

KD11-Z

11/44 CPU/EIS
CKKAAA0

AH-F620A-MC
COPYRIGHT 1980
FICHE 1 OF 2

JAN 1980
digital
MADE IN USA

The main body of the document is a dense grid of approximately 15 columns and 25 rows of small, illegible data tables or charts. Each cell in the grid contains a small-scale version of the data presented in the header, but the text is too small to be read. The overall appearance is that of a technical report or a data visualization tool from the late 1970s or early 1980s.

KD11-Z

11/44 CPU/EIS
CKKAAA0

AH-F620A-MC
COPYRIGHT 1980
FICHE 2 OF 2

JAN 1980
digital
MADE IN USA

I D E N T I F I C A T I O N

Product Code: AC-F618A-MC
Product Name: CKKAAA0 11/44 CPU/EIS
Date Created: MARCH 1979
Maintainer: Diagnostic Group
Author: CHUCK ROBINSON

Copyright (C) 1979
Digital Equipment Corporation, Maynard, Massachusetts 01754

This software is furnished under a license for use only on a single computer system and may be copied only with the inclusion of the above copyright notice. This software, or any other copies thereof, may not be provided or otherwise made available to any other person except for use on such system and to one who agrees to these license terms. Title to and ownership of the software shall at all times remain in DEC.

The information in this software is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

```
*****  
* SUMMARY OF OPERATING INSTRUCTIONS *  
*  
*****
```

The following procedure can be used to run this diagnostic in a standard configuration with at least 8K of memory and a teletype. If the program does not run successfully consult the following document for assistance.

Operating Procedures:

1. Load the program using normal procedures
2. Start the program at Location 200
3. Program should print 'END OF PASS' within the 1st second and repeatably thereafter at approx. 5 sec. intervals WITH CACHE ON.
(APPROX. 10 SEC-INTERVALS WITH CACHE OFF)
4. If the program does not run as described above, consult the full operating instructions which follow.

1.0 GENERAL PROGRAM INFORMATION

1.1 PROGRAM PURPOSE

This diagnostic program is designed to be a comprehensive check of the PDP-11/44 Basic Instruction Set. The program exercises all of the processor logic and microcode for all instructions except the Trap and Memory Management Instructions. The program does not test instructions or hardware related to the Trap or Interrupt mechanisms of the 11/44 (E.G. RTT, RTI, WAIT, RESET, TRAP, EMT).

NOTE: HOWEVER, IF THE OPTIONAL CPU/CIS TESTS ARE RUN THEN THE BREAKPOINT TRAP INSTRUCTION AND THE ABILITY TO TRAP UNDER ILLEGAL INSTRUCTION (TRAP TO 10) WILL BE ASSUMED TO BE FUNCTIONAL. (SEE SECT. 2.3)

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE

PDP-11/44 Processor
16K Memory -- The program uses Locations 0 - 36000

1.2.2 SOFTWARE

This program is written to be run as a Stand-Alone program. However, the program is designed to run under Automated Product Test System (APT) in all three modes.

The program can also be run under the ACT 11 Monitor

1.3 RELATED DOCUMENTS AND STANDARDS

PDP-11/44 MICROCODE LISTING
PDP-11/44 ELECTRICAL SCHEMATICS

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

None

1.5 FAILURE ASSUMPTIONS

None

2.0 OPERATING INSTRUCTIONS

2.1 Loading And Starting Procedures

2.1.1 LOADING

Use normal procedures for Loading Absolute Binary Tapes.

2.1.2 NORMAL START

This is the procedure for normal program running (I.E., starting with Test 1 and executing entire diagnostic).

START AT ADDRESS = 200

2.1.3 SUBTEST START

This is the procedure for starting at a Subtest other than 1.

1. LOAD \$TESTN (in Mailbox Section) with the number of Subtest minus one (in Octal). For example, to start at Subtest 100, \$TESTN=77.

2. Load Starting Address of Subtest in Loc. 216

3. START AT ADDRESS = 204

2.2 SPECIAL ENVIRONMENTS

This program is written to comply with all the requirements of the APT Interface Specification. It will run under APT in either Quick Verify, program or Run-Time modes.

This program is written to comply with all of the requirements of programs to run under the ACT11 Monitor.

THE EXECUTION TIMES PROVIDED IN THE APT SCRIPT THAT FOLLOWS ARE FOR EXECUTION WITH A 11/44 PROCESSOR, CACHE, 16K CORE MEMORY, AND 300 BAUD.

THE FOLLOWING IS A PROGRAM LOAD FILE USED BY APT:

1. E TABLE 'A' IS USED FOR APT DUMP MODE.
 - A. IN ADDITION TO NORMAL CPU DIAGNOSTIC TESTS THIS TABLE WILL SELECT THE OPTIONAL CACHE AND CIS TESTS. (\$SWREG=400)
2. E TABLE 'B' IS USED FOR APT QV MODE WHILE RUNNING ON A MANUFACTURING QV STATION. IT ACCOMPLISHES WHAT ETABLE 'A' DOES BUT ADDITIONALLY SUPPRESSES TYPEOUTS. (\$ENVM=240)
3. ETABLE 'C' IS USED FOR APT QV OR RUNTIME MODES WHILE RUNNING ON SYSTEMS OTHER THAN MFG. QV STATIONS. THIS TABLE DESELECTS THE OPTIONAL CACHE AND CIS TESTS.

	1ST PASS RUN TIME 10	LONGEST TEST TIME 10	ADDITIONAL RUN TIME 0	
.....		E TABLES	
E -MODE/S-MODE (\$ENVM/\$ENV)		A 200/000	B 240/001	C 240/001
SWITCH REGISTER 1 (\$SWREG)		000400	000400	000000
SWITCH REGISTER 2 CPU TYPE/OPTIONS		000000 00/0000	000000 00/0000	000000 00/0000

2.3 PROGRAM OPTIONS

This program is intended to be a Basic Processor Test. It is intended to be the lowest level diagnostic run. HOWEVER, IT DOES PROVIDE FOR OPTIONAL CACHE AND CIS TESTS. THESE TESTS CAN BE SELECTED AND RUN IN MANUFACTURING ON THE MFG. QV STATION. THE TESTS ARE SPECIALLY DESIGNED TO EXERCISE THOSE SIGNALS ON THE CPU DATA PATH/CONTROL MODULES WHICH RELATE TO THE CACHE AND CIS. IT IS ASSUMED THAT CACHE AND CIS MODULES ARE KNOWN GOOD. RUNNING THESE TESTS ELIMINATE HAVING TO RUN THE CACHE AND CIS DIAGNOSTICS IN THE CPU APT QUICK VERIFY SCRIPT.

THE OPTIONAL TESTS ARE SELECTED THROUGH APT SCRIPTING OR BY LOADING BIT08 OF HARDWARE SWITCH REGISTER(177570). SEE SECTION 2.2.1.

2.4 EXECUTION TIMES

The diagnostic completes the first pass in less than 1 sec. Subsequent PRINTING OF END OF PASS MESSAGE REQUIRE APPROXIMATELY. 5 TO 10 SECS. INTERVALS
The program will run continuously until externally Halted.

3.0 ERROR INFORMATION

3.1 ERROR TYPES

There are two basic types of Errors in the diagnostic.

3.1.1 FUNCTIONAL ERRORS

These are Errors which represent a malfunction of an Instruction or Sequence of Instruction. (E.G., the proper condition code not set or improper result of an Arithmetic or Logical Operation).

3.1.2 SEQUENCE ERRORS

The result of a Test being executed out of Sequence. (E.G. Wild Machine or improper Branch or Jump).

3.2 ERROR REPORTING PROCEDURES

The diagnostic responds to the detection of all Errors by storing certain information in Memory and Halting the Processor. The information stored in Memory can be used by the operator to identify the Error detected.

Certain failures will cause the Processor to Hang.

H 1

This type of failure is indicated if the program does not print its END OF PASS indication within a reasonable amount of time. (First message should appear within 1 sec.)

SEQ 0007

3.3 ERROR DESCRIPTOR INFORMATION

The diagnostic Mailbox holds the Error information necessary to identify the detected Error. This information has been designed for compliance with the APT diagnostic interface specification. It is the primary medium for identifying Errors.

3.3.1 \$MSGTRP

This Location is incremented from zero to one before the program comes to a programmed Halt. If this Location is not one, then the diagnostic has come to an unprogrammed Halt. Check the Stack and PC for a clue to the cause. Suspect a Trap.

3.3.2 \$FATAL

This Location is Loaded with a number before a Halt is executed. Each programmed Halt has a unique number associated with it which can be used to identify the Error which has been detected.

3.3.3 \$PASS

This Location is incremented for every complete pass of the diagnostic. Monitoring the Location will indicate whether or not the program is Hung. It will also indicate the number of successful passes completed before the Error Halt. A high pass count might indicate that the Error Halt is associated with an intermittent fault.

3.3.4 \$TESTN

This Location is incremented in each new Subtest. This should indicate the Test being executed when the Error was detected. This Location is also used to detect a Sequence Error.

3.4 ERROR IDENTIFICATION

Because of the overhead associated with each Halt in an APT compatible program the sequence check code will share the Error Halt of Functional Error within each Subtest. To determine which Error is being reported, Locations \$FATAL and \$TESTN are used together. When an Error Halt occurs, check \$FATAL to determine the number of the Error detected. Now, check that the test number where this Error is detected corresponds to the value in \$TESTN. If these agree the Error was a Functional Error as described in the Listings. If these numbers do not agree, then a Sequence Error was detected. In this case \$TESTN will contain one more than the number of the last Test successfully completed. Sequence Errors which share the Error Halts of Functional Errors will always be reported by the last Halt in Subtest in which they were discovered.

4.0 PROGRESS REPORT

The message CKKAAA0 11/44 CPU/EIS is printed on the console Teletype after the first PASS, and following every subsequent 400 PASSES.

5.0 TROUBLE SHOOTING

When the program discovers a fault it will Halt. to determine the cause of the Halt, the diagnostic provides Error information. This information is stored in the API mailbox and is the primary source of Error identification.

Upon finding an Error, the following procedure should aid in isolating the fault.

5.1 CHECK THE MAILBOX

1. \$MSGTY This Location should contain a 1. if the Processor Halts and this Location is zero, then the processor has come to an unexpected Halt. First suspect a Trap. Check the PC and if a Trap check R6 and the stack for the Location of the failing instruction.
2. \$FATAL This location is used to hold the number of the error which has detected. Each Error being checked by the diagnostic is assigned a unique number which is stored in \$FATAL when that Error is detected.

When an Error is detected, check the listing to see that the Error number stored in \$FATAL is one which is detected in the test whose number is in \$TESTN. If there is a disagreement then the Error being reported is a Sequence Error. \$TESTN contains one more than the last test which was successfully completed.

3. \$TESTN This Location is used to indicate the number of the test which was being executed when the fault was detected. \$TESTN is used in conjunction with \$FATAL to distinguish between Sequence and Functional Errors. (See 2. this Section)
4. \$PASS This Location is used to indicate the number of successful passes which the diagnostic has completed. This will give an indication that the diagnostic has not just been Hung in a Loop

If an Error has been detected \$PASS will show whether it was a Hard Error discovered during the first try or whether it was intermittent or developed during the running of the diagnostic.

5.2 SCOPING

While this diagnostic is primarily intended to be a fault detection program, provisions are made to assist a technician who might want to use the program as a trouble shooting test.

The procedure for scoping a Subtest involves modifying several Memory Locations in the test itself. The philosophy is to provide a Scoping Loop which will include the code where the Error was detected. The Loop is set up so that the Loop will not be terminated should the Error intermittantly disappear.

The procedure is as follows:

1. Determine which Error is to be Scoped. Use \$FATAL and \$TESTN for this (See above)
2. Locate the Error routine in the listing.
3. Clear the right Byte of the Conditional Branch Instruction associated with the Error. (This is marked with <=== 's in the listing.)
4. Replace the Instruction following <MOV #XXX,-(R2)> with the Scoping Branch provided in the listing comments.
5. Restart the program. The program may be restarted from the beginning or from the Subtest (See 2.0).

6.0 LISTING

```

97          .TITLE CKKAAA0 11/44 CPU/EIS
98 000000   .ENABLE ABS
99          001000   STBOT=1000
100         .NLIST  CND,MC,MD
101         .LIST   ME
102         000240   SCOPE=NOP
103         000007   R7=%7
104         000006   R6=%6
105         177776   PS=177776
106         177564   TPS=177564
107         177566   TPB=177566
108         140000   USRM=140000
109         030000   PUSRM=30000
110         177772   PIRQ=177772
111         000020   BIT4=20
112         000007   MFPT=7
120         .MCALL  .SAPTHDR,.$APTBL,.$ACT11
121         .MCALL  .SCATCH,1170
122         .SBTTL  BASIC DEFINITIONS
                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100   STACK= 1100          ;;FIRST ADDRESS OF THE STACK
001100   KERSTK= STACK        ;;KERNEL STACK
000700   SUPSTK= STACK-200    ;;SUPERVISOR STACK
000600   USESTK= STACK-300    ;;USER STACK
104000   ERROR=EMT
000004   SCOPE=IOT
177776   PS= 177776          ;;PROCESSOR STATUS WORD
177776   PSW=PS
177774   STKLMT= 177774      ;;STACK LIMIT REGISTER
177772   PIRQ= 177772       ;;PROGRAM INTERRUPT REQUEST REGISTER
177570   DSWR= 177570       ;;HARDWARE SWITCH REGISTER
177570   DDISP= 177570      ;;HARDWARE DISPLAY REGISTER
177546   LKS= 177546        ;;LINE CLOCK (KW11-L) STATUS REGISTER
                ;*MISCELLANEOUS DEFINITIONS
000011   HT= 11             ;;CODE FOR HORIZONTAL TAB
000012   LF= 12             ;;CODE LINE FEED
000015   CR= 15             ;;CODE CARRIAGE RETURN
000200   CRLF= 200          ;;CODE FOR CARRIAGE RETURN-LINE FEED
                ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000   R0= %0             ;;GENERAL REGISTER
000001   R1= %1             ;;GENERAL REGISTER
000002   R2= %2             ;;GENERAL REGISTER
000003   R3= %3             ;;GENERAL REGISTER
000004   R4= %4             ;;GENERAL REGISTER
000005   R5= %5             ;;GENERAL REGISTER
000006   R6= %6             ;;GENERAL REGISTER
000007   R7= %7             ;;GENERAL REGISTER
000000   R10=R0
000001   R11=R1
000002   R12=R2
000003   R13=R3
000004   R14=R4
000005   R15=R5
000006   SP= %6            ;;STACK POINTER
000006   KSP=SP
000006   SSP=SP
000006   USP=SP

```

```
060007      PC=      %7      ;;PROGRAM COUNTER
000000      ;*PRIORITY LEVEL DEFINITIONS
000040      PR0=      0      ;;PRIORITY LEVEL 0
000100      PR1=      40     ;;PRIORITY LEVEL 1
000140      PR2=      100    ;;PRIORITY LEVEL 2
000200      PR3=      140    ;;PRIORITY LEVEL 3
000240      PR4=      200    ;;PRIORITY LEVEL 4
000300      PR5=      240    ;;PRIORITY LEVEL 5
000340      PR6=      300    ;;PRIORITY LEVEL 6
000340      PR7=      340    ;;PRIORITY LEVEL 7
100000      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
040000      SW15=     100000
020000      SW14=     40000
010000      SW13=     20000
004000      SW12=     10000
002000      SW11=     4000
001000      SW10=     2000
000400      SW09=     1000
000200      SW08=     400
000100      SW07=     200
000040      SW06=     100
000020      SW05=     40
000010      SW04=     20
000004      SW03=     10
000002      SW02=     4
000001      SW01=     2
001000      SW00=     1
000400      SW9=SW09
000200      SW8=SW08
000100      SW7=SW07
000040      SW6=SW06
000020      SW5=SW05
000010      SW4=SW04
000004      SW3=SW03
000002      SW2=SW02
000001      SW1=SW01
000001      SW0=SW00
100000      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
040000      BIT15=   100000
020000      BIT14=   40000
010000      BIT13=   20000
004000      BIT12=   10000
002000      BIT11=   4000
001000      BIT10=   2000
000400      BIT09=   1000
000200      BIT08=   400
000100      BIT07=   200
000040      BIT06=   100
000020      BIT05=   40
000010      BIT04=   20
000004      BIT03=   10
000002      BIT02=   4
000001      BIT01=   2
001000      BIT00=   1
000400      BIT9=BIT09
000200      BIT8=BIT08
000200      BIT7=BIT07
```

```

000100          BIT6=BIT06
000040          BIT5=BIT05
000020          BIT4=BIT04
000010          BIT3=BIT03
000004          BIT2=BIT02
000002          BIT1=BIT01
000001          BIT0=BIT00

;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004          ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
000010          RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014          TBITVEC=14        ;;'T' BIT
000014          TRIVEC= 14         ;;TRACE TRAP
000014          BPTVEC= 14         ;;BREAKPOINT TRAP (BPT)
000020          IOTVEC= 20         ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024          PWRVEC= 24         ;;POWER FAIL
000030          EMTVEC= 30         ;;EMULATOR TRAP (EMT) **ERROR**
000034          TRAPVEC=34        ;;'TRAP' TRAP
000060          TKVEC= 60          ;;TTY KEYBOARD VECTOR
000064          TPVEC= 64          ;;TTY PRINTER VECTOR
000100          LKVEC= 100         ;;LINE CLOCK (KW11-!) VECTOR
000114          CACHVEC=114        ;;CACHE ERROR INTERRUPT VECTOR
000240          PIRQVEC=240        ;;PROGRAM INTERRUPT REQUEST VECTOR
000250          MMVEC= 250         ;;MEMORY MANAGEMENT VECTOR

.SBTTL CACHE REGISTER DEFINITIONS
177740          LOADRS = 177740    ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
177742          HIADRS = 177742    ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
177744          MEMERR = 177744    ;;CACHE ERROR REGISTER
177746          CONTRL = 177746    ;;MEMORY CONTROL REGISTER
177750          MAINT = 177750     ;;MEMORY MAINTENANCE REGISTER
177752          HITMIS = 177752   ;;HIT MISS REGISTER '1' IMPLIES HIT IN CACHE

.SBTTL CPU REGISTER DEFINITIONS
177760          SIZELO = 177760    ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
                                ;;TO GET TO THE LAST 32 WORDS OF MEMORY
177762          SIZEHI = 177762    ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
                                ;;CURRENTLY ALL ZERO
177764          SYSTID = 177764    ;;SYSTEM ID REGISTER
177766          CPUERR = 177766    ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
                                ;;THE TRAP TO ERRVEC (000004)

.SBTTL MEMORY MANAGEMENT DEFINITIONS
;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
177572          MMR0= 177572
177574          MMR1= 177574
177576          MMR2= 177576
172516          MMR3= 172516
177572          SR0=MMR0
177574          SR1=MMR1
177576          SR2=MMR2
172516          SR3=MMR3

;*USER 'I' PAGE DESCRIPTOR REGISTERS
177600          UIPDR0= 177600
177602          UIPDR1= 177602
177604          UIPDR2= 177604
177606          UIPDR3= 177606
177610          UIPDR4= 177610
177612          UIPDR5= 177612
177614          UIPDR6= 177614
177616          UIPDR7= 177616
    
```



```
177620      : *USER 'D' PAGE DESCRIPTOR REGISTORS
177622      UDPDR0= 177620
177624      UDPDR1= 177622
177626      UDPDR2= 177624
177630      UDPDR3= 177626
177632      UDPDR4= 177630
177634      UDPDR5= 177632
177636      UDPDR6= 177634
177636      UDPDR7= 177636
177640      : *USER 'I' PAGE ADDRESS REGISTERS
177642      UIPAR0= 177640
177644      UIPAR1= 177642
177646      UIPAR2= 177644
177650      UIPAR3= 177646
177652      UIPAR4= 177650
177654      UIPAR5= 177652
177656      UIPAR6= 177654
177656      UIPAR7= 177656
177660      : *USER 'D' PAGE ADDRESS REGISTERS
177662      UDPAR0= 177660
177664      UDPAR1= 177662
177666      UDPAR2= 177664
177670      UDPAR3= 177666
177672      UDPAR4= 177670
177674      UDPAR5= 177672
177676      UDPAR6= 177674
177676      UDPAR7= 177676
172200      : *SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
172202      SIPDR0= 172200
172204      SIPDR1= 172202
172206      SIPDR2= 172204
172210      SIPDR3= 172206
172212      SIPDR4= 172210
172214      SIPDR5= 172212
172216      SIPDR6= 172214
172216      SIPDR7= 172216
172220      : *SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
172222      SDPDR0= 172220
172224      SDPDR1= 172222
172226      SDPDR2= 172224
172230      SDPDR3= 172226
172232      SDPDR4= 172230
172234      SDPDR5= 172232
172236      SDPDR6= 172234
172236      SDPDR7= 172236
172240      : *SUPERVISOR 'I' PAGE ADDRESS REGISTERS
172242      SIPAR0= 172240
172244      SIPAR1= 172242
172246      SIPAR2= 172244
172250      SIPAR3= 172246
172252      SIPAR4= 172250
172254      SIPAR5= 172252
172256      SIPAR6= 172254
172256      SIPAR7= 172256
172260      : *SUPERVISOR 'D' PAGE ADDRESS REGISTERS
172262      SDPAR0= 172260
172262      SDPAR1= 172262
```

```
172264 SDPAR2= 172264
172266 SDPAR3= 172266
172270 SDPAR4= 172270
172272 SDPAR5= 172272
172274 SDPAR6= 172274
172276 SDPAR7= 172276
;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
172300 KIPDR0= 172300
172302 KIPDR1= 172302
172304 KIPDR2= 172304
172306 KIPDR3= 172306
172310 KIPDR4= 172310
172312 KIPDR5= 172312
172314 KIPDR6= 172314
172316 KIPDR7= 172316
;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
172320 KDPDR0= 172320
172322 KDPDR1= 172322
172324 KDPDR2= 172324
172326 KDPDR3= 172326
172330 KDPDR4= 172330
172332 KDPDR5= 172332
172334 KDPDR6= 172334
172336 KDPDR7= 172336
;*KERNEL 'I' PAGE ADDRESS REGISTERS
172340 KIPAR0= 172340
172342 KIPAR1= 172342
172344 KIPAR2= 172344
172346 KIPAR3= 172346
172350 KIPAR4= 172350
172352 KIPAR5= 172352
172354 KIPAR6= 172354
172356 KIPAR7= 172356
;*KERNEL 'D' PAGE ADDRESS REGISTERS
172360 KDPAR0= 172360
172362 KDPAR1= 172362
172364 KDPAR2= 172364
172366 KDPAR3= 172366
172370 KDPAR4= 172370
172372 KDPAR5= 172372
172374 KDPAR6= 172374
172376 KDPAR7= 172376
.SBTTL UNIBUS MAP REGISTER DEFINITIONS
;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
170200 MAPL00 = 170200
170202 MAPH00 = 170202
170204 MAPL01 = 170204
170206 MAPH01 = 170206
170210 MAPL02 = 170210
170212 MAPH02 = 170212
170214 MAPL03 = 170214
170216 MAPH03 = 170216
170220 MAPL04 = 170220
170222 MAPH04 = 170222
170224 MAPL05 = 170224
170226 MAPH05 = 170226
```

UNIBUS MAP REGISTER DEFINITIONS

170230	MAPL06 = 170230
170232	MAPH06 = 170232
170234	MAPL07 = 170234
170236	MAPH07 = 170236
170240	MAPL10 = 170240
170242	MAPH10 = 170242
170244	MAPL11 = 170244
170246	MAPH11 = 170246
170250	MAPL12 = 170250
170252	MAPH12 = 170252
170254	MAPL13 = 170254
170256	MAPH13 = 170256
170260	MAPL14 = 170260
170262	MAPH14 = 170262
170264	MAPL15 = 170264
170266	MAPH15 = 170266
170270	MAPL16 = 170270
170272	MAPH16 = 170272
170274	MAPL17 = 170274
170276	MAPH17 = 170276
170300	MAPL20 = 170300
170302	MAPH20 = 170302
170304	MAPL21 = 170304
170306	MAPH21 = 170306
170310	MAPL22 = 170310
170312	MAPH22 = 170312
170314	MAPL23 = 170314
170316	MAPH23 = 170316
170320	MAPL24 = 170320
170320	MAPH24 = 170320
170324	MAPL25 = 170324
170326	MAPH25 = 170326
170330	MAPL26 = 170330
170332	MAPH26 = 170332
170334	MAPL27 = 170334
170336	MAPH27 = 170336
170340	MAPL30 = 170340
170342	MAPH30 = 170342
170344	MAPL31 = 170344
170346	MAPH31 = 170346
170350	MAPL32 = 170350
170352	MAPH32 = 170352
170354	MAPL33 = 170354
170356	MAPH33 = 170356
170360	MAPL34 = 170360
170362	MAPH34 = 170362
170364	MAPL35 = 170364
170366	MAPH35 = 170366
170370	MAPL36 = 170370
170372	MAPH36 = 170372
170374	MAPL37 = 170374
170376	MAPH37 = 170376
170200	MAPL0=MAPL00
170202	MAPH0=MAPH00
170204	MAPL1=MAPL01
170206	MAPH1=MAPH01
170210	MAPL2=MAPL02

170212
 170214
 170216
 170220
 170222
 170224
 170226
 170230
 170232
 170234
 170236

MAPH2=MAPH02
 MAPL3=MAPL03
 MAPH3=MAPH03
 MAPL4=MAPL04
 MAPH4=MAPH04
 MAPL5=MAPL05
 MAPH5=MAPH05
 MAPL6=MAPL06
 MAPH6=MAPH06
 MAPL7=MAPL07
 MAPH7=MAPH07

123 000000

.SBITL TRAP CATCHER
 .=0
 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ''.+2,HALT''
 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

000174 000000
 000176 000000

.=174
 DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER

124

.SBITL ACT11 HOOKS
 ;*****

000200
 000046 036040
 000052 000000
 000200
 000300

:HOOKS REQUIRED BY ACT11
 \$SVPC- ;SAVE PC
 .=46
 \$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP
 .-52
 .WORD 0 ;:2)SET LOC.52 TO ZERO
 .=\$\$VPC ;: RESTORE PC
 .-300

125
 126

.SBITL APT MAILBOX-ETABLE
 ;*****

000300
 000300 000000
 000302 000000
 000304 000000
 000306 000000
 000310 000000
 000312 000000
 000314 000000
 000316 000000
 000320
 000320 000
 000321 000
 000322 000000
 000324 000000
 000326 000000

.EVEN
 \$MAIL: ;:APT MAILBOX
 \$MSGTY: .WORD AMSGTY ;:MESSAGE TYPE CODE
 \$FATAL: .WORD AFATAL ;:FATAL ERROR NUMBER
 \$TESTN: .WORD ATESTN ;:TEST NUMBER
 \$PASS: .WORD APASS ;:PASS COUNT
 \$DEVCT: .WORD ADEVCT ;:DEVICE COUNT
 \$UNIT: .WORD AUNIT ;:I/O UNIT NUMBER
 \$MSGAD: .WORD AMSGAD ;:MESSAGE ADDRESS
 \$MSGLG: .WORD AMSGLG ;:MESSAGE LENGTH
 \$ETABLE: ;:APT ENVIRONMENT TABLE
 \$ENV: .BYTE AENV ;:ENVIRONMENT BYTE
 \$ENVM: .BYTE AENVM ;:ENVIRONMENT MODE BITS
 \$SWREG: .WORD ASWREG ;:APT SWITCH REGISTER
 \$USWR: .WORD AUSWR ;:USER SWITCHES
 \$CPUOP: .WORD ACPUOP ;:CPU TYPE,OPTIONS
 ;*
 ;* BITS 15-11=CPU TYPE
 ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
 ;* 11/70=06,PDQ=07,Q=10
 ;*
 ;* BIT 10=REAL TIME CLOCK
 ;* BIT 9=FLOATING POINT PROCESSOR
 ;* BIT 8=MEMORY MANAGEMENT

