGNS
4371 024062 032766 077600 000024  BIT     #077600,24(SP) ;A=0, B=0 TOO ?
4372 024070 001404  BEQ     32$     ;YES, BOTH ZERO
4373 024072 005237 002530  INC     ADDC1   ;NO, CTR:  A=0, B#0
4374 024076 000137 024762  JMP     3$      ;DONE
4375 024102 005237 002534 32$:  INC     ADDC3   ;CTR:  A=0, B=0
4376 024106 000137 024742  JMP     29$     ;DONE, TRUE ZERO RESULT
4377 024112 106166 000001 2$:  ROLB   1(SP)   ;GET $2
4378 024116 112766 000001 000027  MOVB   #1,27(SP) ;INSERT NORMAL BIT
4379 024124 112766 000001 000017  MOVB   #1,17(SP)
    
```

4380	024132	160405			SUB	R4, R5	:R5=E2-E1, R4=E1
4381	024134	003011			BGT	4\$	:JUMP IF E2>E1
4382	024136	016600	000026		MOV	26(SP), R0	:R0=A2
4383	024142	016601	000030		MOV	30(SP), R1	:R1=B2
4384	024146	016602	000032		MOV	32(SP), R2	:R2=C2
4385	024152	016603	000034		MOV	34(SP), R3	:R3=D2
4386	024156	000427			BR	5\$	:GO CHECK SIGNS
4387	024160	060504		4\$:	ADD	R5, R4	:R5=E2-E1, R4=E2, E2>E1
4388	024162	016600	000016		MOV	16(SP), R0	:R0=A1
4389	024166	016601	000020		MOV	20(SP), R1	:R1=B1
4390	024172	016602	000022		MOV	22(SP), R2	:R2=C1
4391	024176	016603	000024		MOV	24(SP), R3	:R3=D1
4392	024202	016666	000026	000016	MOV	26(SP), 16(SP)	
4393	024210	016666	000030	000020	MOV	30(SP), 20(SP)	
4394	024216	016666	000032	000022	MOV	32(SP), 22(SP)	
4395	024224	016666	000034	000024	MOV	34(SP), 24(SP)	
4396	024232	000316			SWAB	(SP)	:EXCHANGE SIGNS
4397	024234	005405			NEG	R5	:E1-E2
4398	024236	126616	000001	5\$:	CMPB	1(SP), (SP)	:COMPARE SIGNS
4399	024242	001412			BEQ	6\$	:SAME, GO CHECK EXPONENT
4400	024244	005403			NEG	R3	:NEGATE OPERAND
4401	024246	005502			ADC	R2	
4402	024250	005501			ADC	R1	
4403	024252	005500			ADC	R0	
4404	024254	005402			NEG	R2	
4405	024256	005501			ADC	R1	
4406	024260	005500			ADC	R0	
4407	024262	005401			NEG	R1	
4408	024264	005500			ADC	R0	
4409	024266	005400			NEG	R0	
4410	024270	005705		6\$:	TST	R5	:CHECK EXPONENTS
4411	024272	001466			BEQ	7\$	:JUMP IF E1=E2
4412	024274	022705	177707		CMP	#-57., R5	:ANY POINT IN SHIFTING?
4413	024300	003413			BLE	8\$	:YES
4414	024302	016600	000016		MOV	16(SP), R0	:NO, ANSWER IS OPERAND
4415	024306	016601	000020		MOV	20(SP), R1	:WITH LARGER EXPONENT
4416	024312	016602	000022		MOV	22(SP), R2	
4417	024316	016603	000024		MOV	24(SP), R3	
4418	024322	005237	002536		INC	ADDC4	:CTR: NO SHIFT
4419	024326	000501			BR	9\$	
4420	024330	022705	177770	8\$:	CMP	#-8., R5	:CHECK # OF BITS TO SHIFT
4421	024334	003437			BLE	10\$	:JUMP IF LESS THAN 1/2 WORD
4422	024336	005700			TST	R0	
4423	024340	006746			SXT	-(SP)	:EXTEND SIGN
4424	024342	022705	177760	12\$:	CMP	#-16., R5	
4425	024346	002411			BLT	11\$	:JUMP IF LESS THAN 1 WORD
4426	024350	010203			MOV	R2, R3	:SHIFT A WORD AT A TIME
4427	024352	010102			MOV	R1, R2	
4428	024354	010001			MOV	R0, R1	
4429	024356	011600			MOV	(SP), R0	:USE EXTENSION
4430	024360	062705	000020		ADD	#16., R5	:ADJUST EXPONENT
4431	024364	001366			BNE	12\$	:TRY AGAIN
4432	024366	005726			TST	(SP)+	:POP EXTENSION
4433	024370	000427			BR	7\$	:SHIFT DONE
4434	024372	022705	177775	11\$:	CMP	#-3, R5	:JUMP IF LESS THAN 4 TO SHIFT
4435	024376	003415			BLE	13\$	



4436	024400	010416			MOV	R4, (SP)	:	SAVE EXP & SHIFT COUNT
4437	024402	010546			MOV	R5, -(SP)	:	
4438	024404	010104			MOV	R1, R4	:	SAVE R1
4439	024406	073005			ASHC	R5, R0	:	SHIFT HIGH ORDER
4440	024410	010205			MOV	R2, R5	:	SAVE R2
4441	024412	073416			ASHC	(SP), R4	:	SHIFT IT
4442	024414	010204			MOV	R2, R4	:	
4443	024416	010502			MOV	R5, R2	:	R2 DONE
4444	024420	010305			MOV	R3, R5	:	SET UP LOW ORDER
4445	024422	073426			ASHC	(SP)+, R4	:	DO LOW ORDER
4446	024424	010503			MOV	R5, R3	:	
4447	024426	012604			MOV	(SP)+, R4	:	RESTORE EXP TO R4
4448	024430	000407			BR	7\$	:	
4449	024432	005726		13\$:	TST	(SP)+	:	POP EXTENSION
4450	024434	006200		10\$:	ASR	R0	:	SHIFT RIGHT
4451	024436	006001			ROR	R1	:	
4452	024440	006002			ROR	R2	:	
4453	024442	006003			ROR	R3	:	
4454	024444	005205			INC	R5	:	COUNT LOOP
4455	024446	002772			BLT	10\$	:	
4456	024450	066603	000024	7\$:	ADD	24(SP), R3	:	FORM SUM
4457	024454	005502			ADC	R2	:	
4458	024456	005501			ADC	R1	:	
4459	024460	005500			ADC	R0	:	
4460	024462	066602	000022		ADD	22(SP), R2	:	
4461	024466	005501			ADC	R1	:	
4462	024470	005500			ADC	R0	:	
4463	024472	066601	000020		ADD	20(SP), R1	:	
4464	024476	005500			ADC	R0	:	
4465	024500	066600	000016		ADD	16(SP), R0	:	
4466	024504	126616	000001		CMPB	1(SP), (SP)	:	CHECK FOR UNEQUAL SIGNS
4467	024510	001161			BNE	14\$	:	CLEAN UP SUBTRACT
4468	024512	030027	001000		BIT	R0, #1000	:	
4469	024516	001405			BEQ	9\$	:	JUMP IF NO NORMAL BIT OVERFLOW
4470	024520	006200			ASR	R0	:	
4471	024522	006001			ROR	R1	:	
4472	024524	006002			ROR	R2	:	
4473	024526	006003			ROR	R3	:	
4474	024530	005204			INC	R4	:	INCREASE EXP
4475	024532	000304		9\$:	SWAB	R4	:	MOVE EXP LEFT
4476	024534	001425			BEQ	16\$	:	JUMP IF NO OVERFLOW
4477	024536	105004			CLRB	R4	:	CLEAR OVERFLOWED BITS
4478	024540	052737	000002	002400	BIS	#02, \$FPS	:	SET V BIT ON OVERFLOW
4479	024546	005237	002540		INC	ADDC5	:	CTR: OVERFLOW
4480	024552	032737	001000	002400	BIT	#001000, \$FPS	:	OVERFLOW ENABLED ?
4481	024560	001464			BEQ	34\$	:	NO, ZERO RESULT
4482	024562	005237	002542		INC	ADDC6	:	CTR: OVERFLOW, ENABLED
4483	024566	052737	100000	002400	BIS	#100000, \$FPS	:	YES, SET ERROR BIT
4484	024574	012737	000010	002402	MOV	#10, \$FEC	:	SET \$FEC
4485	024602	013737	002376	002404	MOV	EXPFEA, \$FEA	:	SET \$FEA
