

KDF11-B

11/23B CPU CLSTR DIAG AH-T039A-MC
CJKDJA0 FICHE 1 OF 1

APR 1982
COPYRIGHT © 1982
MADE IN USA



Table with multiple columns and rows of data, likely representing CPU cluster diagnostic information. The data is too faint to transcribe accurately but appears to be organized in a grid format.

.REM_

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

IDENTIFICATION

PRODUCT CODE: AC-T038A-MC

PRODUCT NAME: CJKDJA0 11/23B CPU CLSTR DIAG

PRODUCT DATE: SEPT-81

MAINTAINER: DIAGNOSTIC ENGINEERING

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

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C): 1982 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DEC/X11

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
105
106
107
108
109

HISTORY

REVISION A FIRST RELEASE OF DIAGNOSTIC

TABLE OF CONTENTS

1.0 GENERAL PROGRAM INFORMATION

1.1 ABSTRACT

1.2 SYSTEM REQUIREMENTS

1.3 RELATED DOCUMENTS AND STANDARDS

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

1.5 ASSUMPTIONS

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURE

2.2 PROGRAM OPTIONS

2.3 EXECUTION TIMES

3.0 ERROR INFORMATION

3.1 ERROR REPORTING PROCEDURES

3.2 ERROR HALTS

4.0 PERFORMANCE AND PROGRESS REPORTS

4.1 PERFORMANCE REPORTS

4.2 PROGRESS REPORTS

5.0 DEVICE INFORMATION TABLES

6.0 PROGRAM DESCRIPTION

6.1 PROGRAM EXECUTION CHARACTERISTICS

6.2 SUBTEST SUMMARIES

6.3 SPECIAL SUBROUTINE DESCRIPTION

7.0 LISTING

1.0 ABSTRACT

THIS PROGRAM IS A GO-NOGO TEST FOR THE 11/23-B CPU BOARD. IT TESTS THE CPU INCLUDING EIS, THE MMU, THE FPP, THE LTC AND BOTH SLU'S. IT DOES NOT CONTAIN THE CAPABILITIES OF SCOPE LOOPING, ERROR RECOVERY OR PRINTING OF ERROR INFORMATION. ERROR HALTS DO INDICATE WHICH DEVICE FAILED TO ALLOW THE TECHNICIAN TO DETERMINE WHICH DIAGNOSTIC TO USE TO FIX THE BOARD OR WHAT FIELD REPLACEABLE UNIT (FRU) MAY FIX THE BOARD. THE PROGRAM WILL RUN UNDER THE ACT AND APT MANUFACTURING SYSTEMS AND IS CHAINABLE UNDER XXDP.

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
159
160
161
162
163
164
165

1.1 SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENTS

- KDF11-B CPU MODULE
- 32K OF MEMORY
- THE SECOND SLU MUST HAVE TURN AROUND CONNECTOR.

B. SOFTWARE ENVIRONMENTS

- APT (MULTI-CPU TESTER)
- ACT
- XXDP (SLIDE)
- STAND-ALONE

1.2 RELATED DOCUMENTS AND STANDARDS

- ASSEMBLED WITH SYSMAC; SEE FIRST PAGE OF LISTING FOR REVISION NUMBER.
- MXXXX MODULE SPECIFICATION
- DIAGNOSTIC ENGINEERING FUNCTIONAL SPECIFICATION FOR SPECIAL MANUFACTURING TEST BGI-79-003-00-U.
- DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS 175-003-009-02.

1.3 PREREQUISITE DIAGNOSTICS

NONE

1.4 ASSUMPTIONS

THIS PROGRAM ASSUMES THE MACHINE IS UP SUFFICIENTLY TO ALLOW PROPER OPERATION OF THE MICRO-ODT OF THE DCF11-AA CHIP SET.

THE SYSTEM MUST HAVE PARITY MEMORY LOCATED IN THE FIRST 32K BLOCK.

THE SOFTWARE ASSUMES THAT THERE IS NO MEMORY OR DEVICES

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
215
216
217
218
219
220
221

LOCATED AT OR BEYOND ADDRESS BIT 17 (64 KW). IF MEMORY IS THERE THE PROGRAM WILL FAIL WHEN IN THE EXTENDED ADDRESS TESTS. IF BIT 7 IS SET IN THE SWITCH REGISTER (176) THIS FAILURE CAN BE PREVENTED SINCE THAT PARTICULAR TEST WILL BE BYPASSED.

SEE PARAGRAPH 2.2.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

TO LOAD AND START THIS PROGRAM USE THE STANDARD PROCEDURES FOR THE DIAGNOSTIC SOFTWARE ENVIRONMENT THAT IS BEING USED.

2.2 PROGRAM OPTIONS

THIS PROGRAM USES THE SOFTWARE SWITCH LOCATION 176 IF PROGRAM IS NOT BEING RUN UNDER APT MODE (BIT 0 SET OF LOCATION \$ENV). IF PROGRAM IS BEING RUN IN APT MODE THE LOCATION \$SWREG IN THE APT ETABLE IS USED TO STORE OPERATING SWITCHES.

WARNING****THIS PROGRAM IS SET TO DO MINIMUM TESTING UNLESS CORRECTIVE ACTION IS TAKEN VIA THE SOFTWARE SWITCH REGISTER (176). BITS 1, 6,7-10 HAVE BEEN SET UP SUCH THAT THE PROGRAM WILL BYPASS CERTAIN TESTS UNLESS THE SWITCH REGISTER BIT IS SET. THIS CONDITION ALSO APPLIES WHEN UNDER CONTROL OF APT. THE APT SWITCH REGISTER, LOCATION 1022, MUST BE CORRECTLY SET AT APT LOAD TIME.

BIT #	DEFINITION
15-11	NOT USED
10	1 - TEST E102 SWITCHES 0 - INHIBIT TESTING E102 SWITCHES
9	1 - TEST PARITY ERROR DETECTION 0 - INHIBIT TESTING PARITY ERROR DETECTION
8	1 - USE THE Q22BE 0 - USE THE QBE IN PLACE OF THE Q22BE

- 222 7 1 - TEST THE UPPER 5 ADDRESS BITS FOR TIME OUT
- 223 0 - INHIBIT TESTING THE UPPER 5 ADRS BITS
- 224
- 225 6 1 - TEST USING A Q BUS EXERCISER (QBE OR Q22BE)
- 226 0 - INHIBIT TESTS THAT USE A Q BUS EXERCISFR
- 227
- 228 5 0 - PROGRAM RESERVED -- PROGRAM WILL CLEAR IF CIS CHIP SET NOT ON BOARD
- 229 1 - PROGRAM RESERVED -- PROGRAM WILL SET IF CIS CHIP SET IS ON BOARD
- 230 4 0 - TEST SLU2 OF 11/23-B
- 231 1 - INHIBIT TESTING OF SLU2
- 232
- 233 3 0 - TEST LTC OF 11/23-B
- 234 1 - INHIBIT TESTING OF LTC
- 235
- 236 2 0 - TEST SLU1 OF 11/23-B
- 237 1 - INHIBIT TESTING OF SLU1
- 238
- 239 1 1 - TEST FPP INSTRUCTION SET
- 240 0 - INHIBIT TESTING OF FPP
- 241
- 242 0 0 - TEST MEMORY MANAGEMENT UNIT
- 243 1 - INHIBIT TESTING OF MMEMORY MANAGEMENT UNIT
- 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
- 271
- 272
- 273
- 274
- 275
- 276
- 277

2.3 EXECUTION TIMES

FIRST PASS RUNTIME (WORST CASE).....45 SFC
LONGEST TEST TIME.....30 SEC
ADDITIONAL RUNTIME (EXTRA UNITS).....NONE
LONGEST PASS TIME.....45 SEC

3.0 ERROR INFORMATION

3.1 ERROR REPORTING PROCEDURES

THE PROGRAM DOES NOT TYPE OUT ANY ERROR REPORTS OF ITS OWN BUT TAKES ADVANTAGE OF THE HARDWARE FEATURE THAT TYPES THE PC WHEN A HALT OCCURS. WHEN AN ERROR IS DETECTED THE PROGRAM JUMPS TO ONE OF SEVEN HALT ROUTINES. THE ROUTINES SIMPLY MOVE A FATAL ERROR NUMBER INTO LOCATION \$FATAL, SET THE FATAL ERROR FLAG IN LOCATION \$MSGTY AND EITHER HALT OR IF ON APT DO A BRANCH DOT. THE OPERATOR HAS THREE WAYS TO DETERMINE THE FAILING DEVICE; 1) BY EXAMINING LOCATION \$FATAL, 2) BY DETERMINING THE HALT ADDRESS AND LOOKING UP THE ADDRESS IN THE LISTING AND 3) BY EXAMINING LOCATION \$TESTN WHICH WILL CONTAIN THE TEST NUMBER BEING EXECUTED.

3.2 ERROR HALTS

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
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333

FOR DISCUSSION SEE SECTION 3.1. THE LABELS FOR THE HALTS AND THE DEVICE THEY INDICATE HAVING FAILED ARE:

CPUHLT: CPU
MMUHLT: MMU
FPPHLT: FPP
LTCHLT: LTC
SL1HLT: SLU1
SL2HLT: SLU2
EXADHT: EXT ADRS TEST
COMHLT: SYSTEM INTERACTION
Q22HLT: Q22BE INTERRUPT TEST
BDVHLT: BDV TEST

UPON RECEIVING THE ERROR HALT ADDRESS (PC) FROM THE MICRO-ODT THE OPERATOR CAN LOOK UP THESE TAGS IN THE SYMBOL TABLE AT THE END OF THE LISTING TO DETERMINE WHICH HALT WAS EXECUTED. NOTE: THE PC SUPPLIED BY THE MICRO-ODT WILL BE THE HALT ADDRESS PLUS 2.

THE OPERATOR CAN DETERMINE WHICH TEST THE ERROR OCCURRED IN BY EXAMINING LOCATION '\$TESTN'. THE TEST NUMBERS EQUATE TO FAILING DEVICES AS FOLLOWS:

DEVICE	CONTENTS OF \$TESTN
CPU	000001
MMU	000002
FPP	000003
SLU1	000004
LTC	000005
SLU2	000006
EXTND ADRS	000007
TESTS	
SYSTEM	
INTERACTION	000010
Q22BE	000011
BDV TESTS	000012

4.0 PERFORMANCE AND PROGRESS REPORTS

THE ONLY REPORT TYPED BY THIS PROGRAM IS THE END PASS MESSAGE WHICH IS:

CJKDJA END OF PASS #XXX

WHERE XXX IS THE DECIMAL NUMBER OF PASSES COMPLETED.

5.0 DEVICE INFORMATION TABLES

SLU1 RCSR

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

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
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389

```

      I           I I I           I
      -----
      I I
RECEIVER DONE -----
RECEIVER INTERRUPT ENABLE -----
  
```

SLU1 RBUF

```

      15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
      -----
      I I I I I           I I I I I I I I I
      -----
      I I I I           I
ERROR- I I I I           I
OVERRUN --- I I           I
FRAME ERROR --- I           I
RECEIVE PARITY ----           I
ERROR
RECEIVED DATA BITS (8) -----
  
```

SLU1 XCSR

```

      15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
      -----
      I           I I I           I
      -----
      I I           I
TRANSMITTER READY -----
TRANSMITTER INTERRUPT ENABLE -----
  
```

SLU1 XBUF

```

      15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
      -----
      I           I I I I I I I I I I
      -----
      I
TRANSMITTER DATA BITS (8) -----
  
```

LTC CSR

```

      15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
      -----
      I           I I           I
      -----
  
```


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
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445

LINE CLOCK
 INTERRUPT ENABLE ----- I

SLU2 RCSR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	I							I	I	I						I

								I		I						
								I		I						
RECEIVER DONE	-----															
INTERRUPT ENABLE	-----															

SLU2 RBUF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	I	I	I	I	I				I	I	I	I	I	I	I	I

		I	I	I	I								I			
		I	I	I	I								I			
ERROR -		I	I	I	I								I			
OVERRUN ---			I	I	I								I			
FRAME ERROR ---				I	I								I			
RECEIVER PARITY ---					I								I			
ERROR													I			
RECEIVER DATA BITS (8)	-----															

SLU2 XCSR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

								I	I	I	I					I

									I	I						I
									I	I						I
TRANSMITTER DONE	-----															
INTERRUPT ENABLE	-----															
BREAK	-----															

SLU2 XBUF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	I								I	I	I	I	I	I	I	I

												I				

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

TRANSMITTER DATA BITS (8) -----

6.0 PROGRAM DESCRIPTION

6.1 PROGRAM EXECUTION CHARACTERISTICS

THIS PROGRAM RUNS THE SAME UNDER ALL DIAGNOSTIC MONITORS. WHEN THE TEST IS STARTED AT ADDRESS 200 OCTAL THE TESTING IS DONE AND ON COMPLETION THE TITLE IS TYPED AS PART OF THE END OF PASS MESSAGE.