000330

\$ETFND:
 .MEXIT

127

.SBITL APT PARAMETER BLOCK
 ;*****

```

:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
000024 000330      .$.X=.      ;;SAVE CURRENT LOCATION
000044 000024      .=-24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000044 000200      200          ;;FOR APT START UP
000044 000044      .-44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044 000330      $APTHDR    ;;POINT TO APT HEADER BLOCK
000044 000330      .=$.X      ;;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC
000330 $APTHD:
000330 000000 $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
000332 000300 $MBADR: .WORD $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
000334 000010 $TSTM: .WORD 10      ;;RUN TIM OF LONGEST TEST
000336 000010 $PASTM: .WORD 10    ;;RUN TIME IN SECS. OF 1,1T PASS ON 1 UNIT (QUICK VERIFY)
000340 000000 $UNITM: .WORD 0      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000342 000014      .WORD $LTEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
:*****
:*****
: SOME POINTERS TO CPU TRAP HANDLERS
:*****
128      .=4
129      T04
130      0
131      T010
132 000004 036370      0
133 000006 000000      T014
134 000010 036400      0
135 000012 000000      .=30
136 000014 036410      T030
137 000016 000000      0
138      000030      T034
139 000030 036420      0
140 000032 000000      .=114
141 000034 036430      T0114
142 000036 000000      0
143      000114      .-244
144 000114 036440      T0244
145 000116 000000      0
146      000244      T0250
147 000244 036450      0
148 000246 000000      T0250
149 000250 036460      0
150 000252 000000      0
151
152
153 :*****
154 :DATA TABLE FOR USE IN ADDRESSING MODE TESTS
155 :*****
156 000370 000370      .=370
156 000370 000000 000000 000000      0,0,0,0,0,0
157 000376 000000 000000 000000      1,1,-1
158 000404 000001 000001 177777      .-1100
159
:*****
:SET UP STARTING ADDRESS
161      001100      .$.X-.
162      000200      .=-200
163 000200 000167 000674      JMP      START
164
165 000204 012706 001000      MOV      #STBOT,R6      ;SET STACK POINTER

```

```

166 000210 012702 000304          MOV    #STESTN,R2          ;SET MAILBOX POINTER
167 000214 000137          JMP    @PC+              ;JUMP TO SUBTEST
168 000216 000000          C                      ;ADDR. OF SUBTEST GOES HERE
169
170          001100
171          000302
172          000304          .=.SX
                                $ERROR-$FATAL
                                $TSTNM=$TESTN
173 001100 012737 036254 000024      START: MOV    #PWRDN,@#24          ;SET UP FOR POWER FAIL
174 001106 012737 000000 000306      MOV    #0,@#SPASS        ;CLEAR PASS COUNT
175 001114 012737 177777 036066      MOV    #-1,@#PASSPT      ;SET PRINT COUNTER
176 001122 012706 001000          RESTRT: MOV   #STBOT,R6    ;INITIALIZE STACK POINTER
177 001126 012702 000304          MOV    #STESTN,R2        ;SET UP POINTER TO MESSAGE TYPE
178 001132 012737 000000 000304      MOV    #0,@#TSTNM        ;CLEAR TEST NUMBER
179 001140 012737 000000 000302      MOV    #0,@#ERROR        ;CLEAR ERROR NUMBER
180 001146 012737 000000 000300      MOV    #0,@#MSGTY        ;CLEAR MESSAGE TYPE (FOR APT)

```

182

:TEST 1 CHECK BRANCHES ON Z BIT

001154 005212
001156 022712 000001
001162 001024
183 001164 000257
184 001166 001401
185 001170 000404

TST1: INC (R2) ;UPDATE TEST NUMBER
CMP #1,(R2) ;SEQUENCE ERROR?
BNE TST2-10 ;BR TO ERROR HALT ON SEQ ERROR
CCC ;CLEAR ALL CONDITION CODES
BEQ BR1 ;SHOULD BRANCH
BR BR2 ;BAD BRANCH OF Z-BIT
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; BRANCH INSTRUCTION AND <====
; REPLACE THE MOVE INSTRUCTION <====
; FOLLOWING W/ 774 <====

186
187
188
189
190 001172
001172 012742 000001
001176 005242
001200 000000
191 001202
001202 001004

BR1: MOV #1,-(R2) ;MOVE TO MAILBOX # ***** 1 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;SHOULD HAVE BRANCHED: Z=0
BR2: BNF BR3
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 767 <====

001204 012742 000002
001210 005242
001212 000000
192 001214 000264
193 001216 001001
194 001220 000404

BR3: MOV #2,-(R2) ;MOVE TO MAILBOX # ***** 2 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
SEZ
BNE BR4
BR BR5
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; BRANCH INSTRUCTION AND <====
; REPLACE THE MOVE INSTRUCTION <====
; FOLLOWING W/ 760 <====

195
196
197
198
199 001222
001222 012742 000003
001226 005242
001230 000000
200 001232
001232 001404

BR4: MOV #3,-(R2) ;MOVE TO MAILBOX # ***** 3 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;SHOULD NOT HAVE BRANCHED HERE ON Z=1
BR5: BFO TST2
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
; CONDITIONAL BRANCH INST. AND < =
; REPLACE THE MOVE INSTRUCTION < -
; WHICH FOLLOWS W/ 753 < -

001234 012742 000004
001240 005242
001242 000000

MOV #4,-(R2) ;MOVE TO MAILBOX # ***** 4 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;SHOULD HAVE BRANCHED ON Z=1
; OR SEQUENCE ERROR

201
202
203
204
205
206
207
208
209
210

:SBTTL DATA PATH TESTS

: THE DATA PATH TESTS ARE USED TO VERIFY THAT VARIOUS
:DATA PATTERNS CAN BE SUCCESSFULLY MOVED THROUGH THE DATA PATHS
:MOVE AND COMPARE MODE 2,3 INSTRUCTIONS ARE USED TO PASS AND
:TEST VARIOUS DATA PATTERNS IN THE DATA PATHS.
: THE TEST EXERCISES THE INTERNAL DATA PATHS, THE UNIBUS
:DATA TRANSCIEVERS, AND AMUX CONTROL FOR ALU AND UBUS INPUTS.

211
212
213
214
215
216

: IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0)
: TO SEE WHICH BITS OF THE DATA PATH ARE FAILING. IF THIS PROVIDES
: INCONCLUSIVE DATA, TRY TO CHECK MODE 3 IR DECODE BY RUNNING
: JUST THE MICROCODE AND IR DECODE TESTS FOR THE MOVE AND COMPARE
: INSTRUCTIONS.

: TEST 2 TEST OF ZEROES IN THE DATA PATH

001244 005212
001246 022712 000002
001252 001006
217 001254 012737 000000 000000
218
219 001262 005737 000000
220 001266 001404

TST2: INC (R2) ;UPDATE TEST NUMBER
CMP #2,(R2) ;SEQUENCE ERROR?
BNE TST3-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #0,@#0 ;MOVE ZEROES THRU ADDRESS LINES, DATA
;LINES AND INTERNAL PATHS
TST @#0 ;SUCCESSFUL?
BEQ TST3

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
: CONDITIONAL BRANCH INST. AND <- --
: REPLACE THE MOVE INSTRUCTION <-
: WHICH FOLLOWS W/ 771 <-
: MOVE TO MAILBOX # ***** 5 *****
: SET MSGTYP TO FATAL ERROR
: DATA INCORRECT
: OR SEQUENCE ERROR

001270 012742 000005
001274 005242
001276 000000

MOV #5,-(R2)
INC -(R2)
HALT

221
222

: TEST 3 TEST OF PATTERN 125252 IN DATA PATH

001300 005212
001302 022712 000003
001306 001007
223 001310 012737 125252 000000
224
225 001316 022737 125252 000000
226 001324 001404

TST3: INC (R2) ;UPDATE TEST NUMBER
CMP #3,(R2) ;SEQUENCE ERROR?
BNE TST4-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #125252,@#0 ;MOVE ALTERNATING ONES AND ZEROES
;THRU DATA PATHS
CMP #125252,@#0 ;SUCCESSFUL
BEQ TST4

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
: CONDITIONAL BRANCH INST. AND <=====
: REPLACE THE MOVE INSTRUCTION <=====
: WHICH FOLLOWS W/ 770 <=====
: MOVE TO MAILBOX # ***** 6 *****
: SET MSGTYP TO FATAL ERROR
: DATA INCORRECT
: OR SEQUENCE ERROR

001326 012742 000006
001332 005242
001334 000000

MOV #6,-(R2)
INC -(R2)
HALT

227
228

: TEST 4 TEST OF PATTERN 052525 IN DATA PATH

001336 005212
001340 022712 000004
001344 001007
229 001346 012737 052525 000000
230
231 001354 022737 052525 000000
232 001362 001404

TST4: INC (R2) ;UPDATE TEST NUMBER
CMP #4,(R2) ;SEQUENCE ERROR?
BNE TST5-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #052525,@#0 ;MOVE ALTERNATING ZEROES AND ONES
;THRU DATA PATH
CMP #052525,@#0 ;SUCCESSFUL?
BEQ TST5

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <==--=
: CONDITIONAL BRANCH INST. AND <==--=
: REPLACE THE MOVE INSTRUCTION <- -
: WHICH FOLLOWS W/ 770 <- -


```

001364 012742 000007      MOV    #7,-(R2)      ;MOVE TO MAILBOX # ***** 7 *****
001370 005242      INC    -(R2)        ;SET MSGTYP TO FATAL ERROR
001372 000000      HALT                ;DATA INCORRECT
                                ; OR SEQUENCE ERROR

```

233
234

```

:*****
:TEST 5 TEST OF ALL ONES IN DATA PATH
:*****

```

```

001374 005212      TST5: INC    (R2)      ;UPDATE TEST NUMBER
001376 022712 000005      CMP    #5,(R2)      ;SEQUENCE ERROR?
001402 001007      BNE   TST6-10      ;BR TO ERROR HALT ON SEQ ERROR
235 001404 012737 177777 000000      MOV    #177777,@#0  ;MOVE ONES THRU DATA PATH
236 001412 022737 177777 000000      CMP    #177777,@#0  ;SUCCESSFUL
237 001420 001404      BEQ   TST6

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 770 <====

```

```

001422 012742 000010      MOV    #10,-(R2)    ;MOVE TO MAILBOX # ***** 10 *****
001426 005242      INC    -(R2)        ;SET MSGTYP TO FATAL ERROR
001430 000000      HALT                ;DATA INCORRECT
                                ; OR SEQUENCE ERROR

```

238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254

```

:*****
:SBTTL B-REGISTER TEST
:

```

```

: THE B-REGISTER SHIFTING LOGIC TESTS ARE USED TO TEST THAT THE
:B-REGISTER CAN HOLD VARIOUS DATA PATTERNS AND THAT THE ASSOCIATED
:LOGIC SUPPORTS THE SHIFTING FUNCTIONS WITHIN THE B-REGISTER AND C-BIT.
:A ONE IS SHIFTED THROUGH EVERY BIT IN THE B-REGISTER AND C-BIT IN
: BOTH DIRECTIONS.
: THE B-REGISTER ITSELF IS TESTED IN ITS ABILITY AS A BUFFER AND AS
:A SHIFT REGISTER. DATA IS ALSO PASSED THROUGH THE DATA PATH AND ALU.
: IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0) TO SEE
: WHICH BITS OF THE B-REGISTER MAY BE FAILING. IF THIS PROVIDES
: INCONCLUSIVE DATA TRY TO CHECK THE MODE 3 IR DECODE BY RUNNING JUST
: THE MICROCODE AND IR DECODE TESTS FOR THE PARTICULAR INSTRUCTIONS.

```

```

:*****
:TEST 6 SHIFT BIT 0 TO BIT 1
:*****

```

```

001432 005212      TST6: INC    (R2)      ;UPDATE TEST NUMBER
001434 022712 000006      CMP    #6,(R2)      ;SEQUENCE ERROR?
001440 001012      BNE   TST7-10      ;BR TO ERROR HALT ON SEQ ERROR
255 001442 000241      CLC                ;CLEAR CARRY BIT
256 001444 012737 000001 000000      MOV    #1,@#0      ;LOAD A 1
257 001452 006137 000000      ROL    @#0          ;SHIFT LEFT
258 001456 022737 000002 000000      CMP    #2,@#0      ;SUCCESSFUL
259 001464 001404      BEQ   TST7

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 765 <====

```

```

001466 012742 000011      MOV    #11,-(R2)   ;MOVE TO MAILBOX # ***** 11 *****
001472 005242      INC    -(R2)        ;SET MSGTYP TO FATAL ERROR
001474 000000      HALT                ;BIT 1 NOT SET
                                ; OR SEQUENCE ERROR

```

260
261

:TEST 7 SHIFT CARRY INTO BIT 0

001476 005212
001500 022712 000007
001504 001017
262 001506 012737 000000 000000
263 001514 000261
264 001516 006137 000000
265 001522 103014

TST7: INC (R2) ;UPDATE TEST NUMBER
CMP #7,(R2) ;SEQUENCE ERROR?
BNE TST10-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #0,@#0 ;CLEAR LOCATION
SEC ;SET CARRY
ROL @#0 ;ROTATE CARRY BIT TO BIT 0
BCC TST10

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 770 <====

001524 012742 000012
001530 005242
001532 000000
266 001534 022737 000001 000000
267 001542 001404

MOV #12,-(R2) ;MOVE TO MAILBOX # ***** 12 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CARRY CLEAR
; OR SEQUENCE ERROR
CMP #1,@#0 ;BIT 0 SET
BFQ TST10

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 760 <====

001544 012742 000013
001550 005242
001552 000000

MOV #13,-(R2) ;MOVE TO MAILBOX # ***** 13 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;BIT 0 NOT SET
; OR SEQUENCE ERROR

268
269

:TEST 10 LEFT SHIFT FROM BIT 0 TO C-BIT

001554 005212
001556 022712 000010
001562 001014
270 001564 012737 000001 000000
271 001572 012700 177757
272 001576 000241
273 001600 005200
274 001602 001404
275 001604 006137 000000
276 001610 103373
277 001612 001404

TST10: INC (R2) ;UPDATE TEST NUMBER
CMP #10,(R2) ;SEQUENCE ERROR?
BNE TST11-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #1,@#0 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
SHL: INC R0 ;INCREMENT BIT COUNTER
BEQ SHLE ;BR TO ERROR HALT IF BIT IS LOST
ROL @#0 ;SHIFT LEFT ONE POSITION
BCC SHL ;BRANCH IF C-BIT NOT SET
BEQ TST11

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 763 <====

001614
001614 012742 000014
001620 005242
001622 000000

SHLE: MOV #14,-(R2) ;MOVE TO MAILBOX # ***** 14 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;LEFT SHIFTING LOGIC FAILED
; OR SEQUENCE ERROR

278
279

:TEST 11 SHIFT BIT 15 TO BIT 14

```

001624 005212          TST11: INC      (R2)          ;UPDATE TEST NUMBER
001626 022712 000011  CMP      #11,(R2)       ;SEQUENCE ERROR?
001632 001012          BNE      TST12-10      ;BR TO ERROR HALT ON SEQ ERROR
280 001634 012737 100000 000000  MOV     #100000,@#0     ;SET BIT 15
281 001642 000241          CLC                    ;CLEAR CARRY
282 001644 006037 000000          ROR     @#0             ;SHIFT BIT 15 TO BIT 14
283 001650 022737 040000 000000  CMP     #40000,@#0     ;SUCCESSFUL
284 001656 001404          BEQ     TST12

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
; CONDITIONAL BRANCH INST. AND <===
; REPLACE THE MOVE INSTRUCTION <= --
; WHICH FOLLOWS W/ 765 <---
; MOVE TO MAILBOX # ***** 15 *****
; SET MSGTYP TO FATAL ERROR
; BIT 14 NOT SET
; OR SEQUENCE ERROR

```

```

001660 012742 000015          MOV     #15,-(R2)
001664 005242          INC     -(R2)
001666 000000          HALT

```

285
286

```

;*****
;TEST 12 RIGHT SHIFT FROM BIT 15 TO C-BIT
;*****

```

```

001670 005212          TST12: INC      (R2)          ;UPDATE TEST NUMBER
001672 022712 000012  CMP     #12,(R2)       ;SEQUENCE ERROR?
001676 001014          BNE     TST13-10      ;BR TO ERROR HALT ON SEQ ERROR
287 001700 012737 100000 000000  MOV     #100000,@#0     ;SET BIT 15
288 001706 012700 177757          MOV     #-21,R0        ;SET BIT COUNTER
289 001712 000241          CLC                    ;CLEAR C-BIT
290 001714 005200          SHR:  INC     R0         ;INCREMENT BIT COUNTER
291 001716 001404          BEQ     SHRE          ;BR TO ERROR HALT IF BIT IS LOST
292 001720 006037 000000          ROR     @#0             ;ROTATE RIGHT ONE POSITION
293 001724 103373          BCC     SHR           ;BRANCH IF C-BIT CLEAR
294 001726 001404          BEQ     TST13

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
; CONDITIONAL BRANCH INST. AND <---
; REPLACE THE MOVE INSTRUCTION < --
; WHICH FOLLOWS W/ 763 < =

```

```

001730          SHRE:  MOV     #16,-(R2)   ;MOVE TO MAILBOX # ***** 16 *****
001730 012742 000016  INC     -(R2)       ;SET MSGTYP TO FATAL ERROR
001734 005242          HALT           ;RIGHT SHIFT LOGIC FAILED
001736 000000          ; OR SEQUENCE ERROR

```

295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311

```

;*****
;SBTTL SCRATCH PAD TESTS
;*****

```

```

; THE SCRATCH PAD TESTS ARE USED TO VERIFY THAT VARIOUS
; DATA PATTERNS CAN BE SUCCESSFULLY HELD IN THE SCRATCH PAD
; CIRCUIT . 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.

```

312
313
314
315
316
317
318
319
320
321
322
323

: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. REGISTERS 10 AND 12 HAVE BEEN ACCESSED BY
: THE INSTRUCTIONS. REGISTERS 13,14,AND 17 WILL BE TESTED LATER IN THE
: MICROCODE TESTS.
: IF THE PATTERN TESTS WITH REGISTER 0 FAIL CHECK THE RESULTANT
: DATA FOR A CLUE TO A FAULT IN THE EXTERNAL CIRCUITRY. IF THE
: PATTERN TESTS WITH R0 ARE SUCCESSFUL BUT THE TESTS WITH THE OTHER
: REGISTERS FAIL, SUSPECT THE REGISTER SELECT LINES AND THEN THE SCRATCH
: PAD ITSELF.

:TEST 13 TEST IF R0 CAN HOLD ALL ZEROES

001740 005212
001742 022712 000013
001746 001004

TST13: INC (R2) ;UPDATE TEST NUMBER
CMP #13,(R2) ;SEQUENCE ERROR?
BNE TST14-10 ;BR TO ERROR HALT ON SEQ ERROR

324
325 001750 012700 000000
326 001754 005700
327 001756 001404

MOV #0,R0 ;MOVE ZEROES TO R0
TST R0 ;SUCCESSFUL?
BEQ TST14

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- -
: CONDITIONAL BRANCH INST. AND <-
: REPLACE THE MOVE INSTRUCTION <
: WHICH FOLLOWS W/ 773 <

001760 012742 000017
001764 005242
001766 000000

MOV #17,-(R2) ;MOVE TO MAILBOX # ***** 17 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;R0 NOT 0
; OR SEQUENCE ERROR

328
329

:TEST 14 TEST IF R0 CAN HOLD ONES AND ZEROES

001770 005212
001772 022712 000014
001776 001005
330 002000 012700 125252
331 002004 020027 125252
332 002010 001404

TST14: INC (R2) ;UPDATE TEST NUMBER
CMP #14,(R2) ;SEQUENCE ERROR?
BNE TST15-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #125252,R0 ;MOVE ALTERNATING ONES AND ZEROES TO R0
CMP R0,#125252 ;SUCCESSFUL?
BEQ TST15

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
: CONDITIONAL BRANCH INST. AND <
: REPLACE THE MOVE INSTRUCTION <
: WHICH FOLLOWS W/ 772 <

002012 012742 000020
002016 005242
002020 000000

MOV #20,-(R2) ;MOVE TO MAILBOX # ***** 20 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;R0 NOT 125252
; OR SEQUENCE ERROR

333
334

:TEST 15 TEST IF R0 CAN HOLD ZEROES AND ONES

002022 005212
002024 022712 000015
002030 001005
335 002032 012700 052525
336 002036 020027 052525
337 002042 001404

TST15: INC (R2) ;UPDATE TEST NUMBER
CMP #15,(R2) ;SEQUENCE ERROR?
BNE TST16-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #052525,R0 ;MOVE ALTERNATING ZEROES AND ONES TO R0
CMP R0,#052525 ;SUCCESSFUL?
BEQ TST16

002044 012742 000021
002050 005242
002052 000000

MOV #21,-(R2)
INC -(R2)
HALT

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 772 <====
: MOVE TO MAILBOX # ***** 21 *****
: SET MSGTYP TO FATAL ERROR
: R0 NOT 52525
: OR SEQUENCE ERROR

338
339

:TEST 16 TEST IF R0 CAN HOLD ALL ONES

002054 005212
002056 022712 000016
002062 001005
340 002064 012700 177777
341 002070 020027 177777
342 002074 001404

TST16: INC (R2)
CMP #16,(R2)
BNE TST17-10
MOV #177777,R0
CMP R0,#177777
BEQ TST17

:UPDATE TEST NUMBER
:SEQUENCE ERROR?
:BR TO ERROR HALT ON SEQ ERROR
:MOVE ALL ONES TO R0
:SUCCESSFUL?

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 772 <====
: MOVE TO MAILBOX # ***** 22 *****
: SET MSGTYP TO FATAL ERROR
: R0 NOT 177777
: OR SEQUENCE ERRO

343
344

:TEST 17 TEST IF R1 CAN HOLD A ONE IN ALL BITS

002106 005212
002110 022712 000017
002114 001012
345 002116 012701 000001
346 002122 012700 177757
347 002126 000241
348 002130 005200
349 002132 001403
350 002134 006101
351 002136 103374
352 002140 001404

TST17: INC (R2)
CMP #17,(R2)
BNE TST20-10
MOV #1,R1
MOV #-21,R0
CLC
REG1: INC R0
SEQ REG1E
ROL R1
BCC REG1
BEQ TST20

:UPDATE TEST NUMBER
:SEQUENCE ERROR?
:BR TO ERROR HALT ON SEQ ERROR
:SET BIT 0
:SET BIT COUNTER
:CLEAR C-BIT
:INCREMENT BIT COUNTER
:BR TO ERROR HALT IF BIT IS LOST
:ROTATE 1 POSITION
:ALL DONE

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 765 <====

002142
002142 012742 000023
002146 005242
002150 000000

REG1E: MOV #23,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 23 *****
:SET MSGTYP TO FATAL ERROR
:FAILURE WITH R1
: OR SEQUENCE ERROR

353
354

:TEST 20 TEST IF R1 CAN HOLD A ZERO IN ALL BITS

002152 005212
002154 022712 000020
002160 001014

TST20: INC (R2)
CMP #20,(R2)
BNE TST21-10

:UPDATE TEST NUMBER
:SEQUENCE ERROR?
:BR TO ERROR HALT ON SEQ ERROR

```

355 002162 012701 177776      MOV    #-2,R1      ;SET ALL ONES IN R1 EXCEPT FOR BIT 0
356 002166 012700 177757      MOV    #-21,R0     ;SET BIT COUNTER
357 002172 000261              SEC                ;SET C-BIT
358 002174 005200              REG1A: INC    R0     ;INCREMENT COUNTER
359 002176 001405              BEQ    R1ERR       ;BR TO ERROR HALT IF COUNTER=0
360 002200 006101              ROL    R1          ;ROTATE 1 POSITION
361 002202 103774              BCS    REG1A       ;CONTINUE UNTIL C-BIT IS CLEAR
362 002204 022701 177777      CMP    #-1,R1      ;CHECK DATA IN R1
363 002210 001404              BEQ    TST21

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 763 <====

```

```

002212
002212 012742 000024      R1ERR: MOV    #24,-(R2) ;MOVE TO MAILBOX # ***** 24 *****
002216 005242              INC    -(R2)       ;SET MSGTYP TO FATAL ERROR
002220 000000              HALT               ;FAILURE WITH R1
; OR SEQUENCE ERROR

```

364

```

;*****
;TEST 21 TEST IF R2 CAN HOLD A ONE IN ALL BITS
;*****

```

```

002222 005212              TST21: INC    (R2)      ;UPDATE TEST NUMBER
002224 022712 000021      CMP    #21,(R2)    ;SEQUENCE ERROR?
002230 001012              BNE    REG2A-14    ;BR TO ERROR HALT ON SEQ ERROR
365 002232 012702 000001      MOV    #1,R2       ;SET BIT 0
366 002236 012700 177757      MOV    #-1,R0      ;SET BIT COUNTER
367 002242 000241              CLC                ;CLEAR C-BIT
368 002244 005200              REG2: INC    R0     ;INCREMENT BIT COUNTER
369 002246 001403              BEQ    REG2A-14    ;BR TO ERROR HALT IF BIT IS LOST
370 002250 006102              ROL    R2          ;ROTATE 1 POSITION
371 002252 103374              BCC    REG2        ;ALL DONE
372 002254 001406              BEQ    REG2A

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; BRANCH INSTRUCTION AND <====
; REPLACE THE MOVE INSTRUCTION <====
; FOLLOWING W/ 771 <====

```

```

377 002256 012702 000304      MOV    #STESTN,R2  ;RESTORE POINTER
378 002262 012742 000025      MOV    #25,-(R2)   ;MOVE TO MAILBOX # ***** 25 *****
002266 005242              INC    -(R2)       ;SET MSGTYP TO FATAL ERROR
002270 000000              HALT               ;FAILURE WITH R2
379 002272 012702 000304      REG2A: MOV    #STESTN,R2 ;RESTORE POINTER

```

380
381

```

;*****
;TEST 22 TEST IF R2 CAN HOLD A ZERO IN ALL BITS
;*****

```

```

002276 005212              TST22: INC    (R2)      ;UPDATE TEST NUMBER
002300 022712 000022      CMP    #22,(R2)    ;SEQUENCE ERROR?
002304 001020              BNE    TST23-10    ;BR TO ERROR HALT ON SEQ ERROR
382 002306 012702 177776      MOV    #-2,R2       ;SET ALL ONES IN R2 EXCEPT FOR BIT 0
383 002312 012700 177757      MOV    #-21,R0     ;SET BIT COUNTER
384 002316 000261              SEC                ;SET C-BIT
385 002320 005200              REG2B: INC    R0     ;INCREMENT BIT COUNTER
386 002322 001407              BEQ    R2ERR       ;BR TO ERROR HALT IF COUNTER=0
387 002324 006102              ROL    R2          ;ROTATE 1 POSITION
388 002326 103774              BCS    REG2B       ;CONTINUE UNTIL C-BIT IS CLEAR
389 002330 022702 177777      CMP    #-1,R2      ;CHECK DATA IN R2
390 002334 001406              BEQ    REG2C