4486	024610	150004		16\$:	BISB	R0, R4	:	INSERT HIGH ORDER FRACTION
4487	024612	006026			ROR	(SP)+	:	INSERT SIGN
4488	024614	006004			ROR	R4	:	
4489	024616	006001			ROR	R1	:	
4490	024620	006002			ROR	R2	:	
4491	024622	006003			ROR	R3	:	

4492	024624	005503				ADC	R3	:
4493	024626	005502				ADC	R2	:
4494	024630	005501				ADC	R1	:
4495	024632	005504				ADC	R4	:
4496	024634	103401				BCS	20\$	: OVERFLOW ON ROUND ?
4497	024636	102024				BVC	30\$	:
4498	024640	052737	000002	002400	20\$:	BIS	#02,\$FPS	: YES - SET V BIT
4499	024646	005237	002540			INC	ADDC5	: CTR: OVERFLOW
4500	024652	032737	001000	002400		BIT	#001000,\$FPS	: OVERFLOW ENABLED ?
4501	024660	001430				BEQ	29\$	: NO, ZERO RESULT
4502	024662	005237	002542			INC	ADDC6	: CTR: OVERFLOW, ENABLED
4503	024666	052737	100000	002400		BIS	#100000,\$FPS	: SET ERROR BIT
4504	024674	012737	000010	002402		MOV	#10,\$FEC	: SET \$FEC
4505	024702	013737	002376	002404		MOV	EXPFEA,\$FEA	: SET \$FEA
4506	024710	010466	000024		30\$:	MOV	R4,24(SP)	: STORE EXP AND SIGN
4507	024714	010166	000026			MOV	R1,26(SP)	: INSERT LOW ORDER FRACTION
4508	024720	010266	000030			MOV	R2,30(SP)	:
4509	024724	010366	000032			MOV	R3,32(SP)	:
4510	024730	000414				BR	3\$	:
4511	024732	005726			34\$:	TST	(SP)+	: FLUSH SIGN
4512	024734	000402				BR	29\$	: AND ZERO RESULT
4513	024736	005237	002544		33\$:	INC	ADDC7	: CTR: UNDERFLOW
4514	024742	005066	000024		29\$:	CLR	24(SP)	: ZERO RESULT
4515	024746	005066	000026			CLR	26(SP)	:
4516	024752	005066	000030			CLR	30(SP)	:
4517	024756	005066	000032			CLR	32(SP)	:
4518	024762	032766	077600	000024	3\$:	BIT	#077600,24(SP)	: SET Z BIT IF EXPONENT ZERO
4519	024770	001003				BNE	17\$	:
4520	024772	052737	000004	002400		BIS	#04,\$FPS	:
4521	025000	005766	000024		17\$:	TST	24(SP)	: SET N BIT IF RESULT NEGATIVE
4522	025004	100003				BPL	18\$	:
4523	025006	052737	000010	002400		BIS	#10,\$FPS	:
4524	025014	012605			18\$:	MOV	(SP)+,R5	:
4525	025016	012604				MOV	(SP)+,R4	:
4526	025020	012603				MOV	(SP)+,R3	: RESTORE REGISTERS
4527	025022	012602				MOV	(SP)+,R2	:
4528	025024	012601				MOV	(SP)+,R1	:
4529	025026	012600				MOV	(SP)+,R0	:
4530	025030	062706	000010			ADD	#8,SP	: POP SECOND ARGUMENT
4531	025034	105737	002400			TSTB	\$FPS	: FOR D MODE?
4532	025040	100404				BMI	26\$	: D MODE
4533	025042	012666	000002			MOV	(SP)+,2(SP)	: F MODE - CONVERT
4534	025046	012666	000002			MOV	(SP)+,2(SP)	:
4535	025052	000134			26\$:	JMP	2(R4)+	: DONE
4536								:
4537								:
4538	025054	005700			14\$:	TST	R0	: CHECK HIGH ORDER FRACTION RESULT
4539	025056	003014				BGT	22\$	: IF + SIGN OK
4540	025060	001453				BEQ	23\$	: CHECK FOR ZERO RESULT
4541	025062	005403				NEG	R3	: ABS VALUE
4542	025064	005502				ADC	R2	:
4543	025066	005501				ADC	R1	:
4544	025070	005500				ADC	R0	:
4545	025072	005402				NEG	R2	:
4546	025074	005501				ADC	R1	:
4547	025076	005500				ADC	R0	:

4548	025100	005401			NEG	R1		
4549	025102	005500			ADC	R0		
4550	025104	005400			NEG	R0		
4551	025106	000316			SWAB	(SP)		EXCHANGE SIGNS
4552	025110	030027	000400	22\$:	BIT	R0, #400		CHECK NORMAL BIT
4553	025114	001427			BEQ	19\$		JUMP IF NONE
4554	025116	005704			TST	R4		CHECK FOR UNDERFLOW
4555	025120	003204			BGT	9\$		
4556	025122	032737	002000	002400	BIT	#002000, \$FPS		UNDERFLOW - IS IT ENABLED ?
4557	025130	001702			BEQ	33\$		NO, MAKE ZERO RESULT
4558	025132	005237	002544		INC	ADDC7		CTR: UNDERFLOW
4559	025136	005237	002546		INC	ADDC8		CTR: UNDERFLOW, ENABLED
4560	025142	052737	100000	002400	BIS	#100000, \$FPS		SET ERROR BIT
4561	025150	012737	000012	002402	MOV	#12, \$FEC		SET \$FEC
4562	025156	013737	002376	002404	MOV	EXPFEA, \$FEA		SET \$FEA
4563	025164	000304			SWAB	R4		MOVE EXP LEFT
4564	025166	105004			CLRB	R4		CLEAR WHERE FRACTION WILL GO
4565	025170	000137	024610		JMP	16\$		ASSEMBLE NUMBER
4566	025174	005304		19\$:	DEC	R4		DECREASE EXP
4567	025176	006303			ASL	R3		DOUBLE FRACTION
4568	025200	006102			ROL	R2		
4569	025202	006101			ROL	R1		
4570	025204	006100			ROL	R0		
4571	025206	000740			BR	22\$		
4572	025210	162704	000010	23\$:	SUB	#8., R4		REDUCE EXP
4573	025214	005701			TST	R1		
4574	025216	001020			BNE	24\$		JUMP IF ONLY R0=0
4575	025220	162704	000020		SUB	#16., R4		
4576	025224	010201			MOV	R2, R1		
4577	025226	001012			BNE	25\$		JUMP IF R2 NON ZERO
4578	025230	162704	000020		SUB	#16., R4		
4579	025234	005703			TST	R3		
4580	025236	001422			BEQ	21\$		ANSWER IS ZERO
4581	025240	150301			BISB	R3, R1		MOVE BYTES TO R0, R1
4582	025242	000301			SWAB	R1		
4583	025244	000303			SWAB	R3		
4584	025246	150300			BISB	R3, R0		
4585	025250	005003			CLR	R3		MAKE ALL OTHERS ZERO
4586	025252	000716			BR	22\$		GO NORMALIZE
4587	025254	010302		25\$:	MOV	R3, R2		
4588	025256	005003			CLR	R3		
4589	025260	000301		24\$:	SWAB	R1		MOVE LEFT
4590	025262	150100			BISB	R1, R0		
4591	025264	105001			CLRB	R1		
4592	025266	000302			SWAB	R2		
4593	025270	150201			BISB	R2, R1		
4594	025272	105002			CLRB	R2		
4595	025274	000303			SWAB	R3		
4596	025276	150302			BISB	R3, R2		
4597	025300	105003			CLRB	R3		
4598	025302	000702			BR	22\$		
4599	025304	005016		21\$:	CLR	(SP)		SET POSITIVE
4600	025306	005004			CLR	R4		
4601	025310	000137	024610		JMP	16\$		
4602								;

; \*END OF ADD/SUB

4603  
4604  
4605  
4606  
4607  
4608  
4609  
4610  
4611  
4612  
4613  
4614  
4615  
4616  
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  
4642  
4643  
4644  
4645  
4646  
4647  
4648  
4649  
4650  
4651  
4652  
4653  
4654  
4655  
4656  
4657  
4658