6.2 SUB-TEST SUMMARIES

6.2.1 CENTRAL PROCESSING UNIT SUBTEST -

THESE TESTS CHECK THE BASIC INSTRUCTION SET AND ADDRESSING MODES, THE EXTENDED ELEVEN INSTRUCTION SET (EIS) AND TRAPS TESTING. IT IS EQUIVALENT TO CJKDB.

474
475
476
477
478
479
480
481
482
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

6.2.2 MEMORY MANAGEMENT UNIT SUBTEST -

THESE TESTS ARE THE SAME AS IN CJKDA, THE KEF11-AA TEST. THE PROGRAM BEGINS BY TESTING SOME OF THE INTERNAL CPU DATA AND ADDRESS PATHS AND ADDRESS DETECTION LOGIC. NEXT THE MEMORY MANAGEMENT REGISTERS ARE CHECKED FOR DATA RELIABILITY, THEN RELOCATION CAPABILITIES (FORMATION OF A PHYSICAL ADDRESS FROM A VIRTUAL ADDRESS AND ASSOCIATED PAGE DESCRIPTOR (PDR) INFORMATION). FINALLY THE ABORT AND STATUS SEGMENTS OF THE LOGIC ARE CHECKED.

6.2.3 EXTENDED ADDRESS BIT TESTING AND PARITY ERROR LOGIC TEST

6.2.4 Q22BE BR LEVEL TESTING

6.2.5 BDV TESTS

6.2.6 FLOATING POINT PROCESSOR SUBTEST -

THE FLOATING POINT PROCESSOR SUBTEST CHECKS FLOATING POINT REGISTERS FIRST USING A LIMITED NUMBER OF FLOATING POINT INSTRUCTIONS. IT THEN VERIFIES THE REST OF THE FLOATING POINT INSTRUCTION SET USING A NUMBER OF DATA PATTERNS FOR EACH INSTRUCTION.

6.2.7 SERIAL LINE UNIT (SLU1) SUBTEST -

THESE TESTS CHECK THE SLU'S REGISTERS FOR ADDRESSING AND DATA HANDLING.

6.2.8 LINE TIME CLOCK (LTC) SUBTEST -

FIRST THE REGISTER IS CHECKED FOR ADDRESSING AND BIT SETTING CAPABILITIES THEN THE INTERRUPT LOGIC IS CHECKED. THERE IS ALSO A

6.2.9 SERIAL LINE UNIT 2 SUBTEST -

THE TESTING DONE HERE IS SIMILIAR AS FOR THE SLU1. AN EXTERNAL JUMPER, TURN AROUND CONNECTOR, MUST BE PRESENT.

530
531
532
533
534
535
536
537
538
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

6.2.10 BLAST SUBTEST

THIS TEST CHECKS THE ABILITY OF THE 11/23-B TO HANDLE SYSTEM INTER-ACTION. THE CPU HAS TO HANDLE DEVICES AT DIFFERENT PRIORITY LEVELS AND ARBITRATE BETWEEN THEM AND ITS OWN PRIORITY. THE TEST SETS UP ALL DEVICES TO INTERRUPT THEN ENABLES THEM ALL AT ONCE. THE SLU'S TRANSFER DATA UNTIL THEY TRANSFER 400(8) BYTES OR UNTIL ONE SECOND (60 TICKS) OF THE LINE CLOCK HAS BEEN RECEIVED. THE PROGRAM THEN VERIFIES THE NUMBER OF TRANSMITTER INTERRUPTS IS EQUAL TO THE NUMBER OF RECEIVER INTERRUPTS. FINALLY THE DATA TRANSFERRED BY EACH DEVICE IS CHECKED.

6.3 SPECIAL SUBROUTINE DESCRIPTIONS

THE ONLY SPECIAL SUBROUTINES ARE THE ERROR ROUTINES EACH SUBTEST HAS IS OWN. THE ROUTINES SIMPLY SET THE FATAL ERROR FLAG IN THE APT MAILBOX AND EITHER 'BRANCH SELF' OR 'HALT'. THIS CHOICE IS DETERMINED IN THE INITIALIZE PORTION OF THE PROGRAM AND IS A 'BRANCH SELF' IF RUNNING UNDER APT OR A 'HALT' IF RUNNING UNDER ANY OTHER MONITOR.

```

.TITLE CJKDJA0 11/23-B CPU CLUSTER DIAG.
.ENABLE ABS
.NLIST CND,MC,MD
.LIST ME
SCOPE=NOP
R7=%7
R6=%6
PS=177776
TPS=177564
TPB=177566
USRM=140000
PUSRM=30000
SP=%6
R6=%6
TAB=%3
LAST=%1
FIFST=%5
R2=%2
HLT=HALT
TRT=3
ITRAP5=4
RTRAP5=4
RTRAP4=14
RTRAP3=30
RTRAP2=20
RTRAP1=34
TTCSR=177564
TRCSR=177560
TKB=177562
TPB=177566
;RESERVED INST AND ILLEGAL ADDRESSES
;FOR TRACE TRAP
;FOR EMULATOR TRAP
;FOR IOT TRAP
;FOR TRAP INST
```

586 000240
587 000240
588 177776
589 000077
590 000010
591 004700
592 000100
593
594 177520
595 177524
596 177522
597
598 177776
599 001000
600 000600
601 104377
602 104777
603 000001
604
605
606
607
608 000400
609 000046
610 000046 133212
611 000052 000052
612 000052 000000
613 000400 000400
614 001000 001000
615
616
617
618
619 001000
620 001000 000000
621 001002 000000
622 001004 000000
623 001006 000000
624 001010 000000
625 001012 000000
626 001014 000000
627 001016 000000
628 001020
629 001020 000
630 001021 000
631 001022 000000
632 001024 000000
633 001026 000000
634
635
636
637
638
639
640 001030
641

BELL=240
NOP=240
STATUS=177776
TRAPA=77
RTRAP=10
ILLA=004700
ILLB=100

PCR=177520
LSREG=177524
RWREG=177522

CC=177776
KERSTK=STBOT
USESTK=STBOT-200
EMTA=104377
TRAPC=104777
APTENV=1
.SBTTL ACT11 HOOKS

:*****
:HOOKS REQUIRED BY ACT11
\$SVPC= ;SAVE PC
.=46
\$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP
.=52
.WORD 0 ;:2)SET LOC.52 TO ZERO
.\$SVPC ;: RESTORE PC
.=1000
.SBTTL APT MAILBOX-ETABLE

:*****
.EVEN
\$MAIL: ;:APT MAILBOX
\$MSGTY: .WORD AMSGTY ;:MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ;:FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ;:TEST NUMBER
\$PASS: .WORD APASS ;:PASS COUNT
\$DEVCT: .WORD ADEVCT ;:DEVICE COUNT
\$UNIT: .WORD AUNIT ;:I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ;:MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG ;:MESSAGE LENGTH
\$ETABLE: ;:APT ENVIRONMENT TABLE
\$ENV: .BYTE AENV ;:ENVIRONMENT BYTE
\$ENVM: .BYTE AENVM ;:ENVIRONMENT MODE BITS
\$SWREG: .WORD ASWREG ;:APT SWITCH REGISTER
\$USWR: .WORD AUSWR ;:USER SWITCHES
\$CPUOP: .WORD ACPUOP ;: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
\$ETEND:
.MEXIT

642
643
644
645
646
647 001030
648 000024
649 000024 000200
650 000044 000044
651 000044 001030
652 001030
653
654
655
656
657 001030
658 001030 000000
659 001032 001000
660 001034 000010
661 001036 000025
662 001040 000000
663 001042 000014
664
665
666
667 000004
668 000004 021336
669 000006 000000
670 000010 021340
671 000012 000000
672 000014 021342
673 000016 000000
674 000020 021344
675 000022 000000
676 000030
677 000030 021346
678 000032 000000
679 000034 021350
680 000036 000000
681 000114
682 000114 021352
683 000116 000000
684 000244
685 000244 021354
686 000246 000000
687 000250 021356
688 000252 000000
689
690 000172
691 000172 000000
692 000174 000000
693 000176 000000
694
695
696
697

```
.SBTTL APT PARAMETER BLOCK

:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
      .SX=      ;;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
      =.SX     ;;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.

$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 10     ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 25     ;;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)
:*****
:SOME POINTERS TO CPU TRAP HANDLERS
:*****
      =4
      T04
      0
      T010
      0
      T014
      0
      T020
      0
      =30
      T030
      0
      T034
      0
      =114
      T0114
      0
      =244
      T0244
      0
      T0250
      0
      =172
MTFLAG: 0      ;;MULTI-TESTER ACTIVE BIT
DISPREG: 0     ;;SOFTWARE DISPLAY REGISTER
SWREG: 0      ;;SOFTWARE SWITCH REGISTER
:*****
:DATA TABLE FOR USE IN ADDRESSING MODE TESTS
:*****
```

```

698          000370          .=370
699 000370 000000 000000 000000      0,0,0,0,0,0
700 000376 000000 000000 000000
701 000404 000001 000001 177777      1,1,-1
702          :*****
703          ;SET UP STARTING ADDRESS
704          .=1000
705 001000 000000 STBOT: .WORD 0 ;STACK POINTER
706          :* *****
707
708
709          000510          .=510          ;Q22BE DEVICE VECTOR
710          ;AREA
711
712          000200          .=200
713 000200 000167 000774      JMP      START
714 000204 012706 001000      MOV      #STBOT,R6          ;SET STACK POINTER
715 000210 012702 001004      MOV      #STESTN,R2        ;SET MAILBOX POINTER
716 000214 000137          JMP      @PC+              ;JUMP TO SUBTEST
717 000216 000000          0          ;ADDR. OF SUBTEST GOES HERE
718
719          001200          .=1200
720          .SBTTL **STARTING OF CPU TEST **
721 001200 032737 000001 001020 START: BIT      #1,@#SENV          ;UNDER APT
722 001206 001004          BNE      CONTIN
723 001210 012700 133431      MOV      #STRMSG,R0        ;START MESSAGE
724 001214 004767 132110      JSR      PC,TYPE
725 001220 012737 000000 001006 CONTIN: MOV      #0,@#SPASS      ;CLEAR PASS COUNT
726 001226 012737 133264 000024 RESTRT: MOV      #PWRDN,@#24 ;SET UP FOR POWER FAIL
727 001234 012706 001000      MOV      #STBOT,R6        ;SET UP STACK
728 001240 012737 001270 000004      MOV      #1$,@#4          ;SET UP FOR TIMEOUT IF NO MULTI TESTER
729 001246 012737 000340 000006      MOV      #340,@#6
730 001254 012737 000002 164000      MOV      #2,@#164000      ;SET BIT1 FOR MULTI TESTER
731 001262 012737 000001 000172      MOV      #1,@#MTFLAG      ;SET FLAG TO INDICATE MULTI-TESTER
732 001270 012737 021336 000004 1$: MOV      #T04,@#4          ;SET TRAP CATCHER
733 001276 012737 000000 000006      MOV      #0,@#6          ;SET HALT BACK IN LOCATION 6
734 001304 012706 001000      MOV      #STBOT,R6        ;INITIALIZE STACK POINTER
735 001310 012737 000001 001004      MOV      #1,@#STESTN      ;SET TEST NUMBER TO 1
736 001316 012737 000000 001002      MOV      #0,@#SFATAL      ;CLEAR ERROR INDICATOR
737 001324 012737 000000 001000      MOV      #0,@#SMSGTY      ;CLEAR MESSAGE TYPE(FOR APT)
738
739 001332 005067 176166      CLR      LSREG            ;THIS WILL TURN ON THE 4 LEDS
740
741 001336 105737 001020      TSTB    @#SENV            ;RUNNING ON APT
742 001342 001036          BNE      TS1              ;IF YES DO BRANCH SELF ON ERROR
743 001344 012737 000000 002164      MOV      #0,@#CPUHLT      ;IF NOT THEN PUT A HALT IN ON ERROR
744 001352 012737 000000 050470      MOV      #0,@#MMUHLT
745 001360 012737 000000 051404      MOV      #0,@#EXADHT
746 001366 012737 000000 053214      MOV      #0,@#Q22HLT
747 001374 012737 000000 124532      MOV      #0,@#FPHLT
748 001402 012737 000000 126674      MOV      #0,@#LTCHLT
749 001410 012737 000000 125754      MOV      #0,@#SL1HLT
750 001416 012737 000000 131540      MOV      #0,@#SL2HLT
751 001424 012737 000000 132504      MOV      #0,@#COMHLT
752 001432 012737 000000 054472      MOV      #0,@#BDVHLT
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 001440
780 001440 012700 021242
781 001444 012704 021300
782 001450 012767 000017 000130
783 001456 012067 000110
784 001462 012401
785 001464 012767 177777 000074
786 001472 012703 000020
787 001476 005267 000064
788 001502 032701 100000
789 001506 013705 177776
790 001512 042705 177773
791 001516 000165 001522
792 001522 000167 000020
793 001526 012767 001610 000042
794 001534 012767 001604 000040
795 001542 000167 000014
796 001546 012767 001604 000022
797 001554 012767 001610 000020
798 001562 006101
799
800 001564 012737
801 001566 000000
802 001570 177776
803 001572 000000
804 001574 000137
805 001576 000000
806 001600 000137
807 001602 000000
808 001604
809 001604 000551