```

```
391 002336 012702 000304          MOV    #STESTN,R2      ;RESTORE POINTER
392 002342          R2ERR:  MOV    #26,-(R2)      ;MOVE TO MAILBOX # ***** 26 *****
      002342 012742 000026          INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
      002346 005242          HALT                   ;FAILURE WITH R2
393 002352 012702 000304          REG2C: MOV   #STESTN,R2      ;RESTORE POINTER
```

:TEST 23 TEST IF R3 CAN HOLD A ONE IN ALL BITS

```
      002356 005212          TST23: INC    (R2)          ;UPDATE TEST NUMBER
      002360 022712 000023          CMP    #23,(R2)      ;SEQUENCE ERROR?
      002364 001012          BNE   TST24-10       ;BR TO ERROR HALT ON SEQ ERROR
396 002366 012703 000001          MOV    #1,R3         ;SET BIT 0
397 002372 012700 177757          MOV    #-21,R0       ;SET BIT COUNTER
398 002376 000241          CLC                    ;CLEAR C-BIT
399 002400 005200          REG3:  INC    R0        ;INCREMENT BIT COUNTER
400 002402 001403          BEQ   REG3E          ;BR TO ERROR HALT IF BIT IS LOST
401 002404 006103          ROL   R3             ;ROTATE 1 POSITION
402 002406 103374          BCC   REG3           ;ALL DONE
403 002410 001404          BEQ   TST24          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
      ;                               CONDITIONAL BRANCH INST. AND <-
      ;                               REPLACE THE MOVE INSTRUCTION .
      ;                               WHICH FOLLOWS W/ 765 <---
```

```
      002412          REG3E:  MOV    #27,-(R2)      ;MOVE TO MAILBOX # ***** 27 *****
      002412 012742 000027          INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
      002416 005242          HALT                   ;FAILURE WITH R3
      002420 000000          ; OR SEQUENCE ERROR
```

404
405 *****
:TEST 24 TEST IF R3 CAN HOLD A ZERO IN ALL BITS

```
      002422 005212          TST24: INC    (R2)          ;UPDATE TEST NUMBER
      002424 022712 000024          CMP    #24,(R2)      ;SEQUENCE ERROR?
      002430 001014          BNE   TST25-10       ;BR TO ERROR HALT ON SEQ ERROR
406 002432 012703 177776          MOV    #-2,R3        ;SET ALL ONES IN R3 EXCEPT FOR BIT 0
407 002436 012700 177757          MOV    #-21,R0       ;SET BIT COUNTER
408 002442 000261          SEC                    ;SET C-BIT
409 002444 005200          REG3A: INC    R0        ;INCREMENT BIT COUNTER
410 002446 001405          BEQ   R3ERR          ;BR TO ERROR HALT IF COUNTER=0
411 002450 006103          ROL   R3             ;ROTATE 1 POSITION
412 002452 103774          BCS   REG3A          ;CONTINUE UNTIL C-BIT IS CLEAR
413 002454 022703 177777          CMP    #-1,R3        ;CHECK DATA
414 002460 001404          BEQ   TST25          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
      ;                               CONDITIONAL BRANCH INST. AND <====
      ;                               REPLACE THE MOVE INSTRUCTION <====
      ;                               WHICH FOLLOWS W/ 763 <====
```

```
      002462          R3ERR:  MOV    #30,-(R2)      ;MOVE TO MAILBOX # ***** 30 *****
      002462 012742 000030          INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
      002466 005242          HALT                   ;FAILURE WITH R3
      002470 000000          ; OR SEQUENCE ERROR
```

415
416 *****

```

;TEST 25          TEST IF R4 CAN HOLD A ONE IN ALL BITS
;*****
TST25:  INC      (R2)          ;UPDATE TEST NUMBER
        CMP      #25,(R2)     ;SEQUENCE ERROR?
        BNE     TST26-10     ;BR TO ERROR HALT ON SEQ ERROR
        MOV     #1,R4        ;SET BIT 0
        MOV     #-21,R0      ;SET BIT COUNTER
        CLC                    ;CLEAR C-BIT
REG4:   INC      R0           ;INCREMENT BIT COUNTER
        BEQ     REG4E        ;BR TO ERROR HALT IF BIT IS LOST
        ROL     R4           ;ROTATE 1 POSITION
        BCC     REG4        ;ALL DONE
        BEQ     TST26
  
```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;           CONDITIONAL BRANCH INST. AND <====
;           REPLACE THE MOVE INSTRUCTION <====
;           WHICH FOLLOWS W/ 765 <====
  
```

```

002526 005212 000025
002526 012742 000031
002526 022712 000026
002532 001012
002534 000000
        MOV     #31,-(R2)    ;MOVE TO MAILBOX # ***** 31 *****
        INC     -(R2)       ;SET MSGTYP TO FATAL ERROR
        HALT                    ;FAILURE WITH R4
                                ; OR SEQUENCE ERROR
  
```

425
426

```

;*****
;TEST 26          TEST IF R4 CAN HOLD A ZERO IN ALL BITS
;*****
TST26:  INC      (R2)          ;UPDATE TEST NUMBER
        CMP      #26,(R2)     ;SEQUENCE ERROR?
        BNE     TST27-10     ;BR TO ERROR HALT ON SEQ ERROR
        MOV     #-2,R4        ;SET ALL ONES IN R4 EXCEPT FOR BIT 0
        MOV     #-21,R0      ;SET BIT COUNTER
        SEC                    ;SET C-BIT
REG4A:  INC      R0           ;INCREMENT BIT COUNTER
        BEQ     R4ERR        ;BR TO ERROR HALT IF COUNTER=0
        ROL     R4           ;ROTATE 1 POSITION
        BCS     REG4A        ;CONTINUE UNTIL C-BIT IS CLEAR
        CMP     #-1,R4       ;CHECK DATA
        BEQ     TST27
  
```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;           CONDITIONAL BRANCH INST. AND <====
;           REPLACE THE MOVE INSTRUCTION <====
;           WHICH FOLLOWS W/ 763 <====
  
```

```

002576 005212 000032
002576 012742 000032
002602 005242
002604 000000
        MOV     #32,-(R2)    ;MOVE TO MAILBOX # ***** 32 *****
        INC     -(R2)       ;SET MSGTYP TO FATAL ERROR
        HALT                    ;FAILURE WITH R4
                                ; OR SEQUENCE ERROR
  
```

436
437
438

```

;*****
;TEST 27          TEST IF R5 CAN HOLD A ONE IN ALL BITS
;*****
TST27:  INC      (R2)          ;UPDATE TEST NUMBER
        CMP      #27,(R2)     ;SEQUENCE ERROR?
        BNE     TST30-10     ;BR TO ERROR HALT ON SEQ ERROR
        MOV     #1,R5        ;SET BIT 0
        MOV     #-21,R0      ;SET BIT COUNTER
  
```



```

441 002626 000241          CLC          :CLEAR C-BIT
442 002630 005200          REG5:  INC      R0          :INCREMENT BIT COUNTER
443 002632 001403          BEQ      REG5E         :BR TO ERROR HALT IF BIT IS LOST
444 002634 006105          ROL      R5           :ROTATE 1 POSITION
445 002636 103374          BCC      REG5         :ALL DONE
446 002640 001404          BEQ      TST30

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 765 <====

```

```

002642          REG5E:
002642 012742 000033      MOV      #33,-(R2)    :MOVE TO MAILBOX # ***** 33 *****
002646 005242          INC      -(R2)     :SET MSGTYP TO FATAL ERROR
002650 000000          HALT                    :FAILURE WITH R5
: OR SEQUENCE ERROR

```

447
448

```

:*****
:TEST 30          TEST IF R5 CAN HOLD A ZERO IN ALL BITS
:*****

```

```

002652 005212          TST30: INC      (R2)          :UPDATE TEST NUMBER
002654 022712 000030      CMP      #30,(R2)     :SEQUENCE ERROR?
002660 001014          BNE      TST31-10    :BR TO ERROR HALT ON SEQ ERROR
449 002662 012705 177776  MOV      #-2,R5       :SET ALL ONES IN R5 EXCEPT FOR BIT 0
450 002666 012700 177757  MOV      #-21,R0      :SET BIT COUNTER
451 002672 000261          SEC                    :SET C-BIT
452 002674 005200          REG5A: INC      R0          :INCREMENT BIT COUNTER
453 002676 001405          BEQ      R5ERR       :BR TO ERROR HALT IF COUNTER=0
454 002700 006105          ROL      R5          :ROTATE 1 POSITION
455 002702 103774          BCS      REG5A        :CONTINUE UNTIL C-BIT IS CLEAR
456 002704 022705 177777  CMP      #-1,R5       :CHECK DATA
457 002710 001404          BEQ      TST31

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 763 <====

```

```

002712          R5ERR:
002712 012742 000034      MOV      #34,-(R2)    :MOVE TO MAILBOX # ***** 34 *****
002716 005242          INC      -(R2)     :SET MSGTYP TO FATAL ERROR
002720 000000          HALT                    :FAILURE WITH R5
: OR SEQUENCE ERROR

```

458
459

```

:*****
:TEST 31          TEST IF R6 CAN HOLD A ONE IN ALL BITS
:*****

```

```

002722 005212          TST31: INC      (R2)          :UPDATE TEST NUMBER
002724 022712 000031      CMP      #31,(R2)     :SEQUENCE ERROR?
002730 001012          BNE      TST32-10    :BR TO ERROR HALT ON SEQ ERROR
460 002732 012706 000001  MOV      #1,R6        :SET BIT 0
461 002736 012700 177757  MOV      #-21,R0      :SET BIT COUNTER
462 002742 000241          REG6: CLC          :CLEAR C-BIT
463 002744 005200          INC      R0          :INCREMENT BIT COUNTER
464 002746 001403          BEQ      REG6E       :BR TO ERROR HALT IF BIT IS LOST
465 002750 006106          ROL      R6          :ROTATE 1 POSITION
466 002752 103374          BCC      REG6        :ALL DONE
467 002754 001404          BEQ      TST32

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====

```

002756
002756 012742 000035
002762 005242
002764 000000

REG6E: MOV #35,-(R2)
INC -(R2)
HALT

: REPLACE THE MOVE INSTRUCTION <===
: WHICH FOLLOWS W/ 765 <===
: MOVE TO MAILBOX # ***** 35 *****
: SET MSGTYP TO FATAL ERROR
: FAILURE WITH P6
: OR SEQUENCE ERROR

468
469

: TEST 32 TEST IF R6 CAN HOLD A ZERO IN ALL BITS

002766 005212
002770 022712 000032
002774 001014
470 002776 012706 177776
471 003002 012700 177757
472 003006 000261
473 003010 005200
474 003012 001405
475 003014 006106
476 003016 103774
477 003020 022706 177777
478 003024 001404

TST32: INC (R2)
CMP #32,(R2)
BNE TST33-10
MOV #-2,R6
MOV #-21,R0
REG6A: INC R0
BEQ R6ERR
ROL R6
BCS REG6A
CMP #-1,R6
BEQ TST33

: UPDATE TEST NUMBER
: SEQUENCE ERROR?
: BR TO ERROR HALT ON SEQ ERROR
: SET ALL ONES IN R6 EXCEPT FOR BIT 0
: SET BIT COUNTER
: SET C-BIT
: INCREMENT BIT COUNT
: BR TO ERROR HALT IF COUNTER-0
: ROTATE 1 POSITION
: CONTINUE UNTIL C-BIT IS CLEAR
: CHECK DATA
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
: CONDITIONAL BRANCH INST. AND <===
: REPLACE THE MOVE INSTRUCTION <===
: WHICH FOLLOWS W/ 763 <===

003026
003026 012742 000036
003032 005242
003034 000000

R6ERR: MOV #36,-(R2)
INC -(R2)
HALT

: MOVE TO MAILBOX # ***** 36 *****
: SET MSGTYP TO FATAL ERROR
: FAILURE WITH R6
: OR SEQUENCE ERROR

479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495

: 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 ITSELF IS TESTED AS WELL AS THE ADDRESS
: SELECT CIRCUITRY. THE AMUX INPUTS TO THE PSW MUX ARE 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 33 TEST IF PSW WILL HOLD ZEROES

003036 005212
003040 022712 000033
003044 001010
496 003046 012706 001000
497 003052 012737 000000 177776
498 003060 005737 177776

TST33: INC (R2)
CMP #33,(R2)
BNE TST34-10
MOV #STBOT,R6
MOV #0,@#PS
TST @#PS

: UPDATE TEST NUMBER
: SEQUENCE ERROR?
: BR TO ERROR HALT ON SEQ ERROR
: SET PSW TO ZERO
: SUCCESSFUL

```

499 003064 001404          BEQ      TST34
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;       CONDITIONAL BRANCH INST. AND <====
                                ;       REPLACE THE MOVE INSTRUCTION <====
                                ;       WHICH FOLLOWS W/ 767 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 37 *****
003066 012742 000037      MOV      #37,-(R2)
003072 005242              INC      -(R2)
003074 000000              HALT
                                ; SET MSGTYP TO FATAL ERROR
                                ; PSW NOT 0
                                ; OR SEQUENCE ERROR
  
```

500
501

```

:*****
:TEST 34      TEST IF PSW WILL HOLD ONES AND ZEROES
:*****
  
```

```

003076 005212              INC      (R2)
003100 022712 000034      CMP      #34,(R2)
003104 001007              BNE     TST35-10
502 003106 012737 000252 177776  MOV     #252,@#PS
503 003114 023727 177776 000252  CMP     @#PS,#252
504 003122 001404          BEQ     TST35
                                ; UPDATE TEST NUMBER
                                ; SEQUENCE ERROR?
                                ; BR TO ERROR HALT ON SEQ ERROR
                                ; MOVE ALT. ONES AND ZEROES TO PSW
                                ; SUCCESSFUL?
                                ;
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;       CONDITIONAL BRANCH INST. AND <====
                                ;       REPLACE THE MOVE INSTRUCTION <====
                                ;       WHICH FOLLOWS W/ 770 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 40 *****
003124 012742 000040      MOV      #40,-(R2)
003130 005242              INC      -(R2)
003132 000000              HALT
                                ; SET MSGTYP TO FATAL ERROR
                                ; PSW NOT 252
                                ; OR SEQUENCE ERROR
  
```

505
506

```

:*****
:TEST 35      TEST IF PSW (EXCEPT T-BIT) WILL HOLD 0'S & 1'S
:*****
  
```

```

003134 005212              INC      (R2)
003136 022712 000035      CMP      #35,(R2)
003142 001007              BNE     TST36-10
507 003144 012737 000105 177776  MOV     #105,@#PS
508 003152 023727 177776 000105  CMP     @#PS,#105
509 003160 001404          BEQ     TST36
                                ; UPDATE TEST NUMBER
                                ; SEQUENCE ERROR?
                                ; BR TO ERROR HALT ON SEQ ERROR
                                ; MOVE ALT. ONES AND ZEROES TO PSW
                                ; SUCCESSFUL?
                                ;
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;       CONDITIONAL BRANCH INST. AND <====
                                ;       REPLACE THE MOVE INSTRUCTION <====
                                ;       WHICH FOLLOWS W/ 770 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 41 *****
003162 012742 000041      MOV      #41,-(R2)
003166 005242              INC      -(R2)
003170 000000              HALT
                                ; SET MSGTYP TO FATAL ERROR
                                ; PSW NOT 105
                                ; OR SEQUENCE ERROR
  
```

510
511

```

:*****
:TEST 36      TEST IF PSW (EXCEPT T-BIT) WILL HOLD ALL ONES
:*****
  
```

```

003172 005212              INC      (R2)
003174 022712 000036      CMP      #36,(R2)
003200 001007              BNE     TST37-10
512 003202 012737 000357 177776  MOV     #357,@#PS
513 003210 023727 177776 000357  CMP     @#PS,#357
514 003216 001404          BEQ     TST37
                                ; UPDATE TEST NUMBER
                                ; SEQUENCE ERROR?
                                ; BR TO ERROR HALT ON SEQ ERROR
                                ; MOVE ONES TO PSW
                                ; SUCCESSFUL
                                ;
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;       CONDITIONAL BRANCH INST. AND <====
  
```

```

003220 012742 000042      MOV    #42,-(R2)      ;REPLACE THE MOVE INSTRUCTION <====
003224 005242              INC    -(R2)         ;WHICH FOLLOWS W/ 770 <----
003226 000000              HALT                ;MOVE TO MAILBOX # ***** 42 *****
                                ;SET MSGTYP TO FATAL ERROR
                                ;PSW NOT 357
                                ; OR SEQUENCE ERROR
  
```

515 .SBTTL CONDITION CODE TEST

516
517
518
519
520
521
522
523
524
525
526
527
528
529
530

```

:*****
:
:   THIS TEST CHECKS THE CONDITIONAL BRANCHES INVOLVING THE Z-BIT.
: THE Z-BIT IS SET WITH ALL OTHER CC BITS ZERO AND BOTH CONDITIONS
: BEQ AND BNE ARE TESTED FOR PROPER EXECUTION. THEN THE Z-BIT IS
: SET WITH ALL OTHER CC BITS CLEAR AND BOTH CONDITIONS ARE TESTED
: AGAIN FOR PROPER OPERATION.
:   THIS TEST CHECKS THE OPERATION OF THE SET AND CLEAR CONDITION
: CODE INSTRUCTIONS AND CHECKS THE CIRCUITRY EXTERNAL TO THE CONDITIONAL
: BRANCH ROM. THE BRANCH MICROCODE FOR ALTERING THE PC AND FOR
: LEAVING THE PC UNALTERED IS TESTED. ONLY THOSE ROM ADDRESSES SPECIFICALLY
: USED IN THE TEST ARE VERIFIED HERE.
:*****
  
```

:TEST 37 TEST BRANCHES AROUND Z-BIT

```

003230 005212 000037      TST37: INC    (R2)          ;UPDATE TEST NUMBER
003232 022712              CMP    #37,(R2)      ;SEQUENCE ERROR?
003236 001014              BNE   TST40-10      ;BR TO ERROR HALT ON SEQ ERROR
531      ;FIRST WITH Z-BIT OFF
532 003240 000257              CCC                ;CC-0100: JUST Z-BIT
533 003242 000264              SEZ
534 003244 001001              BNE   BRZ1          ;CHECK OPPOSITE CONDITION
535 003246 001404              BEQ   BRZ2
  
```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 773 <====
  
```

```

003250 012742 000043      BRZ1: MOV    #43,-(R2)      ;MOVE TO MAILBOX # ***** 43 *****
003250 005242              INC    -(R2)         ;SET MSGTYP TO FATAL ERROR
003256 000000              HALT                ;IMPROPER BR W/ Z=1
536      ;CHECK WITH Z-BIT OFF
537 003260 000277      BRZ2: SCC
538 003262 000244              CLZ
539 003264 001401              BEQ   BRZ3
540 003266 001004              BNE   TST40
  
```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 763 <====
  
```

```

003270 012742 000044      BRZ3: MOV    #44,-(R2)      ;MOVE TO MAILBOX # ***** 44 *****
003270 005242              INC    -(R2)         ;SET MSGTYP TO FATAL ERROR
003276 000000              HALT                ;IMPROPER BR W/ Z=0
                                ; OR SEQUENCE ERROR
  
```

541
542
543

```

:*****
:
  
```

544
545
546
547
548
549
550
551
552
553
554
555

```

: THIS TEST CHECKS THE CONDITIONAL BRANCHES INVOLVING THE N-BIT.
: THE N-BIT IS SET WITH ALL OTHER CC BITS ZERO AND BOTH CONDITIONS
: BMI AND BPL ARE TESTED FOR PROPER EXECUTION. THEN THE N-BIT IS
: SET WITH ALL OTHER CC BITS CLEAR AND BOTH CONDITIONS ARE TESTED
: AGAIN FOR PROPER OPERATION.
: THIS TEST CHECKS THE OPERATION OF THE SET AND CLEAR CONDITION
: CODE INSTRUCTIONS AND CHECKS THE CIRCUITRY EXTERNAL TO THE CONDITIONAL
: BRANCH ROM. THE BRANCH MICROCODE FOR ALTERING THE PC AND FOR
: LEAVING THE PC UNALTERED IS TESTED. ONLY THOSE ROM ADDRESSES SPECIFICALLY
: USED IN THE TEST ARE VERIFIED HERE.
  
```

 : TEST 40 TEST BRANCHES AROUND N-BIT

003300 005212
 003302 022712 000040
 003306 001014
 556
 557 003310 000257
 558 003312 000270
 559 003314 100001
 560 003316 100404

```

TST40: INC (R2) ;UPDATE TEST NUMBER
        CMP #40,(R2) ;SEQUENCE ERROR?
        BNE TST41-10 ;BR TO ERROR HALT ON SEQ ERROR
        ;FIRST WITH N-BIT ON
        CCC ;CC=1000: JUST N-BIT
        SEN
        BPL BRN1 ;CHECK OPPOSITE CONDITION
        BMI BRN2
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < - -
: CONDITIONAL BRANCH INST. AND < - -
: REPLACE THE MOVE INSTRUCTION < - -
: WHICH FOLLOWS W/ 773 < - -
  
```

003320
 003320 012742 000045
 003324 005242
 003326 000000
 561
 562 003330 000277
 563 003332 000250
 564 003334 100401
 565 003336 100004

```

BRN1: MOV #45,-(R2) ;MOVE TO MAILBOX # ***** 45 *****
        INC -(R2) ;SET MSGTYP TO FATAL ERROR
        HALT ;IMPROPER BR W/ N=?
        ;CHECK WITH N-BIT OFF
BRN2: SCC ;CC=0111
        CLN
        BMI BRN3 ;CHECK OPPOSITE CONDITION
        BPL TST41
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < - -
: CONDITIONAL BRANCH INST. AND < -
: REPLACE THE MOVE INSTRUCTION <
: WHICH FOLLOWS W/ 763 <
  
```

003340
 003340 012742 000046
 003344 005242
 003346 000000

```

BRN3: MOV #46,-(R2) ;MOVE TO MAILBOX # ***** 46 *****
        INC -(R2) ;SET MSGTYP TO FATAL ERROR
        HALT ;IMPROPER BR W/ N=0
        ; OR SEQUENCE ERROR
  
```

566
567
568
569
570
571
572
573
574
575
576
577
578

```

: *****
: THIS TEST CHECKS THE CONDITIONAL BRANCHES INVOLVING THE V-BIT.
: THE V-BIT IS SET WITH ALL OTHER CC BITS ZERO AND BOTH CONDITIONS
: BVS AND BVC ARE TESTED FOR PROPER EXECUTION. THEN THE V-BIT IS
: SET WITH ALL OTHER CC BITS CLEAR AND BOTH CONDITIONS ARE TESTED
: AGAIN FOR PROPER OPERATION.
: THIS TEST CHECKS THE OPERATION OF THE SET AND CLEAR CONDITION
: CODE INSTRUCTIONS AND CHECKS THE CIRCUITRY EXTERNAL TO THE CONDITIONAL
: BRANCH ROM. THE BRANCH MICROCODE FOR ALTERING THE PC AND FOR
: LEAVING THE PC UNALTERED IS TESTED. ONLY THOSE ROM ADDRESSES SPECIFICALLY
: USED IN THE TEST ARE VERIFIED HERE.
  
```

579
580

:TEST 41 TEST BRANCHES AROUND V-BIT

003350 005212
003352 022712 000041
003356 001014

TST41: INC (R2) ;UPDATE TEST NUMBER
CMP #41,(R2) ;SEQUENCE ERROR?
BNE TST42-10 ;BR TO ERROR HALT ON SEQ ERROR
;FIRST WITH V-BIT ON
CCC ;CC=0010: JUST V-BIT
SEV
BVC BRV1 ;CHECK OPPOSITE CONDITION
BVS BRV2

581
582 003360 000257
583 003362 000262
584 003364 102001
585 003366 102404

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-- -
: CONDITIONAL BRANCH INST. AND <-
: REPLACE THE MOVE INSTRUCTION <- -
: WHICH FOLLOWS W/ 773 <-

003370
003370 012742 000047
003374 005242
003376 000000

BRV1: MOV #47,-(R2) ;MOVE TO MAILBOX # ***** 47 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;IMPROPER BR W/ V-1
;CHECK WITH V-BIT OFF

586
587 003400 000277
588 003402 000242
589 003404 102401
590 003406 102004

BRV2: SCC ;CC=1101: ALL BVT V-BIT
CLV
BVS BRV3 ;CHECK OPPOSITE CONDITION
BVC TST42

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
: CONDITIONAL BRANCH INST. AND <=====
: REPLACE THE MOVE INSTRUCTION <=====
: WHICH FOLLOWS W/ 763 <=====
:

003410
003410 012742 000050
003414 005242
003416 000000

BRV3: MOV #50,-(R2) ;MOVE TO MAILBOX # ***** 50 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;IMPROPER BR W/ V=0
; OR SEQUENCE ERROR

591
592
593
594
595
596
597
598
599
600
601
602
603
604
605

: THIS TEST CHECKS THE CONDITIONAL BRANCHES INVOLVING THE C-BIT.
: THE C-BIT IS SET WITH ALL OTHER CC BITS ZERO AND BOTH CONDITIONS
: BCS AND BCC ARE TESTED FOR PROPER EXECUTION. THEN THE C-BIT IS
: SET WITH ALL OTHER CC BITS CLEAR AND BOTH CONDITIONS ARE TESTED
: AGAIN FOR PROPER OPERATION.
: THIS TEST CHECKS THE OPERATION OF THE SET AND CLEAR CONDITION
: CODE INSTRUCTIONS AND CHECKS THE CIRCUITRY EXTERNAL TO THE CONDITIONAL
: BRANCH ROM. THE BRANCH MICROCODE FOR ALTERING THE PC AND FOR
: LEAVING THE PC UNALTERED IS TESTED. ONLY THOSE ROM ADDRESSES SPECIFICALLY
: USED IN THE TEST ARE VERIFIED HERE.

:TEST 42 TEST BRANCHES AROUND C-BIT

003420 005212
003422 022712 000042
003426 001014

TST42: INC (R2) ;UPDATE TEST NUMBER
CMP #42,(R2) ;SEQUENCE ERROR?
BNE TST43-10 ;BR TO ERROR HALT ON SEQ ERROR
;FIRST WITH C-BIT ON
CCC ;CC=0001: JUST C-BIT
SEC

606
607 003430 000257
608 003432 000261

```

609 003434 103001          BCC   BRC1          ;CHECK OPPOSITE CONDITION
610 003436 103404          BCS   BRC2          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 773 <====

        003440          BRC1:
003440 012742 000051      MOV   #51,-(R2)      ;MOVE TO MAILBOX # ***** 51 *****
003444 005242          INC   -(R2)      ;SET MSGTYP TO FATAL ERROR
003446 000000          HALT          ;IMPROPER BR W/ C=1

611
612 003450 000277      BRC2:  SCC          ;CC=1110
613 003452 000241      CLC
614 003454 103401      BCS   BRC3          ;CHECK OPPOSITE CONDITION
615 003456 100404      BMI   TST43
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 763 <====

        003460          BRC3:
003460 012742 000052      MOV   #52,-(R2)      ;MOVE TO MAILBOX # ***** 52 *****
003464 005242          INC   -(R2)      ;SET MSGTYP TO FATAL ERROR
003466 000000          HALT          ;IMPROPER BR W/ C=0
; OR SEQUENCE ERROR

```

616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648

:SBTTL MICROCODE TESTS

```

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

```

649
650
651
652

:OF THE SOP INSTRUCTIONS IN THIS TEST. THE DATA IN THIS TEST IS
:OPERATED ON BY EACH INSTRUCTION WITHOUT REINITIALIZING.