```

*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
*           ADDR(OPERAND A)
*           SPUSH
*           ADDR(OPERAND B)
*           SMUL
*           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,
* 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 PDP-11 FORTRAN SOFTWARE
* BY DONALD NORTH, SEPTEMBER, 1975.

```

```

4659 ;*
4660
4661 025314 005237 002550 $MUL: INC MULCO ;CTR: TOTAL NUMBER MUL
4662 025320 042737 000017 002400 BIC #17,$FPS ;CLEAR STATUS BITS N Z V C
4663 025326 105737 002400 TSTB $FPS ;TEST MODE
4664 025332 100402 BMI 15$ ;D-MODE
4665 025334 004737 027672 JSR PC,$CONV ;F-MODE: CONVERT 2 F OPDS TO D
4666 025340 010046 15$: MOV R0,-(SP)
4667 025342 010146 MOV R1,-(SP)
4668 025344 010246 MOV R2,-(SP) ;SAVE ALL REGISTERS
4669 025346 010346 MOV R3,-(SP)
4670 025350 010446 MOV R4,-(SP)
4671 025352 010546 MOV R5,-(SP)
4672 025354 006366 000014 ASL 14(SP) ;SHIFT MULTIPLICAND
4673 025360 006146 ROL -(SP) ;KEEP SIGN
4674 025362 005046 CLR -(SP) ;CLEAR EXPONENT
4675 025364 116616 000021 MOVB 21(SP),(SP) ;KEEP MULTIPLICAND EXP
4676 025370 001436 BEQ 1$ ;ANSWER ZERO
4677 025372 116666 000020 000021 MOVB 20(SP),21(SP) ;SHIFT FRACTION LEFT
4678 025400 000261 SEC ;INSERT NORMAL BIT
4679 025402 006066 000020 ROR 20(SP)
4680 025406 116666 000023 000020 MOVB 23(SP),20(SP)
4681 025414 000366 000022 SWAB 22(SP)
4682 025420 116666 000025 000022 MOVB 25(SP),22(SP)
4683 025426 000366 000024 SWAB 24(SP)
4684 025432 116666 000027 000024 MOVB 27(SP),24(SP)
4685 025440 000366 000026 SWAB 26(SP)
4686 025444 105066 000026 CLRB 26(SP) ;EXTRA BITS
4687 025450 006366 000030 ASL 30(SP) ;SHIFT HIGH MULTIPLIER
4688 025454 005566 000002 ADC 2(SP) ;PRODUCT SIGN
4689 025460 105766 000031 TSTB 31(SP)
4690 025464 001005 BNE 2$ ;JUMP IF NON ZERO
4691 025466 022626 1$: CMP (SP)+,(SP)+ ;FLUSH SIGN AND EXPONENT
4692 025470 005237 002552 INC MULC1 ;CTR: A=0 AND/OR B=0
4693 025474 000137 026150 JMP 3$
4694 025500 005000 2$: CLR R0 ;CLEAR PRODUCT
4695 025502 005001 CLR R1
4696 025504 005002 CLR R2
4697 025506 005003 CLR R3
4698 025510 005005 CLR R5 ;CLEAR C BIT OVERFLOW CATCHER
4699 025512 006066 000030 ROR 30(SP) ;SIGN IS +
4700 025516 012746 000020 MOV #16,-(SP) ;ITERATION COUNT
4701 025522 016604 000040 MOV 40(SP),R4 ;GET LOWEST ORDER MULTIPLIER
4702 025526 001404 BEQ 4$ ;JUMP IF NO BITS HERE
4703 025530 004737 026370 JSR PC,30$
4704 025534 012716 000020 MOV #16,(SP) ;RESTORE COUNT
4705 025540 016604 000036 4$: MOV 36(SP),R4 ;GET NEXT LOWEST FRACTION
4706 025544 001003 BNE 5$ ;WORK TO DO
4707 025546 005766 000040 TST 40(SP)
4708 025552 001406 BEQ 6$ ;NO PRODUCT YET
4709 025554 004737 026364 5$: JSR PC,31$
4710 025560 004737 026272 JSR PC,32$ ;ONE BIT FULL PRECISION
4711 025564 012716 000020 MOV #16,(SP)
4712 025570 016604 000034 6$: MOV 34(SP),R4 ;NEXT TO HIGHEST ORDER FRACTION
4713 025574 001006 BNE 7$
4714 025576 005766 000036 TST 36(SP)

```

4715	025602	001003			BNE	7\$		
4716	025604	005766	000040		TST	40(SP)		
4717	025610	001402			BEQ	8\$		
4718	025612	004737	026272	7\$:	JSR	PC,32\$		
4719	025616	016604	000032	8\$:	MOV	32(SP),R4		GET HIGH ORDER BITS
4720	025622	012716	000007		MOV	#7,(SP)		SEVEN OF THEM
4721	025626	004737	026272		JSR	PC,32\$		
4722	025632	004737	026276		JSR	PC,33\$		NORMAL BIT
4723	025636	005726			TST	(SP)+		FLUSH ITERATION COUNT
4724	025640	062604			ADD	(SP)+,R4		ADD EXP
4725	025642	006303			ASL	R3		SHIFT OUT NORMAL BIT
4726	025644	006102			ROL	R2		
4727	025646	006101			ROL	R1		
4728	025650	006100			ROL	R0		
4729	025652	103405			BCS	9\$		NORMAL BIT FOUND
4730	025654	006303			ASL	R3		
4731	025656	006102			ROL	R2		
4732	025660	006101			ROL	R1		
4733	025662	006100			ROL	R0		HAVE IT
4734	025664	005304			DEC	R4		ADJUST EXP
4735	025666	162704	000200	9\$:	SUB	#200,R4		REMOVE BIAS FROM EXP
4736	025672	003022			BGT	10\$		BR IF NO UNDERFLOW
4737	025674	005237	002560		INC	MULC4		CTR: UNDERFLOW
4738	025700	032737	002000	002400	BIT	#002000,\$FPS		UNDERFLOW - IS IT ENABLED ?
4739	025706	001517			BEQ	12\$		NO, MAKE ZERO RESULT
4740	025710	005237	002562		INC	MULC5		CTR: UNDERFLOW, ENABLED
4741	025714	052737	100000	002400	BIS	#100000,\$FPS		SET ERROR BIT
4742	025722	012737	000012	002402	MOV	#12,\$FEC		SET \$FEC
4743	025730	013737	002376	002404	MOV	EXPFEA,\$FEA		SET \$FEA
4744	025736	000427			BR	11\$		CONTINUE
4745	025740	022704	000377	10\$:	CMP	#377,R4		CHECK FOR OVERFLOW
4746	025744	002024			BGE	11\$		BR IF NO OVERFLOW
4747	025746	052737	000002	002400	BIS	#02,\$FPS		SET V BIT ON OVERFLOW
4748	025754	005237	002554		INC	MULC2		CTR: OVERFLOW
4749	025760	032737	001000	002400	BIT	#001000,\$FPS		OVERFLOW ENABLED ?
4750	025766	001467			BEQ	12\$		NO, ZERO RESULT
4751	025770	005237	002556		INC	MULC3		CTR: OVERFLOW, ENABLED
4752	025774	052737	100000	002400	BIS	#100000,\$FPS		SET ERROR BIT
4753	026002	012737	000010	002402	MOV	#10,\$FEC		SET \$FEC
4754	026010	013737	002376	002404	MOV	EXPFEA,\$FEA		SET \$FEA
4755	026016	105003		11\$:	CLRB	R3		
4756	026020	150203			BISB	R2,R3		SHIFT FRACTION RIGHT
4757	026022	000303			SWAB	R3		
4758	026024	105002			CLRB	R2		
4759	026026	150102			BISB	R1,R2		
4760	026030	000302			SWAB	R2		
4761	026032	105001			CLRB	R1		
4762	026034	150001			BISB	R0,R1		
4763	026036	000301			SWAB	R1		
4764	026040	105000			CLRB	R0		
4765	026042	150400			BISB	R4,R0		
4766	026044	000300			SWAB	R0		
4767	026046	006026			ROR	(SP)+		GET PRODUCT SIGN
4768	026050	006000			ROR	R0		PUT IN RESULT
4769	026052	006001			ROR	R1		
4770	026054	006002			ROR	R2		

4771	026056	006003				ROR	R3	:	
4772	026060	005503				ADC	R3	:	ROUND RESULT
4773	026062	005502				ADC	R2	:	
4774	026064	005501				ADC	R1	:	
4775	026066	005500				ADC	R0	:	
4776	026070	103401				BCS	16\$	:	OVERFLOW ON ROUND ?
4777	026072	102032				BVC	13\$	:	
4778	026074	052737	000002	002400	16\$:	BIS	#02,\$FPS	:	YES - SET V BIT
4779	026102	005237	002554			INC	MULC2	:	CTR: OVERFLOW
4780	026106	032737	001000	002400		BIT	#001000,\$FPS	:	OVERFLOW ENABLED ?
4781	026114	001415				BEQ	3\$	:	NO, ZERO RESULT
4782	026116	005237	002556			INC	MULC3	:	CTR: OVERFLOW, ENABLED
4783	026122	052737	100000	002400		BIS	#100000,\$FPS	:	SET ERROR BIT
4784	026130	012737	000010	002402		MOV	#10,\$FEC	:	SET \$FEC
4785	026136	013737	002376	002404		MOV	EXPFEA,\$FEA	:	SET \$FEA
4786	026144	000405				BR	13\$	:	CONTINUE
4787	026146	005726			12\$:	TST	(SP)+	:	FLUSH SIGN
4788	026150	005000			3\$:	CLR	R0	:	CLEAR RESULT
4789	026152	005001				CLR	R1	:	
4790	026154	005002				CLR	R2	:	
4791	026156	005003				CLR	R3	:	
4792	026160	010066	000024		13\$:	MOV	R0,24(SP)	:	
4793	026164	010166	000026			MOV	R1,26(SP)	:	STUFF RESULT
4794	026170	010266	000030			MOV	R2,30(SP)	:	
4795	026174	010366	000032			MOV	R3,32(SP)	:	
4796	026200	032766	077600	000024		BIT	#077600,24(SP)	:	SET Z BIT IF EXPONENT ZERO
4797	026206	001003				BNE	17\$	:	
4798	026210	052737	000004	002400		BIS	#04,\$FPS	:	
4799	026216	005766	000024		17\$:	TST	24(SP)	:	SET N BIT IF RESULT NEGATIVE
4800	026222	100003				BPL	18\$	:	
4801	026224	052737	000010	002400		BIS	#10,\$FPS	:	
4802	026232	012605			18\$:	MOV	(SP)+,R5	:	
4803	026234	012604				MOV	(SP)+,R4	:	
4804	026236	012603				MOV	(SP)+,R3	:	RESTORE REGISTERS
4805	026240	012602				MOV	(SP)+,R2	:	
4806	026242	012601				MOV	(SP)+,R1	:	
4807	026244	012600				MOV	(SP)+,R0	:	
4808	026246	062706	000010			ADD	#8,SP	:	CLEAR SECOND OPERAND OFF STACK
4809	026252	105737	002400			TSTB	\$FPS	:	F OR D MODE
4810	026256	100404				BMI	14\$	:	D-MODE
4811	026260	012666	000002			MOV	(SP)+,2(SP)	:	F MODE - CONVERT
4812	026264	012666	000002			MOV	(SP)+,2(SP)	:	
4813	026270	000134			14\$:	JMP	2(R4)+	:	RETURN
4814								:	
4815	026272	006204			32\$:	ASR	R4	:	TEST NEXT MULTIPLIER BIT
4816	026274	103022				BCC	34\$	:	JUMP IF ZERO
4817	026276	066603	000032		33\$:	ADD	32(SP),R3	:	ADD IN MULTIPLICAND
4818	026302	005502				ADC	R2	:	
4819	026304	005501				ADC	R1	:	
4820	026306	005500				ADC	R0	:	
4821	026310	005505				ADC	R5	:	SAVE OVERFLOW
4822	026312	066602	000030			ADD	30(SP),R2	:	
4823	026316	005501				ADC	R1	:	
4824	026320	005500				ADC	R0	:	
4825	026322	005505				ADC	R5	:	
4826	026324	066601	000026			ADD	26(SP),R1	:	

4827	026330	005500		ADC	R0	
4828	026332	005505		ADC	R5	
4829	026334	066600	000024	ADD	24(SP),R0	
4830	026340	005505		ADC	R5	
4831	026342	006205		34\$: ASR	R5	:RECOVER OVERFLOW IF ANY
4832	026344	006000		ROR	R0	:SHIFT PRODUCT
4833	026346	006001		ROR	R1	
4834	026350	006002		ROR	R2	
4835	026352	006003		ROR	R3	
4836	026354	005366	000002	DEC	2(SP)	:COUNT LOOP
4837	026360	003344		BGT	32\$	:AGAIN
4838	026362	000207		RTS	PC	:RETURN
4839	026364	005366	000002	31\$: DEC	2(SP)	:ONLY 15 BITS THIS PASS
4840	026370	006204		30\$: ASR	R4	:TEST NEXT MULTIPLIER BIT
4841	026372	103007		BCC	35\$	:JUMP IF ZERO
4842	026374	066601	000026	ADD	26(SP),R1	:USE ONLY HIGH ORDER MULTIPLICAND
4843	026400	005500		ADC	R0	
4844	026402	005505		ADC	R5	
4845	026404	066600	000024	ADD	24(SP),R0	
4846	026410	005505		ADC	R5	
4847	026412	006205		35\$: ASR	R5	:RECOVER ANY OVERFLOW
4848	026414	006000		ROR	R0	
4849	026416	006001		ROR	R1	
4850	026420	006002		ROR	R2	
4851	026422	006003		ROR	R3	
4852	026424	005366	000002	DEC	2(SP)	:COUNT LOOP
4853	026430	003357		BGT	30\$	
4854	026432	000207		RTS	PC	:RETURN
4855						

;\*END OF MUL



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  
4896  
4897  
4898  
4899  
4900  
4901  
4902  
4903  
4904  
4905  
4906  
4907  
4908  
4909  
4910  
4911

\*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:     \$PUSH  
\*                ADDR(OPERAND A)  
\*                \$PUSH  
\*                ADDR(OPERAND B)  
\*                \$DIV  
\*                \$POPX  
\*                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,  
\* 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".  
\* 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, SFEC WILL BE  
\* SET TO 4(8), AND SFEA WILL BE SET TO THE VALUE CONTAINED  
\* IN THE LOCATION "EXPFEA".

# E10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 123  
 DQFPDA.P11 09-FEB-77 10:32 FLOATING POINT SOFTWARE ROUTINES

