KDF11-B 11/23B CPU CLSTR DIAG AH-T039A-MC
CJKDJAO FICHE 1 OF 1

```
1                                       .REM_
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22                                                  IDENTIFICATION
23                                                  --------------
24
25
26
27                                      PRODUCT CODE:  AC-T038A-MC
28
29                                      PRODUCT NAME:   CJKDJA0 11/23B CPU CLSTR DIAG
30
31                                      PRODUCT DATE:  SEPT-81
32
33                                      MAINTAINER:  DIAGNOSTIC ENGINEERING
34
35
36
37
38                                  THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE  WITHOUT  NOTICE
39                                  AND  SHOULD  NOT  BE CONSTRUED AS A COMMITMENT BY DIGITAL CORPORATION.
40                                  DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS
41                                  THAT MAY APPEAR IN THIS DOCUMENT.
42
43                                  NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON
44                                  EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.
45
46                                  COPYRIGHT (C):  1982 BY DIGITAL EQUIPMENT CORPORATION
47
48                                  THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:
49
50                                  DIGITAL         PDP          UNIBUS          MASSBUS
51                                  DEC             DECUS        DECTAPE         DEC/X11
52
53
```

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 PROGESS 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.

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

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

```
                        7       1 - TEST THE UPPER 5 ADDRESS BITS FOR TIME OUT
                                0 - INHIBIT TESTING THE UPPER 5 ADRS BITS

                        6       1 - TEST USING A Q BUS EXERCISER (QBE OR Q22BE)
                                0 - INHIBIT TESTS THAT USE A Q BUS EXERCISER

                        5       0 - PROGRAM RESERVED -- PROGRAM WILL CLEAR IF CIS CHIP SET NOT ON BOARD
                                1 - PROGRAM RESERVED -- PROGRAM WILL SET IF CIS CHIP SET IS ON BOARD
                        4       0 - TEST SLU2 OF 11/23-B
                                1 - INHIBIT TESTING OF SLU2

                        3       0 - TEST LTC OF 11/23-B
                                1 - INHIBIT TESTING OF LTC

                        2       0 - TEST SLU1 OF 11/23-B
                                1 - INHIBIT TESTING OF SLU1

                        1       1 - TEST FPP INSTRUCTION SET
                                0 - INHIBIT TESTING OF FPP

                        0       0 - TEST MEMORY MANAGEMENT UNIT
                                1 - INHIBIT TESTING OF MMEMORY MANAGEMENT UNIT


2.3  EXECUTION TIMES

     FIRST PASS RUNTIME (WORST CASE).....45 SEC
     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
```

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 TESTS	000007
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:

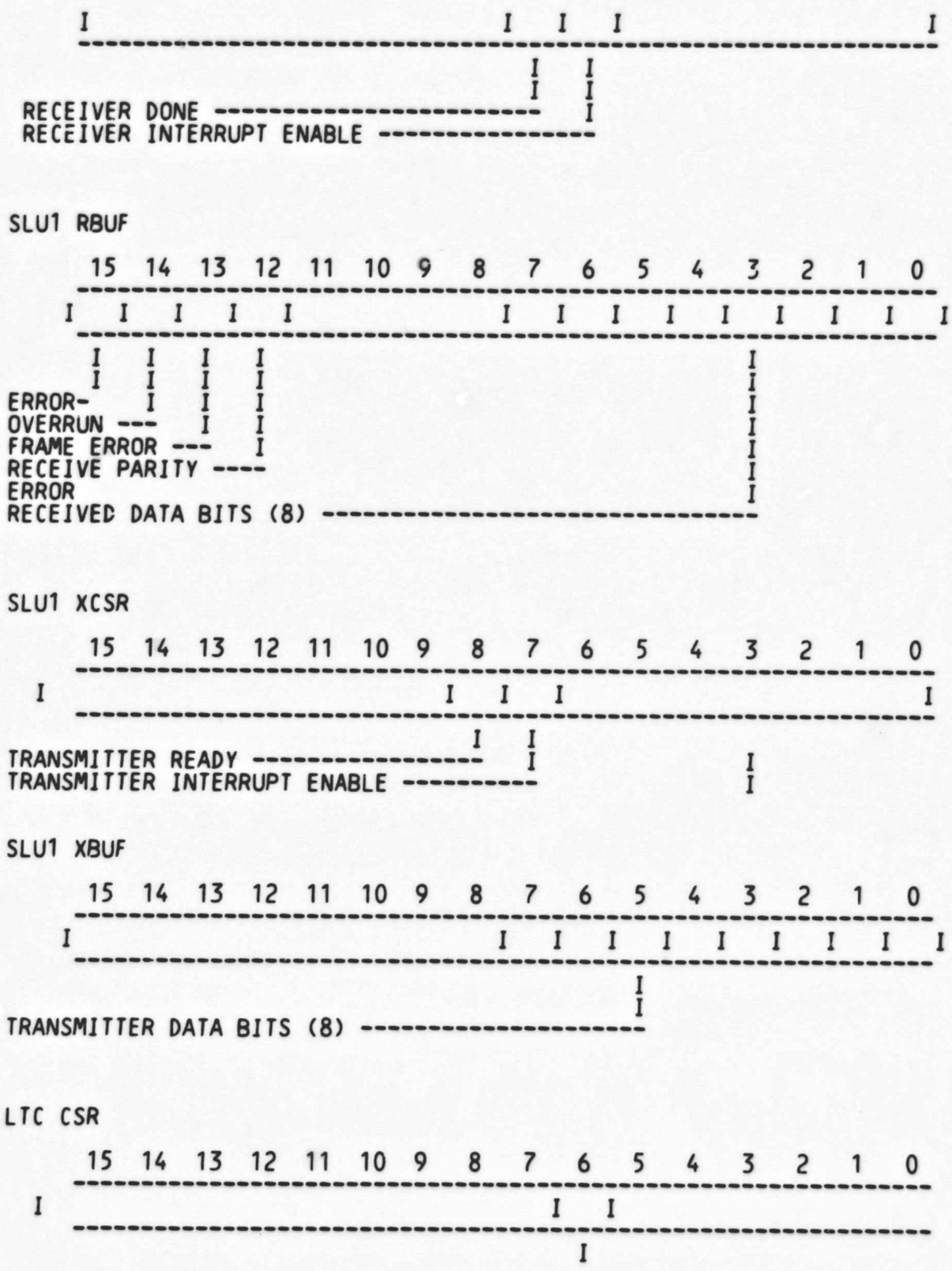
```
CJKDJA END OF PASS #XXX
```

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
-----------------------------------------------------------------
```


RECEIVER DONE
RECEIVER INTERRUPT ENABLE
SLU1 RBUF
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ERROR-
OVERRUN ---
FRAME ERROR ---
RECEIVE PARITY ----
ERROR
RECEIVED DATA BITS (8)
SLU1 XCSR
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
TRANSMITTER READY
TRANSMITTER INTERRUPT ENABLE
SLU1 XBUF
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
TRANSMITTER DATA BITS (8)
LTC CSR
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

```
390
391                                   LINE CLOCK
392                                   INTERRUPT ENABLE ------------------------
393
394
395
396                                   SLU2 RCSR
397
398                                         15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
399                                        ---------------------------------------------------------------
400                                       I                           I   I   I                           I
401                                        ---------------------------------------------------------------
402                                                                   I   I
403                                                                   I   I
404                                   RECEIVER DONE ------------------    I
405                                   INTERRUPT ENABLE ------------------
406
407
408
409                                   SLU2 RBUF
410
411                                         15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
412                                        ---------------------------------------------------------------
413                                       I   I   I   I   I               I   I   I   I   I   I   I   I   I
414                                        ---------------------------------------------------------------
415                                         I   I   I   I                                   I
416                                         I   I   I   I                                   I
417                                   ERROR -   I   I   I                                   I
418                                   OVERRUN ---   I   I                                   I
419                                   FRAME ERROR ---   I                                   I
420                                   RECEIVER PARITY ---                                   I
421                                   ERROR                                                 I
422                                   RECEIVER DATA BITS (8) ------------------------------
423
424
425                                   SLU2 XCSR
426
427                                         15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
428                                        ---------------------------------------------------------------
429                                                               I   I   I   I                       I   I
430                                        ---------------------------------------------------------------
431                                                                   I   I                           I
432                                                                   I   I                           I
433                                   TRANSMITTER DONE ----------------    I                           I
434                                   INTERRUPT ENABLE ------------------------                       I
435                                   BREAK -----------------------------------------------------------
436
437
438
439                                   SLU2 XBUF
440                                         15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
441                                        ---------------------------------------------------------------
442                                       I                       I   I   I   I   I   I   I   I   I   I   I
443                                        ---------------------------------------------------------------
444                                                                               I
445                                                                               I
```

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 DESCRIBTOR (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 000240
561 000007
562 000006
563 177776
564 177564
565 177566
566 140000
567 030000
568 000006
569 000006
570 000003
571 000001
572 000005
573 000002
574 000000
575 000003
576 000004
577 000004
578 000014
579 000030
580 000020
581 000034
582 177564
583 177560
584 177562
585 177566

6.2.10 BLAST SUBTEST

THIS TEST CHECKS THE ABILITY OF THE 11/23-B TO HANDLE SYSTEM INTERACTION. 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 TRANSMTTER 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
                FIRST=%5
                R2=%2
                HLT=HALT
                TRT=3
                ITRAP5=4
                RTRAP5=4                ;RESERVED INST AND ILLEGAL ADDRESSES
                RTRAP4=14               ;FOR TRACE TRAP
                RTRAP3=30               ;FOR EMULATOR TRAP
                RTRAP2=20               ;FOR IOT TRAP
                RTRAP1=34               ;FOR TRAP INST
                TTCSR=177564
                TRCSR=177560
                TKB=177562
                TPB=177566
```

```
                    000240                  BELL=240
                    000240                  NOP=240
                    177776                  STATUS=177776
                    000077                  TRAPA=77
                    000010                  RTRAP=10
                    004700                  ILLA=004700
                    000100                  ILLB=100

                    177520                  PCR=177520
                    177524                  LSREG=177524
                    177522                  RWREG=177522

                    177776                  CC=177776
                    001000                  KERSTK=STBOT
                    000600                  USESTK=STBOT-200
                    104377                  EMTA=104377
                    104777                  TRAPC=104777
                    000001                  APTENV=1
                                    .SBTTL  ACT11 HOOKS

                                    ;*****************************************************************************************
                                    ;HOOKS REQUIRED BY ACT11
                    000400                  $SVPC=.                 ;SAVE PC
                    000046                  .=46
000046              133212                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                    000052                  .=52
000052              000000                  .WORD   0               ;;2)SET LOC.52 TO ZERO
                    000400                  .=$SVPC                 ;; RESTORE PC
                    001000                  .=1000
                                    .SBTTL  APT MAILBOX-ETABLE

                                    ;*****************************************************************************************
                                    .EVEN
001000                              $MAIL:                          ;;APT MAILBOX
001000              000000          $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001002              000000          $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001004              000000          $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001006              000000          $PASS:  .WORD   APASS           ;;PASS COUNT
001010              000000          $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001012              000000          $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
001014              000000          $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
001016              000000          $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
001020                              $ETABLE:                        ;;APT ENVIRONMENT TABLE
001020                  000         $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
001021                  000         $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
001022              000000          $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
001024              000000          $USWR:  .WORD   AUSWR           ;;USER SWITCHES
001026              000000          $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
001030                              $ETEND:
                                    .MEXIT
```

```
                                .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.