```
*****
:
: THIS TEST EXECUTES EVERY POSSIBLE BRANCH WITH EVERY POSSIBLE
: CONDITION CODE COMBINATION.
: THE ROUTINE USES TWO TABLES. THE BRANCH TABLE HCIDS ALL THE
: POSSIBLE BRANCH INSTRUCTIONS, THE OTHER TABLE (YNTAB) HOLDS BIT MAPS FOR
: EACH BRANCH. A ONE IN THE BIT MAP INDICATES THAT THE CORRESPONDING
: BRANCH INSTRUCTION SHOULD BRANCH FOR THE CONDITION CODE SETTING WHICH
: CORRESPONDS TO THE BIT POSITION WITHIN THE MAP. FOR EXAMPLE IF THE LEFT
: MOST BIT IS A ONE THEN THE CORRESPONDING BRANCH INSTRUCTION SHOULD BRANCH
: WHEN THE CONDITION CODES ARE 0.
: THE ROUTINE CONSISTS OF NESTED LOOPS; THE OUTER LOOP SETS UP
: ALL THE POSSIBLE BRANCH INSTRUCTIONS. THE INNER LOOP SETS UP EVERY POSSIBLE
: CONDITION CODE FOR EACH BRANCH.
: THE BIT MAP IS USED TO SET THE ADDRESS LOCATION IN TWO
: JUMP MODE 3 INSTRUCTIONS. THE ADDRESSES ARE CHANGED TO ALLOW THE
: PROGRAM TO CONTINUE OR JUMP TO AN ERROR ROUTINE DEPENDING UPON
: WHETHER IT HANDLED THE BRANCH INSTRUCTION CORRECTLY.
: AT ANY ERROR HALT, LOCATION, BRH, HOLDS THE BRANCH INSTRUCTION
: UNDER TEST AND LOCATION, CC, HOLDS THE VALUE OF THE CONDITION CODES
: AT THE TIME THE BRANCH WAS EXECUTED.
:
: *****
: TEST 1 TEST THE BRANCH ROM
: *****
TS1:
SETUP:  MOV    #BRTAB,R0      ;INITIALIZE BRANCH TABLE POINTER
        MOV    #YNTAB,R4    ;INITIALIZE YES/NO BRANCH MAP POINTER
        MOV    #15.,BRCT    ;INITIALIZE BRANCH TABLE COUNT
SETBR:  MOV    (R0)+,BRH     ;GET NEXT BRANCH INST.
        MOV    (R4)+,R1     ;GET NEXT BRANCH MAP
        MOV    #-1,CC1      ;INITIALIZE CONDITION CODE VALUE
        MOV    #16.,R3      ;INITIALIZE CONDITION CODE COUNT
SETCC:  INC    CC1          ;SET FOR NEXT CC VALUE
        BIT    #100000,R1   ;SEE IF SHOULD BR W/ THESE CC'S
        MOV    @#177776,R5  ;SIMULATE A JNE
        BIC    #177773,R5   ;
        JMP    .+4(R5)      ; (JUMP NOT EQUAL)
        JMP    SET2BR      ; TO SET2BR
        MOV    #CONT,NBR    ;SET TO CONTINUE IF NO BRANCH
        MOV    #ER,YBR      ;SET TO REPORT ERROR IF BRANCH
        JMP    AROUND       ;GO AROUND OPPOSITE CONDITION
SET2BR: MOV    #ER,NBR      ;SET TO REPORT ERROR IF NO BRANCH
        MOV    #CONT,YBR    ;SET TO CONTINUE IF BRANCH
AROUN:  ROL    R1          ;UPDATE BIT MAP
        MOV    (PC)+,@(PC)+ ;SET CONDITION CODE
CC1:    0                  ;NEW CC VALUE GOES HERE
        177776
BRH:    0                  ;BRANCH INST. GOES HERE
        JMP    @ (PC)+      ;THIS JUMP IF NO BRANCH
NBR:    0                  ;WHERE TO GO IF NO BRANCH OCCURS
        JMP    @ (PC)+      ;THIS JUMP IF BRANCH OCCURS
YBR:    0                  ;WHERE TO GO IF BRANCH OCCURS
ER:
BR      ERROR1            ;
```


810 001606 000000
811 001610 005303
812 001612 013705 177776
813 001616 042705 177773
814 001622 000165 001626
815 001626 000167 177644
816 001632 005367 177750
817 001636 013705 177776
818 001642 042705 177773
819 001646 000165 001652
820 001652 000167 177600
821 001656 012700 000357

BRCT: 0
CONT: DEC R3 ;CC'S DONE?
MOV @#177776,R5 ;SIMULATE A JNE
BIC #177773,R5 ; (JUMP NOT EQUAL)
JMP .+4(R5) ; TO SETCC
JMP SETCC
DEC BRCT ;BR'S DONE?
MOV @#177776,R5 ;SIMULATE A JNE
BIC #177773,R5 ; (JUMP NOT EQUAL)
JMP .+4(R5) ; TO SETBR
JMP SETBR
MOV #357,R0 ;IF THIS TEST IS DONE SET UP R0 FOR THE NEXT
;SEVEN TESTS. THIS IS SAVING 4 LOCATIONS PER
;TEST WHICH I NEED BECAUSE BRANCHES WERE OUT
;OF BOUNDS.

822
823
824
825
826
827
828
829

:TEST 2 TEST TRAP OF RESERVED INSTRUCTION

830 001662
831 001662 012706 001000
832 001666 012767 001710 176114
833 001674 010067 176112
834 001700 005067 176072
835 001704 000077
836 001706
837 001706 000510
838 001710 020067 176062
839 001714 001105
840 001716 020627 000774
841 001722 001102
842 001724 022767 001706 177042
843 001732 001076
844 001734 005767 177036
845 001740 001073
846 001742 062706 000004
847 001746 012767 001770 176034
848 001754 005067 176032
849 001760 010037 177776
850 001764 000077
851 001766 000460
852 001770 005767 176002
853 001774 001055
854 001776 020067 176774
855 002002 001052

TS2:
MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
MOV #1\$,RTRAP ;SET UP NEW PC IN VECTOR
MOV R0,RTRAP+2 ;SET UP NEW PSW IN VECTOR
CLR STATUS ;CLEAR PRESENT (OLD) STATUS
TRAPA ;DO TRAP
8\$:
BR ERROR1 ;INSTRUCTION FAILED TO TRAP
1\$:
CMP R0,STATUS ;IS NEW STATUS CORRECT
BNE ERROR1 ;NEW STATUS WRONG
2\$:
CMP SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
BNE ERROR1 ;STACK DID NOT DECREMENT CORRECTLY
3\$:
CMP #8\$,STBOT-4 ;WAS PROPER PC SAVED
BNE ERROR1 ;PROPER PC WAS NOT SAVED
4\$:
TST STBOT-2 ;WAS OLD PSW SAVED
BNE ERROR1 ;WRONG PSW SAVED
5\$:
ADD #4,SP ;RESET STACK POINTER
MOV #6\$,RTRAP ;SET UP NEW PC IN VECTOR
CLR RTRAP+2 ;SET UP NEW PSW IN VECTOR
MOV R0,#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
TRAPA ;DO TRAP
6\$:
BR ERROR1 ;INSTRUCTION FAILED TO TRAP
7\$:
TST STATUS ;IS NEW PSW CORRECT
BNE ERROR1 ;NEW PSW WRONG
CMP R0,STBOT-2 ;WAS OLD STATUS STORED
BNE ERROR1 ;OLD STATUS WRONG

856
857
858
859 002004
860 002004 012706 001000
861 002010 012767 002032 176016
862 002016 010067 176014
863 002022 005067 175750
864 002026 104400
865 002030

:TEST 3 TEST TRAP OF TRAP INSTRUCTION

TS3:
MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
MOV #1\$,RTRAP1 ;SET UP NEW PC IN VECTOR
MOV R0,RTRAP1+2 ;SET UP NEW PSW IN VECTOR
CLR STATUS ;CLEAR PRESENT (OLD) STATUS
TRAP ;DO TRAP
8\$:

```

866 002030 000437
867 002032 020067 175740
868 002036 001034
869 002040 020627 000774
870 002044 001031
871 002046 022767 002030 176720
872 002054 001025
873 002056 005767 176714
874 002062 001022
875 002064 062706 000004
876 002070 012767 002112 175736
877 002076 005067 175734
878 002102 010037 177776
879 002106 104777
880 002110 000407
881 002112 005767 175660
882 002116 001004
883 002120 020067 176652
884 002124 001001
885 002126 000434
886
887
888
889
890 002130 012737 000001 001002
891 002136 012767 000001 176634
892 002144 032737 000001 001020
893 002152 001004
894 002154 012700 002166
895 002160 004767 131144
896 002164 000777
897
898 002166 040506 046111 042105
899 002174 042040 051125 047111
900 002202 020107 050103 020125
901 002210 042524 052123 005123
902 002216 000015
903
904
905
906
907
908 002220
909 002220 012706 001000
910 002224 012767 002246 175566
911 002232 010067 175564
912 002236 005067 175534
913 002242 000004
914 002244
915 002244 000731
916 002246 020067 175524
917 002252 001326
918 002254 020627 000774
919 002260 001323
920 002262 022767 002244 176504
921 002270 001317

      BR      ERROR1      ;INSTRUCTION FAILED TO TRAP
      CMP     RO,STATUS    ;IS NEW STATUS CORRECT
      BNE     ERROR1      ;NEW STATUS WRONG
2$:    CMP     SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
      BNE     ERROR1      ;STACK DID NOT DECREMENT CORRECTLY
3$:    CMP     #8$,STBOT-4 ;WAS PROPER PC SAVED
      BNE     ERROR1      ;PROPER PC WAS NOT SAVED
4$:    TST     STBOT-2    ;WAS OLD PSW SAVED
      BNE     ERROR1      ;WRONG PSW SAVED
5$:    ADD     #4,SP      ;RESET STACK POINTER
      MOV     #6$,RTRAP1  ;SET UP NEW PC IN VECTOR
      CLR     RTRAP1+2    ;SET UP NEW PSW IN VECTOR
      MOV     RO,@#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
      TRAPC   ;DO TRAP WITH LOWER BYTE ALL ONES
      BR      ERROR1      ;INSTRUCTION FAILED TO TRAP
6$:    TST     STATUS     ;IS NEW PSW CORRECT
      BNE     ERROR1      ;NEW PSW WRONG
7$:    CMP     RO,STBOT-2 ;WAS OLD STATUS STORED
      BNE     ERROR1      ;OLD STATUS WRONG
      BR      CPUHLT+34   ;GET OVER ERROR CALL TO NEXT TEST IF NO ERROR
*****
;THIS ERROR IS USED FOR THE ENTIRE CPU,TRAPS AND EIS PORTION OF
;THIS TEST
*****
ERROR1: MOV     #1,@#FATAL ;SET UP FATAL ERROR NUMBER
        MOV     #1,$MSGTY  ;SET FATAL ERROR FLAG
        BIT     #1,@#SENV  ;UNDER APT
        BNE     CPUHLT     ;YES, THEN DO NOT PRINT
        MOV     #CPUMSG,RO
        JSR     PC,TYPE    ;TYPE MSG
CPUHLT: BR      .
CPUMSG: .ASCIZ  /FAILED DURING CPU TESTS/<12><15>

.EVEN
*****
;TEST 4 TEST TRAP OF IOT INSTRUCTION
*****
TS4:   MOV     #STBOT,SP   ;INITIALIZE THE STACK POINTER
        MOV     #1$,RTRAP2 ;SET UP NEW PC IN VECTOR
        MOV     RO,RTRAP2+2 ;SET UP NEW PSW IN VECTOR
        CLR     STATUS    ;CLEAR PRESENT (OLD) STATUS
        IOT      ;DO TRAP
8$:    BR      ERROR1      ;INSTRUCTION FAILED TO TRAP
1$:    CMP     RO,STATUS    ;IS NEW STATUS CORRECT
      BNE     ERROR1      ;NEW STATUS WRONG
2$:    CMP     SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
      BNE     ERROR1      ;STACK DID NOT DECREMENT CORRECTLY
3$:    CMP     #8$,STBOT-4 ;WAS PROPER PC SAVED
      BNE     ERROR1      ;PROPER PC WAS NOT SAVED
    