:TEST 43 TEST MODE 0 USING SOP INST.

003470 005212
003472 022712 000043
003476 001020
653 003500 005000
654 003502 001404

TST43: INC (R2) :UPDATE TEST NUMBER
CMP #43,(R2) :SEQUENCE ERROR?
BNE TST44-10 :BR TO ERROR HALT ON SEQ ERROR
CLR R0 :TRY THE CLEAR INST.
BEQ SOPOA

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 775 <====

003504 012742 000053
003510 005242
003512 000000
655 003514 005200
656 003516 005100
657 003520 005200
658 003522 100404

SOPOA: MOV #53,-(R2) :MOVE TO MAILBOX # ***** 53 *****
INC -(R2) :SET MSGTYP TO FATAL ERROR
HALT :CLR DID NOT SET Z-BIT
SOPOA: INC R0 :TRY THE INCREMENT INST.
COM R0 :TRY COMPLEMENT
INC R0
BMI SOPOB

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 765 <====

003524 012742 000054
003530 005242
003532 000000
659 003534 005100
660 003536 001404

SOPOB: MOV #54,-(R2) :MOVE TO MAILBOX # ***** 54 *****
INC -(R2) :SET MSGTYP TO FATAL ERROR
HALT :NEGATE DID NOT SET N-BIT
SOPOB: COM R0 :TRY COMPLEMENT INST.
BEQ TST44

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 757 <====

003540 012742 000055
003544 005242
003546 000000

MOV #55,-(R2) :MOVE TO MAILBOX # ***** 55 *****
INC -(R2) :SET MSGTYP TO FATAL ERROR
HALT :CUMULATIVE RESULT OF CLR,INC,NEG AND COM INSTS. FAILED
: OR SEQUENCE ERROR

661
662
663
664
665
666
667
668
669
670
671
672

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

:TEST 44 TEST REMAINDER OF SOP INSTS IN MODE 0

003550 005212
003552 022712 000044
003556 001021
673 003560 005000

TST44: INC (R2) :UPDATE TEST NUMBER
CMP #44,(R2) :SEQUENCE ERROR?
BNE TST45-10 :BR TO ERROR HALT ON SEQ ERROR
CLR R0 :INITIALIZE


```

674 003562 005300          DEC      R0          ;TRY DECREMENT INST.
675 003564 100404          BMI      SOPOC          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 774 <====
                                ; MOVE TO MAILBOX # ***** 56 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; N-BIT NOT SET ON DEC
                                ; INITIALIZE CARRY
                                ; TRY ADD CARRY INST
                                ; INITIALIZE CARRY
                                ; TRY SUBTRACT-CARRY INST

003566 012742 000056          MOV      #56,-(R2)
003572 005242              INC      -(R2)
003574 000000              HALT
676 003576 000261          SOPOC: SEC
677 003600 005500          ADC      R0
678 003602 001007          BNE     SOPOD
679 003604 000261          SEC
680 003606 005600          SBC     R0
681 003610 100004          BPL     SOPOD
682 003612 005100          COM     R0
683 003614 005200          INC     R0
684 003616 005300          DEC     R0
685 003620 001404          BEQ     TST45
    
```

```

003622 012742 000057          SOPOD: MOV      #57,-(R2)
003622 005242              INC      -(R2)
003630 000000              HALT
                                ; MOVE TO MAILBOX # ***** 57 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; CUMULATIVE RESULT OF ADC,SBC,COM,INC AND DEC INSTS. FAILE
                                ; OR SEQUENCE ERROR
    
```

```

686
687
688
689
690
691
692
693
:*****
: THIS TEST INTRODUCES THE BYTE CONTROL LOGIC OF THE PROCESSOR.
: THE MODE 0 BYTE MICROCODE IS TESTED. THE METHOD AND SEQUENCE
: OF TESTING IS THE SAME AS THAT USED IN THE SOP MODE 0 TESTS.
:*****
:TEST 45 TEST MODE 0 EVEN BYTE USING SOP INST
:*****
    
```

```

003632 005212 000045          TST45: INC      (R2)
003634 002712              CMP      #45,(R2)
003640 001012              BNE     TST46-10
694 003642 105000          CLR     R0
695 003644 001404          BEQ     SOPBOA
                                ; UPDATE TEST NUMBER
                                ; SEQUENCE ERROR?
                                ; BR TO ERROR HALT ON SEQ ERROR
                                ; TRY CLEARING EVEN BYTE OF REGISTER
    
```

```

003646 012742 000060          SOPBOA: MOV      #60,-(R2)
003652 005242              INC      -(R2)
003654 000000              HALT
696 003656 105100          COM     R0
697 003660 100002          BPL     SOPBOB
698 003662 105200          INCB   R0
699 003664 001404          BEQ     TST46
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 775 <====
                                ; MOVE TO MAILBOX # ***** 60 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; CLRB DID NOT SET Z-BIT
                                ; TRY SETTING EVEN BYTE OF REGISTER
                                ; TRY INCREMENTING EVEN BYTE OF REGISTER>>
    
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
    
```

003666
003666 012742 000061
003672 005242
003674 000000

SOPB08:

MOV #61,-(R2)
INC -(R2)
HALT

WHICH FOLLOWS W/ 765 <====
:MOVE TO MAILBOX # ***** 61 *****
:SET MSGTYP TO FATAL ERROR
:TEST CUMMULATIVE RESULT OF ABOVE BYTE INST.
: OR SEQUENCE ERROR

700
701
702
703
704
705
706
707
708
709

: 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 46 TEST MODE 1 USING SOP INST.

003676 005212
003700 022712 000046
003704 001014
710 003706 005000
711 003710 005010
712 003712 001404

TST46: INC (R2)
CMP #46,(R2)
BNF TST47-10
CLR R0
CLR (R0)
BEQ SOP1A

:UPDATE TEST NUMBER
:SEQUENCE ERROR?
:BR TO ERROR HALT ON SEQ ERROR
:INITIALIZE R0
:TRY CLEAR INST W/MODE 1

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 774 <====

003714 012742 000062
003720 005242
003722 000000
713 003724 005310
714 003726 100003
715 003730 000261
716 003732 005510
717 003734 001404

SOP1A:

MOV #62,-(R2)
INC -(R2)
HALT
DEC (R0)
BPL SOP1B
SEC
ADC (R0)
BEQ TST47

:MOVE TO MAILBOX # ***** 62 *****
:SET MSGTYP TO FATAL ERROR
:CLR DID NOT SET Z-BIT
:TRY DECREMENT INST W/MODE 1

:INITIALIZE CARRY
:TRY ADD-CARRY W/MODE 1

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 763 <====

003736
003736 012742 000063
003742 005242
003744 000000

SOP1B:

MOV #63,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 63 *****
:SET MSGTYP TO FATAL ERROR
:TEST CUMMULATIVE RESULT OF ABOVE INST
: OR SEQUENCE ERROR

718
719
720
721
722
723
724
725
726

: 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 47 TEST MODE 1 EVEN BYTE USING SOP INST

003746 005212

TST47: INC (R2) ;UPDATE TEST NUMBER

```

003750 022712 000047      CMP      #47,(R2)      ;SEQUENCE ERROR?
003754 001020      BNE      TST50-10     ;BR TO ERROR HALT ON SEQ ERROR
727 003756 005000      CLR      R0           ;INITIALIZE R0
728 003760 005010      CLR      (R0)         ;INITIALIZE LOC. 0
729 003762 005110      COM      (R0)
730 003764 105010      CLRB     (R0)         ;TRY TO CLEAR BYTE 0
731 003766 001404      BEQ      SOPB1A

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <--
;          CONDITIONAL BRANCH INST. AND <---
;          REPLACE THE MOVE INSTRUCTION <  =
;          WHICH FOLLOWS W/ 772 <
;
003770 012742 000064      MOV      #64,-(R2)    ;MOVE TO MAILBOX # ***** 64 *****
003774 005242      INC      -(R2)       ;SET MSGTYP TO FATAL ERROR
003776 000000      HALT
732 004000 005210      SOPB1A: INC      (R0)   ;CLRB DID NOT SET Z-BIT
733 004002 100005      BPL      SOPB1B      ;INCREMENT TO TEST WORD
734 004004 105110      COMB     (R0)         ;COMPLEMENT: ODD BYTE = 376
735 004006 105210      INCB     (R0)         ;INC: ODD BYTE = 377
736 004010 100002      BPL      SOPB1B
737 004012 105210      INCB     (R0)         ;INCREMENT ODD BYTE=0
738 004014 001404      BEQ      TST50

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---
;          CONDITIONAL BRANCH INST. AND <---
;          REPLACE THE MOVE INSTRUCTION <  =
;          WHICH FOLLOWS W/ 757 <====

004016 012742 000065      SOPB1B: MOV      #65,-(R2)    ;MOVE TO MAILBOX # ***** 65 *****
004022 005242      INC      -(R2)       ;SET MSGTYP TO FATAL ERROR
004024 000000      HALT                ;CHECK CUMMULATIVE RESULT OF ABOVE INST
; OR SEQUENCE ERROR

```

739
740
741
742
743
744
745
746
747
748
749
750

```

*****
; 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 50 TEST MODE 1 ODD BYTE USING SOP INST
*****

```

```

004026 005212 000050      TST50: INC      (R2)     ;UPDATE TEST NUMBER
004030 022712      CMP      #50,(R2)    ;SEQUENCE ERROR?
004034 001022      BNE      TST51-10     ;BR TO ERROR HALT ON SEQ ERROR
751 004036 005000      CLR      R0           ;INITIALIZE R0
752 004040 005010      CLR      (R0)         ;INITIALIZE LOC. 0
753 004042 005110      COM      (R0)
754 004044 005200      INC      R0           ;R0=ODD BYTE
755 004046 105010      CLRB     (R0)         ;TRY TO CLEAR BYTE 1
756 004050 001404      BEQ      SOPB1C

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 771 <====

```

```

004052 012742 000066          MOV    #66,-(R2)      ;MOVE TO MAILBOX # ***** 66 *****
004056 005242          INC    -(R2)        ;SET MSGTYP TO FATAL ERROR
004060 000000          HALT                ;CLR DID NOT SET Z-BIT
757 004062 005300      SOPB1C: DEC    R0      ;R0=WORD ADDR.
758 004064 005210          INC    (R0)        ;INCREMENT TO TEST WORD
759 004066 005200          INC    R0          ;R0=ODD BYTE
760 004070 105110          COMB   (R0)        ;TRY TO COMPLEMENT BYTE 1
761 004072 105210          INCB   (R0)
762 004074 100002          BPL    SOPB1D
763 004076 105210          INCB   (R0)        ;TRY TO INCREMENT BYTE 1
764 004100 001404          BEQ    TST51

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====

```

```

004102          SOPB1D:
004102 012742 000067          MOV    #67,-(R2)      ;MOVE TO MAILBOX # ***** 67 *****
004106 005242          INC    -(R2)        ;SET MSGTYP TO FATAL ERROR
004110 000000          HALT                ;TEST CUMMULATIVE RESULT OF ABOVE INST.
; OR SEQUENCE ERROR

```

765
766
767
768
769
770
771
772
773
774
775
776

```

:*****
: 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 51 TEST MODE 2 USING SOP INST.

```

004112 005212          TST51: INC    (R2)      ;UPDATE TEST NUMBER
004114 022712 000051          CMP    #51,(R2)     ;SEQUENCE ERROR?
004120 001023          BNE    TST52-10    ;BR TO ERROR HALT ON SEQ ERROR
777 004122 005000          CLR    R0          ;SET R0=400
778 004124 105100          COMB   R0
779 004126 005200          INC    R0
780 004130 005010          CLR    (R0)        ;CLEAR 400
781 004132 005110          COM    (R0)        ;INITIALIZE: 400=-1
782 004134 005020          CLR    (R0)+      ;TRY CLEARING WITH MODE 2
783 004136 001404          BEQ    SOPZA

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 770 <====

```

```

004140 012742 000070          MOV    #70,-(R2)      ;MOVE TO MAILBOX # ***** 70 *****
004144 005242          INC    -(R2)        ;SET MSGTYP TO FATAL ERROR
004146 000000          HALT                ;CLR INST DID NOT SET Z-BIT
784 004150 005300      ,SOPZA: DEC    R0      ;RESET R0
785 004152 005300          DEC    R0
786 004154 005120          COM    (R0)+      ;TRY COMPLEMENTING WITH MODE 2
787 004156 100004          BPL    SOP2B
788 004160 005300          DEC    R0          ;RESET R0
789 004162 005300          DEC    R0

```

```

790 004164 005220      INC      (R0)+      ;TRY INCREMENTING WITH MODE 2
791 004166 001404      BEQ      TST52
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <----
                                ;          WHICH FOLLOWS W/ 754 <----

```

```

004170      SOP2B:
004170 012742 000071      MOV      #71,-(R2)      ;MOVE TO MAILBOX # ***** 71 *****
004174 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
004176 000000      HALT      ;CHECK CUMMULATIVE RESULT OF ABOVE INST
                                ; OR SEQUENCE ERROR

```

792
793
794
795
796
797
798
799
800
801
802
803

```

:*****
: THIS TEST VERIFIES MODE 2 SINGLE OPERAND INSTRUCTIONS WHICH
: ADDRESS EVEN BYTES. R0 IS SET TO 400 AND USED TO INITIALIZE LOCATION
: 400 TO -1. CLRB INSTRUCTION IS THEN EXECUTED ON BYTE 400 WITH
: MODE 2.
: R0 IS THEN DECREMENTED BEFORE EACH OF SEVERAL MODE 2 INSTRUCTIONS
: WHICH ARE USED TO VERIFY THE DATA RESULTS OF THE TEST. THIS PROCEDURE ALSO
: VERIFIES THE PROPER INCREMENTING OF THE REGISTER.
:*****

```

:TEST 52 TEST MODE 2 EVEN BYTE USING SOP INST.

```

004200 005212      TST52: INC      (R2)      ;UPDATE TEST NUMBER
004202 022712 000052      CMP      #52,(R2)      ;SEQUENCE ERROR?
004206 001023      BNE     TST53-10      ;BR TO ERROR HALT ON SEQ ERROR
804 004210 005000      CLR     R0            ;SET R0=400
805 004212 105100      COMB   R0
806 004214 005200      INC     R0
807 004216 005010      CLR     (R0)          ;CLEAR 400
808 004220 005110      COM     (R0)          ;INITIALIZE: 400=-1
809 004222 105020      CLRB   (R0)+         ;TRY TO CLEAT 400 W/MODE 2
810 004224 001404      BEQ     SOPB2A
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <----
                                ;          WHICH FOLLOWS W/ 770 <----

```

```

004226 012742 000072      SOPB2A: MOV     #72,-(R2)      ;MOVE TO MAILBOX # ***** 72 *****
004232 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
004234 000000      HALT    ;CLR DID NOT SET Z-BIT
811 004236 005300      DEC     R0          ;RESULT R0=400
812 004240 005210      INC     (R0)        ;INC 400 TO TEST WORD
813 004242 105110      COMB   (R0)
814 004244 105220      INCB   (R0)+       ;TRY TO INC EVEN BYTE
815 004246 100003      BPL    SOPB2B
816 004250 005300      DEC     R0          ;RESET R0=400
817 004252 105220      INCB   (R0)+       ;TRY INCREMENT OF EVEN BYTE
818 004254 001404      BEQ     TST53
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <----
                                ;          WHICH FOLLOWS W/ 754 <----

```

```

004256      SOPB2B:
004256 012742 000073      MOV     #73,-(R2)      ;MOVE TO MAILBOX # ***** 73 *****
004262 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR

```

004264 000000

HALT

;TEST CUMMULATIVE RESULT OF ABOVE INST.
 ; OR SEQUENCE ERROR

819
 820
 821
 822
 823
 824
 825

 : THIS TEST FOLLOWS THE SAME PROCEDURE DESCRIBED IN THE PREVIOUS
 : TEST. HERE, THE BYTE INSTRUCTION IS USED TO ADDRESS AN ODD BYTE.

:TEST 53 TEST MODE 2 ODD BYTE USING SOP INST.

004266 005212
 004270 022712 000053
 004274 001026
 826 004276 005000
 827 004300 105100
 828 004302 005200
 829 004304 005010
 830 004306 001110
 831 004310 002200
 832 004312 105020
 833 004314 001404

TST53: INC (R2) ;UPDATE TEST NUMBER
 CMP #53,(R2) ;SEQUENCE ERROR?
 BNE TST54-10 ;BR TO ERROR HALT ON SEQ ERROR
 CLR R0 ;SET R0=400
 COMB R0
 INC R0
 CLR (R0) ;CLEAR LOC 400
 COM (R0) ;INITIALIZE: 400=-1
 INC R0 ;R0=ODD BYTE
 CLRB (R0)+ ;TRY TO CLEAR ODD BYTE
 BEQ SOPB2C

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
 : CONDITIONAL BRANCH INST. AND <====
 : REPLACE THE MOVE INSTRUCTION <====
 : WHICH FOLLOWS W/ 767 <====

004316 012742 000074
 004322 005242
 004324 000000
 834 004326 005300
 835 004330 005300
 836 004332 005220

MOV #74,-(R2) ;MOVE TO MAILBOX # ***** 74 *****
 INC -(R2) ;SET MSGTYP TO FATAL ERROR
 HALT ;CLRB DID NOT SET Z-BIT
 SOPB2C: DEC R0 ;R0-WORD ADDR.
 DEC R0
 INC (R0)+ ;INCREMENT WORD

```

838 004334 005300      DEC      R0      ;POINT TO ODD BYTE
839 004336 105110      COMB     (R0)     ;COMPLEMENT ODD BYTE
840 004340 105220      INCB    (R0)+    ;TRY TO INCREMENT ODD BYTE
841 004342 100003      BPL     SOPB2D
842 004344 005300      DEC      R0      ;RESET R0 TO ODD BYTE
843 004346 105220      INCB    (R0)+    ;TRY TO INCREMENT ODD BYTE
844 004350 001404      BEQ     TST54

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-- -
;          CONDITIONAL BRANCH INST. AND <  - -
;          REPLACE THE MOVE INSTRUCTION <  - -
;          WHICH FOLLOWS W/ 751 <-- -

```

```

004352      SOPB2D:
004352 012742 000075      MOV     #75,-(R2) ;MOVE TO MAILBOX # ***** 75 *****
004356 005242      INC     -(R2)    ;SET MSGTYP TO FATAL ERROR
004360 000000      HALT                    ;TEST CUMULATIVE RESULT OF ABOVE INST.
; OR SEQUENCE ERROR

```

845
846
847
848
849
850
851

```

:*****
:
: THESE TESTS CHECK THE NEGATE INSTRUCTION IN ALL MODES. PREVIOUSLY
: TESTED SINGLE-OPERAND INSTRUCTIONS ARE USED TO TEST THE NEGATE INSTRUCTION.
:*****
:TEST 54      TEST MODE 0 USING NEGATE INSTRUCTION
:*****

```

```

004362 005212 000054      TST54: INC     (R2)      ;UPDATE TEST NUMBER
004364 022712      CMP     #54,(R2)   ;SEQUENCE ERROR?
004370 001035      BNE    TST55-10   ;BR TO ERROR HALT ON SEQ ERROR
852 004372 005000      CLR    R0         ;SET R0=0
853 004374 005200      INC    R0         ; R0=1
854 004376 005400      NEG    R0         ;TRY NEGATE MODE 0: R0 -1
855 004400 100003      BPL    NEG00      ;CC 1001?
856 004402 001402      BEQ    NEG00
857 004404 102401      BVS    NEG00
858 004406 103404      BCS    NEG01

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 770 <====

```

```

004410      NEG00:
004410 012742 000076      MOV     #76,-(R2) ;MOVE TO MAILBOX # ***** 76 *****
004414 005242      INC     -(R2)    ;SET MSGTYP TO FATAL ERROR
004416 000000      HALT                    ;NEGATE DID NOT SET CC'S CORRECTLY

```

```

859
860 004420 005200      NEG01: INC     R0         ;TEST DATA RESULT
861 004422 001404      BEQ    NEG02

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 762 <====

```

```

004424 012742 000077      MOV     #77,-(R2) ;MOVE TO MAILBOX # ***** 77 *****
004430 005242      INC     -(R2)    ;SET MSGTYP TO FATAL ERROR
004432 000000      HALT                    ;DATA RESULT C: NEGATE INCORRECT

```

```

862
863 004434 105100      NEG02: COMB    R0         ;R0=377
864 004436 105400      NEGB   R0         ;R0=1
865 004440 100403      BMI    NEG03      ;CC=0001?

```

```

866 004442 001402      BEQ     NEG03
867 004444 102401      BVS     NEG03
868 004446 103404      BCS     NEG04
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <= ==
:          CONDITIONAL BRANCH INST. AND    <- ==
:          REPLACE THE MOVE INSTRUCTION   <= ==
:          WHICH FOLLOWS W/ 750           <---=
  
```

```

004450      NEG03:  MOV     #100,-(R2)  ;MOVE TO MAILBOX # ***** 100 *****
004450 012742 000100  INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
004454 005242      HALT                    ;NEGB DID NOT SET CC'S CORRECTLY
869 004460 005300  NEG04:  DEC     R0        ;TEST DATA RESULT
870 004462 001404      BEQ     TST55
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
:          CONDITIONAL BRANCH INST. AND    <====
:          REPLACE THE MOVE INSTRUCTION   <====
:          WHICH FOLLOWS W/ 742           <---=
  
```

```

004464 012742 000101  MOV     #101,-(R2)  ;MOVE TO MAILBOX # ***** 101 *****
004470 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
004472 000000      HALT                    ;DATA RESULT OF NEGB INCORRECT
: OR SEQUENCE ERROR
  
```

```

871 *****
:TEST 55          TEST MODE 1 USING NEGATE INST.
*****
  
```

```

004474 005212 000055  TST55:  INC     (R2)        ;UPDATE TEST NUMBER
004476 022712      CMP     #55,(R2)     ;SEQUENCE ERROR?
004502 001040      BNE     TST56-10    ;BR TO ERROR HALT ON SEQ ERROR
872 004504 005000      CLR     R0          ;POINT TO LOC. 0
873 004506 005010      CLR     (R0)        ;CLEAR LOC. 0
874 004510 005210      INC     (R0)        ;LOC. 0=1
875 004512 005410      NEG     (R0)        ;TRY NEG. LOC. 0 -1
876 004514 100003      BPL     NEG10       ;CC=1001
877 004516 001402      BEQ     NEG10
878 004520 102401      BVS     NEG10
879 004522 103404      BCS     NEG11
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
:          CONDITIONAL BRANCH INST. AND    <====
:          REPLACE THE MOVE INSTRUCTION   <====
:          WHICH FOLLOWS W/ 767           <---=
  
```

```

004524      NEG10:  MOV     #102,-(R2)  ;MOVE TO MAILBOX # ***** 102 *****
004524 012742 000102  INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
004530 005242      HALT                    ;NEGATE DID NOT SET CC'S CORRECTLY
880 004532 000000
  
```

```

881 004534 005237 000000  NEG11:  INC     @R0      ;TEST DATA RESULT
882 004540 001404      BEQ     NEG12
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
:          CONDITIONAL BRANCH INST. AND    <====
:          REPLACE THE MOVE INSTRUCTION   <====
:          WHICH FOLLOWS W/ 760           <---=
  
```

```

004542 012742 000103  MOV     #103,-(R2)  ;MOVE TO MAILBOX # ***** 103 *****
004546 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
004550 000000      HALT                    ;DATA RESULT OF NEGATE INCORRECT
883 004552 105110  NEG12:  (OMB     (R0)        ;LOC. 0=377
884 004554 105410      NEGB   (R0)        ;TRY NEGB LOC. 0=1
885 004556 100403      BMI     NEG13       ;CC=0001?
886 004560 001402      BEQ     NEG13
  
```



```
887 004562 102401          BVS  NEG13
888 004564 103404          BCS  NEG14
                                : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
                                : CONDITIONAL BRANCH INST. AND <===
                                : REPLACE THE MOVE INSTRUCTION <===
                                : WHICH FOLLOWS W/ 746 <===
                                :
004566          NEG13:    MOV  #104,-(R2)      ;MOVE TO MAILBOX # ***** 104 *****
004566 012742 000104      INC  -(R2)      ;SET MSGTYP TO FATAL ERROR
004572 005242              HALT                    ;NEGB DID NOT SET CC'S CORRECTLY
889 004576 005337 000000  NEG14:  DEC  @#0      ;TEST DATA RESULT
890 004602 001404          BEQ  TST56
                                : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                : CONDITIONAL BRANCH INST. AND <====
                                : REPLACE THE MOVE INSTRUCTION <====
                                : WHICH FOLLOWS W/ 737 <====
                                :
004604 012742 000105      MOV  #105,-(R2)   ;MOVE TO MAILBOX # ***** 105 *****
004610 005242              INC  -(R2)   ;SET MSGTYP TO FATAL ERROR
004612 000000              HALT          ;DATA RESULT OF NEGB INCORRECT
                                : OR SEQUENCE ERROR
891
:*****
:TEST 56          TEST MODE 2 USING NEGATE INSTRUCTION
:*****
004614 005212 000056  TST56:  INC  (R2)      ;UPDATE TEST NUMBER
004616 022712          CMP  #56,(R2)   ;SEQUENCE ERROR?
892 004622 001032          BNE  TST57-10 ;BR TO ERROR HALT ON SEQ ERROR
893 004624 005000          CLR  R0      ;POINT TO LOC. 0
894 004626 005010          CLR  (R0)     ;CLEAR LOC. 0
895 004630 005210          INC  (R0)     ;LOC. 0=1
896 004632 005420          NEG  (R0)+    ;TRY NEG.: LOC. 0 -1
897 004634 100003          BPL  NEG20    ;CC 1001?
898 004636 001402          BFQ  NEG20
899 004640 102401          BVS  NEG20
899 004642 103404          BCS  NEG21
                                : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                : CONDITIONAL BRANCH INST. AND <====
                                : REPLACE THE MOVE INSTRUCTION <====
                                : WHICH FOLLOWS W/ 767 <====
                                :
004644          NEG20:    MOV  #106,-(R2)   ;MOVE TO MAILBOX # ***** 106 *****
004644 012742 000106      INC  -(R2)   ;SET MSGTYP TO FATAL ERROR
004650 005242              HALT                    ;NEGATE DID NOT SET CC'S CORRECTLY
900 004654 105300  NEG21:  DECB  R0      ;R0=LOC. 0
901 004656 105300          DECB  R0
902 004660 105420          NEGB (R0)+    ;BYTE 0=1 R0=1
903 004662 105420          NEGB (R0)+    ;BYTE 1=1 R0=2
904 004664 105340          DECB -(R0)     ;R0=1 LOC. 0=01
905 004666 005300          DEC  R0      ;R0=0
906 004670 001404          BEQ  NEG22
                                : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                : CONDITIONAL BRANCH INST. AND <====
                                : REPLACE THE MOVE INSTRUCTION <====
                                : WHICH FOLLOWS W/ 754 <====
                                :
004672 012742 000107      MOV  #107,-(R2)   ;MOVE TO MAILBOX # ***** 107 *****
004676 005242              INC  -(R2)   ;SET MSGTYP TO FATAL ERROR
004700 000000              HALT                    ;REGISTER NOT INCREMENTED CORRECTLY
907 004702 005337 000000  NEG22:  DEC  @#0      ;LOC. 0=0
```

```

908 004706 001404          BEQ    TST57
                                : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <==--
                                :          CONDITIONAL BRANCH INST. AND  <====
                                :          REPLACE THE MOVE INSTRUCTION  <----=
                                :          WHICH FOLLOWS W/ 745                <= ==
                                :
004710 012742 000110      MOV    #110,-(R2)      ;MOVE TO MAILBOX # ***** 110 *****
004714 005242             INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
004716 000000             HALT                   ;NEG BYTE INSTRUCTIONS FAILED
                                : OR SEQUENCE ERROR
  
```

909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924

```

:*****
:
:      THIS TEST VERIFIES MODE 3 SINGLE OPERAND INSTRUCTIONS. IT
:      USES LOCATION 0 AS ITS TARGET DATA. A TABLE LOCATED AT LOC. 400
:      THRU 402 IS USED TO SUPPLY THE ADDRESS OF LOCATION 0 TO THE
:      INSTRUCTIONS UNDER TEST.
:      R0 IS SET TO 400, THE START OF THE ADDRESS TABLE, AND A CLR
:      INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR LOC. 0. THEN R0
:      IS DECREMENTED BY TWO AND TWO OTHER MODE 3 INSTRUCTIONS OPERATE ON
:      LOC. 0 TO VERIFY THE DATA RESULTS OF THE TEST. THE PROPER INCREMENTING
:      OF THE REGISTER IS ALSO VERIFIED IN THIS MANNER.
:      IF A FAILURE IS DETECTED BE SURE TO VERIFY THAT THE TABLE
:      (LOC. 400-402) HAS THE PROPER VALUES (0).
:*****
  
```

:TEST 57 TEST MODE 3 USING SOP INST.