001030                          $APTHD:
001030  000000                 $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001032  001000                 $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001034  000010                 $TSTM:  .WORD   10      ;;RUN TIM OF LONGEST TEST
001036  000025                 $PASTM: .WORD   25      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001040  000000                 $UNITM: .WORD   0       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001042  000014                         .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
                                ;*****************************************************************************
                                ;SOME POINTERS TO CPU TRAP HANDLERS
                                ;*****************************************************************************
000004                                  .=4
000004  021336                          TO4
000006  000000                          0
000010  021340                          TO10
000012  000000                          0
000014  021342                          TO14
000016  000000                          0
000020  021344                          TO20
000022  000000                          0
000030                                  .=30
000030  021346                          TO30
000032  000000                          0
000034  021350                          TO34
000036  000000                          0
000114                                  .=114
000114  021352                          TO114
000116  000000                          0
000244                                  .=244
000244  021354                          TO244
000246  000000                          0
000250  021356                          TO250
000252  000000                          0

000172                          .=172
000172  000000                 MTFLAG:         0       ;MULTI-TESTER ACTIVE BIT
000174  000000                 DISPREG:        0       ;SOFTWARE DISPLAY REGISTER
000176  000000                 SWREG:          0       ;SOFTWARE SWITCH REGISTER

                                ;*****************************************************************************
                                ;DATA TABLE FOR USE IN ADDRESSING MODE TESTS
                                ;*****************************************************************************
```

```
                000370                          .=370
000370  000000  000000  000000                  0,0,0,0,0,0
000376  000000  000000  000000
000404  000001  000001  177777                  1,1,-1
                        ;*****************************************************************************************
                        ;SET UP STARTING ADDRESS
                000400  .=1000
001000  000000          STBOT:  .WORD 0                         ;STACK POINTER
                        ;* ***************************************************************
                        
                        
                000510                          .=510                   ;Q22BE DEVICE VECTOR
                                                                        ;AREA
                        
                000200                          .=200
000200  000167  000774          JMP     START
000204  012706  001000          MOV     #STBOT,R6               ;SET STACK POINTER
000210  012702  001004          MOV     #$TESTN,R2              ;SET MAILBOX POINTER
000214  000137                  JMP     @(PC)+                  ;JUMP TO SUBTEST
000216  000000                  0                               ;ADDR. OF SUBTEST GOES HERE
                        
                001200          .=1200
                        .SBTTL **STARTING OF CPU TEST **
001200  032737  000001  001020  START:  BIT     #1,@#$ENV               ;UNDER APT
001206  001004                  BNE     CONTIN
001210  012700  133431          MOV     #STRMSG,R0              ;START MESSAGE
001214  004767  132110          JSR     PC,TYPE
001220  012737  000000  001006  CONTIN: MOV     #0,@#$PASS      ;CLEAR PASS COUNT
001226  012737  133264  000024  RESTRT: MOV     #PWRDN,@#24             ;SET UP FOR POWER FAIL
001234  012706  001000          MOV     #STBOT,R6               ;SET UP STACK
001240  012737  001270  000004          MOV     #1$,@#4                 ;SET UP FOR TIMEOUT IF NO MULTI TESTER
001246  012737  000340  000006          MOV     #340,@#6
001254  012737  000002  164000          MOV     #2,@#164000             ;SET BIT1 FOR MULTI TESTER
001262  012737  000001  000172          MOV     #1,@#MTFLAG             ;SET FLAG TO INDICATE MULTI-TESTER
001270  012737  021336  000004  1$:     MOV     #T04,@#4                ;SET TRAP CATCHER
001276  012737  000000  000006          MOV     #0,@#6                  ;SET HALT BACK IN LOCATION 6
001304  012706  001000          MOV     #STBOT,R6               ;INITIALIZE STACK POINTER
001310  012737  000001  001004          MOV     #1,@#$TESTN             ;SET TEST NUMBER TO 1
001316  012737  000000  001002          MOV     #0,@#$FATAL             ;CLEAR ERROR INDICATOR
001324  012737  000000  001000          MOV     #0,@#$MSGTY             ;CLEAR MESSAGE TYPE(FOR APT)
                        
001332  005067  176166          CLR     LSREG                   ;THIS WILL TURN ON THE 4 LEDS
                        
001336  105737  001020          TSTB    @#$ENV                  ;RUNNING ON APT
001342  001036                  BNE     TS1                     ;IF YES DO BRANCH SELF ON ERROR
001344  012737  000000  002164  MOV     #0,@#CPUHLT             ;IF NOT THEN PUT A HALT IN ON ERROR
001352  012737  000000  050470  MOV     #0,@#MMUHLT
001360  012737  000000  051404  MOV     #0,@#EXADHT
001366  012737  000000  053214  MOV     #0,@#Q22HLT
001374  012737  000000  124532  MOV     #0,@#FPHLT
001402  012737  000000  126674  MOV     #0,@#LTCHLT
001410  012737  000000  125754  MOV     #0,@#SL1HLT
001416  012737  000000  131540  MOV     #0,@#SL2HLT
001424  012737  000000  132504  MOV     #0,@#COMHLT
001432  012737  000000  054472  MOV     #0,@#BDVHLT
                        
```

```
                                ;*****************************************************************************************
                                ;
                                ;       THIS TEST EXECUTES EVERY POSSIBLE BRANCH WITH EVERY POSSIBLE
                                ;CONDITION CODE COMBINATION.
                                ;       THE ROUTINE USES TWO TABLES.  THE BRANCH TABLE HOLDS 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
                                ;*****************************************************************************************
001440                          TS1:
001440  012700  021242          SETUP:  MOV     #BRTAB,R0       ;INITIALIZE BRANCH TABLE POINTER
001444  012704  021300                  MOV     #YNTAB,R4       ;INITIALIZE YES/NO BRANCH MAP POINTER
001450  012767  000017  000130          MOV     #15.,BRCT       ;INITIALIZE BRANCH TABLE COUNT
001456  012067  000110          SETBR:  MOV     (R0)+,BRH       ;GET NEXT BRANCH INST.
001462  012401                          MOV     (R4)+,R1        ;GET NEXT BRANCH MAP
001464  012767  177777  000074          MOV     #-1,CC1         ;INITIALIZE CONDITION CODE VALUE
001472  012703  000020                  MOV     #16.,R3         ;INITIALIZE CONDITION CODE COUNT
001476  005267  000064          SETCC:  INC     CC1             ;SET FOR NEXT CC VALUE
001502  032701  100000                  BIT     #100000,R1      ;SEE IF SHOULD BR W/ THESE CC'S
001506  013705  177776                  MOV     @#177776,R5     ;SIMULATE A JNE
001512  042705  177773                  BIC     #177773,R5      ;       (JUMP NOT EQUAL)
001516  000165  001522                  JMP     .+4(R5)         ;       TO SET2BR
001522  000167  000020                  JMP     SET2BR
001526  012767  001610  000042          MOV     #CONT,NBR       ;SET TO CONTINUE IF NO BRANCH
001534  012767  001604  000040          MOV     #ER,YBR         ;SET TO REPORT ERROR IF BRANCH
001542  000167  000014                  JMP     AROUN           ;GO AROUND OPPOSITE CONDITION
001546  012767  001604  000022  SET2BR: MOV     #ER,NBR         ;SET TO REPORT ERROR IF NO BRANCH
001554  012767  001610  000020          MOV     #CONT,YBR       ;SET TO CONTINUE IF BRANCH
001562  006101                  AROUN:  ROL     R1              ;UPDATE BIT MAP
                                ;
001564  012737                          MOV     (PC)+,@(PC)+    ;SET CONDITION CODE
001566  000000          CC1:    0                               ;NEW CC VALUE GOES HERE
001570  177776                  177776
001572  000000          BRH:    0                               ;BRANCH INST. GOES HERE
001574  000137                  JMP     @(PC)+                  ;THIS JUMP IF NO BRANCH
001576  000000          NBR:    0                               ;WHERE TO GO IF NO BRANCH OCCURS
001600  000137                  JMP     @(PC)+                  ;THIS JUMP IF BRANCH OCCURS
001602  000000          YBR:    0                               ;WHERE TO GO IF BRANCH OCCURS
001604                  ER:
001604  000551                  BR      ERROR1          ;
```

```
000000
BRCT:   0
005303  CONT:   DEC     R3              ;CC'S DONE?
013705  177776          MOV     @#177776,R5     ;SIMULATE A JNE
042705  177773          BIC     #177773,R5      ;       (JUMP NOT EQUAL)
000165  001626          JMP     .+4(R5)         ;       TO SETCC
000167  177644          JMP     SETCC
005367  177750          DEC     BRCT            ;BR'S DONE?