```

```
922 002272 005767 176500 4$: TST STBOT-2 ;WAS OLD PSW SAVED
923 002276 001314 BNE ERROR1 ;WRONG PSW SAVED
924 002300 062706 000004 5$: ADD #4,SP ;RESET STACK POINTER
925 002304 012767 002326 175506 MOV #6$,RTRAP2 ;SET UP NEW PC IN VECTOR
926 002312 005067 175504 CLR RTRAP2+2 ;SET UP NEW PSW IN VECTOR
927 002316 010037 177776 MOV RO,@#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
928 002322 000004 IOT ;DO TRAP
929 002324 000701 BR ERROR1 ;INSTRUCTION FAILED TO TRAP
930 002326 005767 175444 6$: TST STATUS ;IS NEW PSW CORRECT
931 002332 001276 BNE ERROR1 ;NEW PSW WRONG
932 002334 020067 176436 7$: CMP RO,STBOT-2 ;WAS OLD STATUS STORED
933 002340 001273 BNE ERROR1 ;OLD STATUS WRONG
934 *****
935 ;TEST 5 TEST TRAP OF EMT INSTRUCTION
936 *****
937 002342 TS5:
938 002342 012706 001000 MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
939 002346 012767 002370 175454 MOV #1$,RTRAP3 ;SET UP NEW PC IN VECTOR
940 002354 010067 175452 MOV RO,RTRAP3+2 ;SET UP NEW PSW IN VECTOR
941 002360 005067 175412 CLR STATUS ;CLEAR PRESENT (OLD) STATUS
942 002364 104000 EMT ;DO TRAP
943 002366 8$:
944 002366 000660 BR ERROR1 ;INSTRUCTION FAILED TO TRAP
945 002370 020067 175402 1$: CMP RO,STATUS ;IS NEW STATUS CORRECT
946 002374 001255 BNE ERROR1 ;NEW STATUS WRONG
947 002376 020627 000774 2$: CMP SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
948 002402 001252 BNE ERROR1 ;STACK DID NOT DECREMENT CORRECTLY
949 002404 022767 002366 176362 3$: CMP #8$,STBOT-4 ;WAS PROPER PC SAVED
950 002412 001246 BNE ERROR1 ;PROPER PC WAS NOT SAVED
951 002414 005767 176356 4$: TST STBOT-2 ;WAS OLD PSW SAVED
952 002420 001243 BNE ERROR1 ;WRONG PSW SAVED
953 002422 062706 000004 5$: ADD #4,SP ;RESET STACK POINTER
954 002426 012767 002450 175374 MOV #6$,RTRAP3 ;SET UP NEW PC IN VECTOR
955 002434 005067 175372 CLR RTRAP3+2 ;SET UP NEW PSW IN VECTOR
956 002440 010037 177776 MOV RO,@#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
957 002444 104377 EMTA ;DO TRAP WITH LOWER BYTE ALL ONES
958 002446 000630 BR ERROR1 ;INSTRUCTION FAILED TO TRAP
959 002450 005767 175322 6$: TST STATUS ;IS NEW PSW CORRECT
960 002454 001225 BNE ERROR1 ;NEW PSW WRONG
961 002456 020067 176314 7$: CMP RO,STBOT-2 ;WAS OLD STATUS STORED
962 002462 001222 BNE ERROR1 ;OLD STATUS WRONG
963 002464 000401 BR ERROR2+2 ;WE MUST GET OVER ERROR CALL AT END OF THIS TEST
964 *****
965 ;THIS ERROR IS NEEDED BECAUSE BRANCHES IN TRAP TESTS BEYOND HERE CAN NOT
966 ;REACH ERROR1.
967 *****
968 002466 000620 ERROR2: BR ERROR1
969
970
971 *****
972 ;TEST 6 TEST TRAP OF TRACE-TRAP INSTRUCTION
973 *****
974 002470 TS6:
975 002470 012706 001000 MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
976 002474 012767 002516 175312 MOV #1$,RTRAP4 ;SET UP NEW PC IN VECTOR
977 002502 010067 175310 MOV RO,RTRAP4+2 ;SET UP NEW PSW IN VECTOR
```

```
978 002506 005067 175264          CLR      STATUS          ;CLEAR PRESENT (OLD) STATUS
979 002512 000003                    TRT          ;DO TRAP
980 002514                    8$:
981 002514 000764                    BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
982 002516 020067 175254          1$:  CMP      R0,STATUS      ;IS NEW STATUS CORRECT
983 002522 001361                    BNE     ERROR2          ;NEW STATUS WRONG
984 002524 020627 000774          2$:  CMP      SP,#STBOT-4     ;DID STACK DECREMENT CORRECTLY
985 002530 001356                    BNE     ERROR2          ;STACK DID NOT DECREMENT CORRECTLY
986 002532 022767 002514 176234  3$:  CMP      #8$,STBOT-4     ;WAS PROPER PC SAVED
987 002540 001352                    BNE     ERROR2          ;PROPER PC WAS NOT SAVED
988 002542 005767 176230          4$:  TST      STBOT-2        ;WAS OLD PSW SAVED
989 002546 001347                    BNE     ERROR2          ;WRONG PSW SAVED
990 002550 062706 000004          5$:  ADD      #4,SP          ;RESET STACK POINTER
991 002554 012767 002576 175232  MOV      #6$,RTRAP4      ;SET UP NEW PC IN VECTOR
992 002562 005067 175230          CLR      RTRAP4+2        ;SET UP NEW PSW IN VECTOR
993 002566 010037 177776          MOV      R0,#STATUS      ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
994 002572 000003                    TRT          ;DO TRAP
995 002574 000734                    BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
996 002576 005767 175174          6$:  TST      STATUS          ;IS NEW PSW CORRECT
997 002602 001331                    BNE     ERROR2          ;NEW PSW WRONG
998 002604 020067 176166          7$:  CMP      R0,STBOT-2     ;WAS OLD STATUS STORED
999 002610 001326                    BNE     ERROR2          ;OLD STATUS WRONG
1000 :PDP-11 ILLEGAL AND ADDRESS INSTRUCTION TEST
1001 :ALL INSTRUCTIONS THAT ARE RESERVED
1002 :SHOULD TRAP TO LOCATION 4, AND THE
1003 :PC THAT POINTS TO THE TRAPPING INSTRUCTION
1004 :SHOULD BE PLACED ON THE STACK
1005
1006 :*****
1007 :TEST 7 TEST TRAP OF ILLEGAL INSTRUCTION
1008 :*****
1009 002612          TS7:
1010 002612 012706 001000          MOV      #STBOT,SP      ;INITIALIZE THE STACK POINTER
1011 002616 012767 002640 175160  MOV      #1$,RTRAP5     ;SET UP NEW PC IN VECTOR
1012 002624 010067 175156          MOV      R0,RTRAP5+2   ;SET UP NEW PSW IN VECTOR
1013 002630 005067 175142          CLR      STATUS        ;CLEAR PRESENT (OLD) STATUS
1014 002634 000100                    JMP      %0             ;DO TRAP
1015 002636                    8$:
1016 002636 000713                    BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
1017 002640 020067 175132          1$:  CMP      R0,STATUS      ;IS NEW STATUS CORRECT
1018 002644 001310                    BNE     ERROR2          ;NEW STATUS WRONG
1019 002646 020627 000774          2$:  CMP      SP,#STBOT-4     ;DID STACK DECREMENT CORRECTLY
1020 002652 001305                    BNE     ERROR2          ;STACK DID NOT DECREMENT CORRECTLY
1021 002654 022767 002636 176112  3$:  CMP      #8$,STBOT-4     ;WAS PROPER PC SAVED
1022 002662 001301                    BNE     ERROR2          ;PROPER PC WAS NOT SAVED
1023 002664 005767 176106          4$:  TST      STBOT-2        ;WAS OLD PSW SAVED
1024 002670 001276                    BNE     ERROR2          ;WRONG PSW SAVED
1025 002672 062706 000004          5$:  ADD      #4,SP          ;RESET STACK POINTER
1026 002676 012767 002720 175100  MOV      #6$,RTRAP5     ;SET UP NEW PC IN VECTOR
1027 002704 005067 175076          CLR      RTRAP5+2        ;SET UP NEW PSW IN VECTOR
1028 002710 010037 177776          MOV      R0,#STATUS      ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
1029 002714 000100                    JMP      %0             ;DO TRAP
1030 002716 000663                    BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
1031 002720 005767 175052          6$:  TST      STATUS          ;IS NEW PSW CORRECT
1032 002724 001260                    BNE     ERROR2          ;NEW PSW WRONG
1033 002726 020067 176044          7$:  CMP      R0,STBOT-2     ;WAS OLD STATUS STORED
```