```

4912
4913
4914
4915
4916
4917
4918
4919
4920 026434 005237 002564          $DIV: INC      DIVCO
4921 026440 042737 000017 002400 BIC      #17,$FPS ;CTR: TOTAL NUMBER OF DIV
4922 026446 105737 002400          TSTB    $FPS  ;CLEAR STATUS BITS N Z V C
4923 026452 100402          BMI      14$  ;TEST MODE
4924 026454 004737 027672          JSR     PC,$CONV ;D-MODE
4925 026460 010046          14$:  MOV    R0,-(SP) ;F-MODE: CONVERT 2 F OPDS TO D
4926 026462 010146          MOV    R1,-(SP)
4927 026464 010246          MOV    R2,-(SP)
4928 026466 010346          MOV    R3,-(SP) ;SAVE REGISTERS
4929 026470 010446          MOV    R4,-(SP)
4930 026472 010546          MOV    R5,-(SP)
4931 026474 032766 077600 000014 BIT     #077600,14(SP) ;DIVIDE BY ZERO ?
4932 026502 001015          BNE     2$
4933 026504 005237 002570          INC     DIVC2 ;CTR: DENOM=0
4934 026510 052737 100000 002400 BIS     #100000,$FPS ;YES, SET ERROR BIT
4935 026516 012737 000004 002402 MOV     #4,$FEC ;SET $FEC
4936 026524 013737 002376 002404 MOV     EXPFEA,$FEA ;SET $FEA
4937 026532 000137 027442          JMP     9$ ;DONE
4938 026536 005000          2$:  CLR    R0
4939 026540 005001          CLR    R1
4940 026542 005002          CLR    R2
4941 026544 005003          CLR    R3
4942 026546 005046          CLR    -(SP)
4943 026550 006366 000026 ASL     26(SP) ;SHIFT NUMERATOR
4944 026554 006116          ROL    (SP) ;NUMERATOR SIGN
4945 026556 005046          CLR    -(SP)
4946 026560 156616 000031 BISB   31(SP),(SP) ;NUMERATOR EXP
4947 026564 001004          BNE     6$ ;NUMERATOR IS ZERO?
4948 026566 005237 002566          INC     DIVC1 ;CTR: NUMER=0
4949 026572 000137 027416          JMP     1$ ;YES, DONE
4950 026576 156600 000030 6$:  BISB   30(SP),R0
4951 026602 000300          SWAB   R0 ;LEFT JUSTIFY NUMERATOR FRACTION
4952 026604 000261          SEC
4953 026606 006000          ROR    R0 ;INSERT NORMAL BIT
4954 026610 156600 000033 BISB   33(SP),R0
4955 026614 156601 000032 BISB   32(SP),R1
4956 026620 000301          SWAB   R1
4957 026622 156601 000035 BISB   35(SP),R1
4958 026626 156602 000034 BISB   34(SP),R2
4959 026632 000302          SWAB   R2
4960 026634 156602 000037 BISB   37(SP),R2
4961 026640 156603 000036 BISB   36(SP),R3
4962 026644 000303          SWAB   R3
4963 026646 006366 000020 ASL     20(SP) ;SHIFT DENOMINATOR
4964 026652 005566 000002 ADC     2(SP) ;RESULT SIGN
4965 026656 005004          CLR    R4
4966 026660 156604 000021 BISB   21(SP),R4 ;DIVISOR EXPONENT
4967 026664 160416          SUB    R4,(SP) ;SUBTRACT EXPONENTS
  
```

# F10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 124  
 DQFPDA.P11 09-FEB-77 10:32 FLOATING POINT SOFTWARE ROUTINES

4968	026666	000366	000020		SWAB	20(SP)	:LEFT JUSTIFY DENOM
4969	026672	000261			SEC		:INSERT NORMAL BIT
4970	026674	006066	000020		ROR	20(SP)	
4971	026700	116666	000023	000020	MOVW	23(SP),20(SP)	
4972	026706	116666	000022	000023	MOVW	22(SP),23(SP)	
4973	026714	116666	000025	000022	MOVW	25(SP),22(SP)	
4974	026722	116666	000024	000025	MOVW	24(SP),25(SP)	
4975	026730	116666	000027	000024	MOVW	27(SP),24(SP)	
4976	026736	116666	000026	000027	MOVW	26(SP),27(SP)	
4977	026744	105066	000026		CLRB	26(SP)	
4978	026750	005066	000030		CLR	30(SP)	:CLEAR QUOTIENT
4979	026754	005066	000032		CLR	32(SP)	
4980	026760	005066	000034		CLR	34(SP)	
4981	026764	020066	000020		CMP	R0,20(SP)	:COMPARE HIGH NUM + DENOM
4982	026770	101020			BHI	3\$	:JUMP IF DENOM LOW
4983	026772	103424			BLO	4\$	:JUMP IF DENOM HI
4984	026774	020166	000022		CMP	R1,22(SP)	:COMPARE LOW ORDER PARTS
4985	027000	101014			BHI	3\$	
4986	027002	103420			BLO	4\$	
4987	027004	020266	000024		CMP	R2,24(SP)	
4988	027010	101010			BHI	3\$	
4989	027012	103414			BLO	4\$	
4990	027014	020366	000026		CMP	R3,26(SP)	
4991	027020	101004			BHI	3\$	
4992	027022	001010			BNE	4\$	
4993	027024	005216			INC	(SP)	:BUMP EXP
4994	027026	005004			CLR	R4	
4995	027030	000443			BR	5\$	
4996							
4997	027032	006000		3\$:	ROR	R0	:HALVE DENOM (C=0)
4998	027034	006001			ROR	R1	:TO INSURE N<D
4999	027036	006002			ROR	R2	
5000	027040	006003			ROR	R3	
5001	027042	005216			INC	(SP)	:COMPENSATE EXP
5002	027044	012705	000011	4\$:	MOV	#9,R5	:FIRST NINE QUOTIENT BITS
5003	027050	004737	027534		JSR	PC,30\$	
5004	027054	110466	000030		MOVW	R4,30(SP)	:SAVE ALL HIGH ORDER Q FRACTION
5005	027060	005705			TST	R5	:DONE?
5006	027062	001025			BNE	10\$	:YES - REST OF NUMBER IS 0
5007	027064	012705	000020		MOV	#16,R5	:16 MORE BITS
5008	027070	004737	027534		JSR	PC,30\$	
5009	027074	010466	000032		MOV	R4,32(SP)	
5010	027100	005705			TST	R5	
5011	027102	001015			BNE	10\$	
5012	027104	012705	000020		MOV	#16,R5	
5013	027110	004737	027534		JSR	PC,30\$	
5014	027114	010466	000034		MOV	R4,34(SP)	
5015	027120	005705			TST	R5	
5016	027122	001005			BNE	10\$	
5017	027124	012705	000020		MOV	#16,R5	
5018	027130	004737	027534		JSR	PC,30\$	
5019	027134	000401			BR	5\$	
5020	027136	005004		10\$:	CLR	R4	:CLEAR LOWEST ORDER QUOTIENT
5021	027140	012605		5\$:	MOV	(SP)+,R5	:PUSH UP EXPONENT
5022	027142	062705	000200		ADD	#200,R5	:INSERT BIAS
5023	027146	003022			BGT	7\$	:BR IF NO UNDERFLOW

## G10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 125  
 DQFPDA.P11 09-FEB-77 10:32 FLGATING POINT SOFTWARE ROUTINES

5024	027150	005237	002576			INC	DIVC5	:CTR: UNDERFLOW
5025	027154	032737	002000	002400		BIT	#002000,\$FPS	:UNDERFLOW - IS IT ENABLED ?
5026	027162	001516				BEQ	15\$	:NO, MAKE ZERO RESULT
5027	027164	005237	002600			INC	DIVC6	:CTR: UNDERFLOW, ENABLED
5028	027170	052737	100000	002400		BIS	#100000,\$FPS	:SET ERROR BIT
5029	027176	012737	000012	002402		MOV	#12,\$FEC	:SET \$FEC
5030	027204	013737	002376	002404		MOV	EXPFEA,\$FEA	:SET \$FEA
5031	027212	000427				BR	11\$	:CONTINUE
5032	027214	022705	000377		7\$:	CMP	#377,R5	:CHECK FOR OVERFLOW
5033	027220	002024				BGE	11\$	:BR IF NO OVERFLOW
5034	027222	052737	000002	002400		BIS	#02,\$FPS	:SET V BIT ON OVERFLOW
5035	027230	005237	002572			INC	DIVC3	:CTR: OVERFLOW
5036	027234	032737	001000	002400		BIT	#001000,\$FPS	:OVERFLOW ENABLED ?
5037	027242	001466				BEQ	15\$	:NO, ZERO RESULT
5038	027244	005237	002574			INC	DIVC4	:CTR: OVERFLOW, ENABLED
5039	027250	052737	100000	002400		BIS	#100000,\$FPS	:SET ERROR BIT
5040	027256	012737	000010	002402		MOV	#10,\$FEC	:SET \$FEC
5041	027264	013737	002376	002404		MOV	EXPFEA,\$FEA	:SET \$FEA
5042	027272	110566	000027		11\$:	MOV#B	R5,27(SP)	:PUT EXPIN RESULT
5043	027276	006026				ROR	(SP)+	:INSERT SIGN
5044	027300	006066	000024			ROR	24(SP)	
5045	027304	006066	000026			ROR	26(SP)	
5046	027310	006066	000030			ROR	30(SP)	
5047	027314	006004				ROR	R4	
5048	027316	005504				ADC	R4	:ROUND
5049	027320	005566	000030			ADC	30(SP)	
5050	027324	005566	000026			ADC	26(SP)	
5051	027330	005566	000024			ADC	24(SP)	
5052	027334	010466	000032			MOV	R4,32(SP)	:INSERT LOW ORDER FRACTION
5053	027340	103401				BCS	16\$	:OVERFLOW ON ROUND ?
5054	027342	102037				BVC	9\$	
5055	027344	052737	000002	002400	16\$:	BIS	#02,\$FPS	:YES - SET V BIT
5056	027352	005237	002572			INC	DIVC3	:CTR: OVERFLOW
5057	027356	032737	001000	002400		BIT	#001000,\$FPS	:OVERFLOW ENABLED ?
5058	027364	001416				BEQ	19\$	:NO, ZERO RESULT
5059	027366	005237	002574			INC	DIVC4	:CTR: OVERFLOW, ENABLED
5060	027372	052737	100000	002400		BIS	#100000,\$FPS	:SET ERROR BIT
5061	027400	012737	000010	002402		MOV	#10,\$FEC	:SET \$FEC
5062	027406	013737	002376	002404		MOV	EXPFEA,\$FEA	:SET \$FEA
5063	027414	000412				BR	9\$	:CONTINUE
5064	027416	005726			1\$:	TST	(SP)+	:FLUSH EXP
5065	027420	005726			15\$:	TST	(SP)+	:FLUSH SIGN
5066	027422	005066	000024		19\$:	CLR	24(SP)	:CLEAR RESULT
5067	027426	005066	000026			CLR	26(SP)	
5068	027432	005066	000030			CLR	30(SP)	
5069	027436	005066	000032			CLR	32(SP)	
5070	027442	032766	077600	000024	9\$:	BIT	#077600,24(SP)	:SET Z BIT IF EXPONENT ZERO
5071	027450	001003				BNE	17\$	
5072	027452	052737	000004	002400		BIS	#04,\$FPS	
5073	027460	005766	000024		17\$:	TST	24(SP)	:SET N BIT IF RESULT NEGATIVE
5074	027464	100003				BPL	18\$	
5075	027466	052737	000010	002400		BIS	#10,\$FPS	
5076	027474	012605			18\$:	MOV	(SP)+,R5	
5077	027476	012604				MOV	(SP)+,R4	
5078	027500	012603				MOV	(SP)+,R3	:RESTORE REGISTERS
5079	027502	012602				MOV	(SP)+,R2	

# H10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 126  
 DQFPDA.P11 09-FEB-77 10:32 FLOATING POINT SOFTWARE ROUTINES

5080	027504	012601		MOV	(SP)+,R1	;
5081	027506	012600		MOV	(SP)+,R0	:
5082	027510	062706	000010	ADD	#8,SP	:FLUSH FIRST ARG
5083	027514	105737	002400	TSTB	\$FPS	:F OR D MODE
5084	027520	100404		BMI	13\$	:D MODE
5085	027522	012666	000002	MOV	(SP)+,2(SP)	:F MODE - CONVERT
5086	027526	012666	000002	MOV	(SP)+,2(SP)	:
5087	027532	000134		13\$: JMP	@(R4)+	:RETURN
5088						
5089	027534	006304		30\$: ASL	R4	:SHIFT QUOTIENT
5090	027536	006303		ASL	R3	:
5091	027540	006102		ROL	R2	:
5092	027542	006101		ROL	R1	:
5093	027544	006100		ROL	R0	:
5094	027546	103420		BCS	31\$	:GUARANTEED TO GO
5095	027550	026600	000022	CMP	22(SP),R0	:COMPARE HIGH DIVISOR AND DIVIDEND
5096	027554	101034		BHI	32\$	:DIVISOR BIGGER
5097	027556	103414		BLO	31\$	:
5098	027560	026601	000024	CMP	24(SP),R1	:CHECK LOW ORDERS
5099	027564	101030		BHI	32\$	:
5100	027566	103410		BLO	31\$	:
5101	027570	026602	000026	CMP	26(SP),R2	:
5102	027574	101024		BHI	32\$	:
5103	027576	103404		BLO	31\$	:
5104	027600	026603	000030	CMP	30(SP),R3	:
5105	027604	101020		BHI	32\$	:
5106	027606	001422		BEQ	33\$	:NUMER=DENOM
5107	027610	166603	000030	31\$: SUB	30(SP),R3	:N=N-D
5108	027614	005602		SBC	R2	:
5109	027616	005601		SBC	R1	:
5110	027620	005600		SBC	R0	:
5111	027622	166602	000026	SUB	26(SP),R2	:
5112	027626	005601		SBC	R1	:
5113	027630	005600		SBC	R0	:
5114	027632	166601	000024	SUB	24(SP),R1	:
5115	027636	005600		SBC	R0	:
5116	027640	166600	000022	SUB	22(SP),R0	:
5117	027644	005204		INC	R4	:INSERT QUOTIENT BIT
5118	027646	005305		32\$: DEC	R5	:COUNT LOOP
5119	027650	003331		BGT	30\$	:
5120	027652	000207		RTS	PC	:RETURN
5121	027654	005204		33\$: INC	R4	:INSERT LAST 1 BIT IN QUOTIENT
5122	027656	000401		BR	34\$	:
5123	027660	006304		35\$: ASL	R4	:FINISH OUT QUOTIENT WITH ZEROS
5124	027662	005305		34\$: DEC	R5	:
5125	027664	003375		BGT	35\$	:
5126	027666	005205		INC	R5	:FLAG NO MORE NUMER
5127	027670	000207		RTS	PC	:RETURN
5128						:*END OF DIV

# I10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 127  
 DQFPDA.P11 09-FEB-77 10:32 FLOATING POINT SOFTWARE ROUTINES