013705  177776          MOV     @#177776,R5     ;SIMULATE A JNE
042705  177773          BIC     #177773,R5      ;       (JUMP NOT EQUAL)
000165  001652          JMP     .+4(R5)         ;       TO SETBR
000167  177600          JMP     SETBR
012700  000357          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.

                        ;*******************************************************************************
                        ;TEST 2 TEST TRAP OF RESERVED INSTRUCTION
                        ;*******************************************************************************
001662  TS2:
012706  001000          MOV     #STBOT,SP       ;INITIALIZE THE STACK POINTER
012767  001710  176114  MOV     #1$,RTRAP               ;SET UP NEW PC IN VECTOR
010067  176112          MOV     R0,RTRAP+2              ;SET UP NEW PSW IN VECTOR
005067  176072          CLR     STATUS          ;CLEAR PRESENT (OLD) STATUS
000077                  TRAPA                   ;DO TRAP
001706  8$:
000510                  BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
020067  176062  1$:     CMP     R0,STATUS       ;IS NEW STATUS CORRECT
001105                  BNE     ERROR1          ;NEW STATUS WRONG
020627  000774  2$:     CMP     SP,#STBOT-4     ;DID STACK DECREMENT CORRECTLY
001102                  BNE     ERROR1          ;STACK DID NOT DECREMENT CORRECTLY
022767  001706  177042  3$:     CMP     #8$,STBOT-4     ;WAS PROPER PC SAVED
001076                  BNE     ERROR1          ;PROPER PC WAS NOT SAVED
005767  177036  4$:     TST     STBOT-2         ;WAS OLD PSW SAVED
001073                  BNE     ERROR1          ;WRONG PSW SAVED
062706  000004  5$:     ADD     #4,SP           ;RESET STACK POINTER
012767  001770  176034          MOV     #6$,RTRAP               ;SET UP NEW PC IN VECTOR
005067  176032          CLR     RTRAP+2         ;SET UP NEW PSW IN VECTOR
010037  177776          MOV     R0,@#STATUS     ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
000077                  TRAPA                   ;DO TRAP
000460                  BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
005767  176002  6$:     TST     STATUS          ;IS NEW PSW CORRECT
001055                  BNE     ERROR1          ;NEW PSW WRONG
020067  176774  7$:     CMP     R0,STBOT-2      ;WAS OLD STATUS STORED
001052                  BNE     ERROR1          ;OLD STATUS WRONG
                        ;*******************************************************************************
                        ;TEST 3 TEST TRAP OF TRAP INSTRUCTION
                        ;*******************************************************************************
002004  TS3:
012706  001000          MOV     #STBOT,SP       ;INITIALIZE THE STACK POINTER
012767  002032  176016          MOV     #1$,RTRAP1              ;SET UP NEW PC IN VECTOR
010067  176014          MOV     R0,RTRAP1+2             ;SET UP NEW PSW IN VECTOR
005067  175750          CLR     STATUS          ;CLEAR PRESENT (OLD) STATUS
104400                  TRAP                    ;DO TRAP
002030  8$:
```

```
                        BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
002032 020067 175740    1$:     CMP     R0,STATUS       ;IS NEW STATUS CORRECT
002036 001034                   BNE     ERROR1          ;NEW STATUS WRONG
002040 020627 000774    2$:     CMP     SP,#STBOT-4     ;DID STACK DECREMENT CORRECTLY
002044 001031                   BNE     ERROR1          ;STACK DID NOT DECREMENT CORRECTLY
002046 022767 002030 176720 3$: CMP     #8$,STBOT-4     ;WAS PROPER PC SAVED
002054 001025                   BNE     ERROR1          ;PROPER PC WAS NOT SAVED
002056 005767 176714    4$:     TST     STBOT-2         ;WAS OLD PSW SAVED
002062 001022                   BNE     ERROR1          ;WRONG PSW SAVED
002064 062706 000004    5$:     ADD     #4,SP           ;RESET STACK POINTER
002070 012767 002112 175736     MOV     #6$,RTRAP1              ;SET UP NEW PC IN VECTOR
002076 005067 175734            CLR     RTRAP1+2                ;SET UP NEW PSW IN VECTOR
002102 010037 177776            MOV     R0,@#STATUS     ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
002106 104777                   TRAPC                   ;DO TRAP WITH LOWER BYTE ALL ONES
002110 000407                   BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
002112 005767 175660    6$:     TST     STATUS          ;IS NEW PSW CORRECT
002116 001004                   BNE     ERROR1          ;NEW PSW WRONG
002120 020067 176652    7$:     CMP     R0,STBOT-2      ;WAS OLD STATUS STORED
002124 001001                   BNE     ERROR1          ;OLD STATUS WRONG
002126 000434                   BR      CPUHLT+34       ;GET OVER ERROR CALL TO NEXT TEST IF NO ERROR
                        ;*******************************************************************************
                        ;THIS ERROR IS USED FOR THE ENTIRE CPU,TRAPS AND EIS PORTION OF
                        ;THIS TEST
                        ;*******************************************************************************
002130 012737 000001 001002 ERROR1: MOV #1,@#$FATAL                     ;SET UP FATAL ERROR NUMBER
002136 012767 000001 176634         MOV #1,$MSGTY                       ;SET FATAL ERROR FLAG
002144 032737 000001 001020         BIT #1,@#$ENV                       ;UNDER APT
002152 001004                       BNE CPUHLT                          ;YES, THEN DO NOT PRINT
002154 012700 002166                MOV #CPUMSG,R0
002160 004767 131144                JSR PC,TYPE                         ;TYPE MSG
002164 000777           CPUHLT: BR  .

002166 040506 046111 042105 CPUMSG: .ASCIZ  /FAILED DURING CPU TESTS/<12><15>
002174 042040 051125 047111
002202 020107 050103 020125
002210 042524 052123 005123
002216 000015

                        .EVEN

                        ;*******************************************************************************
                        ;TEST 4 TEST TRAP OF IOT INSTRUCTION
                        ;*******************************************************************************
002220                  TS4:
002220 012706 001000            MOV     #STBOT,SP       ;INITIALIZE THE STACK POINTER
002224 012767 002246 175566     MOV     #1$,RTRAP2              ;SET UP NEW PC IN VECTOR
002232 010067 175564            MOV     R0,RTRAP2+2             ;SET UP NEW PSW IN VECTOR
002236 005067 175534            CLR     STATUS          ;CLEAR PRESENT (OLD) STATUS
002242 000004                   IOT                     ;DO TRAP
002244                  8$:
002244 000731                   BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
002246 020067 175524    1$:     CMP     R0,STATUS       ;IS NEW STATUS CORRECT
002252 001326                   BNE     ERROR1          ;NEW STATUS WRONG
002254 020627 000774    2$:     CMP     SP,#STBOT-4     ;DID STACK DECREMENT CORRECTLY
002260 001323                   BNE     ERROR1          ;STACK DID NOT DECREMENT CORRECTLY
002262 022767 002244 176504 3$: CMP     #8$,STBOT-4     ;WAS PROPER PC SAVED
002270 001317                   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$,RTRAP2              ;SET UP NEW PC IN VECTOR
                                        CLR     RTRAP2+2                ;SET UP NEW PSW IN VECTOR
                                        MOV     R0,@#STATUS     ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
                                        IOT                     ;DO TRAP
                                        BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
                                6$:     TST     STATUS          ;IS NEW PSW CORRECT
                                        BNE     ERROR1          ;NEW PSW WRONG
                                7$:     CMP     R0,STBOT-2      ;WAS OLD STATUS STORED
                                        BNE     ERROR1          ;OLD STATUS WRONG
                                ;*******************************************************************************************
                                ;TEST 5 TEST TRAP OF EMT INSTRUCTION
                                ;*******************************************************************************************
                                TS5:
                                        MOV     #STBOT,SP       ;INITIALIZE THE STACK POINTER
                                        MOV     #1$,RTRAP3              ;SET UP NEW PC IN VECTOR
                                        MOV     R0,RTRAP3+2             ;SET UP NEW PSW IN VECTOR
                                        CLR     STATUS          ;CLEAR PRESENT (OLD) STATUS
                                        EMT                     ;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$:     ADD     #4,SP           ;RESET STACK POINTER
                                        MOV     #6$,RTRAP3              ;SET UP NEW PC IN VECTOR
                                        CLR     RTRAP3+2                ;SET UP NEW PSW IN VECTOR
                                        MOV     R0,@#STATUS     ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
                                        EMTA                    ;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     R0,STBOT-2      ;WAS OLD STATUS STORED
                                        BNE     ERROR1          ;OLD STATUS WRONG
                                        BR      ERROR2+2        ;WE MUST GET OVER ERROR CALL AT END OF THIS TEST
                                ;*******************************************************************************************
                                ;THIS ERROR IS NEEDED BECAUSE BRANCHES IN TRAP TESTS BEYOND HERE CAN NOT
                                ;REACH ERROR1.