```
1034 002732 001255          BNE      ERROR2          ;OLD STATUS WRONG
1035                          ;*****
1036                          ;TEST 10          TEST TRAP OF ALL ILLEGAL INSTRUCTION
1037                          ;*****
1038                          ;TS10:
1039 002734 012706 001000    MOV      #STBOT,SP          ;INITIALIZE THE STACK POINTER
1040 002740 012767 002762 175036  MOV      #1$,RTRAP5        ;SET UP NEW PC IN VECTOR
1041 002746 010067 175034    MOV      R0,RTRAP5+2      ;SET UP NEW PSW IN VECTOR
1042 002752 005067 175020    CLR      STATUS           ;CLEAR PRESENT (OLD) STATUS
1043 002756 004000          JSR      %0,%0            ;DO TRAP
1044 002760          8$:
1045 002760 000642          BR       ERROR2           ;INSTRUCTION FAILED TO TRAP
1046 002762 020067 175010    1$:  CMP      R0,STATUS        ;IS NEW STATUS CORRECT
1047 002766 001237          BNE      ERROR2           ;NEW STATUS WRONG
1048 002770 020627 000774    2$:  CMP      SP,#STBOT-4     ;DID STACK DECREMENT CORRECTLY
1049 002774 001234          BNE      ERROR2           ;STACK DID NOT DECREMENT CORRECTLY
1050 002776 022767 002760 175770 3$:  CMP      #8$,STBOT-4     ;WAS PROPER PC SAVED
1051 003004 001230          BNE      ERROR2           ;PROPER PC WAS NOT SAVED
1052 003006 005767 175764    4$:  TST      STBOT-2         ;WAS OLD PSW SAVED
1053 003012 001225          BNE      ERROR2           ;WRONG PSW SAVED
1054 003014 062706 000004    5$:  ADD      #4,SP           ;RESET STACK POINTER
1055 003020 012767 003042 174756  MOV      #6$,RTRAP5        ;SET UP NEW PC IN VECTOR
1056 003026 005067 174754    CLR      RTRAP5+2         ;SET UP NEW PSW IN VECTOR
1057 003032 010037 177776    MOV      R0,@#STATUS      ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
1058 003036 004000          JSR      %0,%0            ;DO TRAP
1059 003040 000612          BR       ERROR2           ;INSTRUCTION FAILED TO TRAP
1060 003042 005767 174730    6$:  TST      STATUS          ;IS NEW PSW CORRECT
1061 003046 001207          BNE      ERROR2           ;NEW PSW WRONG
1062 003050 020067 175722    7$:  CMP      R0,STBOT-2      ;WAS OLD STATUS STORED
1063 003054 001204          BNE      ERROR2           ;OLD STATUS WRONG
1064
1065                          ;*****
1066                          ;SBTTL DATA PATH TESTS
1067                          ;
1068                          ; THE DATA PATH TESTS ARE USED TO VEPIFY THAT VARIOUS
1069                          ; DATA PATTERNS CAN BE SUCCESSFULLY MOVED THROUGH THE DATA PATHS
1070                          ; MOVE AND COMPARE MODE 2,3 INSTRUCTIONS ARE USED TO PASS AND
1071                          ; TEST VARIOUS DATA PATTERNS IN THE DATA PATHS.
1072                          ; THE TEST EXERCISES THE INTERNAL DATA PATHS, AND THE UNIBUS
1073                          ; DATA TRANSCIEVERS.
1074                          ; IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0)
1075                          ; TO SEE WHICH BITS OF THE DATA PATH ARE FAILING.
1076 003056 012737 002130 000030  MOV      #ERROR1,@#30     ;SET UP VECTOR FOR ERROR CALLS
1077 003064 012737 000340 000032  MOV      #340,@#32        ;SET UP NEW PSW
1078 003072 012737 021336 000004  MOV      #T04,@#4         ;SET UP FOR UNEXPECTED TRAP TO 4
1079 003100 012737 021340 000010  MOV      #T010,@#10       ;SET UP FOR UNEXPECTED TRAP TO 10
1080 003106 012737 021342 000014  MOV      #T014,@#14       ;SET UP FOR UNEXPECTED TRAP TO 14
1081 003114 012737 021350 000034  MOV      #T034,@#34       ;SET UP FOR UNEXPECTED TRAP TO 34
1082
1083                          ;*****
1084                          ;TEST 11          TEST OF ZEROES IN THE DATA PATH
1085                          ;*****
1086                          ;TS11:
1087 003122          MOV      #0,@#0          ;MOVE ZEROES THRU ADDRESS LINES, DATA
1088 003122 012737 000000 000000  ;LINES AND INTERNAL PATHS
1089
```

1090 003130 005737 000000
1091 003134 001401
1092 003136 104000
1093
1094
1095
1096
1097 003140
1098 003140 012737 125252 000000
1099
1100 003146 022737 125252 000000
1101 003154 001401
1102 003156 104000
1103
1104
1105
1106
1107 003160
1108 003160 012737 052525 000000
1109
1110 003166 022737 052525 000000
1111 003174 001401
1112 003176 104000
1113
1114
1115
1116
1117 003200
1118 003200 012737 177777 000000
1119 003206 022737 177777 000000
1120 003214 001401
1121 003216 104000
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140 003220
1141 003220 000241
1142 003222 012737 000001 000000
1143 003230 006137 000000
1144 003234 022737 000002 000000
1145 003242 001401

TST @#0 ;SUCCESSFUL?
BEQ TS12
EMT ;DATA INCORRECT

:TEST 12 TEST OF PATTERN 125252 IN DATA PATH

TS12:
MOV #125252,@#0 ;MOVE ALTERNATING ONES AND ZEROES
;THRU DATA PATHS
CMP #125252,@#0 ;SUCCESSFUL
BEQ TS13
EMT ;DATA INCORRECT

:TEST 13 TEST OF PATTERN 052525 IN DATA PATH

TS13:
MOV #052525,@#0 ;MOVE ALTERNATING ZEROES AND ONES
;THRU DATA PATH
CMP #052525,@#0 ;SUCCESSFUL?
BEQ TS14
EMT ;DATA INCORRECT

:TEST 14 TEST OF ALL ONES IN DATA PATH

TS14:
MOV #177777,@#0 ;MOVE ONES THRU DATA PATH
CMP #177777,@#0 ;SUCCESSFUL
BEQ TS15
EMT ;DATA INCORRECT

:SBTTL B-REGISTER TEST
:
: THE B-REGISTER (LOCATION 0) SHIFTING LOGIC TESTS ARE USED
: TO TEST THAT THE B-REGISTER CAN HOLD VARIOUS DATA PATTERNS AND THAT
: THE ASSOCIATED LOGIC SUPPORTS THE SHIFTING FUNCTIONS WITHIN THE
: B-REGISTER AND C-BIT.
: A ONE IS SHIFTED THROUGH EVERY BIT IN THE B-REGISTER AND C-BIT IN
: BOTH DIRECTIONS.
: THE B-REGISTER ITSELF IS TESTED IN ITS ABILITY AS A BUFFER AND AS
: A SHIFT REGISTER. DATA IS ALSO PASSED THROUGH THE DATA PATH AND ALU.
: IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0) TO SEE
: WHICH BITS OF THE B-REGISTER MAY BE FAILING.
:*****
:TEST 15 SHIFT BIT 0 TO BIT 1
:*****
TS15:
CLC ;CLEAR CARRY BIT
MOV #1,@#0 ;LOAD A 1
ROL @#0 ;SHIFT LEFT
CMP #2,@#0 ;SUCCESSFUL
BEQ TS16

1146 003244 104000
1147
1148
1149
1150
1151 003246
1152 003246 012737 000000 000000
1153 003254 000261
1154 003256 005137 000000
1155 003262 103001
1156 003264 104000
1157 003266 022737 000001 000000
1158 003274 001401
1159 003276 104000
1160
1161
1162
1163
1164 003300
1165 003300 012737 000001 000000
1166 003306 012700 177757
1167 003312 000241
1168 003314 005200
1169 003316 001404
1170 003320 006137 000000
1171 003324 103373
1172 003326 001401
1173 003330
1174 003330 104000
1175
1176
1177
1178
1179 003332
1180 003332 012737 100000 000000
1181 003340 000241
1182 003342 006037 000000
1183 003346 022737 040000 000000
1184 003354 001401
1185 003356 104000
1186
1187
1188
1189
1190 003360
1191 003360 012737 100000 000000
1192 003366 012700 177757
1193 003372 000241
1194 003374 005200
1195 003376 001404
1196 003400 006037 000000
1197 003404 103373
1198 003406 001401
1199 003410
1200 003410 104000
1201

```
EMT ;BIT 1 NOT SET
:*****
:TEST 16 SHIFT CARRY INTO BIT 0
:*****
TS16:
MOV #0,@#0 ;CLEAR LOCATION
SEC ;SET CARRY
ROL @#0 ;ROTATE CARRY BIT TO BIT 0
BCC CARRY1
EMT ;CARRY CLEAR
CARRY1: CMP #1,@#0 ;BIT 0 SET
BEQ TS17
EMT ;BIT 0 NOT SET
:*****
:TEST 17 LEFT SHIFT FROM BIT 0 TO C-BIT
:*****
TS17:
MOV #1,@#0 ;SET BIT 0
MOV #-21,RO ;SET BIT COUNTER
CLC ;CLEAR C-BIT
SHL: INC RO ;INCREMENT BIT COUNTER
BEQ SHLE ;BR TO ERROR HALT IF BIT IS LOST
ROL @#0 ;SHIFT LEFT ONE POSITION
BCC SHL ;BRANCH IF C-BIT NOT SET
BEQ TS20
SHLE: EMT ;LEFT SHIFTING LOGIC FAILED
:*****
:TEST 20 SHIFT BIT 15 TO BIT 14
:*****
TS20:
MOV #100000,@#0 ;SET BIT 15
CLC ;CLEAR CARRY
ROR @#0 ;SHIFT BIT 15 TO BIT 14
CMP #40000,@#0 ;SUCCESSFUL
BEQ TS21
EMT ;BIT 14 NOT SET
:*****
:TEST 21 RIGHT SHIFT FROM BIT 15 TO C-BIT
:*****
TS21:
MOV #100000,@#0 ;SET BIT 15
MOV #-21,RO ;SET BIT COUNTER
CLC ;CLEAR C-BIT
SHR: INC RO ;INCREMENT BIT COUNTER
BEQ SHRE ;BR TO ERROR HALT IF BIT IS LOST
ROR @#0 ;ROTATE RIGHT ONE POSITION
BCC SHR ;BRANCH IF C-BIT CLEAR
BEQ TS22
SHRE: EMT ;RIGHT SHIFT LOGIC FAILED
```

1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225 003412
1226
1227 003412 012700 000000
1228 003416 005700
1229 003420 001401
1230 003422 104000
1231
1232
1233
1234
1235 003424
1236 003424 012700 125252
1237 003430 020027 125252
1238 003434 001401
1239 003436 104000
1240
1241
1242
1243
1244 003440
1245 003440 012700 052525
1246 003444 020027 052525
1247 003450 001401
1248 003452 104000
1249
1250
1251
1252
1253 003454
1254 003454 012700 177777
1255 003460 020027 177777
1256 003464 001401
1257 003466 104000

```
*****
:SBTTL  SCRATCH PAD TESTS
:
:      THE SCRATCH PAD TESTS ARE USED TO VERIFY THAT VARIOUS
:DATA PATTERNS CAN BE SUCCESSFULLY HELD IN THE SCRATCH PAD
:CIRCUITRY.  MOVE AND COMPARE INSTRUCTIONS ARE USED TO TEST THAT
:RO CAN HOLD VARIOUS DATA PATTERNS.  EACH DATA PATTERN IS
:MOVED AND TESTED IN A SMALL LOOP CONVENIENT FOR SCOPING.  THE
:SUCCESSFUL COMPLETION OF THESE TESTS SHOULD VERIFY THE CIRCUITRY EXTERNAL
:TO THE SCRATCH PAD ITSELF.
:      THE REMAINDER OF THE GENERAL REGISTERS ARE TESTED BY MOVING
:A BIT INTO BIT 0 OF THE REGISTER AND SHIFTING IT LEFT ONE
:BIT AT A TIME INTO THE CARRY BIT.  THE RESULT IS THEN CHECKED TO INSURE THAT
:NO BITS WERE PICKED.  THE PROCEDURE IS THEN REPEATED UNDER OPPOSITE
:CONDITIONS.  THE GENERAL REGISTER AND THE CARRY BIT ARE SET TO
:ALL ONES, AND A ZERO IS SHIFTED LEFT FROM BIT 0 INTO THE CARRY BIT.
:THE RESULT IS THEN CHECKED TO INSURE THAT NO ZEROES WERE PICKED.
:      AT THIS POINT ALL OF THE GENERAL REGISTERS HAVE BEEN EXERCISED
:AS WELL AS REGISTER 11.
*****
:TEST 22      TEST IF RO CAN HOLD ALL ZEROES
*****
TS22:
      MOV      #0,RO      ;MOVE ZEROES TO RO
      TST      RO        ;SUCCESSFUL?
      BEQ      TS23
      EMT
      ;RO NOT 0
*****
:TEST 23      TEST IF RO CAN HOLD ONES AND ZEROES
*****
TS23:
      MOV      #125252,RO  ;MOVE ALTERNATING ONES AND ZEROES TO RO
      CMP      RO,#125252 ;SUCCESSFUL?
      BEQ      TS24
      EMT
      ;RO NOT 125252
*****
:TEST 24      TEST IF RO CAN HOLD ZEROES AND ONES
*****
TS24:
      MOV      #052525,RO  ;MOVE ALTERNATING ZEROES AND ONES TO RO
      CMP      RO,#052525 ;SUCCESSFUL?
      BEQ      TS25
      EMT
      ;RO NOT 52525
*****
:TEST 25      TEST IF RO CAN HOLD ALL ONES
*****
TS25:
      MOV      #177777,RO  ;MOVE ALL ONES TO RO
      CMP      RO,#177777 ;SUCCESSFUL?
      BEQ      TS26
      EMT
      ;RO NOT 177777
```


1258
1259
1260
1261
1262 003470
1263 003470 012701 000001
1264 003474 012700 177757
1265 003500 000241
1266 003502 005200
1267 003504 001403
1268 003506 006101
1269 003510 103374
1270 003512 001401
1271 003514
1272 003514 104000
1273
1274
1275
1276
1277 003516
1278 003516 012701 177776
1279 003522 012700 177757
1280 003526 000261
1281 003530 005200
1282 003532 001405
1283 003534 006101
1284 003536 103774
1285 003540 022701 177777
1286 003544 001401
1287 003546
1288 003546 104000
1289
1290
1291
1292 003550
1293 003550 012702 000001
1294 003554 012700 177757
1295 003560 000241
1296 003562 005200
1297 003564 001403
1298 003566 006102
1299 003570 103374
1300 003572 001401
1301 003574
1302 003574 104000
1303
1304
1305
1306
1307 003576
1308 003576 012700 177776
1309 003602 012700 177757
1310 003606 000261
1311 003610 005200
1312 003612 001405
1313 003614 006102

```
*****
:TEST 26      TEST IF R1 CAN HOLD A ONE IN ALL BITS
*****
TS26:
      MOV      #1,R1          ;SET BIT 0
      MOV      #-21,R0       ;SET BIT COUNTER
      CLC                     ;CLEAR C-BIT
REG1:  INC      R0            ;INCREMENT BIT COUNTER
      BEQ      REG1E         ;BR TO ERROR HALT IF BIT IS LOST
      ROL      R1            ;ROTATE 1 POSITION
      BCC      REG1         ;ALL DONE
      BEQ      TS27
REG1E: EMT                    ;FAILURE WITH R1

*****
:TEST 27      TEST IF R1 CAN HOLD A ZERO IN ALL BITS
*****
TS27:
      MOV      #-2,R1        ;SET ALL ONES IN R1 EXCEPT FOR BIT 0
      MOV      #-21,R0       ;SET BIT COUNTER
      SEC                     ;SET C-BIT
REG1A: INC      R0            ;INCREMENT COUNTER
      BEQ      R1ERR         ;BR TO ERROR HALT IF COUNTER=0
      ROL      R1            ;ROTATE 1 POSITION
      BCS      REG1A         ;CONTINUE UNTIL C-BIT IS CLEAR
      CMP      #-1,R1        ;CHECK DATA IN R1
      BEQ      TS30
R1ERR: EMT                    ;FAILURE WITH R1