```

5129                                     ;*CONVERT TOP 2 OPERANDS ON STACK FROM F MODE
5130                                     ;*   TO D MODE
5131                                     ;*
5132                                     ;* THIS ROUTINE TAKES THE TOP TWO 2-WORD (SINGLE
5133                                     ;* PRECISION) FLOATING POINT NUMBERS ON THE
5134                                     ;* STACK AND CONVERTS THEM BOTH TO 4-WORD
5135                                     ;* (DOUBLE PRECISION) FORMAT BY APPENDING
5136                                     ;* TWO WORDS OF ZEROS AS THE LOW ORDER BIT
5137                                     ;* EXTENSION.
5138                                     ;*
5139                                     ;*
5140 027672 005046          $CONV: CLR      -(SP)          ;CLEAR WORD 3 OF A
5141 027674 016646 000006  MOV      6(SP),-(SP)      ;MOVE WORD 2 OF A
5142 027700 016646 000006  MOV      6(SP),-(SP)      ;MOVE WORD 1 OF A
5143 027704 016646 000006  MOV      6(SP),-(SP)      ;MOVE RETURN ADDR TO TOP
5144 027710 016666 000020 000014  MOV     20(SP),14(SP)    ;MOVE WORD 2 OF B
5145 027716 016666 000016 000012  MOV     16(SP),12(SP)    ;MOVE WORD 1 OF B
5146 027724 005066 000020      CLR     20(SP)          ;CLEAR WORD 4 OF B
5147 027730 005066 000016      CLR     16(SP)          ;CLEAR WORD 3 OF B
5148 027734 005066 000010      CLR     10(SP)          ;CLEAR WORD 4 OF A
5149 027740 000207          RTS      PC
  
```

5150  
5151  
5152  
5153  
5154  
5155  
5156  
5157  
5158  
5159  
5160  
5161  
5162  
5163  
5164  
5165  
5166  
5167  
5168  
5169  
5170  
5171  
5172  
5173  
5174  
5175  
5176  
5177  
5178  
5179  
5180  
5181  
5182  
5183  
5184  
5185  
5186  
5187  
5188  
5189  
5190  
5191  
5192  
5193  
5194  
5195  
5196  
5197  
5198  
5199  
5200  
5201  
5202  
5203  
5204  
5205

027742  
027742  
027742 032777 040000 151176  
027750 001114  
  
027752 000416  
  
027754 013746 000004  
027760 012737 030000 000004  
027766 005737 177060  
027772 012637 000004  
027776 000463  
030000 022626  
030002 012637 000004  
030006 000423  
030010  
030010 032777 000400 151130  
030016 001404  
030020 023737 001112 001102  
030026 001465  
030030 005737 001104  
030034 001421  
030036 023737 001122 001104  
030044 101015  
030046 032777 001000 151072  
030054 001404  
030056 013737 001114 001110  
030064 000446  
030066 005037 001104  
030072 005037 001166  
030076 000415  
030100 032777 004000 151040  
030106 001011  
030110 005737 001210  
030114 001406  
030116 005237 001106  
030122 023737 001166 001106  
030130 002024  
030132 012737 000001 001106  
030140 013737 030216 001166  
030146 005237 001102  
030152 013737 001102 001206  
030160 011637 001110

```
.SBTTL SCOPE HANDLER ROUTINE
;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<15:0>)
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW14=1 LOOP ON TEST
;SW11=1 INHIBIT ITERATIONS
;SW09=1 LOOP ON ERROR
;SW08=1 LOOP ON TEST IN "$LPTST"
;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)
MOV @ERRVEC, -(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5, @ERRVEC ;;SET FOR TIMEOUT
TST @177060 ;;TIME OUT ON XOR?
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 $LPTST,$TSTNM ;;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 #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
$SVLAD: INC $TSTNM ;;COUNT TEST NUMBERS
MOV $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
```

K10

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 129  
DQFPDR.P11 09-FEB-77 10:32 SCOPE HANDLER ROUTINE

5206	030164	011637	001114		MOV	(SP), SLPERR	::: SAVE ERROR LOOP ADDRESS
5207	030170	005037	001170		CLR	\$ESCAPE	::: CLEAR THE ESCAPE FROM ERROR ADDRESS
5208	030174	012737	000001	001122	MOV	#1, \$ERMAX	::: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
5209	030202	013777	001102	150740	\$OVER: MOV	\$STNM, \$DISPLAY	::: DISPLAY TEST NUMBER
5210	030210	013716	001110		MOV	\$LPADR, (SP)	::: FUDGE RETURN ADDRESS
5211	030214	000002			RTI		::: FIXES PS
5212	030216	000400			\$MXCNT: 400		::: MAX. NUMBER OF ITERATIONS



.SBTTL ERROR HANDLER ROUTINE

5213  
5214  
5215  
5216  
5217  
5218  
5219  
5220  
5221  
5222  
5223  
5224  
5225  
5226  
5227  
5228  
5229  
5230  
5231  
5232  
5233  
5234  
5235  
5236  
5237  
5238  
5239  
5240  
5241  
5242  
5243  
5244  
5245  
5246  
5247  
5248  
5249  
5250  
5251  
5252  
5253  
5254  
5255  
5256  
5257  
5258  
5259  
5260  
5261  
5262  
5263  
5264  
5265  
5266  
5267  
5268

\*\*\*\*\*  
: 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 STYPERR 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

ERROR:

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 #4, EREG6  
MOV (SP), EREG7 ; PC -> ERROR CALL INSTR  
INC \$ERFLG ; SET THE ERROR FLAG  
BEQ 7\$ ; DON'T LET THE FLAG GO TO ZERO  
MOV \$STNM, @DISPLAY ; DISPLAY TEST NUMBER  
BIT #BIT10, @SWR ; BELL ON ERROR?  
BEQ 1\$ ; NO - SKIP  
TYPE \$BELL ; RING BELL  
INC \$ERTTL ; COUNT THE NUMBER OF ERRORS  
MOV (SP), \$ERRPC ; GET ADDRESS OF ERROR INSTRUCTION  
SUB #2, \$ERRPC ; STRIP AND SAVE THE ERROR ITEM CODE  
MOVB @ERRPC, \$ITEMB ; SKIP TYPEOUT IF SET  
BIT #BIT13, @SWR ; SKIP TYPEOUTS  
BNE 20\$ ; GO TO USER ERROR ROUTINE  
JSR PC, STYPERR  
TYPE , \$CRLF  
20\$: CMPB #APTENV, \$ENV ; RUNNING IN APT MODE  
BNE 2\$ ; NO SKIP APT ERROR REPORT  
MOVB \$ITEMB, 21\$ ; SET ITEM NUMBER AS ERROR NUMBER  
JSR PC, \$ATY4 ; REPORT FATAL ERROR TO APT  
21\$: .BYTE 0  
.BYTE 0  
22\$: BR 22\$ ; APT ERROR LOOP  
2\$: TST @SWR ; HALT ON ERROR  
BPL 3\$ ; SKIP IF CONTINUE  
HALT ; HALT ON ERROR!  
3\$: BIT #BIT09, @SWR ; LOOP ON ERROR SWITCH SET?  
BEQ 4\$ ; BR IF NO  
MOV \$LPERR, (SP) ; FUDGE RETURN FOR LOOPING  
4\$: TST \$ESCAPE ; CHECK FOR AN ESCAPE ADDRESS  
BEQ 5\$ ; BR IF NONE  
MOV \$ESCAPE, (SP) ; FUDGE RETURN ADDRESS FOR ESCAPE  
5\$:

M10

5269	030446	022737	022160	000042	CMP	#SENDAD,0#42	::ACT-11 AUTO-ACCEPT?
5270	030454	001001			BNE	65	::BRANCH IF NO
5271	030456	000000			HALT		::YES
5272	030460			65:			
5273	030460	000002		645:	RTI		;RETURN

```

5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298 030462
5299 030462 104401
5300 030464 001177
5301 030466 010046
5302 030470 010146
5303 030472 005000
5304 030474 153700 001120
5305 030500 001004
5306
5307 030502 013746 001124
5308 030506 104402
5309 030510 000454
5310 030512 005300
5311 030514 006300
5312 030516 006300
5313 030520 010001
5314 030522 006300
5315 030524 006300
5316 030526 060100
5317 030530 062700 001232
5318 030534 012037 030544
5319 030540 001404
5320 030542 104401
5321 030544 000000
5322 030546 104401 001177
5323 030552 104401 030702
5324 030556 012037 030566
5325 030562 001402
5326 030564 104401
5327 030566 000000
5328 030570 104401 001177
5329 030574 017746 000074

```

```

;*****
.SBTTL ERROR MESSAGE TYPEOUT ROUTINE (MODIFIED SYSMAC)
;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE",
;($ERRTB) 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
;
$STYPERR:
HOTWARM: .WORD    $CRLF
          MOV     RO,-(SP)
          MOV     R1,-(SP)
          CLR     RO
          BISB   2,$ITEMB,RO
          BNE    1$
          MOV     $ERRPC,-(SP)
          TYPOC
          BR     7$
1$:      DEC     RO
          ASL    RO
          ASL    RO
          MOV    RO,R1
          ASL    RO
          ASL    RO
          ADD    R1,RO
          ADD    #,$ERRTB,RO
          MOV    (RO)+,2$
          BEQ   3$
          TYPE
2$:      .WORD    0
          TYPE   , $CRLF
3$:      TYPE   , 11$
          MOV    (RO)+,4$
          BEQ   5$
          TYPE
4$:      .WORD    0
5$:      TYPE   , $CRLF
          MOV    2BS,-(SP)
;
; START WITH MESSAGE PREFIX, HOT OR WARM
; PTR TO "HOT" OR "WARM"
; SAVE RO
; SAVE R1
; PICKUP ITEM INDEX
; IF ITEM NUMBER FROM ERROR 0,
; JUST TYPE PC OF ERROR
; GET ERROR PC FOR TYPEOUT
; TYPE OCTAL, ALL DIGITS
; EXIT
; ADJUST ERROR # FOR TABLE INDEX
; OF 20. BYTES/ENTRY
;
; FORM TABLE PTR
; PICKUP "ERROR MESSAGE" PTR
; SKIP TYPEOUT IF NULL
; TYPE "ERROR MESSAGE"
; "ERROR MESSAGE" PTR HERE
; CR & LF
; "TEST # ERR PC" HEADER
; PICKUP "DATA HEADER" PTR
; SKIP TYPEOUT IF NULL
; TYPE "DATA HEADER"
; "DATA HEADER" PTR HERE
; CR & LF
; ($TESTN)

```

5330	030600	104402				TYP0C		:OCTAL W/ LEADING ZEROS
5331	030602	104401	030700			TYPE	10\$	:<HT>
5332	030606	017746	000064			MOV	29\$,-(SP)	:(SERRPC)
5333	030612	104402				TYP0C		:OCTAL W/ LEADING ZEROS
5334	030614	104401	030700			TYPE	10\$	:<HT>
5335	030620	012001				MOV	(R0)+,R1	:PICKUP "DATA TABLE" PTR
5336	030622	001407				BEQ	7\$	:EXIT IF NULL
5337	030624	013146		6\$:		MOV	2(R1)+,-(SP)	:SAVE ... FOR TYPEOUT
5338	030626	104402				TYP0C		:TYPE OCTAL, ALL DIGITS
5339	030630	005711		14\$:		TST	(R1)	:ANOTHER NUMBER ?
5340	030632	001403				BEQ	7\$	:NO - EXIT
5341	030634	104401	030700			TYPE	10\$	:TAB BETWEEN ELEMENTS
5342	030640	000771				BR	6\$	:LOOP ON DATA TABLE VECTOR
5343	030642	104401	001177		7\$:	TYPE	\$CRLF	:END THE LINE
5344	030646	012001				MOV	(R0)+,R1	:GET OPERAND LIST PTR
5345	030650	001406				BEQ	15\$	:ALL DONE
5346	030652	012137	030662			MOV	(R1)+,12\$	:OPERAND TITLE
5347	030656	001402				BEQ	13\$	:SKIP IF ZERO
5348	030660	104401				TYPE		:TYPE IT
5349	030662	000000		12\$:		.WORD	0	:FROM HERE
5350	030664	000761		13\$:		BR	14\$	:GO FINISH DATA LIST
5351	030666	012601		15\$:		MOV	(SP)+,R1	:RESTORE R1
5352	030670	012600				MOV	(SP)+,R0	:RESTORE R0
5353	030672	000207				RTS	PC	:RETURN
5354	030674	001206		8\$:		.WORD	\$TESTN	
5355	030676	001124		9\$:		.WORD	\$ERRPC	
5356	030700	000011		10\$:		.ASCIZ	<11>	:<HT>
5357	030702	042524	052123 021440	11\$:		.ASCIZ	"TEST # ERR PC	
5358	030710	042411	051122 050040					
5359	030716	004503	000					
5360		030722				.EVEN		