                                ;*******************************************************************************************
                                ERROR2: BR      ERROR1


                                ;*******************************************************************************************
                                ;TEST 6 TEST TRAP OF TRACE-TRAP INSTRUCTION
                                ;*******************************************************************************************
                                TS6:
                                        MOV     #STBOT,SP       ;INITIALIZE THE STACK POINTER
                                        MOV     #1$,RTRAP4              ;SET UP NEW PC IN VECTOR
                                        MOV     R0,RTRAP4+2             ;SET UP NEW PSW IN VECTOR
```

```
                                        CLR     STATUS          ;CLEAR PRESENT (OLD) STATUS
                                        TRT                     ;DO TRAP
                                8$:
                                        BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
                                1$:     CMP     R0,STATUS       ;IS NEW STATUS CORRECT
                                        BNE     ERROR2          ;NEW STATUS WRONG
                                2$:     CMP     SP,#STBOT-4     ;DID STACK DECREMENT CORRECTLY
                                        BNE     ERROR2          ;STACK DID NOT DECREMENT CORRECTLY
                                3$:     CMP     #8$,STBOT-4     ;WAS PROPER PC SAVED
                                        BNE     ERROR2          ;PROPER PC WAS NOT SAVED
                                4$:     TST     STBOT-2         ;WAS OLD PSW SAVED
                                        BNE     ERROR2          ;WRONG PSW SAVED
                                5$:     ADD     #4,SP           ;RESET STACK POINTER
                                        MOV     #6$,RTRAP4              ;SET UP NEW PC IN VECTOR
                                        CLR     RTRAP4+2                ;SET UP NEW PSW IN VECTOR
                                        MOV     R0,@#STATUS     ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
                                        TRT                     ;DO TRAP
                                        BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
                                6$:     TST     STATUS          ;IS NEW PSW CORRECT
                                        BNE     ERROR2          ;NEW PSW WRONG
                                7$:     CMP     R0,STBOT-2      ;WAS OLD STATUS STORED
                                        BNE     ERROR2          ;OLD STATUS WRONG
                        ;PDP-11 ILLEGAL AND ADDRESS INSTRUCTION TEST
                        ;ALL INSTRUCTIONS THAT ARE RESERVED
                        ;SHOULD TRAP TO LOCATION 4, AND THE
                        ;PC THAT POINTS TO THE TRAPPING INSTRUCTION
                        ;SHOULD BE PLACED ON THE STACK

                        ;*****************************************************************************************
                        ;TEST 7 TEST TRAP OF ILLEGAL INSTRUCTION
                        ;*****************************************************************************************
002612                  TS7:
002612  012706  001000          MOV     #STBOT,SP       ;INITIALIZE THE STACK POINTER
002616  012767  002640  175160        MOV     #1$,RTRAP5              ;SET UP NEW PC IN VECTOR
002624  010067  175156              MOV     R0,RTRAP5+2             ;SET UP NEW PSW IN VECTOR
002630  005067  175142              CLR     STATUS          ;CLEAR PRESENT (OLD) STATUS
002634  000100                      JMP     %0                      ;DO TRAP
002636                          8$:
002636  000713                      BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
002640  020067  175132          1$: CMP     R0,STATUS       ;IS NEW STATUS CORRECT
002644  001310                      BNE     ERROR2          ;NEW STATUS WRONG
002646  020627  000774          2$: CMP     SP,#STBOT-4     ;DID STACK DECREMENT CORRECTLY
002652  001305                      BNE     ERROR2          ;STACK DID NOT DECREMENT CORRECTLY
002654  022767  002636  176112  3$: CMP     #8$,STBOT-4     ;WAS PROPER PC SAVED
002662  001301                      BNE     ERROR2          ;PROPER PC WAS NOT SAVED
002664  005767  176106          4$: TST     STBOT-2         ;WAS OLD PSW SAVED
002670  001276                      BNE     ERROR2          ;WRONG PSW SAVED
002672  062706  000004          5$: ADD     #4,SP           ;RESET STACK POINTER
002676  012767  002720  175100        MOV     #6$,RTRAP5              ;SET UP NEW PC IN VECTOR
002704  005067  175076              CLR     RTRAP5+2                ;SET UP NEW PSW IN VECTOR
002710  010037  177776              MOV     R0,@#STATUS     ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
002714  000100                      JMP     %0                      ;DO TRAP
002716  000663                      BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
002720  005767  175052          6$: TST     STATUS          ;IS NEW PSW CORRECT
002724  001260                      BNE     ERROR2          ;NEW PSW WRONG
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  002734                        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 VERIFY 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                                ;*****************************************************************************
1085                                ;TEST 11        TEST OF ZEROES IN THE DATA PATH
1086                                ;*****************************************************************************
1087  003122                        TS11:
1088  003122  012737  000000  000000        MOV     #0,@#0          ;MOVE ZEROES THRU ADDRESS LINES, DATA
1089                                                                ;LINES AND INTERNAL PATHS
```

```
1090  003130  005737  000000          TST     @#0             ;SUCCESSFUL?
1091  003134  001401                  BEQ     TS12
1092  003136  104000                  EMT                     ;DATA INCORRECT
1093
1094                                  ;*************************************************************************
1095                                  ;TEST 12        TEST OF PATTERN 125252 IN DATA PATH
1096                                  ;*************************************************************************
1097  003140                          TS12:
1098  003140  012737  125252  000000          MOV     #125252,@#0     ;MOVE ALTERNATING ONES AND ZEROES
1099                                                                  ;THRU DATA PATHS
1100  003146  022737  125252  000000          CMP     #125252,@#0     ;SUCCESSFUL
1101  003154  001401                          BEQ     TS13
1102  003156  104000                          EMT                     ;DATA INCORRECT
1103
1104                                  ;*************************************************************************
1105                                  ;TEST 13        TEST OF PATTERN 052525 IN DATA PATH
1106                                  ;*************************************************************************
1107  003160                          TS13:
1108  003160  012737  052525  000000          MOV     #052525,@#0     ;MOVE ALTERNATING ZEROES AND ONES
1109                                                                  ;THRU DATA PATH
1110  003166  022737  052525  000000          CMP     #052525,@#0     ;SUCCESSFUL?
1111  003174  001401                          BEQ     TS14
1112  003176  104000                          EMT                     ;DATA INCORRECT
1113
1114                                  ;*************************************************************************
1115                                  ;TEST 14        TEST OF ALL ONES IN DATA PATH
1116                                  ;*************************************************************************
1117  003200                          TS14:
1118  003200  012737  177777  000000          MOV     #177777,@#0     ;MOVE ONES THRU DATA PATH
1119  003206  022737  177777  000000          CMP     #177777,@#0     ;SUCCESSFUL
1120  003214  001401                          BEQ     TS15
1121  003216  104000                          EMT                     ;DATA INCORRECT
1122
1123                                  ;*************************************************************************
1124                                  .SBTTL  B-REGISTER TEST
1125                                  ;
1126                                  ;       THE B-REGISTER (LOCATION 0) SHIFTING LOGIC TESTS ARE USED
1127                                  ;TO TEST THAT THE B-REGISTER CAN HOLD VARIOUS DATA PATTERNS AND THAT
1128                                  ;THE ASSOCIATED LOGIC SUPPORTS THE SHIFTING FUNCTIONS WITHIN THE
1129                                  ;B-REGISTER AND C-BIT.
1130                                  ;A ONE IS SHIFTED THROUGH EVERY BIT IN THE B-REGISTER AND C-BIT IN
1131                                  ;BOTH DIRECTIONS.
1132                                  ;       THE B-REGISTER ITSELF IS TESTED IN ITS ABILITY AS A BUFFER AND AS
1133                                  ;A SHIFT REGISTER.  DATA IS ALSO PASSED THROUGH THE DATA PATH AND ALU,
1134                                  ;       IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0) TO SEE
1135                                  ;WHICH BITS OF THE B-REGISTER MAY BE FAILING.
1136                                  ;
1137                                  ;*************************************************************************
1138                                  ;TEST 15        SHIFT BIT 0 TO BIT 1
1139                                  ;*************************************************************************
1140  003220                          TS15:
1141  003220  000241                          CLC                     ;CLEAR CARRY BIT
1142  003222  012737  000001  000000          MOV     #1,@#0          ;LOAD A 1
1143  003230  006137  000000                  ROL     @#0             ;SHIFT LEFT
1144  003234  022737  000002  000000          CMP     #2,@#0          ;SUCCESSFUL
1145  003242  001401                          BEQ     TS16
```

```
1146  003244  104000                          EMT                     ;BIT 1 NOT SET
1147
1148                                          ;*************************************************************************************
1149                                          ;TEST 16        SHIFT CARRY INTO BIT 0
1150                                          ;*************************************************************************************
1151  003246                                  TS16:
1152  003246  012737  000000  000000          MOV     #0,@#0          ;CLEAR LOCATION
1153  003254  000261                          SEC                     ;SET CARRY
1154  003256  006137  000000                  ROL     @#0             ;ROTATE CARRY BIT TO BIT 0
1155  003262  103001                          BCC     CARRY1
1156  003264  104000                          EMT                     ;CARRY CLEAR
1157  003266  022737  000001  000000  CARRY1: CMP     #1,@#0          ;BIT 0 SET
1158  003274  001401                          BEQ     TS17
1159  003276  104000                          EMT                     ;BIT 0 NOT SET
1160
1161                                          ;*************************************************************************************
1162                                          ;TEST 17        LEFT SHIFT FROM BIT 0 TO C-BIT
1163                                          ;*************************************************************************************
1164  003300                                  TS17:
1165  003300  012737  000001  000000          MOV     #1,@#0          ;SET BIT 0
1166  003306  012700  177757                  MOV     #-21,R0         ;SET BIT COUNTER
1167  003312  000241                          CLC                     ;CLEAR C-BIT
1168  003314  005200                  SHL:    INC     R0              ;INCREMENT BIT COUNTER
1169  003316  001404                          BEQ     SHLE            ;BR TO ERROR HALT IF BIT IS LOST
1170  003320  006137  000000                  ROL     @#0             ;SHIFT LEFT ONE POSITION
1171  003324  103373                          BCC     SHL             ;BRANCH IF C-BIT NOT SET
1172  003326  001401                          BEQ     TS20
1173  003330                                  SHLE:
1174  003330  104000                          EMT                     ;LEFT SHIFTING LOGIC FAILED
1175
1176                                          ;*************************************************************************************
1177                                          ;TEST 20        SHIFT BIT 15 TO BIT 14
1178                                          ;*************************************************************************************
1179  003332                                  TS20:
1180  003332  012737  100000  000000          MOV     #100000,@#0     ;SET BIT 15
1181  003340  000241                          CLC                     ;CLEAR CARRY
1182  003342  006037  000000                  ROR     @#0             ;SHIFT BIT 15 TO BIT 14
1183  003346  022737  040000  000000          CMP     #40000,@#0      ;SUCCESSFUL
1184  003354  001401                          BEQ     TS21
1185  003356  104000                          EMT                     ;BIT 14 NOT SET
1186
1187                                          ;*************************************************************************************
1188                                          ;TEST 21        RIGHT SHIFT FROM BIT 15 TO C-BIT
1189                                          ;*************************************************************************************
1190  003360                                  TS21:
1191  003360  012737  100000  000000          MOV     #100000,@#0     ;SET BIT 15
1192  003366  012700  177757                  MOV     #-21,R0         ;SET BIT COUNTER
1193  003372  000241                          CLC                     ;CLEAR C-BIT
1194  003374  005200                  SHR:    INC     R0              ;INCREMENT BIT COUNTER
1195  003376  001404                          BEQ     SHRE            ;BR TO ERROR HALT IF BIT IS LOST
1196  003400  006037  000000                  ROR     @#0             ;ROTATE RIGHT ONE POSITION
1197  003404  103373                          BCC     SHR             ;BRANCH IF C-BIT CLEAR
1198  003406  001401                          BEQ     TS22
1199  003410                                  SHRE:
1200  003410  104000                          EMT                     ;RIGHT SHIFT LOGIC FAILED
1201
```

```
                        ;*****************************************************************************************
                        .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
                        ;R0 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 R0 CAN HOLD ALL ZEROES
                        ;*****************************************************************************************
003412                  TS22:

003412  012700  000000          MOV     #0,R0           ;MOVE ZEROES TO R0
003416  005700                  TST     R0              ;SUCCESSFUL?