*****
:TEST 30      TEST IF R2 CAN HOLD A ONE IN ALL BITS
*****
TS30:
      MOV      #1,R2          ;SET BIT 0
      MOV      #-21,R0       ;SET BIT COUNTER
      CLC                     ;CLEAR C-BIT
REG2:  INC      R0            ;INCREMENT BIT COUNTER
      BEQ      REG2E         ;BR TO ERROR HALT IF BIT IS LOST
      ROL      R2            ;ROTATE 1 POSITION
      BCC      REG2         ;ALL DONE
      BEQ      TS31
REG2E: EMT                    ;FAILURE WITH R2

*****
:TEST 31      TEST IF R2 CAN HOLD A ZERO IN ALL BITS
*****
TS31:
      MOV      #-2,R2        ;SET ALL ONES IN R2 EXCEPT FOR BIT 0
      MOV      #-21,R0       ;SET BIT COUNTER
      SEC                     ;SET C-BIT
REG2B: INC      R0            ;INCREMENT BIT COUNTER
      BEQ      R2ERR         ;BR TO ERROR HALT IF COUNTER=0
      ROL      R2            ;ROTATE 1 POSITION
```

1314 003616 103774
1315 003620 022702 177777
1316 003624 001401
1317 003626
1318 003626 104000
1319
1320
1321
1322
1323 003630
1324 003630 012703 000001
1325 003634 012700 177757
1326 003640 000241
1327 003642 005200
1328 003644 001403
1329 003646 006103
1330 003650 103374
1331 003652 001401
1332 003654
1333 003654 104000
1334
1335
1336
1337
1338 003656
1339 003656 012703 177776
1340 003662 012700 177757
1341 003666 000261
1342 003670 005200
1343 003672 001405
1344 003674 006103
1345 003676 103774
1346 003700 022703 177777
1347 003704 001401
1348 003706
1349 003706 104000
1350
1351
1352
1353
1354 003710
1355 003710 012704 000001
1356 003714 012700 177757
1357 003720 000241
1358 003722 005200
1359 003724 001403
1360 003726 006104
1361 003730 103374
1362 003732 001401
1363 003734
1364 003734 104000
1365
1366
1367
1368
1369 003736

```
BCS REG2B ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R2 ;CHECK DATA IN R2
BEQ TS32
R2ERR: EMT ;FAILURE WITH R2

;*****
;TEST 32 TEST IF R3 CAN HOLD A ONE IN ALL BITS
;*****
TS32:
MOV #1,R3 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG3: INC R0 ;INCREMENT BIT COUNTER
BEQ REG3E ;BR TO ERROR HALT IF BIT IS LOST
ROL R3 ;ROTATE 1 POSITION
BCC REG3 ;ALL DONE
BEQ TS33
REG3E: EMT ;FAILURE WITH R3

;*****
;TEST 33 TEST IF R3 CAN HOLD A ZERO IN ALL BITS
;*****
TS33:
MOV #-2,R3 ;SET ALL ONES IN R3 EXCEPT FOR BIT 0
MOV #-21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG3A: INC R0 ;INCREMENT BIT COUNTER
BEQ R3ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R3 ;ROTATE 1 POSITION
BCS REG3A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R3 ;CHECK DATA
BEQ TS34
R3ERR: EMT ;FAILURE WITH R3

;*****
;TEST 34 TEST IF R4 CAN HOLD A ONE IN ALL BITS
;*****
TS34:
MOV #1,R4 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG4: INC R0 ;INCREMENT BIT COUNTER
BEQ REG4E ;BR TO ERROR HALT IF BIT IS LOST
ROL R4 ;ROTATE 1 POSITION
BCC REG4 ;ALL DONE
BEQ TS35
REG4E: EMT ;FAILURE WITH R4

;*****
;TEST 35 TEST IF R4 CAN HOLD A ZERO IN ALL BITS
;*****
TS35:
```

1370 003736 012704 177776
1371 003742 012700 177757
1372 003746 000261
1373 003750 005200
1374 003752 001405
1375 003754 006104
1376 003756 103774
1377 003760 022704 177777
1378 003764 00401
1379 003766
1380 003766 104000

```
MOV #2,R4 ;SET ALL ONES IN R4 EXCEPT FOR BIT 0
MOV #-21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG4A: INC R0 ;INCREMENT BIT COUNTER
BEQ R4ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R4 ;ROTATE 1 POSITION
BCS REG4A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R4 ;CHECK DATA
BEQ TS36
R4ERR: EMT ;FAILURE WITH R4
```

1381
1382
1383
1384
1385
1386 003770
1387 003770 012705 000001
1388 003774 012700 177757
1389 004000 000241
1390 004002 005200
1391 004004 001403
1392 004006 006105
1393 004010 103374
1394 004012 001401
1395 004014
1396 004014 104000

```
*****
:TEST 36 TEST IF R5 CAN HOLD A ONE IN ALL BITS
*****
TS36: MOV #1,R5 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG5: INC R0 ;INCREMENT BIT COUNTER
BEQ REG5E ;BR TO ERROR HALT IF BIT IS LOST
ROL R5 ;ROTATE 1 POSITION
BCC REG5 ;ALL DONE
BEQ TS37
REG5E: EMT ;FAILURE WITH R5
```

1397
1398
1399
1400
1401 004016
1402 004016 012705 177776
1403 004022 012700 177757
1404 004026 000261
1405 004030 005200
1406 004032 001405
1407 004034 006105
1408 004036 103774
1409 004040 022705 177777
1410 004044 001401
1411 004046
1412 004046 104000

```
*****
:TEST 37 TEST IF R5 CAN HOLD A ZERO IN ALL BITS
*****
TS37: MOV #-2,R5 ;SET ALL ONES IN R5 EXCEPT FOR BIT 0
MOV #-21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG5A: INC R0 ;INCREMENT BIT COUNTER
BEQ R5ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R5 ;ROTATE 1 POSITION
BCS REG5A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R5 ;CHECK DATA
BEQ TS40
R5ERR: EMT ;FAILURE WITH R5
```

1413
1414
1415
1416
1417 004050
1418 004050 012706 000001
1419 004054 012700 177757
1420 004060 000241
1421 004062 005200
1422 004064 001403
1423 004066 006106
1424 004070 103374
1425 004072 001401

```
*****
:TEST 40 TEST IF R6 CAN HOLD A ONE IN ALL BITS
*****
TS40: MOV #1,R6 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG6: INC R0 ;INCREMENT BIT COUNTER
BEQ REG6E ;BR TO ERROR HALT IF BIT IS LOST
ROL R6 ;ROTATE 1 POSITION
BCC REG6 ;ALL DONE
BEQ TS41
```

1426	004074			
1427	004074	104000		
1428				
1429				
1430				
1431				
1432	004076			
1433	004076	012706	177776	
1434	004102	012700	177757	
1435	004106	030261		
1436	004110	005200		
1437	004112	001405		
1438	004114	006106		
1439	004116	103774		
1440	004120	022706	177777	
1441	004124	001401		
1442	004126			
1443	004126	104000		
1444				
1445				
1446				
1447				
1448				
1449				
1450				
1451				
1452				
1453				
1454				
1455				
1456				
1457				
1458				
1459				
1460				
1461				
1462	004130			
1463	004130	012706	001000	
1464	004134	012737	000000	177776
1465	004142	005737	177776	
1466	004146	001401		
1467	004150	104000		
1468				
1469				
1470				
1471				
1472	004152			
1473	004152	012737	000252	177776
1474	004160	023727	177776	000252
1475	004166	001401		
1476	004170	104000		
1477				
1478				
1479				
1480				
1481	004172			

```

REG6E:
      EMT                                ;FAILURE WITH R6

:*****
:TEST 41      TEST IF R6 CAN HOLD A ZERO IN ALL BITS
:*****
TS41:
      MOV      #-2,R6                    ;SET ALL ONES IN R6 EXCEPT FOR BIT 0
      MOV      #-21,R0                   ;SET BIT COUNTER
      SEC                                ;SET C-BIT
REG6A:  INC     R0                        ;INCREMENT BIT COUNT
      BEQ     R6ERR                       ;BR TO ERROR HALT IF COUNTER=0
      ROL     R6                          ;ROTATE 1 POSITION
      BCS     REG6A                       ;CONTINUE UNTIL C-BIT IS CLEAR
      CMP     #-1,R6                      ;CHECK DATA
      BEQ     TS42

R6ERR:
      EMT                                ;FAILURE WITH R6

:*****
:SBTTL PSW TESTS
:
:      THE PSW TESTS ARE USED TO VERIFY THAT VARIOUS DATA
: PATTERNS CAN BE SUCCESSFULLY HELD IN THE PSW AND THAT THE
: PSW ADDRESSING LOGIC IS FUNCTIONING. MOVE AND COMPARE INSTRUCTIONS
: ARE USED TO TEST THAT THE PSW CAN HOLD VARIOUS DATA PATTERNS.
: EACH DATA PATTERN IS MOVED AND TESTED IN A SMALL LOOP CONVENIENT FOR
: SCOPING.
:
:      THE PSW REGISTER IS TESTED, THE CC INPUTS ARE TESTED
: LATER IN THE MICROCODE TESTS. SETTING OF THE T-BIT BY THE
: TEST PATTERNS IS PURPOSELY AVOIDED. TESTING OF THE
: T-BIT TRAP CIRCUITRY IS LEFT FOR THE TRAP TEST.
:
:*****
:TEST 42      TEST IF PSW WILL HOLD ZEROES
:*****
TS42:
      MOV      #STBOT,R6                 ;SET PSW TO ZERO
      MOV      #0,@#PS                   ;SUCCESSFUL
      TST     @#PS
      BEQ     TS43
      EMT                                ;PSW NOT 0

:*****
:TEST 43      TEST IF PSW WILL HOLD ONES AND ZEROES
:*****
TS43:
      MOV      #252,@#PS                 ;MOVE ALT. ONES AND ZEROES TO PSW
      CMP     @#PS,#252                  ;SUCCESSFUL?
      BEQ     TS44
      EMT                                ;PSW NOT 252

:*****
:TEST 44      TEST IF PSW (EXCEPT T-BIT) WILL HOLD ZEROES AND ONES
:*****
TS44:

```

1482 004172 012737 000105 177776
1483 004200 023727 177776 000105
1484 004206 001401
1485 004210 104000
1486
1487
1488
1489
1490 004212
1491 004212 012737 000357 177776
1492 004220 023727 177776 000357
1493 004226 001401
1494 004230 104000
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529 004232
1530 004232 005000
1531 004234 001401
1532 004236 104000
1533 004240 005200
1534 004242 005100
1535 004244 005200
1536 004246 100401
1537 004250 104000

```
MOV #105,@#PS ;MOVE ALT. ONES AND ZEROES TO PSW
CMP @#PS,#105 ;SUCCESSFUL?
BEQ TS45
EMT ;PSW NOT 105

:*****
:TEST 45 TEST IF PSW (EXCEPT T-BIT) WILL HOLD ALL ONES
:*****
TS45:
MOV #357,@#PS ;MOVE ONES TO PSW
CMP @#PS,#357 ;SUCCESSFUL
BEQ TS46
EMT ;PSW NOT 357

:*****
:SBTTL MICROCODE TESTS
:
: THE TEST EXERCISES BRANCHES IN THE MICROCODE BY
: TESTING AT LEAST ONE INSTRUCTION FROM EVERY CLASS OF INSTRUCTION IN
: ALL POSSIBLE MODES. FOR EXAMPLE, TO TEST THE SINGLE OPERAND INSTRUCTIONS,
: AT LEAST ONE SINGLE OPERAND INSTRUCTION IS VERIFIED IN ALL UNIQUE
: ADDRESSING MODES. BYTE MODES ARE ALSO TESTED. AS EACH NEW
: MODE IS INTRODUCED THE SAME INSTRUCTION IS TRIED AND TESTED IN
: A SMALL LOOP CONVENIENT FOR SCOPING. THE TEST IS SET UP USING
: ONLY INSTRUCTIONS AND ADDRESSING MODES WHICH HAVE BEEN PREVIOUSLY
: VERIFIED.
: IF THESE TESTS FAIL, CHECK THE RESULTS FOR A CLUE TO THE
: FAULT.
:*****