```

5361 .SBTTL TYPE ROUTINE
5362
5363 *****
5364 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5365 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5366 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5367 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5368 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5369
5370 *
5371 *CALL:
5372 *1) USING A TRAP INSTRUCTION
5373 * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
5374 *OR
5375 * TYPE
5376 * MESADR
5377 *
5378 030722 105737 001165 $TYPE: TSTB $TFPLG ;; IS THERE A TERMINAL?
5379 030726 100002 BPL 1$ ;; BR IF YES
5380 030730 000000 HALT ;; HALT HERE IF NO TERMINAL
5381 030732 000430 BR 3$ ;; LEAVE
5382 030734 010046 1$: MOV RO,-(SP) ;; SAVE RO
5383 030736 017600 000002 MOV 22(SP),RO ;; GET ADDRESS OF ASCIZ STRING
5384 030742 122737 000001 001222 CMPB #APTENV,$ENV ;; RUNNING IN APT MODE
5385 030750 001011 BNE 62$ ;; NO, GO CHECK FOR APT CONSOLE
5386 030752 132737 000100 001223 BITB #APTPOOL,$ENVM ;; SPOOL MESSAGE TO APT
5387 030760 001405 BEQ 62$ ;; NO, GO CHECK FOR CONSOLE
5388 030762 010037 030772 MOV RO,61$ ;; SETUP MESSAGE ADDRESS FOR APT
5389 030766 004737 031212 JSR PC,$ATY3 ;; SPOOL MESSAGE TO APT
5390 030772 000000 61$: .WORD 0 ;; MESSAGE ADDRESS
5391 030774 132737 000040 001223 62$: BITB #APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
5392 031002 001003 BNE 60$ ;; YES, SKIP TYPE OUT
5393 031004 112046 2$: MOVB (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
5394 031006 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
5395 031010 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
5396 031012 012600 60$: MOV (SP)+,RO ;; RESTORE RO
5397 031014 062716 000002 3$: ADD #2,(SP) ;; ADJUST RETURN PC
5398 031020 000002 RTI ;; RETURN
5399 031022 122716 000011 4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
5400 031026 001430 BEQ 8$
5401 031030 122716 000200 CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
5402 031034 001006 BNE 5$
5403 031036 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
5404 031040 104401 TYPE ;; TYPE A CR AND LF
5405 031042 001177 $CRLF
5406 031044 105037 031200 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
5407 031050 000755 BR 2$ ;; GET NEXT CHARACTER
5408 031052 004737 031134 5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
5409 031056 123726 001164 6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
5410 031062 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
5411 031064 013746 001162 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
5412 ;; AND THE NULL CHAR.
5413 031070 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
5414 031074 002770 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
5415 031076 004737 031134 JSR PC,$TYPEC ;; GO TYPE A NULL
5416 031102 105337 031200 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT

```

```

5417 031106 000770          BR      7$          ;;LOOP
5418
5419          ;HORIZONTAL TAB PROCESSOR
5420
5421 031110 112716 000040      8$:      MOVB      #' (SP)          ;;REPLACE TAB WITH SPACE
5422 031114 004737 031134      9$:      JSR        PC,$TYPEC          ;;TYPE A SPACE
5423 031120 132737 000007 031200      BITB      #',$SCHARCNT          ;;BRANCH IF NOT AT
5424 031126 001372          BNE      9$          ;;TAB STOP
5425 031130 005726          TST      (SP)+          ;;POP SPACE OFF STACK
5426 031132 000724          BR        2$          ;;GET NEXT CHARACTER
5427 031134 105777 150016      $TYPEC: TSTB      2$TPS          ;;WAIT UNTIL PRINTER IS READY
5428 031140 100375          BPL      $TYPEC
5429 031142 116677 000002 150010      MOVB      2(SP),2$TPB          ;;LOAD CHAR TO BE TYPED INTO DATA REG.
5430 031150 122766 000015 000002      CMPB      #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
5431 031156 001003          BNE      1$          ;;BRANCH IF NO
5432 031160 105037 031200      CLRB      $SCHARCNT          ;;YES--CLEAR CHARACTER COUNT
5433 031164 000406          BR        $TYPEX          ;;EXIT
5434 031166 122766 000012 000002 1$:      CMPB      #LF,2(SP)          ;;IS CHARACTER A LINE FEED?
5435 031174 001402          BEQ      $TYPEX          ;;BRANCH IF YES
5436 031176 105227          INCB      (PC)+          ;;COUNT THE CHARACTER
5437 031200 000000      $SCHARCNT: .WORD 0          ;;CHARACTER COUNT STORAGE
5438 031202 000207      $TYPEX: RTS      PC
5439
  
```

```

5440          .SBTTL  APT COMMUNICATIONS ROUTINE
5441
5442          ::*****
5443 031204 112737 000001 031450 $ATY1:  MOVB  #1,$FFLG      ;; TO REPORT FATAL ERROR
5444 031212 112737 000001 031446 $ATY3:  MOVB  #1,$MFLG      ;; TO TYPE A MESSAGE
5445 031220 000403                BR      $ATYC
5446 031222 112737 000001 031450 $ATY4:  MOVB  #1,$FFLG      ;; TO ONLY REPORT FATAL ERROR
5447 031230                $ATYC:
5448 031230 010046                MOV    RO,-(SP)      ;; PUSH RO ON STACK
5449 031232 010146                MOV    R1,-(SP)      ;; PUSH R1 ON STACK
5450 031234 105737 031446                TSTB  $MFLG      ;; SHOULD TYPE A MESSAGE?
5451 031240 001450                BEQ   5$          ;; IF NOT: BR
5452 031242 122737 000001 001222                CMPB  #APTENV,$ENV      ;; OPERATING UNDER APT?
5453 031250 001031                BNE   3$          ;; IF NOT: BR
5454 031252 132737 000100 001223                BITB  #APTPOOL,$ENVM    ;; SHOULD SPOOL MESSAGES?
5455 031260 001425                BEQ   3$          ;; IF NOT: BR
5456 031262 017600 000004                MOV    24(SP),RO      ;; GET MESSAGE ADDR.
5457 031266 062766 000002 000004                ADD    #2,4(SP)      ;; BUMP RETURN ADDR.
5458 031274 005737 001202          1$:  TST   $MSGTYPE      ;; SEE IF DONE W/ LAST XMISSION?
5459 031300 001375                BNE   1$          ;; IF NOT: WAIT
5460 031302 010037 001216                MOV    RO,$MSGAD      ;; PUT ADDR IN MAILBOX
5461 031306 105720          2$:  TSTB  (RO)+          ;; FIND END OF MESSAGE
5462 031310 001376                BNE   2$
5463 031312 163700 001216                SUB   $MSGAD,RO      ;; SUB START OF MESSAGE
5464 031316 006200                ASR   RO            ;; GET MESSAGE LNTH IN WORDS
5465 031320 010037 001220                MOV    RO,$MSGGLT      ;; PUT LENGTH IN MAILBOX
5466 031324 012737 000004 001202                MOV    #4,$MSGTYPE      ;; TELL APT TO TAKE MSG.
5467 031332 000413                BR    5$
5468 031334 017637 000004 031360          3$:  MOV    24(SP),4$      ;; PUT MSG ADDR IN JSR LINKAGE
5469 031342 062766 000002 000004                ADD    #2,4(SP)      ;; BUMP RETURN ADDRESS
5470 031350 013746 177776                MOV    177776,-(SP)   ;; PUSH 177776 ON STACK
5471 031354 004737 030722                JSR   PC,$TYPE      ;; CALL TYPE MACRO
5472 031360 000000          4$:  .WORD  0
5473 031362          5$:
5474 031362 105737 031450          10$:  TSTB  $FFLG      ;; SHOULD REPORT FATAL ERROR?
5475 031366 001416                BEQ   12$          ;; IF NOT: BR
5476 031370 005737 001222                TST   $ENV          ;; RUNNING UNDER APT?
5477 031374 001413                BEQ   12$          ;; IF NOT: BR
5478 031376 005737 001202          11$:  TST   $MSGTYPE      ;; FINISHED LAST MESSAGE?
5479 031402 001375                BNE   11$          ;; IF NOT: WAIT
5480 031404 017637 000004 001204                MOV    24(SP),$FATAL  ;; GET ERROR #
5481 031412 062766 000002 000004                ADD    #2,4(SP)      ;; BUMP RETURN ADDR.
5482 031420 005237 001202                INC   $MSGTYPE      ;; TELL APT TO TAKE ERROR
5483 031424 105037 031450          12$:  CLRB  $FFLG      ;; CLEAR FATAL FLAG
5484 031430 105037 031447                CLRB  $LFLG      ;; CLEAR LOG FLAG
5485 031434 105037 031446                CLRB  $MFLG      ;; CLEAR MESSAGE FLAG
5486 031440 012601                MOV    (SP)+,R1      ;; POP STACK INTO R1
5487 031442 012600                MOV    (SP)+,RO      ;; POP STACK INTO RO
5488 031444 000207                RTS   PC            ;; RETURN
5489 031446          000          $MFLG: .BYTE  0      ;; MESSG. FLAG
5490 031447          000          $LFLG: .BYTE  0      ;; LOG FLAG
5491 031450          000          $FFLG: .BYTE  0      ;; FATAL FLAG
5492          031452                .EVEN
5493          000200          APTSIZE=200
5494          000001          APTENV=001
5495          000100          APTPOOL=100

```

F11

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 137  
DQFPDA.P11 09-FEB-77 10:32 APT COMMUNICATIONS ROUTINE

5496

000040

APTC SUP=040



# G11

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 138  
 DQFPDA.P11 09-FEB-77 10:32 BINARY TO OCTAL (ASCII) AND TYPE

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

5497  
5498  
5499  
5500  
5501  
5502  
5503  
5504  
5505  
5506  
5507  
5508  
5509  
5510  
5511  
5512  
5513  
5514  
5515  
5516  
5517  
5518  
5519  
5520  
5521  
5522  
5523  
5524  
5525  
5526  
5527  
5528  
5529  
5530  
5531  
5532  
5533  
5534  
5535  
5536  
5537  
5538  
5539  
5540  
5541  
5542  
5543  
5544  
5545  
5546  
5547  
5548  
5549  
5550  
5551  
5552

031452 017646 000000  
 031456 116637 000001 031675  
 031464 112637 031677  
 031470 062716 000002  
 031474 000406  
 031476 112737 000001 031675  
 031504 112737 000006 031677  
 031512 112737 000005 031674  
 031520 010346  
 031522 010446  
 031524 010546  
 031526 113704 031677  
 031532 005404  
 031534 062704 000006  
 031540 110437 031676  
 031544 113704 031675  
 031550 016605 000012  
 031554 005003  
 031556 006105  
 031560 070404  
 031562 006105  
 031564 006105  
 031566 006105  
 031570 010503  
 031572 006103  
 031574 105337 031676  
 031600 100016  
 031602 042703 177770  
 031606 001002  
 031610 005704  
 031612 001403