003420  001401                  BEQ     TS23
003422  104000                  EMT                     ;R0 NOT 0

                        ;*****************************************************************************************
                        ;TEST 23        TEST IF R0 CAN HOLD ONES AND ZEROES
                        ;*****************************************************************************************
003424                  TS23:
003424  012700  125252          MOV     #125252,R0      ;MOVE ALTERNATING ONES AND ZEROES TO R0
003430  020027  125252          CMP     R0,#125252      ;SUCCESSFUL?
003434  001401                  BEQ     TS24
003436  104000                  EMT                     ;R0 NOT 125252

                        ;*****************************************************************************************
                        ;TEST 24        TEST IF R0 CAN HOLD ZEROES AND ONES
                        ;*****************************************************************************************
003440                  TS24:
003440  012700  052525          MOV     #052525,R0      ;MOVE ALTERNATING ZEROES AND ONES TO R0
003444  020027  052525          CMP     R0,#052525      ;SUCCESSFUL?
003450  001401                  BEQ     TS25
003452  104000                  EMT                     ;R0 NOT 52525

                        ;*****************************************************************************************
                        ;TEST 25        TEST IF R0 CAN HOLD ALL ONES
                        ;*****************************************************************************************
003454                  TS25:
003454  012700  177777          MOV     #177777,R0      ;MOVE ALL ONES TO R0
003460  020027  177777          CMP     R0,#177777      ;SUCCESSFUL?
003464  001401                  BEQ     TS26
003466  104000                  EMT                     ;R0 NOT 177777
```

```
1258
1259                                    ;*****************************************************************************
1260                                    ;TEST 26        TEST IF R1 CAN HOLD A ONE IN ALL BITS
1261                                    ;*****************************************************************************
1262  003470                            TS26:
1263  003470  012701  000001                   MOV     #1,R1           ;SET BIT 0
1264  003474  012700  177757                   MOV     #-21,R0         ;SET BIT COUNTER
1265  003500  000241                           CLC                     ;CLEAR C-BIT
1266  003502  005200                    REG1:  INC     R0              ;INCREMENT BIT COUNTER
1267  003504  001403                           BEQ     REG1E           ;BR TO ERROR HALT IF BIT IS LOST
1268  003506  006101                           ROL     R1              ;ROTATE 1 POSITION
1269  003510  103374                           BCC     REG1            ;ALL DONE
1270  003512  001401                           BEQ     TS27
1271  003514                            REG1E:
1272  003514  104000                           EMT                     ;FAILURE WITH R1
1273
1274                                    ;*****************************************************************************
1275                                    ;TEST 27        TEST IF R1 CAN HOLD A ZERO IN ALL BITS
1276                                    ;*****************************************************************************
1277  003516                            TS27:
1278  003516  012701  177776                   MOV     #-2,R1          ;SET ALL ONES IN R1 EXCEPT FOR BIT 0
1279  003522  012700  177757                   MOV     #-21,R0         ;SET BIT COUNTER
1280  003526  000261                           SEC                     ;SET C-BIT
1281  003530  005200                    REG1A: INC     R0              ;INCREMENT COUNTER
1282  003532  001405                           BEQ     R1ERR           ;BR TO ERROR HALT IF COUNTER=0
1283  003534  006101                           ROL     R1              ;ROTATE 1 POSITION
1284  003536  103774                           BCS     REG1A           ;CONTINUE UNTIL C-BIT IS CLEAR
1285  003540  022701  177777                   CMP     #-1,R1          ;CHECK DATA IN R1
1286  003544  001401                           BEQ     TS30
1287  003546                            R1ERR:
1288  003546  104000                           EMT                     ;FAILURE WITH R1
1289                                    ;*****************************************************************************
1290                                    ;TEST 30        TEST IF R2 CAN HOLD A ONE IN ALL BITS
1291                                    ;*****************************************************************************
1292  003550                            TS30:
1293  003550  012702  000001                   MOV     #1,R2           ;SET BIT 0
1294  003554  012700  177757                   MOV     #-21,R0         ;SET BIT COUNTER
1295  003560  000241                           CLC                     ;CLEAR C-BIT
1296  003562  005200                    REG2:  INC     R0              ;INCREMENT BIT COUNTER
1297  003564  001403                           BEQ     REG2E           ;BR TO ERROR HALT IF BIT IS LOST
1298  003566  006102                           ROL     R2              ;ROTATE 1 POSITION
1299  003570  103374                           BCC     REG2            ;ALL DONE
1300  003572  001401                           BEQ     TS31
1301  003574                            REG2E:
1302  003574  104000                           EMT                     ;FAILURE WITH R2
1303
1304                                    ;*****************************************************************************
1305                                    ;TEST 31        TEST IF R2 CAN HOLD A ZERO IN ALL BITS
1306                                    ;*****************************************************************************
1307  003576                            TS31:
1308  003576  012702  177776                   MOV     #-2,R2          ;SET ALL ONES IN R2 EXCEPT FOR BIT 0
1309  003602  012700  177757                   MOV     #-21,R0         ;SET BIT COUNTER
1310  003606  000261                           SEC                     ;SET C-BIT
1311  003610  005200                    REG2B: INC     R0              ;INCREMENT BIT COUNTER
1312  003612  001405                           BEQ     R2ERR           ;BR TO ERROR HALT IF COUNTER=0
1313  003614  006102                           ROL     R2              ;ROTATE 1 POSITION
```

```
1314  003616  103774                  BCS     REG2B           ;CONTINUE UNTIL C-BIT IS CLEAR
1315  003620  022702  177777          CMP     #-1,R2          ;CHECK DATA IN R2
1316  003624  001401                  BEQ     TS32
1317  003626                  R2ERR:
1318  003626  104000                  EMT                     ;FAILURE WITH R2
1319
1320                          ;*****************************************************************************
1321                          ;TEST 32        TEST IF R3 CAN HOLD A ONE IN ALL BITS
1322                          ;*****************************************************************************
1323  003630                  TS32:
1324  003630  012703  000001          MOV     #1,R3           ;SET BIT 0
1325  003634  012700  177757          MOV     #-21,R0         ;SET BIT COUNTER
1326  003640  000241                  CLC                     ;CLEAR C-BIT
1327  003642  005200          REG3:   INC     R0              ;INCREMENT BIT COUNTER
1328  003644  001403                  BEQ     REG3E           ;BR TO ERROR HALT IF BIT IS LOST
1329  003646  006103                  ROL     R3              ;ROTATE 1 POSITION
1330  003650  103374                  BCC     REG3            ;ALL DONE
1331  003652  001401                  BEQ     TS33
1332  003654                  REG3E:
1333  003654  104000                  EMT                     ;FAILURE WITH R3
1334
1335                          ;*****************************************************************************
1336                          ;TEST 33        TEST IF R3 CAN HOLD A ZERO IN ALL BITS
1337                          ;*****************************************************************************
1338  003656                  TS33:
1339  003656  012703  177776          MOV     #-2,R3          ;SET ALL ONES IN R3 EXCEPT FOR BIT 0
1340  003662  012700  177757          MOV     #-21,R0         ;SET BIT COUNTER
1341  003666  000261                  SEC                     ;SET C-BIT
1342  003670  005200          REG3A:  INC     R0              ;INCREMENT BIT COUNTER
1343  003672  001405                  BEQ     R3ERR           ;BR TO ERROR HALT IF COUNTER=0
1344  003674  006103                  ROL     R3              ;ROTATE 1 POSITION
1345  003676  103774                  BCS     REG3A           ;CONTINUE UNTIL C-BIT IS CLEAR
1346  003700  022703  177777          CMP     #-1,R3          ;CHECK DATA
1347  003704  001401                  BEQ     TS34
1348  003706                  R3ERR:
1349  003706  104000                  EMT                     ;FAILURE WITH R3
1350
1351                          ;*****************************************************************************
1352                          ;TEST 34        TEST IF R4 CAN HOLD A ONE IN ALL BITS
1353                          ;*****************************************************************************
1354  003710                  TS34:
1355  003710  012704  000001          MOV     #1,R4           ;SET BIT 0
1356  003714  012700  177757          MOV     #-21,R0         ;SET BIT COUNTER
1357  003720  000241                  CLC                     ;CLEAR C-BIT
1358  003722  005200          REG4:   INC     R0              ;INCREMENT BIT COUNTER
1359  003724  001403                  BEQ     REG4E           ;BR TO ERROR HALT IF BIT IS LOST
1360  003726  006104                  ROL     R4              ;ROTATE 1 POSITION
1361  003730  103374                  BCC     REG4            ;ALL DONE
1362  003732  0C1401                  BEQ     TS35
1363  003734                  REG4E:
1364  003734  104000                  EMT                     ;FAILURE WITH R4