:*****
: THE CLR INSTRUCTION IS USED TO INTRODUCE EACH ADDRESSING
: MODE WITH THE SINGLE OPERAND INSTRUCTION. FOLLOWING THE SEQUENCE CHECK,
: THE CLR INSTRUCTION IS EXECUTED AND A BRANCH TEST IS EXECUTED WHICH
: CHECKS THAT THE Z-BIT WAS PROPERLY SET. THIS TEST CAN CHECK IR DECODE
: AND MICROCODE FOR SOP INSTRUCTIONS WITH MODE 0. FOLLOWING THIS TEST
: SEVERAL OTHER SOP INSTRUCTIONS ARE INTRODUCED WITH MODE 0. THESE
: INSTRUCTIONS MAINPULATE DATA AND SERVE TO CHECK THE DATA RESULTS
: OF THE SOP INSTRUCTIONS IN THIS TEST. THE DATA IN THIS TEST IS
: OPERATED ON BY EACH INSTRUCTION WITHOUT REINITIALIZING.
:*****
:TEST 46 TEST MODE 0 USING SOP INST.
:*****
TS46:
CLR R0 ;TRY THE CLEAR INST.
BEQ SOPOA
EMT ;CLR DID NOT SET Z-BIT
SOP0A: INC R0 ;TRY THE INCREMENT INST.
COM R0 ;TRY COMPLEMENT
INC R0
BMI SOPOB
EMT ;NEGATE DID NOT SET N-BIT
```

1538 004252 005100
1539 004254 001401
1540 004256 104000
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555 004260
1556 004260 005000
1557 004262 005300
1558 004264 100401
1559 004266 104000
1560 004270 000261
1561 004272 005500
1562 004274 001007
1563 004276 000261
1564 004300 005600
1565 004302 100004
1566 004304 005100
1567 004306 005200
1568 004310 005300
1569 004312 001401
1570 004314
1571 004314 104000
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582 004316
1583 004316 105000
1584 004320 001401
1585 004322 104000
1586 004324 105100
1587 004326 100002
1588 004330 105200
1589 004332 001401
1590 004334
1591 004334 104000
1592
1593

SOP0B: COM R0 ;TRY COMPLEMENT INST.
BEQ TS47
EMT ;CUMMULATIVE RESULT OF CLR,INC,NEG AND COM INSTS. FAILED

THIS TEST INTRODUCES THE REMAINING SOP INSTRUCTIONS AND TESTS
THEM IN MODE 0. THE PURPOSE IS TO PROVIDE A BASELINE OF
INSTRUCTIONS FOR USE IN THE SUBSEQUENT TESTS. SINCE THE MICROCODE FOR
THESE INSTRUCTIONS IS IDENTICAL TO THAT ALREADY TESTED, ANY TROUBLE
SHOOTING EFFORTS SHOULD BE AIMED AT THE ACTUAL IR DECODE AND ALU
FUNCTIONING.

TEST 47 TEST REMAINDER OF SOP INSTS IN MODE 0

TS47:

CLR R0 ;INITIALIZE
DEC R0 ;TRY DECREMENT INST.
BMI SOPOC
EMT ;N-BIT NOT SET ON DEC
SOP0C: SEC ;INITIALIZE CARRY
ADC R0 ;TRY ADD CARRY INST
BNE SOPOD
SEC ;INITIALIZE CARRY
SBC R0 ;TRY SUBTRACT-CARRY INST
BPL SOPOD
COM R0
INC R0
DEC R0
SOP0D: BEQ TS50
EMT ; CUMMULATIVE RESULT OF ADC,SBC,COM,INC AND DEC INSTS. F

THIS TEST INTRODUCES THE BYTE CONTROL LOGIC OF THE PROCESSOR.
THE MODE 0 BYTE MICROCODE IS TESTED. THE METHOD AND SEQUENCE
OF TESTING IS THE SAME AS THAT USED IN THE SOP MODE 0 TESTS.

TEST 50 TEST MODE 0 EVEN BYTE USING SOP INST

TS50:

CLRB R0 ;TRY CLEARING EVEN BYTE OF REGISTER
BEQ SOPBOA
EMT ;CLRB DID NOT SET Z-BIT
SOPBOA: COMB R0 ;TRY SETTING EVEN BYTE OF REGISTER
BPL SOPBOB
INCB R0 ;TRY INCREMENTING EVEN BYTE OF REGISTER>>
BEQ TS51
SOPBOB: EMT ;TEST CUMMULATIVE RESULT OF ABOVE BYTE INST.

1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604 004336
1605 004336 005000
1606 004340 005010
1607 004342 001401
1608 004344 104000
1609 004346 005310
1610 004350 100003
1611 004352 000261
1612 004354 005510
1613 004356 001401
1614 004360
1615 004360 104000
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627 004362
1628 004362 005000
1629 004364 005010
1630 004366 005110
1631 004370 105010
1632 004372 001401
1633 004374 104000
1634 004376 005210
1635 004400 100005
1636 004402 105110
1637 004404 105210
1638 004406 100002
1639 004410 105210
1640 004412 001401
1641 004414
1642 004414 104000
1643
1644
1645
1646
1647
1648
1649

: THIS TEST USES THE CLR INSTRUCTION TO INTRODUCE AND TEST
: SINGLE OPERAND MODE 1 INSTRUCTIONS. AGAIN, THE CLR INSTRUCTION
: IS USED TO INTRODUCE THE MICROCODE AND TO TEST THAT THE PROPER
: CONDITION CODES ARE SET. OTHER SOP INSTRUCTIONS ARE USED TO MANIPULATE
: COMMON DATA TO VERIFY THAT THE CORRECT DATA IS PRODUCED.

: TEST 51 TEST MODE 1 USING SOP INST.

TS51:
CLR R0 ;INITIALIZE R0
CLR (R0) ;TRY CLEAR INST W/MODE 1
BEQ SOP1A
EMT ;CLR DID NOT SET Z-BIT
SOP1A: DEC (R0) ;TRY DECREMENT INST W/MODE 1
BPL SOP1B
SEC ;INITIALIZE CARRY
ADC (R0) ;TRY ADD-CARRY W/MODE 1
BEQ TS52
SOP1B: EMT ;TEST CUMMULATIVE RESULT OF ABOVE INST

: THIS TEST VERIFIES THE BYTE INSTRUCTION MICROCODE FOR MODE 1
: SINGLE OPERAND INSTRUCTIONS.
: THIS IS THE FIRST PLACE THE SIGN EXTEND LOGIC IS EXERCISED
: AND VERIFIED.

: TEST 52 TEST MODE 1 EVEN BYTE USING SOP INST

TS52:
CLR R0 ;INITIALIZE R0
CLR (R0) ;INITIALIZE LOC. 0
COM (R0)
CLRB (R0) ;TRY TO CLEAR BYTE 0
BEQ SOPB1A
EM1 ;CLRB DID NOT SET Z-BIT
SOPB1A: INC (R0) ;INCREMENT TO TEST WORD
BPL SOPB1B
COMB (R0) ;COMPLEMENT: ODD BYTE = 376
INCB (R0) ;INC: ODD BYTE = 377
BPL SOPB1B
INCB (R0) ;INCREMENT ODD BYTE=0
BEQ TS53
SOPB1B: EMT ;CHECK CUMMULATIVE RESULT OF ABOVE INST

: THIS TEST VERIFIES THAT SINGLE OPERAND BYTE INSTRUCTIONS WILL
: FUNCTION CORRECTLY FOR ODD BYTES.
: THIS IS THE FIRST TIME THAT ADDRESS LINE 0 HAS BEEN

1650
1651
1652
1653
1654
1655
1656
1657 004416
1658 004416 005000
1659 004420 005010
1660 004422 005110
1661 004424 005200
1662 004426 105010
1663 004430 001401
1664 004432 104000
1665 004434 005300
1666 004436 005210
1667 004440 005200
1668 004442 105110
1669 004444 105210
1670 004446 100002
1671 004450 105210
1672 004452 001401
1673 004454
1674 004454 104000
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689 004456
1690 004456 005000
1691 004460 105100
1692 004462 005200
1693 004464 005010
1694 004466 005110
1695 004470 005020
1696 004472 001401
1697 004474 104000
1698 004476 005300
1699 004500 005300
1700 004502 005120
1701 004504 100004
1702 004506 005300
1703 004510 005300
1704 004512 005220
1705 004514 001401

:EXERCISED. CHECKS ARE MADE THAT THE PROPER BYTE IS MODIFIED AND
:THE CONDITION CODES ARE CHECKED. IT IS ALSO VERIFIED THAT THE UNADDRESSED
:BYTE IS NOT ALTERED BY THE INSTRUCTION.

:TEST 53 TEST MODE 1 ODD BYTE USING SOP INST

TS53:

```
      CLR      R0          ;INITIALIZE R0
      CLR      (R0)       ;INITIALIZE LOC. 0
      COM      (R0)
      INC      R0          ;R0=ODD BYTE
      CLRB     (R0)       ;TRY TO CLEAR BYTE 1
      BEQ      SOPB1C
      EMT
SOPB1C: DEC      R0          ;CLRB DID NOT SET Z-BIT
      INC      (R0)       ;R0=WORD ADDR.
      INC      R0          ;INCREMENT TO TEST WORD
      INC      R0          ;R0=ODD BYTE
      COMB     (R0)       ;TRY TO COMPLEMENT BYTE 1
      INCB     (R0)
      BPL      SOPB1D
      INCB     (R0)       ;TRY TO INCREMENT BYTE 1
      BEQ      TS54
SOPB1D: EMT          ;TEST CUMMULATIVE RESULT OF ABOVE INST.
```

: THIS TEST VERIFIES MODE 2 SINGLE-OPERAND INSTRUCTIONS. PREVIOUSLY
:TESTED INSTRUCTIONS ARE USED TO SET A POINTER IN R0 TO LOC. 400.
:LOC. 400 IS INITIALIZED TO -1 BEFORE A CLR MODE 2 IS EXECUTED.
: THEN R0 IS DECREMENTED BY TWO TO AGAIN POINT TO 400 BEFORE EACH
:OF SEVERAL MODE 2 INSTRUCTIONS ARE USED TO VERIFY THE DATA RESULTS OF
:THE TEST. THIS PROCEDURE ALSO VERIFIES THE PROPER INCREMENTING OF THE
:REGISTER.

:TEST 54 TEST MODE 2 USING SOP INST.

TS54:

```
      CLR      R0          ;SET R0=400
      COMB     R0
      INC      R0
      CLR      (R0)       ;CLEAR 400
      COM      (R0)       ;INITIALIZE: 400=-1
      CLR      (R0)+      ;TRY CLEARING WITH MODE 2
      BEQ      SOPZA
      EMT
SOPZA: DEC      R0          ;CLR INST DID NOT SET Z-BIT
      DEC      R0          ;RESET R0
      COM      (R0)+      ;TRY COMPLEMENTING WITH MODE 2
      BPL      SOP2B
      DEC      R0          ;RESET R0
      DEC      R0
      INC      (R0)+      ;TRY INCREMENTING WITH MODE 2
      BEQ      TS55
```


1706 004516
1707 004516 104000
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722 004520
1723 004520 005000
1724 004522 105100
1725 004524 005200
1726 004526 005010
1727 004530 005110
1728 004532 105020
1729 004534 001401
1730 004536 104000
1731 004540 005300
1732 004542 005210
1733 004544 105110
1734 004546 105220
1735 004550 100003
1736 004552 005300
1737 004554 105220
1738 004556 001401
1739 004560
1740 004560 104000
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750 004562
1751 004562 005000
1752 004564 105100
1753 004566 005200
1754 004570 005010
1755 004572 005110
1756 004574 005200
1757 004576 105020
1758 004600 001401
1759 004602 104000
1760 004604 005300
1761 004606 005300

```
SOP2B: EMT ;CHECK CUMMULATIVE RESULT OF ABOVE INST

:*****
: THIS TEST VERIFIES MODE 2 SINGLE OPERAND INSTRUCTIONS WHICH
: ADDRESS EVEN BYTES. R0 IS SET TO 400 AND USED TO INITIALIZE LOCATION
: 400 TO -1. CLRB INSTRUCTION IS THEN EXECUTED ON BYTE 400 WITH
: MODE 2.
: R0 IS THEN DECREMENTED BEFORE EACH OF SEVERAL MODE 2 INSTRUCTIONS
: WHICH ARE USED TO VERIFY THE DATA RESULTS OF THE TEST. THIS PROCEDURE ALSO
: VERIFIES THE PROPER INCREMENTING OF THE REGISTER.
:*****
: TEST 55 TEST MODE 2 EVEN BYTE USING SOP INST.
:*****
TS55: CLR R0 ;SET R0=400
      COMB R0
      INC R0
      CLR (R0) ;CLEAR 400
      COM (R0) ;INITIALIZE: 400=-1
      CLRB (R0)+ ;TRY TO CLEAT 400 W/MODE 2
      BEQ SOPB2A
SOPB2A: EMT ;CLR DID NOT SET Z-BIT
        DEC R0 ;RESULT R0=400
        INC (R0) ;INC 400 TO TEST WORD
        COMB (R0)
        INCB (R0)+ ;TRY TO INC EVEN BYTE
        BPL SOPB2B
        DEC R0 ;RESET R0=400
        INCB (R0)+ ;TRY INCREMENT OF EVEN BYTE
        BEQ TS56
SOPB2B: EMT ;TEST CUMMULATIVE RESULT OF ABOVE INST.