```

004720 005212 000057      TST57: INC    (R2)          ;UPDATE TEST NUMBER
004722 022712             CMP    #57,(R2)       ;SEQUENCE ERROR?
004726 001020             BNE   TST60-10      ;BR TO ERROR HALT ON SEQ ERROR
925 004730 005000             CLR   R0            ;SET R0=400
926 004732 105100             COMB  R0
927 004734 005200             INC   R0
928 004736 005010             CLR   (R0)          ;CLEAR LOC 400
929 004740 005030             CLR   @(R0)+         ;TRY TO CLEAR LOC 0 USING MODE 3 ;R0 402
930 004742 001404             BEQ   SOP3A
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <-
:          CONDITIONAL BRANCH INST. AND  <
:          REPLACE THE MOVE INSTRUCTION  <
:          WHICH FOLLOWS W/ 771          <-
:
  
```

```

004744 012742 000111      MOV    #111,-(R2)      ;MOVE TO MAILBOX # ***** 111 *****
004750 005242             INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
004752 000000             HALT                   ;CLR DID NOT SET Z-BIT
931 004754 005300      SOP3A: DEC   R0            ;RESET R0=400
932 004756 005300             DEC   R0
933 004760 005130             COM   @(R0)+         ;TRY TO COMPLEMENT LOC 0 OF MODE 3 ;R0-402
934 004762 100002             BPL   SOP3B
935 004764 005230             INC   @(R0)+         ;TRY TO INCREMENT LOC 0 W/MODE 3 ;R0=404
936 004766 001404             BEQ   TST60
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
:          CONDITIONAL BRANCH INST. AND  <====
:          REPLACE THE MOVE INSTRUCTION  <====
:          WHICH FOLLOWS W/ 757          <====
:
  
```

```

004770 012742 000112      SOP3B: MOV    #112,-(R2)      ;MOVE TO MAILBOX # ***** 112 *****
004774 005242             INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
004776 000000             HALT                   ;CUMMULATIVE RESULT OF ABOVE INST FAILED
  
```

; OR SEQUENCE ERROR

937
 938
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 950

```

:*****
:      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 60 TEST MODE 3 EVEN BYTE USING SOP INST.

005000 005212
 005002 022712 000060
 005006 001026
 951 005010 005004
 952 005012 105104
 953 005014 005204
 954 005016 005000
 955 005020 005010
 956 005022 005110
 957 005024 105034
 958 005026 001404

```

TST60:  INC      (R2)          ;UPDATE TEST NUMBER
        CMP      #60,(R2)     ;SEQUENCE ERROR?
        BNE     TST61-10     ;BR TO ERROR HALT ON SEQ ERROR
        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
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <== =
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 767 <====
  
```

005030 012742 000113
 005034 005242
 005036 000000
 959 005040 005304
 960 005042 005304
 961 005044 005234
 962 005046 100006
 963 005050 105434
 964 005052 100004
 965 005054 005304
 966 005056 005304
 967 005060 105234
 968 005062 001404

```

SOPB3A: MOV      #113,-(R2)    ;MOVE TO MAILBOX # ***** 113 *****
        INC     -(R2)
        HALT
        DEC     R4
        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
        DEC     R4
        INCB   @(R4)+      ;TRY TO INCREMENT EVEN BYTE ;LOC. 0=17400
        BEQ    TST61
  
```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
:          CONDITIONAL BRANCH INST. AND <--
:          REPLACE THE MOVE INSTRUCTION <---
:          WHICH FOLLOWS W/ 751 <==
  
```

005064
 005064 012742 000114
 005070 005242
 005072 000000

```

SOPB3B: MOV      #114,-(R2)    ;MOVE TO MAILBOX # ***** 114 *****
        INC     -(R2)
        HALT
        CUMULATIVE RESULT OF ABOVE INST FAILED
        ; OR SEQUENCE ERROR
  
```

969
 970
 971

```

:*****
:
:*****
  
```

972
973
974
975
976
977
978
979
980
981
982
983

```

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

```

```

*****
:TEST 61 TEST MODE 3 ODD BYTE USING SOP INST.
*****

```

005074 005212
005076 022712 000061
005102 001024
984 005104 005000
985 005106 105100
986 005110 005200
987 005112 005030
988 005114 005130
989 005116 105030
990 005120 001404

```

TST61: INC (R2) ;UPDATE TEST NUMBER
        CMP #61,(R2) ;SEQUENCE ERROR?
        BNE TST62-10 ;BR TO ERROR HALT ON SEQ ERROR
        CLR R0 ;SET R0=400
        COMB R0
        INC R0
        CLR @ (R0)+ ;INITIALIZE
        COM @ (R0)+ ;LOC 0=-1 R0=404
        CLRB @ (R0)+ ;TRY TO CLEAR ODD BYTE LOC. 0=377 R0=406
        BEQ SOPB3C

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
: CONDITIONAL BRANCH INST. AND < -
: REPLACE THE MOVE INSTRUCTION <-
: WHICH FOLLOWS W/ 770 <--

```

005122 012742 0001
005126 005242
005130 000000
991 005132 005300
992 005134 005300
993 005136 005300
994 005140 005300
995 005142 005230
996 005144 105430
997 005146 100002
998 005150 105230
999 005152 001404

```

MOV #115,-(R2) ;MOVE TO MAILBOX # ***** 115 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CLRB DID NOT SET Z-BIT
SOPB3C: DEC R0 ;RESET R0=402
        DEC R0
        DEC R0 ;POINT TO EVEN BYTE ADDR.
        DEC R0
        INC @ (R0)+ ;INCREMENT WORD LOC. 0=400 R0=404
        NEGB @ (R0)+ ;TRY TO NEGATE ODD BYTE LOC. 0=177400 R0=406
        BPL SCOR70
        INCB @ (R0)+ ;TRY TO INCREMENT ODD BYTE LOC.0=0 R0=410
        BEQ TST62

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 753 <====

```

005154
005154 012742 000116
005160 005242
005162 000000

```

SOPB3D: MOV #116,-(R2) ;MOVE TO MAILBOX # ***** 116 *****
        INC -(R2) ;SET MSGTYP TO FATAL ERROR
        HALT ;CUMULATIVE RESULT OF ABOVE INSTS FAILED
        ; OR SEQUENCE ERROR

```

1000

```

*****
:TEST 62 TEST MODE 3 USING NEGATE INSTRUCTION
*****

```

005164 005212
005166 022712 000062
005172 001054
1001 005174 005000
1002 005176 105100

```

TST62: INC (R2) ;UPDATE TEST NUMBER
        CMP #62,(R2) ;SEQUENCE ERROR?
        BNE TST63-10 ;BR TO ERROR HALT ON SEQ ERROR
        CLR R0 ;R0=400
        COMB R0

```

```

1003 005200 005200      INC      R0
1004 005202 005010      CLR      (R0)      ;LOC. 400=0
1005 005204 005004      CLR      R4        ;R4=0
1006 005206 005014      CLR      (R4)      ;LOC. 0=0
1007 005210 005214      INC      (R4)      ;LOC. 0=1
1008 005212 005430      NEG      @ (R0)+    ;TRY NEGATE   LOC. 0=-1  R0=402
1009 005214 100003      BPL     NEG30      ;CC=1001?
1010 005216 001402      BEQ     NEG30
1011 005220 102401      BVS     NEG30
1012 005222 103404      BCS     NEG31

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
; CONDITIONAL BRANCH INST. AND           <====
; REPLACE THE MOVE INSTRUCTION          <====
; WHICH FOLLOWS W/ 763                  <====

      005224      NEG30:  MOV      #117,-(R2)  ;MOVE TO MAILBOX # ***** 117 *****
      005224 012742 000117      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
      005230 005242      HALT     ;NEG DID NOT SET CC'S CORRECTLY
      005232 000000      NEG31:  INC      (R4)      ;LOC. 0=0
1013 005234 005214      BEQ     NEG32
1014 005236 001404

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
; CONDITIONAL BRANCH INST. AND           <====
; REPLACE THE MOVE INSTRUCTION          <====
; WHICH FOLLOWS W/ 755                  <====

      005240 012742 000120      MOV      #120,-(R2) ;MOVE TO MAILBOX # ***** 120 *****
      005244 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
      005246 000000      HALT     ;DATA RESULT OF NEG INCORRECT
1015 005250 105137 000001      NEG32:  COMB    @#1      ;LOC 0=177400
1016 005254 005237 000000      INC      @#0      ;LOC. 0=177401
1017 005260 105430      NEGB   @ (R0)+    ;TRY NEGB LOC. 0=177777  R0=404
1018 005262 100404      BMI     NEG33

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
; CONDITIONAL BRANCH INST. AND           <====
; REPLACE THE MOVE INSTRUCTION          <====
; WHICH FOLLOWS W/ 743                  <====

      005264 012742 000121      MOV      #121,-(R2) ;MOVE TO MAILBOX # ***** 121 *****
      005270 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
      005272 000000      HALT     ;NEGB FAILED WITH EVEN BYTE
1019 005274 105430      NEG33:  NEGB   @ (R0)+    ;TRY NEGB LOC.0=777  R0=406
1020 005276 100004      BPL     NEG34

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
; CONDITIONAL BRANCH INST. AND           <====
; REPLACE THE MOVE INSTRUCTION          <====
; WHICH FOLLOWS W/ 735                  <====

      005300 012742 000122      MOV      #122,-(R2) ;MOVE TO MAILBOX # ***** 122 *****
      005304 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
      005306 000000      HALT     ;NEGB FAILED WITH ODD BYTE
1021 005310 105137 000001      NEG34:  COMB    @#1      ;LOC. 0=177377
1022 005314 105237 000001      INCB   @#1      ;LOC. 0=177777
1023 005320 005214      INC      (R4)      ;LOC. 0=0
1024 005322 001404      BEQ     TST63

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
; CONDITIONAL BRANCH INST. AND           <====
; REPLACE THE MOVE INSTRUCTION          <====
; WHICH FOLLOWS W/ 723                  <====

      005324 012742 000123      MOV      #123,-(R2) ;MOVE TO MAILBOX # ***** 123 *****
      005330 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
  
```

005332 000000

HALT

;DATA RESULT OF NEGB'S INCORRECT
; OR SEQUENCE ERROR

1025
1026
1027
1028
1029
1030
1031
1032
1033
1034

: THIS TEST VERIFIES MODE 4 SINGLE OPERAND INSTRUCTIONS.
: RO IS SET TO 400. A CLR INSTRUCTION IS EXECUTED IN MODE 4 TO CLEAR
: LOC. 376. RO IS RESET TO 400 AND A COM INSTRUCTION USING MODE 4
: COMPLEMENTS LOC.376.
: TWO INC INSTRUCTIONS AND A MODE 4 INSTRUCTION ARE EXECUTED
: TO COMPLETE THE TEST.

:TEST 63 TEST MODE 4 USING SOP INSTS

005334 005212
005336 022712 000063
005342 001021
1035 005344 005000
1036 005346 105100
1037 005350 005200
1038 005352 005040
1039 005354 001404

TST63: INC (R2) ;UPDATE TEST NUMBER
CMP #63,(R2) ;SEQUENCE ERROR?
BNE TST64-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;SET R0=400
COMB R0
INC R0
CLR -(R0) ;TRY TO CLEAR USING MODE 4
BEQ SOP4A

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-- -
: CONDITIONAL BRANCH INST. AND <--
: REPLACE THE MOVE INSTRUCTION <-
: WHICH FOLLOWS W/ 772 < -

005356 012742 000124
005362 005242
005364 000000
1040 005366 005200
1041 005370 005200
1042 005372 005140
1043 005374 100004
1044 005376 005200
1045 005400 005200
1046 005402 005240
1047 005404 001404

MOV #124,-(R2) ;MOVE TO MAILBOX # ***** 124 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CLR DID NOT SET Z-BIT
SOP4A: INC R0 ;RESET R0
INC R0
COM -(R0) ;TRY TO COMPLEMENT USING MODE 4
BPL SOP4B
INC R0 ;MOVE POINTER
INC R0
INC -(R0)
BEQ TST64

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
: CONDITIONAL BRANCH INST. AND <--
: REPLACE THE MOVE INSTRUCTION <=
: WHICH FOLLOWS W/ 756 < -

005406
005406 012742 000125
005412 005242
005414 000000

SOP4A: MOV #125,-(R2) ;MOVE TO MAILBOX # ***** 125 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CHECK CUMMULATIVE RESULT OF ABOVE INST.
; OR SEQUENCE ERROR

1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058

: THIS TEST VERIFIES MODE 5 SINGLE OPERAND INSTRUCTIONS. IT
: USES LOCATION 0 AS ITS TARGET DATA. A TABLE LOCATED AT LOC. 372
: THRU 374 IS USED TO SUPPLY THE ADDRESS OF LOCATION 0 TO THE
: INSTRUCTIONS UNDER TEST.
: RO IS SET TO 376, (THE START OF THE ADDRESS TABLE) +2,
: AND A CLR INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR
: LOC. 0. THEN RO IS INCREMENTED BY TWO AND TWO OTHER MODE 3
: INSTRUCTIONS OPERATE ON LOC. 0 TO VERIFY THE DATA RESULTS OF

1059
1060
1061
1062
1063
1064

:THE TEST. THE PROPER DECREMENTING OF THE REGISTER IS ALSO
:VERIFIED IN THIS MANNER.
: IF A FAILURE IS DETECTED BE SURE TO VERIFY THAT THE TABLE
:(LOC. 372 THRU 374) HAS THE PROPER VALUES (0).

:TEST 64 TEST MODE 5 USING SOP INSTS

005416 005212
005420 022712 000064
005424 001017
1065 005426 005000
1066 005430 005020
1067 005432 105400
1068 005434 005050
1069 005436 001404

TST64: INC (R2) ;UPDATE TEST NUMBER
CMP #64,(R2) ;SEQUENCE ERROR?
BNE TST65-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;SET R0=376
CLR (R0)+
NEGB R0
CLR @-(R0) ;TRY TO CLEAR LOC 0 W/MODE 5
BEQ SOP5A

005440 012742 000126
005444 005242
005446 000000
1070 005450 005200
1071 005452 005200
1072 005454 005150
1073 005456 100002
1074 005460 005250
1075 005462 001404

MOV #126,-(R2) ;MOVE TO MAILBOX # ***** 126 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CLR DID NOT SET Z-BIT
SOP5A: INC R0 ;RESET R0
INC R0
COM @-(R0) ;TRY TO COMPLEMENT LOC. 0 W/MODE 5
BPL SOP5B
INC @-(R0) ;TRY TO INCREMENT LOC. 0 W/MODE 5
BEQ TST65

005464
005464 012742 000127
005470 005242
005472 000000

SOP5B: MOV #127,-(R2) ;MOVE TO MAILBOX # ***** 127 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;TEST CUMULATIVE RESULT OF ABOVE INSTS
; OR SEQUENCE ERROR

1076
1077
1078
1079
1080
1081
1082
1083
1084
1085

: THIS TEST VERIFIES MODE 6 SINGLE OPERAND INSTRUCTIONS. IT
: USES LOCATION 0 AS ITS TARGET DATA. R0 IS SET TO 400 USING
: PREVIOUSLY TESTED INSTRUCTIONS AND A MODE 6 CLR INSTRUCTION IS
: EXECUTED ON LOC. 0 USING R0 AND A -400 OFFSET. COM AND INC
: INSTRUCTIONS ARE THEN USED TO VERIFY THE DATA.

:TEST 65 TEST MODE 6 USING SOP INSTS

005474 005212
005476 022712 000065
005502 001020
1086 005504 005000
1087 005506 105100
1088 005510 005200
1089 005512 005060 177400

TST65: INC (R2) ;UPDATE TEST NUMBER
CMP #65,(R2) ;SEQUENCE ERROR?
BNE TST66-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;SET R0=400
COMB R0
INC R0
CLR -400(R0) ;TRY TO CLEAR LOCATION 0 W/MODE 6

```

1090 005516 001404          BEQ      SOP6A          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 771 <====
      005520 012742 000130      MOV      #130,-(R2)      ;MOVE TO MAILBOX # ***** 130 *****
      005524 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
      005526 000000          HALT          ;CLR DID NOT SET Z-BIT
1091 005530 005160 177400      SOP6A: COM      -400(R0)      ;TRY TO COMPLEMENT LOCATION 0 W/MODE 6
1092 005534 100003          BPL      SOP6B
1093 005536 005260 177400      INC      -400(R0)      ;TRY TO INCREMENT LOCATION 0 W/MODE 6
1094 005542 001404          BEQ      TST66
;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 757 <====
      005544          SOP6B: MOV      #131,-(R2)      ;MOVE TO MAILBOX # ***** 131 *****
      005544 012742 000131      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
      005550 005242          HALT          ;TEST CUMULATIVE RESULT OF ABOVE INSTS
      005552 000000          ; OR SEQUENCE ERROR

```

```

1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
:*****
: THIS TEST VERIFIES MODE 7 SINGLE OPERAND INSTRUCTIONS. IT USES
: THE POINTER TO LOC. 0 WHICH IS STORED AT LOC. 402.
: R0 IS SET TO 400 AND A MODE 7 CLR INSTRUCTION IS
: EXECUTED WITH A +2 OFFSET TO CLEAR LOC. 0.
: SEVERAL OTHER MODE 7 INSTRUCTIONS ARE THEN USED ON THE COMMON
: LOCATION TO VERIFY THE DATA RESULTS.
:*****

```

```

:TEST 66          TEST MODE 7 USING SOP INST.
:*****
TST66:  INC      (R2)          ;UPDATE TEST NUMBER
      005554 005212          CMP      #66,(R2)      ;SEQUENCE ERROR?
      005556 022712 000066      BNE      TST67-10      ;BR TO ERROR HALT ON SEQ ERROR
1106 005564 005000          CLR      R0          ;SET R0=400
1107 005566 105100          COMB    R0
1108 005570 005200          INC      R0
1109 005572 005210          INC      (R0)          ;R0=1
1110 005574 005070 000002      CLR      @2(R0)        ;TRY TO CLEAR LOC. 0 W/MODE 7
1111 005600 001404          BEQ      SOP7A
;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 770 <====
      005602 012742 000132      MOV      #132,-(R2)      ;MOVE TO MAILBOX # ***** 132 *****
      005606 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
      005610 000000          HALT          ;CLR DID NOT SET Z-BIT
1112 005612 005170 000002      SOP7A: COM      @2(R0)      ;TRY TO COMPLEMENT LOC. 0 W/MODE 7
1113 005616 100003          BPL      SOP7B
1114 005620 005270 000002      INC      @2(R0)        ;TRY TO INCREMENT LOC. 0 W/MODE 7
1115 005624 001404          BEQ      TST67
;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====

```


005626 012742 000133 SOP7B: MOV #133,-(R2) ; MOVE TO MAILBOX # ***** 133 *****
005632 005242 INC -(R2) ; SET MSGTYP TO FATAL ERROR
005634 000000 HALT ; TEST CUMMULATIVE RESULT OF ABOVE INSTS.
; OR SEQUENCE ERROR

1116
1117

005636 005212 000067 TST67: INC (R2) ; UPDATE TEST NUMBER
005640 022712 CMP #67,(R2) ; SEQUENCE ERROR?
005644 001024 BNE TST70-10 ; BR TO ERROR HALT ON SEQ ERROR
1118 005646 005000 CLR R0
1119 005650 005010 CLR (R0)
1120 005652 005120 COM (R0)+ ; LOC. 0=177777, R0=2
1121 005654 005440 NEG -(R0) ; TRY NEGATE, LOC. 0=1
1122 005656 100403 RMI NEG40 ; CC=0001?
1123 005660 001402 BEQ NEG40
1124 005662 102401 BVS NEG40
1125 005664 103404 BCS NEG41

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 767

005666 012742 000134 NEG40: MOV #134,-(R2) ; MOVE TO MAILBOX # ***** 134 *****
005672 005242 INC -(R2) ; SET MSGTYP TO FATAL ERROR
005674 000000 HALT ; NEG DID NOT SET CC'S CORRECTLY
1126 005676 005400 NEG41: NEG R0 ; TST R0 WITH A NEG.
1127 005700 001404 BEQ NEG42

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 761

005702 012742 000135 NEG42: MOV #135,-(R2) ; MOVE TO MAILBOX # ***** 135 *****
005706 005242 INC -(R2) ; SET MSGTYP TO FATAL ERROR
005710 000000 HALT ; R0 NOT DECREMENTED PROPERLY
1128 005712 005310 DEC (R0) ; TEST DTA RESULT OF NEG
1129 005714 001404 BEQ TST70

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 753

005716 012742 000136 MOV #136,-(R2) ; MOVE TO MAILBOX # ***** 136 *****
005722 005242 INC -(R2) ; SET MSGTYP TO FATAL ERROR
005724 000000 HALT ; DATA RESULT OF NEG INCORRECT
; OR SEQUENCE ERROR

1130

005726 005212 000070 TST70: INC (R2) ; UPDATE TEST NUMBER
005730 022712 CMP #70,(R2) ; SEQUENCE ERROR?
005734 001031 BNE TST71-10 ; BR TO ERROR HALT ON SEQ ERROR
1131 005736 005000 CLR R0 ; R0=0
1132 005740 005010 CLR (R0) ; LOC. 0=0
1133 005742 105100 COMB R0 ; R0=377

```
1134 005744 005200      INC      R0      ;R0=400
1135 005746 005010      CLR      (R0)    ;SET 400 = 0
1136 005750 005004      CLR      R4      ;R4=0
1137 005752 005314      DEC      (R4)    ;LOC. 0=177777
1138 005754 005450      NEG      @-(R0)  ;TRY NEGATE: LOC. 0=1
1139 005756 100403      BMI     NEG50    ;CC=0001?
1140 005760 001402      BEQ     NEG50
1141 005762 102401      BVS     NEG50
1142 005764 103404      BCS     NEG51
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 763 <====
```

```
005766      NEG50:  MOV      #137,-(R2) ;MOVE TO MAILBOX # ***** 137 *****
005766 012742 000137      INC      -(R2)    ;SET MSGTYP TO FATAL ERROR
005772 005242      HALT     ;NEG DID NOT SET CC'S CORRECTLY
005774 000000
1143 005776 005314      NEG51:  DEC      (R4)
*144 006000 001404      BEQ     NEG52
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 755 <====
```

```
006002 012742 000140      MOV      #140,-(R2) ;MOVE TO MAILBOX # ***** 140 *****
006006 005242      INC      -(R2)    ;SET MSGTYP TO FATAL ERROR
006010 000000      HALT     ;DATA RESULT OF NEG INCORRECT
1145 006012 105100      NEG52:  COMB    R0
1146 006014 005300      DEC      R0
1147 006016 001404      BEQ     TST71
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 746 <====
```

```
006020 012742 000141      MOV      #141,-(R2) ;MOVE TO MAILBOX # ***** 141 *****
006024 005242      INC      -(R2)    ;SET MSGTYP TO FATAL ERROR
006026 000000      HALT     ;REGISTER NOT DECREMENTED PROPERLY
: OR SEQUENCE ERROR
```