1365
1366                          ;*****************************************************************************
1367                          ;TEST 35        TEST IF R4 CAN HOLD A ZERO IN ALL BITS
1368                          ;*****************************************************************************
1369  003736                  TS35:
```

```
1370  003736  012704  177776          MOV     #-2,R4          ;SET ALL ONES IN R4 EXCEPT FOR BIT 0
1371  003742  012700  177757          MOV     #-21,R0         ;SET BIT COUNTER
1372  003746  000261                  SEC                     ;SET C-BIT
1373  003750  005200          REG4A:  INC     R0              ;INCREMENT BIT COUNTER
1374  003752  001405                  BEQ     R4ERR           ;BR TO ERROR HALT IF COUNTER=0
1375  003754  006104                  ROL     R4              ;ROTATE 1 POSITION
1376  003756  103774                  BCS     REG4A           ;CONTINUE UNTIL C-BIT IS CLEAR
1377  003760  022704  177777          CMP     #-1,R4          ;CHECK DATA
1378  003764  001401                  BEQ     TS36
1379  003766                  R4ERR:
1380  003766  104000                  EMT                     ;FAILURE WITH R4
1381
1382
1383                            ;*****************************************************************************
1384                            ;TEST 36        TEST IF R5 CAN HOLD A ONE IN ALL BITS
1385                            ;*****************************************************************************
1386  003770                  TS36:
1387  003770  012705  000001          MOV     #1,R5           ;SET BIT 0
1388  003774  012700  177757          MOV     #-21,R0         ;SET BIT COUNTER
1389  004000  000241                  CLC                     ;CLEAR C-BIT
1390  004002  005200          REG5:   INC     R0              ;INCREMENT BIT COUNTER
1391  004004  001403                  BEQ     REG5E           ;BR TO ERROR HALT IF BIT IS LOST
1392  004006  006105                  ROL     R5              ;ROTATE 1 POSITION
1393  004010  103374                  BCC     REG5            ;ALL DONE
1394  004012  001401                  BEQ     TS37
1395  004014                  REG5E:
1396  004014  104000                  EMT                     ;FAILURE WITH R5
1397
1398
1399                            ;*****************************************************************************
1400                            ;TEST 37        TEST IF R5 CAN HOLD A ZERO IN ALL BITS
1401  004016                            ;*****************************************************************************
1402  004016  012705  177776  TS37:
1403  004022  012700  177757          MOV     #-2,R5          ;SET ALL ONES IN R5 EXCEPT FOR BIT 0
1404  004026  000261                  MOV     #-21,R0         ;SET BIT COUNTER
1405  004030  005200                  SEC                     ;SET C-BIT
1406  004032  001405          REG5A:  INC     R0              ;INCREMENT BIT COUNTER
1407  004034  006105                  BEQ     R5ERR           ;BR TO ERROR HALT IF COUNTER=0
1408  004036  103774                  ROL     R5              ;ROTATE 1 POSITION
1409  004040  022705  177777          BCS     REG5A           ;CONTINUE UNTIL C-BIT IS CLEAR
1410  004044  001401                  CMP     #-1,R5          ;CHECK DATA
1411  004046                          BEQ     TS40
1412  004046  104000          R5ERR:
1413                                  EMT                     ;FAILURE WITH R5
1414
1415                            ;*****************************************************************************
1416                            ;TEST 40        TEST IF R6 CAN HOLD A ONE IN ALL BITS
1417  004050                            ;*****************************************************************************
1418  004050  012706  000001  TS40:
1419  004054  012700  177757          MOV     #1,R6           ;SET BIT 0
1420  004060  000241                  MOV     #-21,R0         ;SET BIT COUNTER
1421  004062  005200                  CLC                     ;CLEAR C-BIT
1422  004064  001403          REG6:   INC     R0              ;INCREMENT BIT COUNTER
1423  004066  006106                  BEQ     REG6E           ;BR TO ERROR HALT IF BIT IS LOST
1424  004070  103374                  ROL     R6              ;ROTATE 1 POSITION
1425  004072  001401                  BCC     REG6            ;ALL DONE
                                        BEQ     TS41
```

```
                                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
                                        MOV     #0,@#PS         ;SET PSW TO ZERO
                                        TST     @#PS            ;SUCCESSFUL
                                        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          MOV     #105,@#PS       ;MOVE ALT. ONES AND ZEROES TO PSW
1483  004200  023727  177776  000105          CMP     @#PS,#105       ;SUCCESSFUL?
1484  004206  001401                          BEQ     TS45
1485  004210  104000                          EMT                     ;PSW NOT 105
1486
1487                                  ;*************************************************************************************
1488                                  ;TEST 45        TEST IF PSW (EXCEPT T-BIT) WILL HOLD ALL ONES
1489                                  ;*************************************************************************************
1490  004212
1491  004212  012737  000357  177776  TS45:   MOV     #357,@#PS       ;MOVE ONES TO PSW
1492  004220  023727  177776  000357          CMP     @#PS,#357       ;SUCCESSFUL
1493  004226  001401                          BEQ     TS46
1494  004230  104000                          EMT                     ;PSW NOT 357
1495                                  ;*************************************************************************************
1496                                  .SBTTL  MICROCODE TESTS
1497                                  ;
1498                                  ;       THE TEST EXERCISES  BRANCHES IN THE MICROCODE BY
1499                                  ;TESTING AT LEAST ONE INSTRUCTION FROM EVERY CLASS OF INSTRUCTION IN
1500                                  ;ALL POSSIBLE MODES.  FOR EXAMPLE, TO TEST THE SINGLE OPERAND INSTRUCTIONS,
1501                                  ;AT LEAST ONE SINGLE OPERAND INSTRUCTION IS VERIFIED IN ALL UNIQUE
1502                                  ;ADDRESSING MODES.  BYTE MODES ARE ALSO TESTED.  AS EACH NEW
1503                                  ;MODE IS INTRODUCED THE SAME INSTRUCTION IS TRIED AND TESTED IN
1504                                  ;A SMALL LOOP CONVENIENT FOR SCOPING.  THE TEST IS SET UP USING
1505                                  ;ONLY INSTRUCTIONS AND ADDRESSING MODES WHICH HAVE BEEN PREVIOUSLY
1506                                  ;VERIFIED.
1507                                  ;       IF THESE TESTS FAIL, CHECK THE RESULTS FOR A CLUE TO THE
1508                                  ;FAULT.
1509                                  ;
1510                                  ;*************************************************************************************
1511
1512
1513
1514                                  ;*************************************************************************************
1515                                  ;
1516                                  ;       THE CLR INSTRUCTION IS USED TO INTRODUCE EACH ADDRESSING
1517                                  ;MODE WITH THE SINGLE OPERAND INSTRUCTION.  FOLLOWING THE SEQUENCE CHECK,
1518                                  ;THE CLR INSTRUCTION IS EXECUTED AND A BRANCH TEST IS EXECUTED WHICH
1519                                  ;CHECKS THAT THE Z-BIT WAS PROPERLY SET.  THIS TEST CAN CHECK IR DECODE
1520                                  ;AND MICROCODE FOR SOP INSTRUCTIONS WITH MODE 0.  FOLLOWING THIS TEST
1521                                  ;SEVERAL OTHER SOP INSTRUCTIONS ARE INTRODUCED WITH MODE 0.  THESE
1522                                  ;INSTRUCTIONS MAINPULATE DATA AND SERVE TO CHECK THE DATA RESULTS
1523                                  ;OF THE SOP INSTRUCTIONS IN THIS TEST.  THE DATA IN THIS TEST IS
1524                                  ;OPERATED ON BY EACH INSTRUCTION WITHOUT REINITIALIZING.
1525                                  ;
1526                                  ;*************************************************************************************
1527                                  ;TEST 46        TEST MODE 0 USING SOP INST.
1528                                  ;*************************************************************************************
1529  004232
1530  004232  005000                  TS46:   CLR     R0              ;TRY THE CLEAR INST.
1531  004234  001401                          BEQ     SOP0A
1532  004236  104000                          EMT                     ;CLR DID NOT SET Z-BIT
1533  004240  005200                  SOP0A:  INC     R0              ;TRY THE INCREMENT INST.
1534  004242  005100                          COM     R0              ;TRY COMPLEMENT
1535  004244  005200                          INC     R0
1536  004246  100401                          BMI     SOP0B
1537  004250  104000                          EMT                     ;NEGATE DID NOT SET N-BIT
```

```
1538  004252  005100  SOP0B:  COM     R0              ;TRY COMPLEMENT INST.
1539  004254  001401          BEQ     TS47
1540  004256  104000          EMT                     ;CUMMULATIVE RESULT OF CLR,INC,NEG AND COM INSTS. FAILED
1541
1542
1543                          ;*****************************************************************************
1544                          ;
1545                          ;       THIS TEST INTRODUCES THE REMAINING SOP INSTRUCIONS AND TESTS
1546                          ;THEM IN MODE 0.  THE PURPOSE IS TO PROVIDE A BASELINE OF
1547                          ;INSTRUCTIONS FOR USE IN THE SUBSEQUENT TESTS.  SINCE THE MICROCODE FOR
1548                          ;THESE INSTRUCTIONS IS IDENTICAL TO THAT ALREADY TESTED, ANY TROUBLE
1549                          ;SHOOTING EFFORTS SHOULD BE AIMED AT THE ACTUAL IR DECODE AND ALU
1550                          ;FUNCTIONING.