:*****
: THIS TEST FOLLOWS THE SAME PROCEDURE DESCRIBED IN THE PREVIOUS
: TEST. HERE, THE BYTE INSTRUCTION IS USED TO ADDRESS AN ODD BYTE.
:*****
: TEST 56 TEST MODE 2 ODD BYTE USING SOP INST.
:*****
TS56: CLR R0 ;SET R0=400
      COMB R0
      INC R0
      CLR (R0) ;CLEAR LOC 400
      COM (R0) ;INITIALIZE: 400=-1
      INC R0 ;R0=ODD BYTE
      CLRB (R0)+ ;TRY TO CLEAR ODD BYTE
      BEQ SOPB2C
SOPB2C: EMT ;CLRB DID NOT SET Z-BIT
        DEC R0 ;R0=WORD ADDR.
        DEC R0
```

1762 004610 005220
1763 004612 005300
1764 004614 105110
1765 004616 105220
1766 004620 100033
1767 004622 005300
1768 004624 105220
1769 004626 001401
1770 004630
1771 004630 104000
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781 004632
1782 004632 005000
1783 004634 005200
1784 004636 005400
1785 004640 100003
1786 004642 001402
1787 004644 102401
1788 004646 103401
1789 004650
1790 004650 104000
1791
1792 004652 005200
1793 004654 001401
1794 004656 104000
1795
1796 004660 105100
1797 004662 105400
1798 004664 100403
1799 004666 001402
1800 004670 102401
1801 004672 103401
1802 004674
1803 004674 104000
1804 004676 005300
1805 004700 001401
1806 004702 104000
1807
1808
1809
1810 004704
1811 004704 005000
1812 004706 005010
1813 004710 005210
1814 004712 005410
1815 004714 100003
1816 004716 001402
1817 004720 102401

```
INC (R0)+ ;INCREMENT WORD
DEC R0 ;POINT TO ODD BYTE
COMB (R0) ;COMPLEMENT ODD BYTE
INCB (R0)+ ;TRY TO INCREMENT ODD BYTE
BPL SOPB2D
DEC R0 ;RESET R0 TO ODD BYTE
INCB (R0)+ ;TRY TO INCREMENT ODD BYTE
BEQ TS57

SOPB2D: EMT ;TEST CUMMULATIVE RESULT OF ABOVE INST.

:*****
: THESE TESTS CHECK THE NEGATE INSTRUCTION IN ALL MODES. PREVIOUSLY
: TESTED SINGLE-OPERAND INSTRUCTIONS ARE USED TO TEST THE NEGATE INSTRUCTION.
:*****
:TEST 57 TEST MODE 0 USING NEGATE INSTRUCTION
:*****
TS57: CLR R0 ;SET R0=0
INC RC ; R0=1
NEG R0 ;TRY NEGATE MODE 0: R0=-1
BPL NEG00 ;CC=1001?
BEQ NEG00
BVS NEG00
BCS NEG01

NEG00: EMT ;NEGATE DID NOT SET CC'S CORRECTLY

NEG01: INC R0 ;TEST DATA RESULT
BEQ NEG02
EMT ;DATA RESULT OF NEGATE INCORRECT

NEG02: COMB R0 ;R0=377
NEGB R0 ;R0=1
BMI NEG03 ;CC=0001?
BEQ NEG03
BVS NEG03
BCS NEG04

NEG03: EMT ;NEGB DID NOT SET CC'S CORRECTLY

NEG04: DEC R0 ;TEST DATA RESULT
BEQ TS60
EMT ;DATA RESULT OF NEGB INCORRECT

:*****
:TEST 60 TEST MODE 1 USING NEGATE INST.
:*****
TS60: CLR R0 ;POINT TO LOC. 0
CLR (R0) ;CLEAR LOC. 0
INC (R0) ;LOC. 0=1
NEG (R0) ;TRY NEG. LOC. 0=-1
BPL NEG10 ;CC=1001
BEQ NEG10
BVS NEG10
```

1818 004722 103401
1819 004724
1820 004724 104000
1821
1822 004726 005237 000000
1823 004732 001401
1824 004734 104000
1825 004736 105110
1826 004740 105410
1827 004742 100403
1828 004744 001402
1829 004746 102401
1830 004750 103401
1831 004752
1832 004752 104000
1833 004754 005337 000000
1834 004760 001401
1835 004762 104000
1836
1837
1838
1839 004764
1840 004764 005000
1841 004766 005010
1842 004770 005210
1843 004772 005420
1844 004774 100003
1845 004776 001402
1846 005000 102401
1847 005002 103401
1848 005004
1849 005004 104000
1850 005006 105300
1851 005010 105300
1852 005012 105420
1853 005014 105420
1854 005016 105340
1855 005020 005300
1856 005022 001401
1857 005024 104000
1858 005026 005337 000000
1859 005032 001401
1860 005034 104000
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873

BCS NEG11
NEG10: EMT ;NEGATE DID NOT SET CC'S CORRECTLY
NEG11: INC @#0 ;TEST DATA RESULT
BEQ NEG12
EMT ;DATA RESULT OF NEGATE INCORRECT
NEG12: COMB (R0) ;LOC. 0=377
NEGB (R0) ;TRY NEGB LOC. 0=1
BMI NEG13 ;CC=0001?
BEQ NEG13
BVS NEG13
BCS NEG14
NEG13: EMT ;NEGB DID NOT SET CC'S CORRECTLY
NEG14: DEC @#0 ;TEST DATA RESULT
BEQ TS61
EMT ;DATA RESULT OF NEGB INCORRECT
:*****
:TEST 61 TEST MODE 2 USING NEGATE INSTRUCTION
:*****
TS61:
CLR R0 ;POINT TO LOC. 0
CLR (R0) ;CLEAR LOC. 0
INC (R0) ;LOC. 0=1
NEG (R0)+ ;TRY NEG.: LOC. 0=-i
BPL NEG20 ;CC=1001?
BEQ NEG20
BVS NEG20
BCS NEG21
NEG20: EMT ;NEGATE DID NOT SET CC'S CORRECTLY
NEG21: DECB R0 ;R0=LOC. 0
DECB R0
NEGB (R0)+ ;BYTE 0=1 R0=1
NEGB (R0)+ ;BYTE 1=1 R0=2
DECB -(R0) ;R0=1 LOC. 0=01
DEC R0 ;R0=0
BEQ NEG22
EMT ;REGISTER NOT INCREMENTED CORRECTLY
NEG22: DEC @#0 ;LOC. 0=0
BEQ TS62
EMT ;NEG BYTE INSTRUCTIONS FAILED
:*****
: THIS TEST VERIFIES MODE 3 SINGLE OPERAND INSTRUCTIONS. IT
: USES LOCATION 0 AS ITS TARGET DATA. A TABLE LOCATED AT LOC. 400
: THRU 402 IS USED TO SUPPLY THE ADDRESS OF LOCATION 0 TO THE
: INSTRUCTIONS UNDER TEST.
: RC IS SET TO 400, THE START OF THE ADDRESS TABLE, AND A CLR
: INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR LOC. 0. THEN R0
: IS DECREMENTED BY TWO AND TWO OTHER MODE 3 INSTRUCTIONS OPERATE ON
: LOC. 0 TO VERIFY THE DATA RESULTS OF THE TEST. THE PROPER INCREMENTING
: OF THE REGISTER IS ALSO VERIFIED IN THIS MANNER.
: IF A FAILURE IS DETECTED BE SURE TO VERIFY THAT THE TABLE

1874
1875
1876
1877
1878
1879 005036
1880 005036 005000
1881 005040 105100
1882 005042 005200
1883 005044 005010
1884 005046 005030
1885 005050 001401
1886 005052 104000
1887 005054 005300
1888 005056 005300
1889 005060 005130
1890 005062 100002
1891 005064 005230
1892 005066 001401
1893 005070
1894 005070 104000
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911 005072
1912 005072 005004
1913 005074 105104
1914 005076 005204
1915 005100 005000
1916 005102 005010
1917 005104 005110
1918 005106 105034
1919 005110 001401
1920 005112 104000
1921 005114 005304
1922 005116 005304
1923 005120 005234
1924 005122 100006
1925 005124 105434
1926 005126 100004
1927 005130 005304
1928 005132 005304
1929 005134 105234

;(LOC. 400-402) HAS THE PROPER VALUES (0).
:*****
:TEST 62 TEST MODE 3 USING SOP INST.
:*****
TS62:
CLR R0 ;SET R0=400
COMB R0
INC R0
CLR (R0) ;CLEAR LOC 400
CLR @(R0)+ ;TRY TO CLEAR LOC 0 USING MODE 3 ;R0=402
BEQ SOP3A
EMT ;CLR DID NOT SET Z-BIT
SOP3A: DEC R0 ;RESET R0=400
DEC R0
COM @(R0)+ ;TRY TO COMPLEMENT LOC 0 OF MODE 3 ;R0=402
BPL SOP3B
INC @(R0)+ ;TRY TO INCREMENT LOC 0 W/MODE 3 ;R0=404
BEQ TS63
SOP3B: EMT ;CUMULATIVE RESULT OF ABOVE INST FAILED
:*****
: THIS TEST VERIFIES MODE 3 SINGLE OPERAND BYTE INSTRUCTIONS
: WHICH ADDRESS EVEN BYTES. AGAIN, THE TARGET LOCATION 0 IS USED
: AND THE SAME TABLE AT 400 IS EMPLOYED.
: AFTER POINTING R4 TO THE TABLE (400) AND SETTING LOCATION
: 0 TO -1, A CLRB INSTRUCTION IS USED TO CLEAR BYTE 0.
: SEVERAL OTHER MODE 3 INSTRUCTIONS ARE THEN USED WITH THE TABLE
: TO VERIFY THE DATA RESULTS AND THE PROPER INCREMENTING OF THE REGISTER.
: IF A FAILURE IS DETECTED, BE SURE THAT THE TABLE (LOCATION 400-402) HAS
: THE PROPER VALUES (0).
:*****
:TEST 63 TEST MODE 3 EVEN BYTE USING SOP INST.
:*****
TS63:
CLR R4 ;SET R4=400
COMB R4
INC R4
CLR R0 ;INITIALIZE LOC. 0=-1
CLF (R0)
COM (R0) ;LOC. 0=-1
CLRB @(R4)+ ;TRY TO CLEAR EVEN BYTE ;LOC. 0=177400 R4=402
BEQ SOPB3A
EMT ;CLRB DID NOT SET Z-BIT
SOPB3A: DEC R4 ;RESET POINTER R4=400
DEC R4
INC @(R4)+ ;TRY INCREMENTING WORD LOC.0=177401 R4=402
BPL SOPB3B
NEGB @(R4)+ ;TRY TO NEGATE EVEN BYTE ;LOC.0=-1 R4=404
BPL SOPB3B
DEC R4 ;R4=402
DEC R4
INCB @(R4)+ ;TRY TO INCREMENT EVEN BYTE ;LOC. 0=17400

1930 005136 001401
1931 005140
1932 005140 104000
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946

BEQ TS64
SOPB3B:
EMT

;CUMMULATIVE RESULT OF ABOVE INST FAILED

:*****
: THIS TEST VERIFIES MODE 3 SINGLE OPERAND BYTE INSTRUCTIONS
: WHICH ADDRESS ODD BYTES. THE TARGET IS BYTE 1. A TABLE AT
: LOC. 400-406 IS USED. R0 SERVES AS THE TABLE POINTER.
: R0 IS INITIALIZED TO 400. LOC. 0 IS SET TO -1 USING THE
: FIRST TWO TABLE ENTRIES. A CLRB MODE 3 IS EXECUTED ON BYTE 1 USING
: TABLE ADDRESS AT 404. R0 IS DECREMENTED TO 402 AND SEVERAL SOP
: MODE 3 INSTRUCTIONS ARE USED TO VERIFY DATA RESULTS AND PROPER
: REGISTER INCREMENTING.
: THE TABLE (400-406) SHOULD CONTAIN 0,0,1,1 BEFORE AND
: AFTER THE TEST IS RUN.
:

1947
1948
1949
1950 005142
1951 005142 005000
1952 005144 105100
1953 005146 005200
1954 005150 005030
1955 005152 005130
1956 005154 105030
1957 005156 001401
1958 005160 104000
1959 005162 005300
1960 005164 005300
1961 005166 005300
1962

```
*****  
;TEST 64 TEST MODE 3 ODD BYTE USING SOP INST.  
*****  
TS64:  
      CLR      R0          ;SET R0=400  
      COMB     R0  
      INC      R0  
      CLR      @(R0)+      ;INITIALIZE  
      COM      @(R0)+      ;LOC 0=-1 R0=404  
      CLRB     @(R0)+      ;TRY TO CLEAR ODD BYTE LOC. 0=377 R0=406  
      BEQ      SOPB3C  
      EMT  
SOPB3C: DEC      R0          ;CLRB DID NOT SET Z-BIT  
          DEC      R0          ;RESET R0=402  
          DEC      R0          ;POINT TO EVEN BYTE ADDR.
```