```

*****
*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 $TYPOC
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   ;;CALL FOR TYPEOUT
*$TYPOC---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     $TYPON
*$TYPOC: MOV     #1,SOFILL   ;;SET THE ZERO FILL SWITCH
        MOV     #6,SOMODE+1  ;;SET FOR SIX(6) DIGITS
*$TYPON: MOV     #5,SOCNT    ;;SET THE ITERATION COUNT
        MOV     R3,-(SP)     ;;SAVE R3
        MOV     R4,-(SP)     ;;SAVE R4
        MOV     R5,-(SP)     ;;SAVE R5
        MOV     SOMODE+1,R4  ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG     R4
        ADD     #6,R4        ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV     R4,SOMODE    ;;SAVE IT FOR USE
        MOV     SOFILL,R4    ;;GET THE ZERO FILL SWITCH
        MOV     12(SP),R5    ;;PICKUP THE INPUT NUMBER
        CLR     R3          ;;CLEAR THE OUTPUT WORD
1$:     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
  
```

# H11

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 139  
 DQFPDA.P11 09-FEB-77 10:32 BINARY TO OCTAL (ASCII) AND TYPE

5553	031614	005204		4\$:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
5554	031616	052703	000060		BIS	#'0,R3	:: MAKE THIS DIGIT ASCII
5555	031622	052703	000040	5\$:	BIS	#',R3	:: MAKE ASCII IF NOT ALREADY
5556	031626	110337	031672		MOVB	R3,8\$	:: SAVE FOR TYPING
5557	031632	104401	031672		TYPE	8\$	:: GO TYPE THIS DIGIT
5558	031636	105337	031674	7\$:	DECB	\$OCNT	:: COUNT BY 1
5559	031642	003347			BGT	2\$	:: BR IF MORE TO DO
5560	031644	002402			BLT	6\$	:: BR IF DONE
5561	031646	005204			INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
5562	031650	000744			BR	2\$	:: GO DO THE LAST DIGIT
5563	031652	012605		6\$:	MOV	(SP)+,R5	:: RESTORE R5
5564	031654	012604			MOV	(SP)+,R4	:: RESTORE R4
5565	031656	012603			MOV	(SP)+,R3	:: RESTORE R3
5566	031660	016666	000002 000004		MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
5567	031666	012616			MOV	(SP)+,(SP)	
5568	031670	000002			RTI		:: RETURN
5569	031672	000		8\$:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
5570	031673	000			.BYTE	0	:: TERMINATOR FOR TYPE ROUTINE
5571	031674	000		\$OCNT:	.BYTE	0	:: OCTAL DIGIT COUNTER
5572	031675	000		\$OFILL:	.BYTE	0	:: ZERO FILL SWITCH
5573	031676	000000		\$OMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE

5574  
5575  
5576  
5577  
5578  
5579  
5580  
5581  
5582  
5583  
5584  
5585  
5586  
5587  
5588  
5589  
5590  
5591  
5592  
5593  
5594  
5595  
5596  
5597  
5598  
5599  
5600  
5601  
5602  
5603  
5604  
5605  
5606  
5607  
5608  
5609  
5610

.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
        .TYPOC ;;CALL=TYPOC   TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        .TYPOS ;;CALL=TYPOS   TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        .TYPON ;;CALL=TYPON   TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
    
```

.SBTTL POWER DOWN AND UP ROUTINES

```

5611
5612
5613
5614
5615 031746 012737 032120 000024 $PWRDN: MOV $SILLUP, @PWRVEC ;; SET FOR FAST UP
5616 031754 012737 000340 000026 MOV @340, @PWRVEC+2 ;; PRIO:7
5617 031762 010046 MOV R0, -(SP) ;; PUSH R0 ON STACK
5618 031764 010146 MOV R1, -(SP) ;; PUSH R1 ON STACK
5619 031766 010246 MOV R2, -(SP) ;; PUSH R2 ON STACK
5620 031770 010346 MOV R3, -(SP) ;; PUSH R3 ON STACK
5621 031772 010446 MOV R4, -(SP) ;; PUSH R4 ON STACK
5622 031774 010546 MOV R5, -(SP) ;; PUSH R5 ON STACK
5623 031776 017746 147144 MOV @SWR, -(SP) ;; PUSH @SWR ON STACK
5624 032002 010637 032124 MOV SP, $SAVR6 ;; SAVE SP
5625 032006 012737 032020 000024 MOV $PWRUP, @PWRVEC ;; SET UP VECTOR
5626 032014 000000 HALT
5627 032016 000776 BR .-2 ;; HANG UP
5628
5629
5630
5631 032020 012737 032120 000024 $PWRUP: MOV $SILLUP, @PWRVEC ;; SET FOR FAST DOWN
5632 032026 013706 032124 MOV $SAVR6, SP ;; GET SP
5633 032032 005037 032124 CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
5634 032036 005237 032124 15: INC $SAVR6 ;; WAIT FOR THE INC
5635 032042 001375 BNE 15 OF WORD
5636 032044 011600 MOV (SP), R0 ;; GET SAVED SWR OFF STACK
5637 032046 076600 000226 MED 226 ;; RESTORE SWR CONTENTS
5638 032052 012677 147070 MOV (SP)+, @SWR ;; POP STACK INTO @SWR
5639 032056 012605 MOV (SP)+, R5 ;; POP STACK INTO R5
5640 032060 012604 MOV (SP)+, R4 ;; POP STACK INTO R4
5641 032062 012603 MOV (SP)+, R3 ;; POP STACK INTO R3
5642 032064 012602 MOV (SP)+, R2 ;; POP STACK INTO R2
5643 032066 012601 MOV (SP)+, R1 ;; POP STACK INTO R1
5644 032070 012600 MOV (SP)+, R0 ;; POP STACK INTO R0
5645 032072 012737 031746 000024 MOV $PWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
5646 032100 012737 000340 000026 MOV @340, @PWRVEC+2 ;; PRIO:7
5647 032106 104401 TYPE ;; REPORT THE POWER FAILURE
5648 032110 032126 $PWRMG: .WORD $POWER ;; POWER FAIL MESSAGE POINTER
5649 032112 012716 MOV (PC)+, (SP) ;; RESTART AT START
5650 032114 002734 $PWRAD: .WORD START ;; RESTART ADDRESS
5651 032116 000002 RTI
5652 032120 000000 $SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
5653 032122 000776 BR .-2 ;; BEFORE THE POWER DOWN WAS COMPLETE
5654 032124 000000 $SAVR6: 0 ;; PUT THE SP HERE
5655 032126 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
5656 032134 000122
5657
.EVEN

```

# K11

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 142  
 DQFPDA.P11 09-FEB-77 10:32 ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC

```

5658 .SBTTL ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC
5659
5660 ;MESSAGE PREFIX
5661 032136 047510 035124 000040 ASCHOT: .ASCIZ "HOT: "
5662 032144 040527 046522 020072 ASCWRM: .ASCIZ "WARM: "
5663 032152 000
5664
5665 ;ERROR MESSAGES HERE
5666 032153 106 046455 042117 EMA: .ASCIZ "F-MODE EXERCISER - FPS ERROR"
5667 032160 020105 054105 051105
5668 032166 044503 042523 020122
5669 032174 020055 050106 020123
5670 032202 051105 047522 000122
5671 032210 026504 047515 042504 EMB: .ASCIZ "D-MODE EXERCISER - FPS ERROR"
5672 032216 042440 042530 041522
5673 032224 051511 051105 026440
5674 032232 043040 051520 042440
5675 032240 051122 051117 000
5676 032245 106 046455 042117 EMC: .ASCIZ "F-MODE EXERCISER - RESULT ERROR"
5677 032252 020105 054105 051105
5678 032260 044503 042523 020122
5679 032266 020055 042522 052523
5680 032274 052114 042440 051122
5681 032302 051117 000
5682 032305 104 046455 042117 EMD: .ASCIZ "D-MODE EXERCISER - RESULT ERROR"
5683 032312 020105 054105 051105
5684 032320 044503 042523 020122
5685 032326 020055 042522 052523
5686 032334 052114 042440 051122
5687 032342 051117 000
5688 032345 101 042104 043050 EME: .ASCIZ "ADD(F/D) - RESULT ERROR"
5689 032352 042057 020051 020055
5690 032360 042522 052523 052114
5691 032366 042440 051122 051117
5692 032374 000
5693 032375 123 041125 043050 EMF: .ASCIZ "SUB(F/D) - RESULT ERROR"
5694 032402 042057 020051 020055
5695 032410 042522 052523 052114
5696 032416 042440 051122 051117
5697 032424 000
5698 032425 115 046125 043050 EMG: .ASCIZ "MUL(F/D) - RESULT ERROR"
5699 032432 042057 020051 020055
5700 032440 042522 052523 052114
5701 032446 042440 051122 051117
5702 032454 000
5703 032455 104 053111 043050 EMH: .ASCIZ "DIV(F/D) - RESULT ERROR"
5704 032462 042057 020051 020055
5705 032470 042522 052523 052114
5706 032476 042440 051122 051117
5707 032504 000
5708 032505 125 042516 050130 EMI: .ASCIZ "UNEXPECTED FLOATING POINT TRAP, IGNORED AND CONTINUING"
5709 032512 041505 042524 020104
5710 032520 046106 040517 044524
5711 032526 043516 050040 044517
5712 032534 052116 052040 040522
5713 032542 026120 044440 047107
  
```

L11

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11  
DQFPDA.P11 09-FEB-77 10:32

27(1006) 09-FEB-77 10:34 PAGE 143  
ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC

5714	032550	051117	042105	040440	
5715	032556	042116	041440	047117	
5716	032564	044524	052516	047111	
5717	032572	000107			
5718	032574	042101	043104	026440	EMJ: .ASCIZ "ADDF - FPS ERROR"
5719	032602	043040	051520	042440	
5720	032610	051122	051117	000	
5721	032615	123	041125	020106	EMK: .ASCIZ "SUBF - FPS ERROR"
5722	032622	020055	050106	020123	
5723	032630	051105	047522	000122	
5724	032636	052515	043114	026440	EML: .ASCIZ "MULF - FPS ERROR"
5725	032644	043040	051520	042440	
5726	032652	051122	051117	000	
5727	032657	104	053111	020106	EMM: .ASCIZ "DIVF - FPS ERROR"
5728	032664	020055	050106	020123	
5729	032672	051105	047522	000122	
5730	032700	042101	042104	026440	EMN: .ASCIZ "ADDD - FPS ERROR"
5731	032706	043040	051520	042440	
5732	032714	051122	051117	000	
5733	032721	123	041125	020104	EMO: .ASCIZ "SUBD - FPS ERROR"
5734	032726	020055	050106	020123	
5735	032734	051105	047522	000122	
5736	032742	052515	042114	026440	EMP: .ASCIZ "MULD - FPS ERROR"
5737	032750	043040	051520	042440	
5738	032756	051122	051117	000	
5739	032763	104	053111	020104	EMQ: .ASCIZ "DIVD - FPS ERROR"
5740	032770	020055	050106	020123	
5741	032776	051105	047522	000122	
5742	033004	042101	043104	026440	EMR: .ASCIZ "ADDF - FEC/FEA ERROR"
5743	033012	043040	041505	043057	
5744	033020	040505	042440	051122	
5745	033026	051117	000		
5746	033031	123	041125	020106	EMS: .ASCIZ "SUBF - FEC/FEA ERROR"
5747	033036	020055	042506	027503	
5748	033044	042506	020101	051105	
5749	033052	047522	000122		
5750	033056	052515	043114	026440	EMT: .ASCIZ "MULF - FEC/FEA ERROR"
5751	033064	043040	041505	043057	
5752	033072	040505	042440	051122	
5753	033100	051117	000		
5754	033103	104	053111	020106	EMU: .ASCIZ "DIVF - FEC/FEA ERROR"
5755	033110	020055	042506	027503	
5756	033116	042506	020101	051105	
5757	033124	047522	000122		
5758	033130	042101	042104	026440	EMV: .ASCIZ "ADDD - FEC/FEA ERROR"
5759	033136	043040	041505	043057	
5760	033144	040505	042440	051122	
5761	033152	051117	000		
5762	033155	123	041125	020104	EMW: .ASCIZ "SUBD - FEC/FEA ERROR"
5763	033162	020055	042506	027503	
5764	033170	042506	020101	051105	
5765	033176	047522	000122		
5766	033202	052515	042114	026440	EMX: .ASCIZ "MULD - FEC/FEA ERROR"
5767	033210	043040	041505	043057	
5768	033216	040505	042440	051122	
5769	033224	051117	000		