1551                          ;
1552                          ;*****************************************************************************
1553                          ;TEST 47        TEST REMAINDER OF SOP INSTS IN MODE 0
1554                          ;*****************************************************************************
1555  004260                  TS47:
1556  004260  005000          CLR     R0              ;INITIALIZE
1557  004262  005300          DEC     R0              ;TRY DECREMENT INST.
1558  004264  100401          BMI     SOP0C
1559  004266  104000          EMT                     ;N-BIT NOT SET ON DEC
1560  004270  000261  SOP0C:  SEC                     ;INITIALIZE CARRY
1561  004272  005500          ADC     R0              ;TRY ADD CARRY INST
1562  004274  001007          BNE     SOP0D
1563  004276  000261          SEC                     ;INITIALIZE CARRY
1564  004300  005600          SBC     R0              ;TRY SUBTRACT-CARRY INST
1565  004302  100004          BPL     SOP0D
1566  004304  005100          COM     R0
1567  004306  005200          INC     R0
1568  004310  005300          DEC     R0
1569  004312  001401          BEQ     TS50
1570  004314                  SOP0D:
1571  004314  104000          EMT                     ; CUMMULATIVE RESULT OF ADC,SBC,COM,INC AND DEC INSTS. F
1572
1573                          ;*****************************************************************************
1574                          ;
1575                          ;       THIS TEST INTRODUCES THE BYTE CONTROL LOGIC OF THE PROCESSOR.
1576                          ;THE MODE 0 BYTE MICROCODE IS TESTED.  THE METHOD AND SEQUENCE
1577                          ;OF TESTING IS THE SAME AS THAT USED IN THE SOP MODE 0 TESTS.
1578                          ;
1579                          ;*****************************************************************************
1580                          ;TEST 50        TEST MODE 0 EVEN BYTE USING SOP INST
1581                          ;*****************************************************************************
1582  004316                  TS50:
1583  004316  105000          CLRB    R0              ;TRY CLEARING EVEN BYTE OF REGISTER
1584  004320  001401          BEQ     SOPB0A
1585  004322  104000          EMT                     ;CLRB DID NOT SET Z-BIT
1586  004324  105100  SOPB0A: COMB    R0              ;TRY SETTING EVEN BYTE OF REGISTER
1587  004326  100002          BPL     SOPB0B
1588  004330  105200          INCB    R0              ;TRY INCREMENTING EVEN BYTE OF REGISTER>>
1589  004332  001401          BEQ     TS51
1590  004334                  SOPB0B:
1591  004334  104000          EMT                     ;TEST CUMMULATIVE RESULT OF ABOVE BYTE INST.
1592
1593                          ;*****************************************************************************
```

```
                                ;
                                ;       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.
                                ;*****************************************************************************
004336                          TS51:
004336  005000                          CLR     R0              ;INITIALIZE R0
004340  005010                          CLR     (R0)            ;TRY CLEAR INST W/MODE 1
004342  001401                          BEQ     SOP1A
004344  104000                          EMT                     ;CLR DID NOT SET Z-BIT
004346  005310                  SOP1A:  DEC     (R0)            ;TRY DECREMENT INST W/MODE 1
004350  100003                          BPL     SOP1B
004352  000261                          SEC                     ;INITIALIZE CARRY
004354  005510                          ADC     (R0)            ;TRY ADD-CARRY W/MODE 1
004356  001401                          BEQ     TS52
004360                          SOP1B:
004360  104000                          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
                                ;*****************************************************************************
004362                          TS52:
004362  005000                          CLR     R0              ;INITIALIZE R0
004364  005010                          CLR     (R0)            ;INITIALIZE LOC. 0
004366  005110                          COM     (R0)
004370  105010                          CLRB    (R0)            ;TRY TO CLEAR BYTE 0
004372  001401                          BEQ     SOPB1A
004374  104000                          EMT                     ;CLRB DID NOT SET Z-BIT
004376  005210                  SOPB1A: INC     (R0)            ;INCREMENT TO TEST WORD
004400  100005                          BPL     SOPB1B
004402  105110                          COMB    (R0)            ;COMPLEMENT:  ODD BYTE = 376
004404  105210                          INCB    (R0)            ;INC:  ODD BYTE = 377
004406  100002                          BPL     SOPB1B
004410  105210                          INCB    (R0)            ;INCREMENT ODD BYTE=0
004412  001401                          BEQ     TS53
004414                          SOPB1B:
004414  104000                          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
```

```
                        ;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
                        ;*****************************************************************************
004416                  TS53:
004416  005000                  CLR     R0              ;INITIALIZE R0
004420  005010                  CLR     (R0)            ;INITIALIZE LOC. 0
004422  005110                  COM     (R0)
004424  005200                  INC     R0              ;R0=ODD BYTE
004426  105010                  CLRB    (R0)            ;TRY TO CLEAR BYTE 1
004430  001401                  BEQ     SOPB1C
004432  104000                  EMT                     ;CLRB DID NOT SET Z-BIT
004434  005300          SOPB1C: DEC     R0              ;R0=WORD ADDR.
004436  005210                  INC     (R0)            ;INCREMENT TO TEST WORD
004440  005200                  INC     R0              ;R0=ODD BYTE
004442  105110                  COMB    (R0)            ;TRY TO COMPLEMENT BYTE 1
004444  105210                  INCB    (R0)
004446  100002                  BPL     SOPB1D
004450  105210                  INCB    (R0)            ;TRY TO INCREMENT BYTE 1
004452  001401                  BEQ     TS54
004454                  SOPB1D:
004454  104000                  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.
                        ;*****************************************************************************
004456                  TS54:
004456  005000                  CLR     R0              ;SET R0=400
004460  105100                  COMB    R0
004462  005200                  INC     R0
004464  005010                  CLR     (R0)            ;CLEAR 400
004466  005110                  COM     (R0)            ;INITIALIZE: 400=-1
004470  005020                  CLR     (R0)+           ;TRY CLEARING WITH MODE 2
004472  001401                  BEQ     SOPZA
004474  104000                  EMT                     ;CLR INST DID NOT SET Z-BIT
004476  005300          SOPZA:  DEC     R0              ;RESET R0
004500  005300                  DEC     R0
004502  005120                  COM     (R0)+           ;TRY COMPLEMENTING WITH MODE 2
004504  100004                  BPL     SOP2B
004506  005300                  DEC     R0              ;RESET R0
004510  005300                  DEC     R0
004512  005220                  INC     (R0)+           ;TRY INCREMENTING WITH MODE 2
004514  001401                  BEQ     TS55
```

```
1706  004516
1707  004516  104000                  SOP2B:
                                              EMT                     ;CHECK CUMMULATIVE RESULT OF ABOVE INST
1708
1709                                  ;*************************************************************************************
1710                                  ;
1711                                  ;       THIS TEST VERIFIES MODE 2 SINGLE OPERAND INSTRUCTIONS WHICH
1712                                  ;ADDRESS EVEN BYTES.  R0 IS SET TO 400 AND USED TO INITIALIZE LOCATION
1713                                  ;400 TO -1.  CLRB INSTRUCTION IS THEN EXECUTED ON BYTE 400 WITH
1714                                  ;MODE 2.
1715                                  ;       R0 IS THEN DECREMENTED BEFORE EACH OF SEVERAL MODE 2 INSTRUCTIONS
1716                                  ;WHICH ARE USED TO VERIFY THE DATA RESULTS OF THE TEST.  THIS PROCEDURE ALSO
1717                                  ;VERIFIES THE PROPER INCREMENTING OF THE REGISTER.
1718                                  ;
1719                                  ;*************************************************************************************
1720                                  ;TEST 55        TEST MODE 2 EVEN BYTE USING SOP INST.
1721                                  ;*************************************************************************************
1722  004520                          TS55:
1723  004520  005000                          CLR     R0              ;SET R0=400
1724  004522  105100                          COMB    R0
1725  004524  005200                          INC     R0
1726  004526  005010                          CLR     (R0)            ;CLEAR 400
1727  004530  005110                          COM     (R0)            ;INITIALIZE: 400=-1
1728  004532  105020                          CLRB    (R0)+           ;TRY TO CLEAT 400 W/MODE 2
1729  004534  001401                          BEQ     SOPB2A
1730  004536  104000                          EMT                     ;CLR DID NOT SET Z-BIT
1731  004540  005300                  SOPB2A: DEC     R0              ;RESULT R0=400
1732  004542  005210                          INC     (R0)            ;INC 400 TO TEST WORD
1733  004544  105110                          COMB    (R0)
1734  004546  105220                          INCB    (R0)+           ;TRY TO INC EVEN BYTE
1735  004550  100003                          BPL     SOPB2B
1736  004552  005300                          DEC     R0              ;RESET R0=400
1737  004554  105220                          INCB    (R0)+           ;TRY INCREMENT OF EVEN BYTE
1738  004556  001401                          BEQ     TS56
1739  004560                          SOPB2B:
1740  004560  104000                          EMT                     ;TEST CUMMULATIVE RESULT OF ABOVE INST.
1741
1742                                  ;*************************************************************************************
1743                                  ;
1744                                  ;       THIS TEST FOLLOWS THE SAME PROCEDURE DESCRIBED IN THE PREVIOUS
1745                                  ;TEST.  HERE, THE BYTE INSTRUCTION IS USED TO ADDRESS AN ODD BYTE.
1746                                  ;
1747                                  ;*************************************************************************************
1748                                  ;TEST 56        TEST MODE 2 ODD BYTE USING SOP INST.
1749                                  ;*************************************************************************************
1750  004562                          TS56:
1751  004562  005000                          CLR     R0              ;SET R0=400
1752  004564  105100                          COMB    R0
1753  004566  005200                          INC     R0
1754  004570  005010                          CLR     (R0)            ;CLEAR LOC 400
1755  004572  005110                          COM     (R0)            ;INITIALIZE: 400=-1
1756  004574  005200                          INC     R0              ;R0=ODD BYTE
1757  004576  105020                          CLRB    (R0)+           ;TRY TO CLEAR ODD BYTE
1758  004600  001401                          BEQ     SOPB2C
1759  004602  104000                          EMT                     ;CLRB DID NOT SET Z-BIT
1760  004604  005300                  SOPB2C: DEC     R0              ;R0=WORD ADDR.