```
1148 *****
:TEST 71 TEST MODE 6 WITH NEGATE
*****
```

```
TST71:  INC      (R2)      ;UPDATE TEST NUMBER
006030 005212 000071      CMP      #71,(R2)  ;SEQUENCE ERROR?
006032 022712      BNE     TST72-10  ;BR TO ERROR HALT ON SEQ ERROR
006036 001022
1149 006040 005000      CLR      R0      ;R0=0
1150 006042 005004      CLR      R4      ;R4=0
1151 006044 105100      COMB    R0      ;R0=377
1152 006046 005014      CLR      (R4)    ;LOC. 0=0
1153 006050 105024      CLR     (R4)+    ;LOC. 0=177777, R4 1
1154 006052 105114      COMB    (R4)    ;LOC. 0=177400
1155 006054 005460 177401      NEG     -377(R0) ;LOC. 0=400
1156 006060 100403      BMI     NEG60    ;CC=0001
1157 006062 001402      BEQ     NEG60
1158 006064 102401      BVS     NEG60
1159 006066 103404      BCS     NEG61
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
```

006070
006070 012742 000142
006074 005242
006076 000000
1160 006100 105314
1161 006102 001404

NEG60: MOV #142,-(R2)
INC -(R2)
HALT
NEG61: DECB (R4)
BEQ TST72

: WHICH FOLLOWS W/ 763 <===-
: MOVE TO MAILBOX # ***** 142 *****
: SET MSGTYP TO FATAL ERROR
: NEG DID NOT SET CC'S CORRECTLY

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===-
: CONDITIONAL BRANCH INST. AND <===
: REPLACE THE MOVE INSTRUCTION <===
: WHICH FOLLOWS W/ 755 <===

006104 012742 000143
006110 005242
006112 000000

MOV #143,-(R2)
INC -(R2)
HALT

: MOVE TO MAILBOX # ***** 143 *****
: SET MSGTYP TO FATAL ERROR
: DATA RESULT OF NEG INCORRECT
: OR SEQUENCE ERROR

1162

: TEST 72 TEST MODE 7 W/ NEGATE

006114 005212
006116 022712 000072
006122 001024
1163 006124 005000
1164 006126 005010
1165 006130 005110
1166 006132 105100
1167 006134 105470 000005
1168 006140 100403
1169 006142 001402
1170 006144 102401
1171 006146 103404

TST72: INC (R2)
CMP #72,(R2)
BNE TST73-10
CLR R0
CLR (R0)
COM (R0)
COMB R0
NEGB @5(R0)
BMI NEG70
BEQ NEG70
BVS NEG70
BCS NEG71

: UPDATE TEST NUMBER
: SEQUENCE ERROR?
: BR TO ERROR HALT ON SEQ ERROR
: R0=0
: LOC. 0=0
: LOC. 0=177777
: R0=377
: R0+5=404, 404 1, LOC. 0-777
: CC=0001?

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
: CONDITIONAL BRANCH INST. AND <===
: REPLACE THE MOVE INSTRUCTION <===
: WHICH FOLLOWS W/ 765 <===

006150
006150 012742 000144
006154 005242
006156 000000
1172 006160 105100
1173 006162 105120
1174 006164 105310
1175 006166 005467 171606
1176 006172 001404

NEG70: MOV #144,-(R2)
INC -(R2)
HALT
NEG71: COMB R0
COMB (R0)+
DECB (R0)
NEG 0
BEQ TST73

: MOVE TO MAILBOX # ***** 144 *****
: SET MSGTYP TO FATAL ERROR
: NEG DID NOT SET CC'S CORRECTLY
: R0=0
: LOC. 0=400, R0=1
: LOC. 0-0
: USE NEG MODE 67 TO TEST FOR ZERO

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
: CONDITIONAL BRANCH INST. AND <===
: REPLACE THE MOVE INSTRUCTION <===
: WHICH FOLLOWS W/ 753 <===

006174 012742 000145
006200 005242
006202 000000

MOV #145,-(R2)
INC -(R2)
HALT

: MOVE TO MAILBOX # ***** 145 *****
: SET MSGTYP TO FATAL ERROR
: DATA RESULT OF NEG WAS INCORRECT
: OR SEQUENCE ERROR

1177
1178
1179
1180
1181
1182

: THIS TEST VERIFIES PROGRAM COUNTER ADDRESSING WITH SOP
: INSTRUCTIONS. CLR MODE 77 IS USED TO CLEAR THE LOCATION FOLLOWING THE
: INSTRUCTION (SOPX). THEN SINGLE OPERAND INSTRUCTIONS WITH MODES 37, 67, AND

1183 :77, USING INDIRECT POINTER SOPXAD ARE USED TO VERIFY THE DATA RESULTS
 1184 :OF THESE INSTRUCTIONS.
 1185 :
 1186 :*****

:TEST 73 TEST SOP OPCODE MODES 2,3,6,7 WITH P.L.
 :*****

006204	005212		TST73:	INC	(R2)	:UPDATE TEST NUMBER
006206	022712	000073		CMP	#73,(R2)	:SEQUENCE ERROR?
006212	001017			BNE	SOPB	:BR TO ERROR HALT ON SEQ ERROR
1187 006214	005027			CLR	(R7)+	:CLEAR NEXT LOCATION: (SOPX)
1188 006216	177777		SOPX:	-1		:USE MODE 27
1189 006220	001404			BEQ	SOPA	

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
 : CONDITIONAL BRANCH INST. AND <====
 : REPLACE THE MOVE INSTRUCTION <====
 : WHICH FOLLOWS W/ 774 <====

006222	012742	000146		MOV	#146,-(R2)	:MOVE TO MAILBOX # ***** 146 *****
006226	005242			INC	-(R2)	:SET MSGTYP TO FATAL ERROR
006230	000000			HALT		:CLR DID NOT SET Z-BIT
1190 006232	005237	006216	SOPA:	INC	@#SOPX	:INC SOPX W/MODE 37
1191 006236	005467	177754		NEG	SOPX	:NEGATE SOPX W/MODE 67
1192 006242	100003			BPL	SOPB	
1193 006244	005277	000012		INC	@SOPXAD	:INC SOPX W/MODE 77
1194 006250	001405			BEQ	TST74	

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
 : CONDITIONAL BRANCH INST. AND <====
 : REPLACE THE MOVE INSTRUCTION <====
 : WHICH FOLLOWS W/ 760 <====

006252			SOPB:	MOV	#147,-(R2)	:MOVE TO MAILBOX # ***** 147 *****
006252	012742	000147		INC	-(R2)	:SET MSGTYP TO FATAL ERROR
006256	005242			HALT		:INC DID NOT SET Z-BIT
006260	000000					: OR SEQUENCE ERROR

1195 006262 006216 SOPXAD: SOPX :INDIRECT ADDRESS OF SOPX

1196 :
 1197 :
 1198 :
 1199 :
 1200 :
 1201 :
 1202 :
 1203 :
 1204 :
 1205 :
 :*****

: THIS TEST VERIFIES SINGLE OPERAND NON-MODIFYING INSTRUCTIONS
 : USING MODE 0. R0 IS SET TO ZERO AND THE CONDITION CODES ARE SET
 : TO THE COMPLEMENT OF THAT EXPECTED BY THE INSTRUCTION. A TST INSTRUCTION
 : IS EXECUTED AND CONDITIONAL BRANCHES ARE USED TO TEST THE CONDITION
 : CODES.
 :*****

:TEST 74 TEST MODE 0 SOP NON-MODIFYING
 :*****

006264	005212		TST74:	INC	(R2)	:UPDATE TEST NUMBER
006266	022712	000074		CMP	#74,(R2)	:SEQUENCE ERROR?
006272	001010			BNE	TST75-10	:BR TO ERROR HALT ON SEQ ERROR
1206 006274	005000			CLR	R0	:INITIALIZE R0=0
1207 006276	000277			SCC		:SET CC=1011
1208 006300	000244			CLZ		
1209 006302	005700			TST	R0	:TRY TST W/ MODE 0
1210 006304	102403			BVS	SNMOA	:CHECK THAT CC=0100
1211 006306	100402			BMI	SNMOA	
1212 006310	103401			BCS	SNMOA	
1213 006312	001404			BEQ	TST75	

006314
006314 012742 000150
006320 005242
006322 000000

SNMOA: MOV #150,-(R2) ;MOVE TO MAILBOX # ***** 150 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CONDITION CODES NOT SET PROPERLY
; OR SEQUENCE ERROR

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 767 <====

1214
1215
1216
1217
1218
1219
1220
1221
1222
1223

: THIS TEST VERIFIES SINGLE OPERAND NON-MODIFYING BYTE INSTRUCTIONS WITH MODE 0.
: R0 IS SET TO 377 AND COMPLEMENT OF THE EXPECTED CONDITION CODES
: IS LOADED IN PSW. A TSTB INSTRUCTION IS EXECUTED AND THE RESULTS
: ARE CHECKED WITH SEVERAL CONDITIONAL BRANCH INSTRUCTIONS.
: THIS VERIFIES THAT THE PROPER BYTE WAS TESTED.

:TEST 75 TEST MODE 0 EVEN BYTE W/ SOP NON-MODIFYING

006324 005212
006326 022712 000075
006332 001010
1224 006334 005000
1225 006336 105100
1226 006340 000277
1227 006342 000250
1228 006344 105700
1229 006346 102402
1230 006350 101401
1231 006352 100404

TST75: INC (R2) ;UPDATE TEST NUMBER
CMP #75,(R2) ;SEQUENCE ERROR?
BNE TST76-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;INITIALIZE
COMB R0 ;R0=377
SCC ;SET CC=0111
CLN
TSTB R0 ;TRY TST EVEN BYTE
BVS SNMBOA ;CHECK CC=1000
BLOS SNMBOA
BMI TST76

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 767 <====

006354
006354 012742 000151
006360 005242
006362 000000

SNMBOA: MOV #151,-(R2) ;MOVE TO MAILBOX # ***** 151 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CONDITION CODES NOT SFT PROPERLY
; OR SEQUENCE ERROR

1232
1233
1234
1235
1236
1237
1238
1239
1240
1241

: THIS TEST VERIFIES SINGLE OPERAND INSTRUCTIONS WITH MODE 1.
: R0 IS USED TO POINT TO AND CLEAR LOC. 0. THE COMPLEMENT OF THE
: EXPECTED CONDITION CODES ARE LOADED IN THE PSW. A TST INSTRUCTION
: IS THEN EXECUTED ON LOC. 0 USING R0 AND CONDITIONAL BRANCHES TEST
: THE RESULTS.

:TEST 76 TEST MODE 1 SOP NON-MODIFYING

006364 005212
006366 022712 000076
006372 001011
1242 006374 005000

TST76: INC (R2) ;UPDATE TEST NUMBER
CMP #76,(R2) ;SEQUENCE ERROR?
BNE TST77-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;POINT TO LOC 0

```

1243 006376 005010          CLR      (R0)          ;CLEAR LOC 0
1244 006400 000277          SCC              ;INITIALIZE
1245 006402 000244          CLZ              ;CC=1011
1246 006404 005710          TST      (R0)          ;TRY TST W/ MODE 1
1247 006406 102403          BVS      SNM1A        ;CHECK CC=0100
1248 006410 103402          BCS      SNM1A
1249 006412 100401          BMI      SNM1A
1250 006414 001404          BEQ      TST77

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 766 <====

```

```

006416          SNM1A:
006416 012742 000152          MOV      #152,-(R2)    ;MOVE TO MAILBOX # ***** 152 *****
006422 005242          INC      -(R2)        ;SET MSGTYP TO FATAL ERROR
006424 000000          HALT              ;CC'S NOT SET PROPERLY
: OR SEQUENCE ERROR

```

1251
1252
1253
1254
1255
1256
1257
1258
1259

```

:*****
: THIS TEST SETS LOCATION 0 TO 377 AND THEN USES R0 TO TEST
: THE EVEN BYTE AND THE ODD BYTE USING SOP BYTE INSTRUCTIONS WITH MODE 1.
: AGAIN, CONDITIONAL BRANCHES ARE USED TO VERIFY THE SETTING OF THE
: PROPER CONDITION CODE BITS.
:*****

```

:TEST 77 TEST MODE 1 BYTE INST. NON-MODIFYING

```

006426 005212 000077          TST77: INC      (R2)          ;UPDATE TEST NUMBER
006430 022712          CMP      #77,(R2)     ;SEQUENCE ERROR?
006434 001026          BNE      TST100-10    ;BR TO ERROR HALT ON SEQ ERROR
1260 006436 005000          CLR      R0           ;POINT TO LOC 0
1261 006440 005010          CLR      (R0)         ;CLEAR LOC 0
1262 006442 105110          COMB     (R0)         ;COMPLEMENT BYTE 0
1263 006444 000277          SCC              ;SET CC=0111
1264 006446 000250          CLN              ;
1265 006450 105710          TSTB     (R0)         ;TRY TST ON EVEN BYTE
1266 006452 102402          BVS      SNMB1A
1267 006454 101401          BLOS     SNMB1A
1268 006456 100404          BMI      SNMB1B

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 766 <====

```

```

006460          SNMB1A:
006460 012742 000153          MOV      #153,-(R2)    ;MOVE TO MAILBOX # ***** 153 *****
006464 005242          INC      -(R2)        ;SET MSGTYP TO FATAL ERROR
006466 000000          HALT              ;CC'S NOT CORRECT

```

```

1269 006470 005000          SNMB1B: CLR      R0
1270 006472 005200          INC      R0
1271 006474 000277          SCC              ;SET CC=1011
1272 006476 000244          CLZ              ;
1273 006500 105710          TSTB     (R0)         ;TRY TO TST AN ODD BYTE
1274 006502 102403          BVS      SNMB1C
1275 006504 103402          BCS      SNMB1C
1276 006506 100401          BMI      SNMB1C
1277 006510 001404          BEQ      TST100

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 751 <====

```

```

006512
006512 012742 000154
006516 005242
006520 000000

```

```

SNMB1C:
MOV #154,-(R2) ;MOVE TO MAILBOX # ***** 154 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CC'S NOT CORRECT
; OR SEQUENCE ERROR

```

1278
1279
1280
1281
1282
1283
1284
1285
1286

```

:*****
: THIS TEST VERIFIES THE SINGLE-OPERAND NON-MODIFYING INSTRUCTIONS
: USING MODE 2. IT USES THE IDENTICAL PROCEDURE EMPLOYED IN THE
: MODE 1 TESTS. ADDITIONALLY, THE REGISTER IS CHECKED TO ASSURE THAT
: IT IS INCREMENTED PROPERLY.
:*****

```

```

:TEST 100 TEST MODE 2 WITH SOP NON-MODIFYING
:*****

```

```

006522 005212
006524 022712 000100
006530 001020
1287 006532 005000
1288 006534 005010
1289 006536 000277
1290 006540 000244
1291 006542 005720
1292 006544 102403
1293 006546 103402
1294 006550 100401
1295 006552 001404

```

```

TST100: INC (R2) ;UPDATE TEST NUMBER
CMP #100,(R2) ;SEQUENCE ERROR?
BNE TST101-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;INITIALIZE R0=0
CLR (R0) ;CLEAR LOC 0
SCC ;SET CC=1011
CLZ
TST (R0)+ ;TRY TST W/ MODE 2
BVS SNM2A ;CHECK CC=0100
BCS SNM2A
BMI SNM2A
BEQ SNM2B

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 766 <====

```

```

006554
006554 012742 000155
006560 005242
006562 000000
1296 006564 005300
1297 006566 005300
1298 006570 001404

```

```

SNM2A:
MOV #155,-(R2) ;MOVE TO MAILBOX # ***** 155 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CC'S NOT CORRECT
SNM2B:
DEC R0 ;RESET R0
DEC R0
BEQ TST101

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 757 <====

```

```

006572 012742 000156
006576 005242
006600 000000

```

```

MOV #156,-(R2) ;MOVE TO MAILBOX # ***** 156 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;MODE 2 DID NOT INC REG CORRECTLY
; OR SEQUENCE ERROR

```

1299
1300
1301
1302
1303
1304

```

:*****
: THIS TEST VERIFIES MODE 2 SINGLE OPERAND NON-MODIFYING BYTE
: INSTRUCTIONS IT USES R0 TO POINT TO LOC. 0. WITH LOCATION 0
: SET TO 377, THE EVEN AND ODD BYTE IS TESTED WITH TSTB INSTRUCTIONS
:*****

```

1305
1306
1307
1308

:TO VERIFY THE CORRECT CC ARE SET. THE REGISTER IS CHECKED FOR
:PROPER INCREMENTING.

:TEST 101 TEST MODE 2 - BYTE W/ SOP NON-MODIFYING

006602 005212
006604 022712 000101
006610 001042
1309 006612 005000
1310 006614 005010
1311 006616 105110
1312 006620 000277
1313 006622 000250
1314 006624 105720
1315 006626 102402
1316 006630 101401
1317 006632 100404

TST101: INC (R2) ;UPDATE TEST NUMBER
CMP #101,(R2) ;SEQUENCE ERROR?
BNE TST102-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;CLEAR R0
CLR (R0) ;CLEAR LOC 0
COMB (R0) ;SET LOC 0=377
SCC ;SET CC=0111
CLN
TSIB (R0)+ ;TRY TST OF EVEN BYTE
BVS SNMB2A
BLOS SNMB2A
BMI SNMB2B

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 766 <====

006634
006634 012742 000157
006640 005242
006642 000000
1318 006644 005300
1319 006646 001404

SNMB2A: MOV #157,-(R2) ;MOVE TO MAILBOX # ***** 157 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CC'S NOT SET CORRECTLY
SNMB2B: DEC R0 ;DECREMENT R0
BEQ SNMB2C

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 760 <====

006650 012742 000160
006654 005242
006656 000000
1320 006660 005200
1321 006662 000277
1322 006664 000244
1323 006666 105720
1324 006670 102403
1325 006672 103402
1326 006674 100401
1327 006676 001404

SNMB2C: MOV #160,-(R2) ;MOVE TO MAILBOX # ***** 160 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;MODE 2 DID NOT INC REG CORRECTLY
SNMB2C: INC R0 ;POINT TO ODD BYTE
SCC ;SET CC=1011
CLZ
TSTB (R0)+ ;TRY TST OF ODD BYTE
BVS SNMB2D ;CHECK CC'S=0100
BCS SNMB2D
BMI SNMB2D
BEQ SNMB2E

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 744 <====

006700
006700 012742 000161
006704 005242
006706 000000
1328 006710 005300
1329 006712 005300
1330 006714 001404

SNMB2D: MOV #161,-(R2) ;MOVE TO MAILBOX # ***** 161 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CC'S NOT CORRECT
SNMB2E: DEC R0
DEC R0
BEQ TST102

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====


```
006716 012742 000162      MOV    #162,-(R2)      ; WHICH FOLLOWS W/ 735      <====  
006722 005242              INC    -(R2)          ;MOV# TO MAILBOX # ***** 162 *****  
006724 000000              HALT                   ;SET MSGTYP TO FATAL ERROR  
                                ;R0 DID NOT INCREMENT PROPERLY  
                                ; OR SEQUENCE ERROR
```

1331
1332
1333
1334
1335
1336
1337
1338
1339

```
*****  
: THIS TEST VERIFIES MODE 3 SINGLE OPERAND NON-MODIFYING INSTRUCTIONS.  
: A POINTER IN A TABLE AT LOC. 376 IS USED TO TEST LOCATION 0.  
: THE CC'S AND THE REGISTER ARE CHECKED FOLLOWING THE  
: TST MODE 3 INSTRUCTION.  
*****
```

```
TEST 102 TEST MODE 3 W/ SOP NON-MODIFYING INSTS  
*****
```

```
006726 005212 000102  
006730 022712  
006734 001022  
1340 006736 005000  
1341 006740 005010  
1342 006742 105100  
1343 006744 005300  
1344 006746 000277  
1345 006750 000244  
1346 006752 005730  
1347 006754 102403  
1348 006756 103402  
1349 006760 100401  
1350 006762 001404
```

```
TST102: INC    (R2)          ;UPDATE TEST NUMBER  
        CMP    #102,(R2)    ;SEQUENCE ERROR?  
        BNE   TST103-10    ;BR TO ERROR HALT ON SEQ ERROR  
        CLR   R0           ;R0=0  
        CLR   (R0)         ;CLEAR LOC 0  
        COMB  R0           ;R0=376  
        DEC   R0           ;  
        SCC                ;SET CC=1011  
        CLZ                ;  
        TST   @(R0)+       ;TRY TST W/ MODE 3  
        BVS   SNM3A        ;CHECK CC=0100  
        BCS   SNM3A  
        BMI   SNM3A  
        BEQ   SNM3B
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
: CONDITIONAL BRANCH INST. AND <====  
: REPLACE THE MOVE INSTRUCTION <====  
: WHICH FOLLOWS W/ 764 <====
```

```
006764  
006764 012742 000163  
006770 005242  
006772 000000  
1351 006774 005300  
1352 006776 105100  
1353 007000 001404
```

```
SNM3A: MOV    #163,-(R2)    ;MOVE TO MAILBOX # ***** 163 *****  
        INC    -(R2)      ;SET MSGTYP TO FATAL ERROR  
        HALT                   ;CC'S NOT CORRECT  
SNM3B: DEC    R0           ;R0=377  
        COMB  R0           ;R0=0  
        BEG   TST103
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
: CONDITIONAL BRANCH INST. AND <====  
: REPLACE THE MOVE INSTRUCTION <====  
: WHICH FOLLOWS W/ 755 <====
```

```
007002 012742 000164  
007006 005242  
007010 000000
```

```
MOV    #164,-(R2)    ;MOVE TO MAILBOX # ***** 164 *****  
INC    -(R2)        ;SET MSGTYP TO FATAL ERROR  
HALT                   ;MODE 3 DID NOT INC REG CORRECTLY  
                                ; OR SEQUENCE ERROR
```

1354
1355
1356
1357
1358
1359
1360
1361

```
*****  
: THIS TEST VERIFIES SOP NON-MODIFYING BYTE INSTRUCTIONS MODE 3  
: LOC. 0 IS SET TO 377. TABLE AT LOC. 402-404 IS USED TO TEST  
: BYTE 0 AND BYTE 1. THE REGISTER IS CHECKED FOR PROPER INCREMENTING AND  
: THE CC'S ARE VERIFIED.  
: THE TABLE AT LOC. 402-404 SHOULD CONTAIN 0 AND 1 BEFORE AND
```

1362
1363
1364

:AFTER THE TEST IS RUN.

:TEST 103 TEST MODE 3 - BYTES W/ SOP NON-MODIFYING INSTS.

007012 005212
007014 022712 000103
007020 001036
1365 007022 005000
1366 007024 005010
1367 007026 105110
1368 007030 105100
1369 007032 005200
1370 007034 005720
1371 007036 000277
1372 007040 000250
1373 007042 105730
1374 007044 102402
1375 007046 101401
1376 007050 100404

TST103: INC (R2) ;UPDATE TEST NUMBER
CMP #103,(R2) ;SEQUENCE ERROR?
BNE TST104-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;R0=0
CLR (R0) ;CLEAR LOC 0
COMB (R0) ;LOC. 0 =377
COMB R0
INC R0
TST (R0)+ ;R0=402
SCC ;CC=0111
CLN
TSTB @(R0)+ ;TRY TST OF EVEN BYTE
BVS SNMB3A ;CHECK CC=1000
BLOS SNMB3A
BMI SNMB3B

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 763 <--- =

007052
007052 012742 000165
007056 005242
007060 000000
1377 007062 000277
1378 007064 000244
1379 007066 105730
1380 007070 102403
1381 007072 103402
1382 007074 100401
1383 007076 001404

SNMB3A: MOV #165,-(R2) ;MOVE TO MAILBOX # ***** 165 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CC'S NOT CORRECT
SNMB3B: SCC ;SET CC=1011
CLZ
TSTB @(R0)+ ;TRY TST OF ODD BYTE
BVS SNMB3C ;CHECK CC=0100
BCS SNMB3C
BMI SNMB3C
BEQ SNMB3D

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-
: CONDITIONAL BRANCH INST. AND <=
: REPLACE THE MOVE INSTRUCTION <-
: WHICH FOLLOWS W/ 750 <

007100
007100 012742 000166
007104 005242
007106 000000
1384 007110 005720
1385 007112 005710
1386 007114 100404

SNMB3C: MOV #166,-(R2) ;MOVE TO MAILBOX # ***** 166 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CC'S NOT CORRECT
SNMB3D: TST (R0)+ ;R0=410
TST (R0)
BMI TST104

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
: CONDITIONAL BRANCH INST. AND <
: REPLACE THE MOVE INSTRUCTION <
: WHICH FOLLOWS W/ 741 < -

007116 012742 000167
007122 005242
007124 000000

MOV #167,-(R2) ;MOVE TO MAILBOX # ***** 167 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;TSTB DID NOT INCREMENT R0 CORRECTLY
; OR SEQUENCE ERROR

1387
1388
1389

: THIS TEST VERIFIES MODE 4 SOP NON-MODIFYING INSTRUCTIONS.

1390 :LOC. 0 IS SET TO -1 AND THE CC'S ARE SET TO THE COMPLEMENT OF THE
 1391 :EXPECTED RESULTS. R0 AND SET TO 2 AND A TST MODE 4 IS EXECUTED.
 1392 :THE CC'S ARE CHECKED WITH CONDITIONAL BRANCH INSTRUCTIONS AND THE REGISTER
 1393 :IS CHECKED FOR PROPER DECREMENTING.
 1394 :
 1395 :*****

:TEST 104 TEST MODE 4 W/ SOP NON-MODIFYING INSTS
 :*****

007126	005212		TST104: INC	(R2)	:UPDATE TEST NUMBER
007130	022712	000104	CMP	#104,(R2)	:SEQUENCE ERROR?
007134	001017		BNE	TST105-10	:BR TO ERROR HALT ON SEQ ERROR
1396 007136	005000		CLR	R0	:R0=0
1397 007140	005010		CLR	(R0)	:LOC 0=0
1398 007142	005120		COM	(R0)+	:LOC 0=-1
1399 007144	000277		SCC		:SET CC=1011
1400 007146	000244		CLZ		
1401 007150	005740		TST	-(R0)	:TRY TST W/ MODE 4
1402 007152	102402		BVS	SNM4A	:CHECK CC=0100
1403 007154	101401		BLOS	SNM4A	
1404 007156	100404		BMI	SNM4B	

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
 : CONDITIONAL BRANCH INST. AND <====
 : REPLACE THE MOVE INSTRUCTION <====
 : WHICH FOLLOWS W/ 766 <====

007160			SNM4A:		
007160	012742	000170	MOV	#170,-(R2)	:MOVE TO MAILBOX # ***** 170 *****
007164	005242		INC	-(R2)	:SET MSGTYP TO FATAL ERROR
007166	000000		HALT		:CC'S NOT CORRECT
1405 007170	005700		SNM4B:		
1406 007172	001404		TST	R0	
			BEQ	TST105	

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-- ==
 : CONDITIONAL BRANCH INST. AND <== -
 : REPLACE THE MOVE INSTRUCTION <-- ==
 : WHICH FOLLOWS W/ 760 < --=

007174	012742	000171	MOV	#171,-(R2)	:MOVE TO MAILBOX # ***** 171 *****
00720C	005242		INC	-(R2)	:SET MSGTYP TO FATAL ERROR
007202	000000		HALT		:TST MODE 4 DID NOT DEC R0 CORRECTLY
					: OR SEQUENCE ERROR

1407
 1408
 1409
 1410
 1411
 1412
 1413
 1414
 1415

:*****
 : THIS TEST VERIFIES MODE 5 SOP NON-MODIFYING INSTRUCTIONS.
 : IT USES A POINTER AT LOC. 376 TO TEST LOC. 0. R0 IS SET
 : TO 400, A TST MODE 5 INSTRUCTION IS EXECUTED AND THE CC'S CHECKED.
 : R0 IS CHECKED TO INSURE PROPER DECREMENTING.
 :*****

:TEST 105 TEST MODE 5 W/ SOP NON-MODIFYING INSTS
 :*****

007204	005212		TST105: INC	(R2)	:UPDATE TEST NUMBER
007206	022712	000105	CMP	#105,(R2)	:SEQUENCE ERROR?
007212	001022		BNE	TST106-10	:BR TO ERROR HALT ON SEQ ERROR
1416 007214	005000		CLR	R0	:R0=0
1417 007216	005010		CLR	(R0)	:LOC 0=0
1418 007220	005110		COM	(R0)	:LOC 0=-1
1419 007222	105100		COMB	R0	:R0=377
1420 007224	005200		INC	R0	:R0=400

```
1421 007226 000277          SCC          ;SET CC=0111
1422 007230 000250          CLN
1423 007232 005750          TST          @-(R0)      ;TRY TST W/ MODE 5
1424 007234 102402          BVS          SNM5A      ;CHECK CC=1000
1425 007236 101401          BLOS         SNM5A
1426 007240 100404          BMI          SNM5B