M11

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 144  
 DQFPDA.P11 09-FEB-77 10:32 ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC

5770	033227	104	053111	020104	EMY:	.ASCIZ	"DIVD - FEC/FEA ERROR"
5771	033234	020055	042506	027503			
5772	033242	042506	020101	051105			
5773	033250	047522	000122				
5774							
5775							
5776							
5777	033254	054105	023520	004504	DHA:	.ASCIZ	"EXP'D RCV'D"
5778	033262	041522	023526	000104			
5779	033270	026455	042455	050130	DHB:	.ASCIZ	"---EXPECTED--- ---RECEIVED---
5780	033276	041505	042524	026504			
5781	033304	026455	026411	026455			
5782	033312	042522	042503	053111			
5783	033320	042105	026455	000055			
5784	033326	026455	026455	026455	DHC:	.ASCIZ	"-----EXPECTED----- RECEIVED-----"
5785	033334	026455	026455	042455			
5786	033342	050130	041505	042524			
5787	033350	026504	026455	026455			
5788	033356	026455	026455	026455			
5789	033364	026411	026455	026455			
5790	033372	026455	026455	026455			
5791	033400	042522	042503	053111			
5792	033406	042105	026455	026455			
5793	033414	026455	026455	026455			
5794	033422	000055					
5795	033424	046117	020104	041520	DHD:	.ASCIZ	"OLD PC OLD PS FPS FEC FEA \$FPS \$FEC \$FEA"
5796	033432	047411	042114	050040			
5797	033440	004523	043040	051520			
5798	033446	020011	042506	004503			
5799	033454	043040	040505	020011			
5800	033462	043044	051520	020011			
5801	033470	043044	041505	020011			
5802	033476	043044	040505	000			
5803	033503	105	050130	042047	DHE:	.ASCIZ	"EXP'D-FEC-RCV'D EXP'D-FEA-RCV'D"
5804	033510	043055	041505	051055			
5805	033516	053103	042047	042411			
5806	033524	050130	042047	043055			
5807	033532	040505	051055	053103			
5808	033540	042047	000				
5809							
5810							
5811							
5812		033544					
5813	033544	002400	002362	000000	DTA:	.WORD	\$FPS,FPS,0
5814	033552	002402	002364	002404	DTB:	.WORD	\$FEC,FEC,\$FEA,FEA,0
5815	033560	002366	000000				
5816	033564	002416	002420	002406	DTD:	.WORD	ANS2,ANS2+2,ANS1,ANS1+2,0
5817	033572	002410	000000				
5818	033576	002416	002420	002422	DTE:	.WORD	ANS2,ANS2+2,ANS2+4,ANS2+6
5819	033604	002424					
5820	033606	002406	002410	002412			
5821	033614	002414	000000				
5822	033620	002370	002372	002362	DTF:	.WORD	FPPOPC,FPPOPS,FPS,FEC,FEA,\$FPS,\$FEC,\$FEA,0
5823	033626	002364	002366	002400			
5824	033634	002402	002404	000000			
5825							

# N11

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 145  
 DQFPDA.P11 09-FEB-77 10:32 ERROR MESSAGES, DATA HEADERS, DATA VECTORS, OPERAND VECTORS, ETC

```

5826                                     ; OPERAND VECTORS HERE
5827                                     .EVEN
5828 033642 034102 002426 002430 LOD:   .WORD  XLO,LONUM,LONUM+2,LONUM+4,LONUM+6,0
5829 033650 002432 002434 000000
5830 033656 034112 002436 002440 HID:   .WORD  XHI,HINUM,HINUM+2,HINUM+4,HINUM+6,0
5831 033664 002442 002444 000000
5832 033672 034102 002426 002430 LOF:   .WORD  XLO,LONUM,LONUM+2,0
5833 033700 000000
5834 033702 034112 002436 002440 HIF:   .WORD  XHI,HINUM,HINUM+2,0
5835 033710 000000
5836 033712 034122 002446 002450 OP1F: .WORD  XOP1,OP1,OP1+2,0
5837 033720 000000
5838 033722 034130 002456 002460 OP2F: .WORD  XOP2,OP2,OP2+2,0
5839 033730 000000
5840 033732 034136 002466 002470 OP3F: .WORD  XOP3,OP3,OP3+2,0
5841 033740 000000
5842 033742 034144 002476 002500 OP4F: .WORD  XOP4,OP4,OP4+2,0
5843 033750 000000
5844 033752 034160 002506 002510 OP5F: .WORD  XOP5,OP5,OP5+2,0
5845 033760 000000
5846 033762 034152 002516 002520 OP6F: .WORD  XOP6,OP6,OP6+2,0
5847 033770 000000
5848 033772 034122 002446 002450 OP1D: .WORD  XOP1,OP1,OP1+2,OP1+4,OP1+6,0
5849 034000 002452 002454 000000
5850 034006 034130 002456 002460 OP2D: .WORD  XOP2,OP2,OP2+2,OP2+4,OP2+6,0
5851 034014 002462 002464 000000
5852 034022 034136 002466 002470 OP3D: .WORD  XOP3,OP3,OP3+2,OP3+4,OP3+6,0
5853 034030 002472 002474 000000
5854 034036 034144 002476 002500 OP4D: .WORD  XOP4,OP4,OP4+2,OP4+4,OP4+6,0
5855 034044 002502 002504 000000
5856 034052 034160 002506 002510 OP5D: .WORD  XOP5,OP5,OP5+2,OP5+4,OP5+6,0
5857 034060 002512 002514 000000
5858 034066 034152 002516 002520 OP6D: .WORD  XOP6,OP6,OP6+2,OP6+4,OP6+6,0
5859 034074 002522 002524 000000
5860
5861
5862                                     ; OPERAND TITLES
5863 034102 047514 052516 035115 XLO:   .ASCIZ  "LONUM:"<11>
5864 034110 000011
5865 034112 044510 052516 035115 XHI:   .ASCIZ  "HINUM:"<11>
5866 034120 000011
5867 034122 050117 035061 000011 XOP1:  .ASCIZ  "OP1:"<11>
5868 034130 050117 035062 000011 XOP2:  .ASCIZ  "OP2:"<11>
5869 034136 050117 035063 000011 XOP3:  .ASCIZ  "OP3:"<11>
5870 034144 050117 035064 000011 XOP4:  .ASCIZ  "OP4:"<11>
5871 034152 050117 035066 000011 XOP6:  .ASCIZ  "OP6:"<11>
5872 034160 050117 035065 000011 XOP5:  .ASCIZ  "OP5:"<11>
5873                                     ; THE END
5874 000001                                     .END
  
```



ABASE =	000000	993			
ACDM1 =	000000	993			
ACDM2 =	000000	993			
ACPUOP =	000000	993	1008		
ADDC0	002526	1197#	1305	4030	4329#
ADDC1	002530	1198#	4030	4373#	
ADDC2	002532	1199#	4030	4352#	
ADDC3	002534	1200#	4030	4375#	
ADDC4	002536	1201#	4030	4418#	
ADDC5	002540	1202#	4030	4479#	4499#
ADDC6	002542	1203#	4030	4482#	4502#
ADDC7	002544	1204#	4030	4513#	4558#
ADDC8	002546	1205#	4030	4559#	
ADDW0 =	000000	993			
ADDW1 =	000000	993			
ADDW10 =	000000	993			
ADDW11 =	000000	993			
ADDW12 =	000000	993			
ADDW13 =	000000	993			
ADDW14 =	000000	993			
ADDW15 =	000000	993			
ADDW2 =	000000	993			
ADDW3 =	000000	993			
ADDW4 =	000000	993			
ADDW5 =	000000	993			
ADDW6 =	000000	993			
ADDW7 =	000000	993			
ADDW8 =	000000	993			
ADDW9 =	000000	993			
ADEVCT =	000000	993	999		
ADEVH =	000000	993			
AEND1	004036	1439	1446	1450#	
AEND10	006412	1864	1873	1880	1884#
AEND11	006676	1937	1944	1948#	
AEND12	007172	2001	2010	2014#	
AEND13	007420	2055	2062	2066#	
AEND14	007656	2107	2116	2120#	
AEND15	010164	2173	2180	2185	2189#
AEND16	010512	2242	2251	2258	2262#
AEND17	010762	2303	2310	2315	2319#
AEND2	004332	1503	1512	1516#	
AEND20	011252	2360	2369	2376	2380#
AEND3	004560	1557	1564	1568#	
AEND4	005016	1609	1618	1622#	
AEND5	005324	1675	1682	1687	1691#
AEND6	005652	1745	1754	1761	1765#
AEND7	006122	1807	1814	1819	1823#
AENV =	000000	993	1004		
AENVH =	000000	993	1005		
AERR1	003736	1421	1425#		
AERR11	006576	1919	1923#		
AERR12	007062	1983	1987#		
AERR15	010042	2155	2159#		
AERR16	010350	2224	2228#		
AERR2	004222	1485	1489#		
AERR5	005202	1657	1661#		



APTCSU=	000040	5391	5496#			
APTEMV=	000001	5252	5384	5452	5494#	
APTSIZ=	000200	1295	5493#			
APTSPO=	000100	5386	5454	5495#		
ARET1	003712	1392	1418#			
ARET10	006262	1829	1855#			
ARET11	006552	1890	1916#			
ARET12	007036	1954	1980#			
ARET13	007332	2020	2046#			
ARET14	007560	2072	2098#			
ARET15	010016	2126	2152#			
ARET16	010324	2195	2221#			
ARET17	010652	2268	2294#			
ARET2	004176	1456	1482#			
ARET20	011122	2325	2351#			
ARET3	004472	1522	1548#			
ARET4	004720	1574	1600#			
ARET5	005156	1628	1654#			
ARET6	005464	1698	1724#			
ARET7	006012	1772	1798#			
ASCHOT	032136	1384	5661#			
ASCHRM	032144	1386	5662#			
ASWREG=	000000	993	1006			
ATESTN=	000000	993	997			
ATST1	003774	1426	1434	1437#		
ATST10	006306	1858	1862#			
ATST11	006634	1924	1932	1935#		
ATST12	007120	1988	1996	1999#		
ATST13	007356	2049	2053#			
ATST14	007604	2101	2105#			
ATST15	010100	2160	2168	2171#		
ATST16	010406	2229	2237	2240#		
ATST17	010676	2297	2301#			
ATST2	004260	1490	1498	1501#		
ATST20	011146	2354	2358#			
ATST3	004516	1551	1555#			
ATST4	004744	1603	1607#			
ATST5	005240	1662	1670	1673#		
ATST6	005546	1732	1740	1743#		
ATST7	006036	1801	1805#			
AUNIT =	000000	993	1000			
AUSWR =	000000	993	1007			
AVECT1=	000000	993				
AVECT2=	000000	993				
BGNMES	002622	1233#	1310			
BIT0 =	000001	862#				
BIT00 =	000001	852#	862			
BIT01 =	000002	851#	861			
BIT02 =	000004	850#	860			
BIT03 =	000010	849#	859			
BIT04 =	000020	848#	858	1316	1366	3930
BIT05 =	000040	847#	857			
BIT06 =	000100	846#	856			
BIT07 =	000200	845#	855			
BIT08 =	000400	844#	854	5179		
BIT09 =	001000	843#	853	5187	5262	

































F13

FPU ADD/SUB/MUL/DIV RANDOM EXER MACY11 27(1006) 09-FEB-77 10:34 PAGE 165  
DQFPDA.P11 09-FEB-77 10:32 CROSS REFERENCE TABLE -- MACRO NAMES

DSKZ:DQFPDA, DSKZ:DQFPDA.SEQ/SOL/CRF=DQFPDA.MEM, DQFPDA.MAC, DQFPDA.P11  
RUN-TIME: 20 18 1 SECONDS  
RUN-TIME RATIO: 126/40=3.1  
CORE USED: 32K (64 PAGES)