1761  004606  005300                          DEC     R0
```

```
1762  004610  005220          INC     (R0)+           ;INCREMENT WORD
1763  004612  005300          DEC     R0              ;POINT TO ODD BYTE
1764  004614  105110          COMB    (R0)            ;COMPLEMENT ODD BYTE
1765  004616  105220          INCB    (R0)+           ;TRY TO INCREMENT ODD BYTE
1766  004620  100003          BPL     SOPB2D
1767  004622  005300          DEC     R0              ;RESET R0 TO ODD BYTE
1768  004624  105220          INCB    (R0)+           ;TRY TO INCREMENT ODD BYTE
1769  004626  001401          BEQ     TS57
1770  004630                  SOPB2D:
1771  004630  104000          EMT                     ;TEST CUMMULATIVE RESULT OF ABOVE INST.
1772
1773                          ;*****************************************************************************
1774                          ;
1775                          ;       THESE TESTS CHECK THE NEGATE INSTRUCTION IN ALL MODES.  PREVIOUSLY
1776                          ;TESTED SINGLE-OPERAND INSTRUCTIONS ARE USED TO TEST THE NEGATE INSTRUCTION.
1777                          ;
1778                          ;*****************************************************************************
1779                          ;TEST 57        TEST MODE 0 USING NEGATE INSTRUCTION
1780                          ;*****************************************************************************
1781  004632                  TS57:
1782  004632  005000          CLR     R0              ;SET R0=0
1783  004634  005200          INC     R0              ;    R0=1
1784  004636  005400          NEG     R0              ;TRY NEGATE MODE 0:  R0=-1
1785  004640  100003          BPL     NEG00           ;CC=1001?
1786  004642  001402          BEQ     NEG00
1787  004644  102401          BVS     NEG00
1788  004646  103401          BCS     NEG01
1789  004650                  NEG00:
1790  004650  104000          EMT                     ;NEGATE DID NOT SET CC'S CORRECTLY
1791
1792  004652  005200  NEG01:  INC     R0              ;TEST DATA RESULT
1793  004654  001401          BEQ     NEG02
1794  004656  104000          EMT                     ;DATA RESULT OF NEGATE INCORRECT
1795
1796  004660  105100  NEG02:  COMB    R0              ;R0=377
1797  004662  105400          NEGB    R0              ;R0=1
1798  004664  100403          BMI     NEG03           ;CC=0001?
1799  004666  001402          BEQ     NEG03
1800  004670  102401          BVS     NEG03
1801  004672  103401          BCS     NEG04
1802  004674                  NEG03:
1803  004674  104000          EMT                     ;NEGB DID NOT SET CC'S CORRECTLY
1804  004676  005300  NEG04:  DEC     R0              ;TEST DATA RESULT
1805  004700  001401          BEQ     TS60
1806  004702  104000          EMT                     ;DATA RESULT OF NEGB INCORRECT
1807                          ;*****************************************************************************
1808                          ;TEST 60        TEST MODE 1 USING NEGATE INST.
1809                          ;*****************************************************************************
1810  004704                  TS60:
1811  004704  005000          CLR     R0              ;POINT TO LOC. 0
1812  004706  005010          CLR     (R0)            ;CLEAR LOC. 0
1813  004710  005210          INC     (R0)            ;LOC. 0=1
1814  004712  005410          NEG     (R0)            ;TRY NEG. LOC. 0=-1
1815  004714  100003          BPL     NEG10           ;CC=1001
1816  004716  001402          BEQ     NEG10
1817  004720  102401          BVS     NEG10
```

```
1818  004722  103401                  BCS     NEG11
1819  004724                  NEG10:
1820  004724  104000                  EMT                     ;NEGATE DID NOT SET CC'S CORRECTLY
1821
1822  004726  005237  000000  NEG11:  INC     @#0             ;TEST DATA RESULT
1823  004732  001401                  BEQ     NEG12
1824  004734  104000                  EMT                     ;DATA RESULT OF NEGATE INCORRECT
1825  004736  105110          NEG12:  COMB    (R0)            ;LOC. 0=377
1826  004740  105410                  NEGB    (R0)            ;TRY NEGB LOC. 0=1
1827  004742  100403                  BMI     NEG13           ;CC=0001?
1828  004744  001402                  BEQ     NEG13
1829  004746  102401                  BVS     NEG13
1830  004750  103401                  BCS     NEG14
1831  004752                  NEG13:
1832  004752  104000                  EMT                     ;NEGB DID NOT SET CC'S CORRECTLY
1833  004754  005337  000000  NEG14:  DEC     @#0             ;TEST DATA RESULT
1834  004760  001401                  BEQ     TS61
1835  004762  104000                  EMT                     ;DATA RESULT OF NEGB INCORRECT
1836                          ;*********************************************************************************
1837                          ;TEST 61        TEST MODE 2 USING NEGATE INSTRUCTION
1838                          ;*********************************************************************************
1839  004764                  TS61:
1840  004764  005000                  CLR     R0              ;POINT TO LOC. 0
1841  004766  005010                  CLR     (R0)            ;CLEAR LOC. 0
1842  004770  005210                  INC     (R0)            ;LOC. 0=1
1843  004772  005420                  NEG     (R0)+           ;TRY NEG.: LOC. 0=-1
1844  004774  100003                  BPL     NEG20           ;CC=1001?
1845  004776  001402                  BEQ     NEG20
1846  005000  102401                  BVS     NEG20
1847  005002  103401                  BCS     NEG21
1848  005004                  NEG20:
1849  005004  104000                  EMT                     ;NEGATE DID NOT SET CC'S CORRECTLY
1850  005006  105300          NEG21:  DECB    R0              ;R0=LOC. 0
1851  005010  105300                  DECB    R0
1852  005012  105420                  NEGB    (R0)+           ;BYTE 0=1   R0=1
1853  005014  105420                  NEGB    (R0)+           ;BYTE 1=1   R0=2
1854  005016  105340                  DECB    -(R0)           ;R0=1  LOC. 0=01
1855  005020  005300                  DEC     R0              ;R0=0
1856  005022  001401                  BEQ     NEG22
1857  005024  104000                  EMT                     ;REGISTER NOT INCREMENTED CORRECTLY
1858  005026  005337  000000  NEG22:  DEC     @#0             ;LOC. 0=0
1859  005032  001401                  BEQ     TS62
1860  005034  104000                  EMT                     ;NEG BYTE INSTRUCTIONS FAILED
1861
1862                          ;*********************************************************************************
1863                          ;
1864                          ;       THIS TEST VERIFIES MODE 3 SINGLE OPERAND INSTRUCTIONS.  IT
1865                          ;USES LOCATION 0 AS ITS TARGET DATA.  A TABLE LOCATED AT LOC. 400
1866                          ;THRU 402 IS USED TO SUPPLY THE ADDRESS OF LOCATION 0 TO THE
1867                          ;INSTRUCTIONS UNDER TEST.
1868                          ;       R0 IS SET TO 400, THE START OF THE ADDRESS TABLE, AND A CLR
1869                          ;INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR LOC. 0.  THEN R0
1870                          ;IS DECREMENTED BY TWO AND TWO OTHER MODE 3 INSTRUCTIONS OPERATE ON
1871                          ;LOC. 0 TO VERIFY THE DATA RESULTS OF THE TEST.  THE PROPER INCREMENTING
1872                          ;OF THE REGISTER IS ALSO VERIFIED IN THIS MANNER.
1873                          ;       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                     ;CUMMULATIVE 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
        CLR     (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          BEQ     TS64
1931  005140
1932  005140  104000  SOPB3B:
                              EMT                     ;CUMMULATIVE RESULT OF ABOVE INST FAILED
1933
1934  ;*************************************************************************************
1935  ;
1936  ;       THIS TEST VERIFIES MODE 3 SINGLE OPERAND BYTE INSTRUCTIONS
1937  ;WHICH ADDRESS ODD BYTES.  THE TARGET IS BYTE 1.  A TABLE AT
1938  ;LOC. 400-406 IS USED.  R0 SERVES AS THE TABLE POINTER.
1939  ;       R0 IS INITIALIZED TO 400.  LOC. 0 IS SET TO -1 USING THE
1940  ;FIRST TWO TABLE ENTRIES.  A CLRB MODE 3 IS EXECUTED ON BYTE 1 USING
1941  ;TABLE ADDRESS AT 404.  R0 IS DECREMENTED TO 402 AND SEVERAL SOP
1942  ;MODE 3 INSTRUCTIONS ARE USED TO VERIFY DATA RESULTS AND PROPER
1943  ;REGISTER INCREMENTING.
1944  ;       THE TABLE (400-406) SHOULD CONTAIN 0,0,1,1 BEFORE AND
1945  ;AFTER THE TEST IS RUN.
1946  ;
```

```
1947
1948
1949
1950  005142
1951  005142  005000          ;**********************************************************************************
1952  005144  105100          ;TEST 64        TEST MODE 3 ODD BYTE USING SOP INST.
1953  005146  005200          ;**********************************************************************************
1954  005150  005030          TS64:
1955  005152  005130                  CLR     R0              ;SET R0=400
1956  005154  105030                  COMB    R0
1957  005156  001401                  INC     R0
1958  005160  104000                  CLR     @(R0)+          ;INITIALIZE
1959  005162  005300                  COM     @(R0)+          ;LOC 0=-1 R0=404
1960  005164  005300                  CLRB    @(R0)+          ;TRY TO CLEAR ODD BYTE LOC. 0=377 R0=406
1961  005166  005300                  BEQ     SOPB3C
1962                                   EMT                     ;CLRB DID NOT SET Z-BIT
                         SOPB3C: DEC     R0              ;RESET R0=402
                                 DEC     R0
                                 DEC     R0              ;POINT TO EVEN BYTE ADDR.
```