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 764 <====

007242          SNM5A:
007242 012742 000172        MOV          #172,-(R2)   ;MOVE TO MAILBOX # ***** 172 *****
007246 005242              INC          -(R2)       ;SET MSGTYP TO FATAL ERROR
007250 000000              HALT
1427 007252 005200          SNM5B: INC          R0      ;CC'S NOT SET PROPERLY
1428 007254 105100          COMB         R0      ;R0=377
1429 007256 001404          BEQ          TST106     ;R0=0

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====

007260 012742 000173        MOV          #173,-(R2)   ;MOVE TO MAILBOX # ***** 173 *****
007264 005242              INC          -(R2)       ;SET MSGTYP TO FATAL ERROR
007266 000000              HALT
;MODE 5 DID NOT DEC R0 CORRECTLY
; OR SEQUENCE ERROR
```

1430
1431
1432
1433
1434
1435
1436
1437
1438

```
*****
: THIS TEST VERIFIES MODE 6 SOP NON-MODIFYING INSTRUCTIONS.
: R0 IS SET TO 377 AND A MODE 6 TST INSTRUCTION IS EXECUTED
: USING R0 AND AN OFFSET OF -377. THE CC'S ARE CHECKED AS WELL
: AS R0 TO INSURE IT WAS NOT ALTERED.
*****
```

TEST 106 TEST MODE 6 W/ SOP NON-MODIFYING INSTS

```
007270 005212 000106        TST106: INC          (R2)      ;UPDATE TEST NUMBER
007272 022712              CMP          #106,(R2)   ;SEQUENCE ERROR?
007276 001021              BNE          TST107-10  ;BR TO ERROR HALT ON SEQ ERROR
1439 007300 005000          CLR          R0        ;R0=0
1440 007302 005010          CLR          (R0)      ;LOC 0=0
1441 007304 005110          COM          (R0)      ;LOC 0=-1
1442 007306 105100          COMB         R0        ;R0=377
1443 007310 000277          SCC          ;SET CC=0111
1444 007312 000250          CLN
1445 007314 005760 177401  TST          -377(R0)   ;TRY TST W/ MODE 6
1446 007320 102402          BVS          SNM6A      ;CHECK CC=1000
1447 007322 101401          BLOS         SNM6A
1448 007324 100404          BMI          SNM6B

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 764 <====

007326          SNM6A:
007326 012742 000174        MOV          #174,-(R2)   ;MOVE TO MAILBOX # ***** 174 *****
007332 005242              INC          -(R2)       ;SET MSGTYP TO FATAL ERROR
007334 000000              HALT
;CC'S INCORRECT
```

```

1449 007336 105100          SNM6B: COMB  R0          ;R0=0
1450 007340 001404          BEQ    TST107
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <==
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 756 <===
007342 012742 000175          MOV    #175,-(R2) ;MOVE TO MAILBOX # ***** 175 *****
007346 005242          INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
007350 000000          HALT           ;TST MODE 6 INCORRECTLY CHANGED R0
; OR SEQUENCE ERROR

```

1451
1452
1453
1454
1455
1456
1457
1458
1459

```

:*****
: THIS TEST VERIFIES MODE 7 SOP NON-MODIFYING INSTRUCTIONS.
: IT USES A POINTER TO LOC. 0 STORED AT LOC. 400 TO TST LOC. 0.
: R0 IS SET TO 377 AND LOC. 0 IS TESTED THRU THE POINTER AT 400 USING
: R0 AND AN OFFSET OF 1.
:*****

```

```

:TEST 107 TEST MODE 7 W/ SOP NON-MODIFYING INSTS.
:*****

```

```

007352 005212          TST107: INC    (R2)          ;UPDATE TEST NUMBER
007354 022712 000107          CMP    #107,(R2)        ;SEQUENCE ERROR?
007360 001021          BNE    TST110-10       ;BR TO ERROR HALT ON SEQ ERROR
1460 007362 005000          CLR    R0              ;R0=0
1461 007364 005010          CLR    (R0)            ;LOC 0=0
1462 007366 005110          COM    (R0)            ;LOC 0=-1
1463 007370 105100          COMB   R0              ;R0=377
1464 007372 000277          SCC           ;CC=0111
1465 007374 000250          CLN           ;
1466 007376 005770 000001          TST    @1(R0)          ;TRY TST W/ MODE 7
1467 007402 102402          BVS    SNM7A           ;CHECK CC=1000
1468 007404 101401          BLOS   SNM7A
1469 007406 100404          BMI    SNM7B
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 764 <====

```

```

007410          SNM7A: MOV    #176,-(R2)        ;MOVE TO MAILBOX # ***** 176 *****
007410 012742 000176          INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
007414 005242          HALT           ;CC'S NOT CORRECT
007416 000000          ;R0=0
1470 007420 105100          SNM7B: COMB   R0
1471 007422 001404          BEQ    TST110
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 756 <====

```

```

007424 012742 000177          MOV    #177,-(R2)        ;MOVE TO MAILBOX # ***** 177 *****
007430 005242          INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
007432 000000          HALT           ;TST MODE 7 INCORRECTLY CHANGED R0
; OR SEQUENCE ERROR

```

1472
1473
1474
1475
1476

```

:*****
: THIS TEST VERIFIES MODE 0 DOUBLE OPERAND INSTRUCTIONS. IT SETS
: DATA IN R0 AND R4 AND USES THE ADD INSTRUCTION TO TEST THE DOP
:*****

```

1477
1478
1479

:MICROCODE.

:*****
:TEST 110 TEST MODE 0 DOUBLE-OPERAND (DOP) INSTS.
:*****

007434 005212
007436 022712 000110
007442 001006
1480 007444 005000
1481 007446 005100
1482 007450 005004
1483 007452 060004
1484 007454 005204
1485 007456 001404

TST110: INC (R2) ;UPDATE TEST NUMBER
CMP #110,(R2) ;SEQUENCE ERROR?
BNE TST111-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;R0=0
COM R0 ;R0=-1
CLR R4 ;R4=0
ADD R0,R4 ;TRY ADD: R4=-1
INC R4 ;R4=0
BEQ TST111

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 771 <====
; MOVE TO MAILBOX # ***** 200 *****
; SET MSGTYP TO FATAL ERROR
; ADD INST. FAILED W/ MODE 0
; OR SEQUENCE ERROR

007460 012742 000200
007464 005242
007466 000000

MOV #200,-(R?)
INC -(R2)
HALT

1486
1487
1488
1489
1490
1491
1492

:*****
: THIS TEST VERIFIES THE MOVE INSTRUCTION WITH MODE 0 TO MODE 0.
: THIS TEST IS NECESSARY BECAUSE THIS PARTICULAR INSTRUCTION UTILIZES UNIQUE
: MICROCODE.
:

1494
 007470 005212
 007472 022712 000111
 007476 001006
 1495 007500 005000
 1496 007502 005004
 1497 007504 005100
 1498 007506 010004
 1499 007510 005204
 1500 007512 001404

```

:*****
:TEST 111      MOV MODE 0 TO MODE 0
:*****
TST111: INC      (R2)          ;UPDATE TEST NUMBER
        CMP      #111,(R2)    ;SEQUENCE ERROR?
        BNE     TST112-10    ;BR TO ERROR HALT ON SEQ ERROR
        CLR     R0           ;R0=0
        CLR     R4           ;R4=0
        COM     R0           ;R0=-1
        MOV     R0,R4        ;TRY MOVE -1 TO R4
        INC     R4           ;INC R4
        BEQ     TST112

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 771 <====
; MOVE TO MAILBOX # ***** 201 *****
; SET MSGTYP TO FATAL ERROR
; MOVE FAILED MODE 0 TO MODE 0
; OR SEQUENCE ERROR

```

007514 012742 000201
 007520 005242
 007522 000000

```

MOV     #201,-(R2)
INC     -(R2)
HALT

```

1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508

```

:*****
: THIS TEST VERIFIES THE SUBTRACT INSTRUCTION WITH MODE 0,0.
: THIS TEST IS NECESSARY BECAUSE THIS PARTICULAR INSTRUCTION UTILIZES SOME
: UNIQUE MICROCODE.
:*****

```

```

:TEST 112      TEST SUB MODE 0,0
:*****

```

007524 005212
 007526 022712 000112
 007532 001016
 1509 007534 005000
 1510 007536 005004
 1511 007540 005204
 1512 007542 160400
 1513 007544 100003
 1514 007546 001402
 1515 007550 102401
 1516 007552 103404

```

TST112: INC      (R2)          ;UPDATE TEST NUMBER
        CMP      #112,(R2)    ;SEQUENCE ERROR?
        BNE     TST113-10    ;BR TO ERROR HALT ON SEQ ERROR
        CLR     R0           ;R0=0
        CLR     R4           ;R4=0
        INC     R4           ;R4=1
        SUB     R4,R0        ;TRY SUB 0,0  R0--1
        BPL     SUB0         ;CC=1001
        BEQ     SUB0
        BVS     SUB0
        BCS     SUB0A

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 767 <====

```

007554
 007554 012742 000202
 007560 005242
 007562 000000
 1517 007564 005200
 1518 007566 001404

```

SUB0:   MOV     #202,-(R2)    ;MOVE TO MAILBOX # ***** 202 *****
        INC     -(R2)
        HALT
SUB0A:  INC     R0
        BEQ     TST113

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 761 <====
; MOVE TO MAILBOX # ***** 203 *****
; SET MSGTYP TO FATAL ERROR

```

007570 012742 000203
 007574 005242

```

MOV     #203,-(R2)
INC     -(R2)

```

007576 000000

HALT

;DATA RESULT OF SUB FAILED
; OR SEQUENCE ERROR

1519
1520
1521
1522
1523
1524
1525
1526
1527
1528

: THIS TEST QUICKLY VERIFIES THE REMAINING DOP MODIFYING INSTRUCTIONS
: WITH MODE 0,0 TO PROVIDE A BASELINE FOR SUBSEQUENT TESTS.
: SINGLE OPERAND INSTRUCTIONS ARE USED TO SET UP DATA IN R0 AND R4
: BEFORE EACH OF THE SEVERAL DOP MODIFYING INSTRUCTIONS ARE USED AND
: VERIFIED.

TEST 113 TEST ALL THE DOP INSTRUCTIONS W/ SOURCE MODE 0,0

007600 005212
007602 022712 000113
007606 001051
1529 007610 005000
1530 007612 010004
1531 007614 001404

TST113: INC (R2) ;UPDATE TEST NUMBER
CMP #113,(R2) ;SEQUENCE ERROR?
BNE TST114-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;R0=0
MOV R0,R4 ;TRY MOVE MODE 0,0
BEQ DOPOA

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 774 <====

007616 012742 000204
007622 005242
007624 000000
1532 007626 005200
1533 007630 005100
1534 007632 005104
1535 007634 040004
1536 007636 005304
1537 007640 001404

DOPOA: MOV #204,-(R2) ;MOVE TO MAILBOX # ***** 204 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;Z-BIT NOT SET
INC R0 ;R0=1
COM R0 ;R0=177776
COM R4 ;R4=177777
BIC R0,R4 ;TRY BIC: R4-1
DEC R4 ;R4=0
BEQ DOPOB

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 762 <====

007642 012742 000205
007646 005242
007650 000000
1538 007652 050004
1539 007654 005204
1540 007656 005204
1541 007660 001404

DOPOB: MOV #205,-(R2) ;MOVE TO MAILBOX # ***** 205 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;BIC CLEAR RESULT INCORRECT
BIS R0,R4 ;TRY BIS: R4=177777
INC R4
INC R4 ;R4=0
BEQ DOPOC

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 752 <====

007662 012742 000206
007666 005242
007670 000000
1542 007672 005000
1543 007674 105100
1544 007676 005004
1545 007700 005104
1546 007702 040004
1547 007704 060004

DOPOC: MOV #206,-(R2) ;MOVE TO MAILBOX # ***** 206 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;RESULT OF BIS INCORRECT
CLR R0 ;R0=0
COMB R0 ;R0=377
CLR R4 ;R4=0
COM R4 ;R4=177777
BIC R0,R4 ;R4=177400
ADD R0,R4 ;TRY ADD: R4=177777


```
010022 012742 000212     MOV   #212,-(R2)      ; MOVE TO MAILBOX # ***** 212 ***** <----
010026 005242     INC   -(R2)          ; SET MSGTYP TO FATAL ERROR
010030 000000     HALT                ; DATA RESULT INCORRECT
                                ; OR SEQUENCE ERROR
```

1574
1575
1576
1577
1578
1579
1580

```
*****
: THIS TEST VERIFIES MODE 0,0 DOP NON-MODIFYING INSTRUCTIONS.
: R0 AND R4 ARE PRESET TO 0 AND 1 RESPECTIVELY. COMPARE INSTRUCTIONS ARE
: THEN EXECUTED AND CHECKED. FIRST R4 IS COMPARED TO R0 THEN R0 TO R4.
*****
```

```
*****
: TEST 115 TEST DOP NON-MODIFYING INST. W/ SOURCE MODE 0,0
*****
```

```
010032 005212     INC   (R2)           ; UPDATE TEST NUMBER
010034 022712 000115   CMP   #115,(R2)      ; SEQUENCE ERROR?
010040 001042     BNE  TST116-10       ; BR TO ERROR HALT ON SEQ ERROR
1581 010042 005000     CLR  R0              ; R0=0
1582 010044 005004     CLR  R4              ; R4=0
1583 010046 005204     INC  R4              ; R4-1
1584 010050 020400     CMP  R4,R0           ; TRY COMPARE R4 TO R0
1585 010052 003004     BGT  DNM1
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 772 <====
```

```
010054 012742 000213     MOV   #213,-(R2)      ; MOVE TO MAILBOX # ***** 213 *****
010060 005242     INC   -(R2)          ; SET MSGTYP TO FATAL ERROR
010062 000000     HALT                ; CC'S NOT CORRECT FOR CMP
1586 010064 020004     DNM1: CMP  R0,R4     ; TRY COMPARE R0 TO R4
1587 010066 002404     BLT  DNM2
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 764 <====
```

```
010070 012742 000214     MOV   #214,-(R2)      ; MOVE TO MAILBOX # ***** 214 *****
010074 005242     INC   -(R2)          ; SET MSGTYP TO FATAL ERROR
010076 000000     HALT                ; CC'S NOT CORRECT FOR CMP
1588 010100 005200     DNM2: INC  R0        ; R0=1
1589 010102 020400     CMP  R4,R0           ; TRY COMPARE R4=1 TO R0-1
1590 010104 001404     BEQ  DNM3
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 755 <====
```

```
010106 012742 000215     MOV   #215,-(R2)      ; MOVE TO MAILBOX # ***** 215 *****
010112 005242     INC   -(R2)          ; SET MSGTYP TO FATAL ERROR
010114 000000     HALT                ; CC'S NOT CORRECT (Z=1) FOR CMP
1591 010116 005000     DNM3: CLR  R0        ; R0=0
1592 010120 005100     COM  R0              ; R0=177777
1593 010122 005004     CLR  R4              ; R4=0
1594 010124 030004     BIT  R0,R4           ; TRY BIT R0 TO R4
1595 010126 001404     BEQ  DNM4
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 744 <====
```

```
010130 012742 000216      MOV    #216,-(R2)      ;MOVE TO MAILBOX # ***** 216 *****
010134 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
010136 000000              HALT                    ;CC'S NOT CORRECT FOR BIT
1596 010140 005304      DNM4: DEC    R4          ;R4=177777
1597 010142 030004      BIT    R0,R4          ;TRY BIT AGAIN
1598 010144 100404      BMI   TST116
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
:          CONDITIONAL BRANCH INST. AND <-
:          REPLACE THE MOVE INSTRUCTION <
:          WHICH FOLLOWS W/ 735 <-
```

```
010146 012742 000217      MOV    #217,-(R2)      ;MOVE TO MAILBOX # ***** 217 *****
010152 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
010154 000000              HALT                    ;CC'S NOT CORRECT FOR BIT
: OR SEQUENCE ERROR
```

1599
1600
1601
1602
1603
1604

```
.....
: THIS TEST VERIFIES MODE 0,X DOUBLE OPERAND NON-MODIFYING INSTRUCTIONS.
: IT SETS DATA IN R0 AND LOCATION 0 AND COMPARES THEM USING DOPNM INSTRUCTIONS.
:
: TEST 116      TEST MODE 0,X DOUBLE-OPERAND NON-MODIFYING INSTS.
:
:.....
```

```
010156 005212              TST116: INC    (R2)          ;UPDATE TEST NUMBER
010160 022712 000116      CMP    #116,(R2)       ;SEQUENCE ERROR?
010164 001022              BNE   TST117-10        ;BR TO ERROR HALT ON SEQ ERROR
1605 010166 005000              CLR    R0              ;R0=0
1606 010170 005010              CLR    (R0)            ;LOC. 0=0
1607 010172 005110              COM    (R0)            ;LOC. 0=177777
1608 010174 005200              INC    R0              ;R0=1
1609 010176 020037 000000      CMP    R0,R#0          ;TRY CMP MODE 0,3
1610 010202 100403              BMI   DNM03A           ;CC=0001
1611 010204 001402              BEQ   DNM03A
1612 010206 102401              BVS   DNM03A
1613 010210 103404              BCS   DNM03B
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < ==
:          CONDITIONAL BRANCH INST. AND <- -
:          REPLACE THE MOVE INSTRUCTION <===
:          WHICH FOLLOWS W/ 765 <===
```

```
010212 012742 000220      DNM03A: MOV    #220,-(R2)      ;MOVE TO MAILBOX # ***** 220 *****
010212 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
010220 000000              HALT                    ;CC'S NOT SET CORRECTLY
1614 010222 005300      DNM03B: DEC    R0          ;
1615 010224 001002      BNE   DNM03C           ;
1616 010226 005210      INC    (R0)            ;
1617 010230 001404      BEQ   TST117           ;
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
:          CONDITIONAL BRANCH INST. AND <===
:          REPLACE THE MOVE INSTRUCTION <===
:          WHICH FOLLOWS W/ 755 <===
```

```
010232 012742 000221      DNM03C: MOV    #221,-(R2)      ;MOVE TO MAILBOX # ***** 221 *****
010232 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
010240 000000              HALT                    ;DATA INCORRECTLY MODIFIED BY CMP
: OR SEQUENCE ERROR
```

1618
1619

```
.....
:
```

1620
1621
1622
1623
1624
1625

THIS TEST VERIFIES MODE 1 DOP INSTRUCTIONS. R0 IS SET TO -1
AND LOC 0 TO 1. R4 IS THEN CLEARED AND USED TO POINT TO LOC 0.
IN THE ADD MODE 1 INSTRUCTION, LOC 0 IS ADDED TO R0 AND THE
RESULTS VERIFIED.

:TEST 117 TEST MODE 1 W/ DOP INST.

010242 005212
010244 022712 000117
010250 001007
1626 010252 005000
1627 010254 005100
1628 010256 005004
1629 010260 005014
1630 010262 005214
1631 010264 061400
1632 010266 001404

TST117: INC (R2) ;UPDATE TEST NUMBER
CMP #117,(R2) ;SEQUENCE ERROR?
BNE TST120-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;R0=0
COM R0 ;R0=177777
CLR R4 ;R4=0
CLR (R4) ;LOC 0=0
INC (R4) ;LOC 0=1
ADD (R4),R0 ;TRY ADD SOURCE MODE 1
BEQ TST120

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
: CONDITIONAL BRANCH INST. AND <
: REPLACE THE MOVE INSTRUCTION <
: WHICH FOLLOWS W/ 770 <
: MOVE TO MAILBOX # ***** 222 *****
: SET MSGTYP TO FATAL ERROR
: RESULT OF ADD INCORRECT
: OR SEQUENCE ERROR

010270 012742 000222
010274 005242
010276 000000

MOV #222,-(R2)
INC -(R2)
HALT

1633
1634
1635
1636
1637
1638
1639
1640

THIS TEST VERIFIES MODE 1 DOP BYTE INSTRUCTIONS WHICH ADDRESS
EVEN BYTES. LOC. 0 IS SET TO -1 AND R4 IS CLEARED. THEN R4 IS
SET TO -1 USING A BISB THRU R0 WITH MODE 1.

:TEST 120 TEST MODE 1 - EVEN BYTE W/ DOP INSTS.

010300 005212
010302 022712 000120
010306 001007
1641 010310 005000
1642 010312 005010
1643 010314 005110
1644 010316 005004
1645 010320 151004
1646 010322 105104
1647 010324 001404

TST120: INC (R2) ;UPDATE TEST NUMBER
CMP #120,(R2) ;SEQUENCE ERROR?
BNE TST121-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
CLR R4 ;R4=0
BISB (R0),R4 ;TRY MODE 1- EVEN BYTE W/ DOP
COMB R4 ;R4=0
BEQ TST121

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
: CONDITIONAL BRANCH INST. AND <===
: REPLACE THE MOVE INSTRUCTION <===
: WHICH FOLLOWS W/ 770 <===
: MOVE TO MAILBOX # ***** 223 *****
: SET MSGTYP TO FATAL ERROR
: RESULT OF BISB IS INCORRECT
: OR SEQUENCE ERROR

010326 012742 000223
010332 005242
010334 000000

MOV #223,-(R2)
INC -(R2)
HALT

1648
1649
1650

:

1651
 1652
 1653
 1654
 1655
 1656

: THIS TEST VERIFIES MODE 1 DOP NON-MODIFYING INSTRUCTIONS
 : WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO -1 AND R0 IS CLEARED
 : AND USED AS THE ADDRESSING REGISTER. R4 IS SET TO 377 AND A
 : MODE 1,0 CMPB INSTRUCTION IS USED THE RESULTS VERIFIED.

 : TEST 121 TEST MODE 1 - EVEN BYTE W/ DOP NON-MOD INST

010336 005212
 010340 022712 000121
 010344 001007
 1657 010346 005000
 1658 010350 005010
 1659 010352 005110
 1660 010354 005004
 1661 010356 105104
 1662 010360 121004
 1663 010362 001404

TST121: INC (R2) ;UPDATE TEST NUMBER
 CMP #121,(R2) ;SEQUENCE ERROR?
 BNE TST122-10 ;BR TO ERROR HALT ON SEQ ERROR
 CLR R0 ;R0=0
 CLR (R0) ;LOC 0=0
 COM (R0) ;LOC 0-177777
 CLR R4 ;R4=0
 COMB R4 ;R4=377
 CMPB (R0),R4 ;TRY MODE 1 - EVEN BYTE W/ DOP NON-MODIFYING
 BEQ TST122

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
 : CONDITIONAL BRANCH INST. AND <
 : REPLACE THE MOVE INSTRUCTION <- -
 : WHICH FOLLOWS W/ 770 <
 : MOVE TO MAILBOX # ***** 224 *****
 : SET MSGTYP TO FATAL ERROR
 : RESULT OF CMPB INCORRECT
 : OR SEQUENCE ERROR

010364 012742 000224
 010370 005242
 010372 000000

MOV #224,-(R2)
 INC -(R2)
 HALT

1664
 1665
 1666
 1667
 1668
 1669
 1670
 1671
 1672
 1673
 1674
 1675
 1676

 : THIS TEST VERIFIES MODE 1,0 MOVB INSTRUCTIONS
 : WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO 177400 R0 IS CLEARED AND
 : R4 IS SET TO -1. MOVB ARE USED TO MOVE BYTE 0 TO THIS
 : VERIFIES THAT THE PROPER BYTE WAS SELECTED AND THAT = SIGN-X-TEND
 : FUNCTION WITH MODE 0.
 : THEN LOC. 0 IS COMPLEMENTED AND THE SAME PROCEDURE EXERCISES
 : THE LOGIC FOR COMPLEMENTARY DATA.
 : THIS TEST EXERCISES UNIQUE MICROCODE.

: TEST 122 TEST MOV INSTRUCTION MODE 1,0 EVEN BYTE

010374 005212
 010376 022712 000122
 010402 001020
 1677 010404 005000
 1678 010406 005010
 1679 010410 105110
 1680 010412 005110
 1681 010414 005004
 1682 010416 005104
 1683 010420 111004
 1684 010422 005704
 1685 010424 001404

TST122: INC (R2) ;UPDATE TEST NUMBER
 CMP #122,(R2) ;SEQUENCE ERROR?
 BNE TST123-10 ;BR TO ERROR HALT ON SEQ ERROR
 CLR R0 ;R0=0
 CLR (R0) ;LOC 0=0
 COMB (R0) ;LOC 0=177400
 COM (R0)
 CLR R4 ;R4=0
 COM R4 ;R4=177777
 MOVB (R0),R4 ;R4=0
 TST R4 ;CHECK SIGN OF WORD
 BEQ DOP1

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
 : CONDITIONAL BRANCH INST. AND <
 : REPLACE THE MOVE INSTRUCTION <=
 : WHICH FOLLOWS W/ 766 <.

```
010426 012742 000225      MOV      #225,-(R2)      ;MOVE TO MAILBOX # ***** 225 *****
010432 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
010434 000000              HALT                    ;MOVB SHOULD SIGN X-TEND
1686 010436 005110      DOP1:  COM      (R0)      ;LOC 0=177777
1687 010440 111004      MOV      (R0),R4        ;DO MOVB W/ EVEN BYTE
1688 010442 100404      BMI     TST123
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-- -
;          CONDITIONAL BRANCH INST. AND <
;          REPLACE THE MOVE INSTRUCTION <-- -
;          WHICH FOLLOWS W/ 757 <- -
010444 012742 000226      MOV      #226,-(R2)      ;MOVE TO MAILBOX # ***** 226 *****
010450 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
010452 000000              HALT                    ;MOVB SHOULD SIGN X-TEND
; OR SEQUENCE ERROR
```

1689
1690
1691
1692
1693
1694
1695
1696
1697

```
*****
: THIS TEST VERIFIES MODE 1 DOP INSTRUCTIONS WHICH REFERENCE
: ODD BYTES. LOC. 0 IS SET TO 177400. R0 IS SET TO 0 AND R4 IS
: SET TO 1. THE BISB INSTRUCTION USES THE DATA IN BYTE 1 TO SET BYTE 0.
: THE RESULT IS CHECKED BY INCREMENTING THE WORD (LOC. 0) TO ZERO.
:
: *****
: TEST 123      TEST MODE 1-ODD BYTE W/ DOP INSTS.
: *****
```

```
010454 005212              TST123: INC      (R2)          ;UPDATE TEST NUMBER
010456 022712 000123      CMP      #123,(R2)      ;SEQUENCE ERROR?
010462 001010              BNE     TST124-10      ;BR TO ERROR HALT ON SEQ ERROR
1698 010464 005000              CLR     R0              ;R0=0
1699 010466 005010              CLR     (R0)            ;LOC. 0=0
1700 010470 005004              CLR     R4              ;R4=0
1701 010472 005204              INC     R4              ;R4=1
1702 010474 105114      COMB    (R4)            ;LOC. 0=177400
1703 010476 151410      BISB   (R4),(R0)        ;TRY TO BIS LOW ORDER BITS W/ MODE 1
1704 010500 005210              INC     (R0)            ;CHECK RESULT
1705 010502 001404      BEQ    TST124
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 767 <====
010504 012742 000227      MOV      #227,-(R2)      ;MOVE TO MAILBOX # ***** 227 *****
010510 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
010512 000000              HALT                    ;RESULT OF BISB INCORRECT
; OR SEQUENCE ERROR
```

1706
1707
1708
1709
1710
1711
1712
1713
1714

```
*****
: THIS TEST VERIFIES MODE 2 DOP INSTRUCTIONS. LOC. 0 IS SET TO -1.
: R0 IS CLEARED AND USED AS THE MODE 2 ADDRESSING REGISTER TO MOVE LOC. 0
: TO R7. THE DATA RESULTS ARE VERIFIED AND THE INCREMENTING OF THE REGISTER
: IS CHECKED.
:
: *****
: TEST 124      TEST MODE 2 W/ DOP INSTS.
: *****
```

```
010514 005212              TST124: INC      (R2)          ;UPDATE TEST NUMBER
010516 022712 000124      CMP      #124,(R2)      ;SEQUENCE ERROR?
```

```

1715 010522 001015      BNE      TST125-10      ;BR TO ERROR HALT ON SEQ ERROR
1716 010524 005000      CLR      R0              ;R0=0
1717 010526 005010      CLR      (R0)           ;LOC. 0=0
1718 010530 005110      COM      (R0)           ;LOC. 0=177777
1719 010532 012004      MOV      (R0)+,R4       ;TRY MOVE MODE 2,0
1720 010534 005204      INC      R4              ;CHECK R4
1720 010536 001404      BEQ      DOP2

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 771
<====
<====
<====
<====

```

```

010540 012742 000230      MOV      #230,-(R2)     ;MOVE TO MAILBOX # ***** 230 *****
010544 005242      INC      -(R2)         ;SET MSGTYP TO FATAL ERROR
010546 000000      HALT                    ;RESULT OF MOV INST INCORRECT
1721 010550 005300      DOP2: DEC      R0       ;TEST R0 AFTER MODE 2
1722 010552 005300      DEC      R0
1723 010554 001404      BEQ      TST125

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 762
<====
<====
<====
<====

```

```

010556 012742 000231      MOV      #231,-(R2)     ;MOVE TO MAILBOX # ***** 231 *****
010562 005242      INC      -(R2)         ;SET MSGTYP TO FATAL ERROR
010564 000000      HALT                    ;REGISTER NOT INCREMENTED IN MODE 2
; OR SEQUENCE ERROR

```

1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734

```

*****
; THIS TEST VERIFIES MODE 2 DOP BYTE INSTRUCTIONS, WHICH ADDRESS
; EVEN BYTES. LOC. 0 IS SET TO -1. R0 IS CLEARED AND USED AS THE
; ADDRESSING REGISTER IN A TEST WHICH TRIES TO CLEAR BYTE 1 USING
; BYTE 0 DATA AND A BICB. UNIQUE IN THIS TEST IS USE OF THE
; SAME ADDRESSING REGISTER FOR BOTH SOURCE AND DESTINATION. THE SOURCE AND
; DESTINATION IS CHECKED TO INSURE PROPER FUNCTIONING.
*****

```

TEST 125 TEST MODE 2 - EVEN BYTE W/ DOP INST.

```

010566 005212 000125      TST125: INC      (R2)      ;UPDATE TEST NUMBER
010570 022712      CMP      #125,(R2)     ;SEQUENCE ERROR?
010574 001016      BNE      TST126-10     ;BR TO ERROR HALT ON SEQ ERROR
1735 010576 005000      CLR      R0              ;R0=0
1736 010600 010010      MOV      R0,(R0)       ;LOC. 0=0
1737 010602 005110      COM      (R0)           ;LOC. 0=177777
1738 010604 142010      BICB     (R0)+,(R0)     ;TRY TO CLEAR BYTE 1 FROM BYTE 0 W/ BICB
1739 010606 105737 000001      TSTB     @#1           ;CHECK RESULT
1740 010612 001404      BEQ      DOPB2A

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 770
<====
<====
<====
<====

```

```

010614 012742 000232      MOV      #232,-(R2)     ;MOVE TO MAILBOX # ***** 232 *****
010620 005242      INC      -(R2)         ;SET MSGTYP TO FATAL ERROR
010622 000000      HALT                    ;BICB DESTINATION INCORRECT
1741 010624 105137 000000      DOPB2A: COMB     @#0    ;CHECK BICB SOURCE
1742 010630 001404      BEQ      TST126

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
<====

```

010632 012742 000233
010636 005242
010640 000000

MOV #233,-(R2)
INC -(R2)
HALT

CONDITIONAL BRANCH INST. AND <====
REPLACE THE MOVE INSTRUCTION <----
WHICH FOLLOWS W/ 761 <====
:MOVE TO MAILBOX # ***** 233 *****
:SET MSGTYP TO FATAL ERROR
:BICB SOURCE INCORRECTLY CHANGED
: OR SEQUENCE ERROR

1743
1744
1745
1746
1747
1748
1749
1750

: THIS TEST VERIFIES MODE 2 DOP BYTE INSTRUCTIONS WHICH REFERENCE
: ODD BYTES. R0 IS SET TO 1, LOC. 0 IS SET TO 177400, AND R4 IS CLEARED.
: A MODE 2 MOVVB USES R0 TO MOVE BYTE 1 TO R4. AN INCREMENT
: IS USED TO CHECK THAT THE PROPER BYTE WAS MOVED AND SIGN X-TENDED.

:TEST 126 TEST MODE 2 - ODD BYTE W/ DOP INST.

010642 005212 000126
010644 022712
010650 001017
1751 010652 005000
1752 010654 005004
1753 010656 005010
1754 010660 005110
1755 010662 105120
1756 010664 112004
1757 010666 005204
1758 010670 001404

TST126: INC (R2) ;UPDATE TEST NUMBER
CMP #126,(R2) ;SEQUENCE ERROR?
BNE TST127-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR R0 ;R0=0
CLR R4 ;R4=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
COMB (R0)+ ;LOC 0=177400; R0=1
MOVVB (R0)+,R4 ;TRY DOP MODE 2 W/ ODD BYTE
INC R4 ;CHECK RESULT OF MOVVB
BEQ DOPB2B

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-
: CONDITIONAL BRANCH INST. AND <-
: REPLACE THE MOVE INSTRUCTION <----
: WHICH FOLLOWS W/ 767 <--

010672 012742 000234
010676 005242
010700 000000
1759 010702 005740
1760 010704 005700
1761 010706 001404

MOV #234,-(R2)
INC -(R2)
HALT
DOPB2B: TST -(R0)
TST R0
BEQ TST127

:MOVE TO MAILBOX # ***** 234 *****
:SET MSGTYP TO FATAL ERROR
:RESULT OF MOVVB INCORRECT
:BUMP R0 DOWN BY 2
:CHECK R0

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-
: CONDITIONAL BRANCH INST. AND <-
: REPLACE THE MOVE INSTRUCTION <-
: WHICH FOLLOWS W/ 760 <-

010710 012742 000235
010714 005242
010716 000000

MOV #235,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 235 *****
:SET MSGTYP TO FATAL ERROR
:MODE 2 BYTE DID NOT INCREMENT REG. CORRECTLY
: OR SEQUENCE ERROR

1762
1763
1764
1765
1766
1767
1768
1769
1770

: THIS TEST VERIFIES MODE 3 DOUBLE-OPERAND INSTRUCTIONS.
: LOC. 0 IS LOADED WITH ALTERNATING ZEROES AND ONES; AND R0 IS LOADED
: WITH ALTERNATING ONES AND ZEROES. A MODE 3 BIS IS USED TO SET R0
: TO -1 BY USING LOC. 0 AS THE SOURCE TO BIS THE ZEROES IN R0. THE
: RESULT IS TESTED BY INCREMENTING R0 AND CHECKING FOR ZERO.

:TEST 127 TEST MODE 3 W/ DOP INSTS.

010720 005212
010722 022712 000127
010726 001011
1771 010730 012737 052525 000000
1772 010736 012700 125252
1773 010742 053700 000000
1774 010746 005200
1775 010750 001404

TST127: INC (R2)
CMP #127,(R2)
BNE TST130-10
MOV #052525,@#0
MOV #125252,R0
BIS @#0,R0
INC R0
BEQ TST130

:UPDATE TEST NUMBER
:SEQUENCE ERROR?
:BR TO ERROR HALT ON SEQ ERROR
:MOVE 52525 TO LOC. 0
:SET ALT. ONE AND ZERO IN R0
:TRY TO SET ALL OTHER BITS W/ MODE 3
:TEST RFSULT

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 766 <====

010752 012742 000236
010756 005242
010760 000000

MOV #236,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 236 *****
:SET MSGTYP TO FATAL ERROR
:BIS W/ MODE 3 INCORRECT RESULT
: OR SEQUENCE ERROR

1776
1777
1778
1779
1780
1781
1782
1783

: THIS TEST VERIFIES MODE 3 DOUBLE OPERAND BYTE INSTRUCTIONS WHICH
: ADDRESS EVEN BYTES. BYTE 0 IS SET TO ALTERNATING 1'S AND 0'S; BYTE 1,
: ALTERNATING 0'S AND 1'S. R0 IS CLEARED AND A BISB IS USED TO
: SET THE LOW BYTE OF R0 TO 252.

:TEST 130 TEST MODE 3 - EVEN BYTE W/ DOP INSTS.

010762 005212
010764 022712 000130
010770 001011
1784 010772 012737 052652 000000
1785 011000 005000
1786 011002 153700 000000
1787 011006 022700 000252
1788 011012 001404

TST130: INC (R2)
CMP #130,(R2)
BNE TST131-10
MOV #52652,@#0
CLR R0
BISB @#0,R0
CMP #252,R0
BEQ TST131

:UPDATE TEST NUMBER
:SEQUENCE ERROR?
:BR TO ERROR HALT ON SEQ ERROR
:MOVE 1'S AND 0' PATTERN TO LOC. 0
:R0=0
:TRY R0=252 W/ MODE 3 - EVEN BYTE
:BISB W/ EVEN BYTE SUCCESSFUL?

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 766 <====

011014 012742 000237
011020 005242
011022 000000

MOV #237,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 237 *****
:SET MSGTYP TO FATAL ERROR
:BISB W/ MODE 3 - EVEN BYTE FAILED
: OR SEQUENCE ERROR

1789
1790
1791
1792
1793
1794
1795
1796

: THIS TEST VERIFIES MODE 3 DOUBLE OPERAND BYTE INSTRUCTIONS
: WHICH ADDRESS ODD BYTES. THE SAME PROCEDURE USED IN PREVIOUS
: TEST IS USED HERE. THIS TIME BYTE 1 IS USED AS THE SOURCE BYTE.
: THE EXPECTED RESULT IS: R0 = 125.

:TEST 131 TEST MODE 3 - ODD BYTE W/ DOP INSTS.

011024 005212
011026 022712 000131
011032 001011
1797 011034 012737 052652 000000
1798 011042 005000

TST131: INC (R2)
CMP #131,(R2)
BNE TST132-10
MOV #52652,@#0
CLR R0

:UPDATE TEST NUMBER
:SEQUENCE ERROR?
:BR TO ERROR HALT ON SEQ ERROR
:MOVE 1'S AND 0'S PATTERN TO LOC 0
:R0=0

```
1799 011044 153700 000001      BISB  @#1,R0      ;TRY R0=152 W/ MODE 3 - ODD BYTE
1800 011050 022700 000125      CMP   #125,R0    ;R0=125?
1801 011054 001404              BEQ   TST132     ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <----
;          CONDITIONAL BRANCH INST. AND <----
;          REPLACE THE MOVE INSTRUCTION <----
;          WHICH FOLLOWS W/ 766 <----
011056 012742 000240      MOV   #240,-(R2) ;MOVE TO MAILBOX # ***** 240 *****
011062 005242              INC   -(R2)      ;SET MSGTYP TO FATAL ERROR
011064 000000              HALT              ;BISB W/ MODE 3 - ODD BYTE FAILED
; OR SEQUENCE ERROR
```

1802
1803

```
*****  
;TEST 132 TEST DEST. MODE 0-BYTE W/ DOP NON-MODIFYING MST  
*****  
011066 005212 000132      TST132: INC (R2) ;UPDATE TEST NUMBER  
011070 022712 000132      CMP   #132,(R2) ;SEQUENCE ERROR?  
011074 001017              BNE   TST133-10 ;BR TO ERROR HALT ON SEQ ERROR  
1804 011076 005000      CLR   R0 ;R0=0  
1805 011100 105100      COMB R0 ;R0=377  
1806 011102 000263      +SEC!SEV ;SET C AND V BITS  
1807 011104 132700 000200      BITB #200,R0 ;TRY DOPNM DEST. MODE 0-BYTE  
1808 011110 001403      BEQ   DNMB0A ;BR TO ERROR IF Z BIT SET  
1809 011112 102402      BVS   DNMB0A ;BR TO ERROR IF V BIT SET  
1810 011114 103001      BCC   DNMB0A ;BR TO ERROR IF C BIT CLEAR.  
1811 011116 100404      BMI   DNMB0B
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <  
;          CONDITIONAL BRANCH INST. AND <  
;          REPLACE THE MOVE INSTRUCTION <  
;          WHICH FOLLOWS W/ 766 <
```

```
011120 012742 000241      DNMB0A: MOV #241,-(R2) ;MOVE TO MAILBOX # ***** 241 *****  
011120 012742 000241      INC   -(R2) ;SET MSGTYP TO FATAL ERROR  
011124 005242              HALT ;CC'S INCORRECT  
1812 011130 105100      DNMB0B: COMB R0 ;CHECK DESTINATION DATA  
1813 011132 001404      BEQ   TST133
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====  
;          CONDITIONAL BRANCH INST. AND <=====  
;          REPLACE THE MOVE INSTRUCTION <=====  
;          WHICH FOLLOWS W/ 760 <=====  
011134 012742 000242      MOV   #242,-(R2) ;MOVE TO MAILBOX # ***** 242 *****  
011140 005242              INC   -(R2) ;SET MSGTYP TO FATAL ERROR  
011142 000000              HALT ;DEST. DATA MODIFIED  
; OR SEQUENCE ERROR
```

1814
1815

```
*****  
;TEST 133 TEST DEST. MODE 1 W/ DOP NON-MODIFYING INST  
*****  
011144 005212 000133      TST133: INC (R2) ;UPDATE TEST NUMBER  
011146 022712 000133      CMP   #133,(R2) ;SEQUENCE ERROR?  
011152 001017              BNE   TST134-10 ;BR TO ERROR HALT ON SEQ ERROR  
1816 011154 005000      CLR   R0 ;R0=0  
1817 011156 005010      CLR   (R0) ;LOC. 0=0  
1818 011160 000241      CLC ;CLEAR C BIT  
1819 011162 032710 177777      BIT  #177777,(R0) ;TRY DOPNM DEST. MODE 1  
1820 011166 100403      BMI   DNM1A ;BR TO ERROR IF N BIT SET  
1821 011170 102402      BVS   DNM1A ;BR TO ERROR IF V BIT SET
```

```

1822 011172 103401      BCS      DNM1A      ;BR TO ERROR IF C BIT SET
1823 011174 001404      BEQ      DNM1B
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 766 <====

```

```

011176      DNM1A:
011176 012742 000243      MOV      #243,-(R2) ;MOVE TO MAILBOX # ***** 243 *****
011202      INC      -(R2) ;SET MSGTYP TO FATAL ERROR
011204      HALT     ;COND. CODES INCORRECT
1824 011206 005710      DNM1B: TST      (R0) ;CHECK TEST DATA
1825 011210 001404      BEQ      TST134
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 760 <====

```

```

011212 012742 000244      MOV      #244,-(R2) ;MOVE TO MAILBOX # ***** 244 *****
011216      INC      -(R2) ;SET MSGTYP TO FATAL ERROR
011220      HALT     ;DESTINATION DATA MODIFIED
; OR SEQUENCE ERROR

```

```

1826
1827 ;*****
;TEST 134 TEST DEST, MODE 2 W/ DOP NON-MODIFYING INST.
;*****

```

```

011222 005212      TST134: INC      (R2) ;UPDATE TEST NUMBER
011224 022712 000134      CMP      #134,(R2) ;SEQUENCE ERROR?
011230 001027      BNE     TST135-10 ;BR TO ERROR HALT ON SEQ ERROR
1828 011232 005000      CLR      R0 ;R0=0
1829 011234 005010      CLR      (R0) ;LOC. 0=0
1830 011236 052710 125252      BIS     #125252,(R0) ;LOC. 0=125252
1831 011242 032720 077777      BIT     #77777,(R0)+ ;TRY DOPNM INST W/ MODE 2
1832 011246 102402      BVS     DNM2A ;BR TO ERROR IF V BIT SET
1833 011250 001401      BEQ     DNM2A ;BR TO ERROR IF Z-BIT SET
1834 011252 100004      BPL     DNM2B
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 766 <====

```

```

011254      DNM2A:
011254 012742 000245      MOV      #245,-(R2) ;MOVE TO MAILBOX # ***** 245 *****
011260      INC      -(R2) ;SET MSGTYP TO FATAL ERROR
011262      HALT     ;COND. CODES INCORRECT
1835 011264 005300      DNM2B: DEC      R0 ;DECREMENT R0 TO CHECK IT.
1836 011266 005300      DEC      R0
1837 011270 001404      BEQ     DNM2D
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 757 <====

```

```

011272      DNM2C:
011272 012742 000246      MOV      #246,-(R2) ;MOVE TO MAILBOX # ***** 246 *****
011276      INC      -(R2) ;SET MSGTYP TO FATAL ERROR
011300      HALT     ;MODE 2 REGISTER NOT INCREMENTED BY 2
1838 011302 022710 125252      DNM2D: CMP      #125252,(R0) ;CHECK DEST. DATA
1839 011306 001404      BEQ     TST135
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====

```

```

:                                     REPLACE THE MOVE INSTRUCTION <====
:                                     WHICH FOLLOWS W/ 750 <====
011310 012742 000247          MOV    #247,-(R2)      ;MOVE TO MAILBOX # ***** 247 *****
011314 005242                INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
011316 000000                HALT                ;DEST. DATA MODIFIED
:                                     ; OR SEQUENCE ERROR

```

1840
1841

```

;*****
;TEST 135          TEST DEST. MODE 2-BYTE, W/DOP NON-MODIFYING INST
;*****

```

```

C11320 005212          ST135: INC    (R2)          ;UPDATE TEST NUMBER
011322 022712 000135    CMP    #135,(R2)       ;SEQUENCE ERROR?
011326 001051          BNE    TST136-10      ;BR TO ERROR HALT ON SEQ ERROR
1842 011330 005000      CLR    R0             ;R0=0
1843 011332 005010      CLR    (R0)          ;LOC. 0=0
1844 011334 052710 052652  BIS    #52652,(R0)    ;LOC. 0=52652
1845 011340 000263      +SEC!SEV            ;SET C AND V BITS
1846 011342 132720 000201  BITB   #201,(R0)+    ;TRY DOPNM INST. W/ MODE 2 EVEN BYTE
1847 011346 001403      BEQ    DNMB2A         ;BR TO ERROR IF Z-BIT SET
1848 011350 103002      BCC    DNMB2A         ;BR TO ERROR IF C-BIT CLEAR
1849 011352 102401      BVS    DNMB2A         ;BR TO ERROR IF V-BIT SET
1850 011354 100404      BMI    DNMB2B

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:                                     CONDITIONAL BRANCH INST. AND <====
:                                     REPLACE THE MOVE INSTRUCTION <====
:                                     WHICH FOLLOWS W/ 764 <====

```

```

011356 012742 000250    DNMB2A: MOV    #250,-(R2)    ;MOVE TO MAILBOX # ***** 250 *****
011356 005242          INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
011362 000000          HALT                ;COND. CODES INCORRECT
1851 011366 005300    DNMB2B: DEC    R0             ;CHECK DEST. REGISTER.
1852 011370 001404      BEQ    DNMB2C

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:                                     CONDITIONAL BRANCH INST. AND <====
:                                     REPLACE THE MOVE INSTRUCTION <====
:                                     WHICH FOLLOWS W/ 756 <====

```

```

011372 012742 000251          MOV    #251,-(R2)    ;MOVE TO MAILBOX # ***** 251 *****
011376 005242          INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
011400 000000          HALT                ;DEST. REGISTER NOT INCREMENTED BY 1
1853 011402 005200    DNMB2C: INC    R0             ;R0=1
1854 011404 132720 000201  BITB   #201,(R0)+    ;TRY DOPNM INST. W/MODE 2-ODD BYTE
1855 011410 001402      BEQ    DNMB2D         ;BR TO ERROR IF Z-BIT SET
1856 011412 102401      BVS    DNMB2D         ;BR TO ERROR IF V-BIT SET
1857 011414 100004      BPL    DNMB2E

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:                                     CONDITIONAL BRANCH INST. AND <====
:                                     REPLACE THE MOVE INSTRUCTION <====
:                                     WHICH FOLLOWS W/ 744 <====

```

```

011416 012742 000252    DNMB2D: MOV    #252,-(R2)    ;MOVE TO MAILBOX # ***** 252 *****
011416 005242          INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
011422 000000          HALT                ;COND. CODES INCORRECT
1858 011426 005300    DNMB2E: DEC    R0             ;DEC R0 TO CHECK IT.
1859 011430 005300      DEC    R0
1860 011432 001404      BEQ    DNMB2F

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:                                     CONDITIONAL BRANCH INST. AND <====

```

```

:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 735 < ---
011434 012742 000253          MOV      #253,-(R2)  ;MOVE TO MAILBOX # ***** 253 *****
011440 005242          INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
011442 000000          HALT                    ;DEST. REGISTER NOT INCREMENTED BY 1
1861 011444 022710 052652  DNMB2F: CMP      #52652,(R0) ;CHECK DEST. DATA IS UNMODIFIED
1862 011450 001404          BEQ      TST136

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 726 <====

011452 012742 000254          MOV      #254,-(R2)  ;MOVE TO MAILBOX # ***** 254 *****
011456 005242          INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
011460 000000          HALT                    ;DEST. DATA WAS MODIFIED.
: OR SEQUENCE ERROR

```

1863
1864
1865

```

:*****
:TEST 136          TEST DEST. MODE 3-BYTES W/DOP NON-MODIFYING INST.
:*****

```

```

TST136: INC      (R2)          ;UPDATE TEST NUMBER
011462 005212          CMP      #136,(R2)      ;SEQUENCE ERROR?
011464 022712 000136          BNE     TST137-10      ;BR TO ERROR HALT ON SEQ ERROR
1866 011472 005000          CLR     R0            ;R0=0
1867 011474 005010          CLR     (R0)          ;LOC. 0=0
1868 011476 052710 125125          BIS     #125125,(R0)  ;LOC. 0=125125
1869 011502 105100          COMB   R0            ;R0=377
1870 011504 005200          INC     R0            ;R0=400
1871 011506 005010          CLR     (R0)          ;LOC. 400=0
1872 011510 000263          +SEC!SEV ;C-BIT=V-BIT=1
1873 011512 132730 000201          BITB   #201,a(R0)+    ;TRY DOPNM W/MODE 3-EVEN BYTE
1874 011516 001403          BEQ     DNMB3A        ;BR TO ERROR IF Z BIT SET
1875 011520 102402          BVS    DNMB3A        ;BR TO ERROR IF V BIT SET
1876 011522 103001          BCC    DNMB3A        ;BR TO ERROR IF C BIT CLEAR
1877 011524 100004

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <----
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 761 <----

```

```

DNMB3A: MOV      #255,-(R2)  ;MOVE TO MAILBOX # ***** 255 *****
011526 012742 000255          INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
011532 005242          HALT                    ;COND. CODES INCORRECT
1878 011536 022700 000402  DNMB3B: CMP      #402,R0  ;CHECK DEST. REGISTER INC. BY 2 AND INC BY 2 AGAIN
1879 011542 001404          BEQ     DNMB3C

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < ---
:          CONDITIONAL BRANCH INST. AND < ---
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 752 <====

011544 012742 000256          MOV      #256,-(R2)  ;MOVE TO MAILBOX # ***** 256 *****
011550 005242          INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
011552 000000          HALT                    ;DEST. REGISTER NOT INCREMENTED BY 2
1880 011554 005200          DNMB3C: INC     R0            ;R0=404
1881 011556 005200          INC     R0
1882 011560 132730 000201          BITB   #201,a(R0)+    ;TRY DOPNM DEST MODE 3-BYTE(ODD)
1883 011564 001402          BEQ     DNMB3D        ;BR TO ERROR IF Z BIT SET
1884 011566 102401          BVS    DNMB3D        ;BR TO ERROR IF V BIT SET

```

```

1885 011570 100404      BMI      DNMB3E      ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- -
;                                     CONDITIONAL BRANCH INST. AND <
;                                     REPLACE THE MOVE INSTRUCTION <- -
;                                     WHICH FOLLOWS W/ 737 <- -
      011572      DNMB3D:
011572 012742 000257      MOV      #257,-(R2)      ;MOVE TO MAILBOX # ***** 257 *****
011576 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
1886 011600 000000      HALT      ;COND. CODES INCORRECT
1887 011602 005004      DNMB3E: CLR      R4      ;R4=0
1888 011604 022714 125125      CMP      #125125,(R4) ;CHECK DEST. DATA
1888 011610 001404      BEQ      TST137
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 727 <====
      011612 012742 000260      MOV      #260,-(R2)      ;MOVE TO MAILBOX # ***** 260 *****
011616 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
011620 000000      HALT      ;DEST. DATA MODIFIED
; OR SEQUENCE ERROR

1889
1890
;*****
;TEST 137 TEST DEST. MODE 4 W/DOP NON-MODIFYING INST.
;*****
      011622 005212      TST137: INC      (R2)      ;UPDATE TEST NUMBER
011624 022712 000137      CMP      #137,(R2)      ;SEQUENCE ERROR?
1891 011630 001033      BNE      TST140-10      ;BR TO ERROR HALT ON SEQ ERROR
1892 011632 005000      CLR      R0      ;R0=0
1893 011634 005010      CLR      (R0)      ;LOC. 0=0
1894 011636 052710 125252      BIS      #125252,(R0) ;LOC. 0=125125
1895 011642 052700 000002      BIS      #2,R0      ;R0=2
1896 011646 000277      SCC      ;SET ALL COND. CODE BITS
1897 011650 032740 020000      BIT      #20000,-(R0) ;TRY DOPNM W/ MODE 4
1898 011654 100403      BMI      DNM4A      ;BR TO ERROR IF N-BIT SET
1899 011656 102402      BVS      DNM4A      ;BR TO ERROR IF V-BIT SET
1900 011660 103001      BCC      DNM4A      ;BR TO ERROR IF C-BIT CHAR
1900 011662 001004      BNE      DNM4B
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 762 <====
      011664      DNM4A:
011664 012742 000261      MOV      #261,-(R2)      ;MOVE TO MAILBOX # ***** 261 *****
011670 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
011672 000000      HALT      ;COND. CODES INCORRECT
1901 011674 005700      DNM4B: TST      R0      ;CHECK DEST. REGISTER
1902 011676 001404      BEQ      DNM4C
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 754 <====
      011700 012742 000262      MOV      #262,-(R2)      ;MOVE TO MAILBOX # ***** 262 *****
011704 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
011706 000000      HALT      ;DEST. REGISTER NOT DECFMENTED BY 2
1903 011710 022737 125252 000000      DNM4C: CMP      #125252,@#0 ;CHECK DEST. DATA
1904 011716 001404      BEQ      TST140
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- -

```

```

:          CONDITIONAL BRANCH INST. AND <----
:          REPLACE THE MOVE INSTRUCTION <----
:          WHICH FOLLOWS W/ 744 <----
011720 012742 000263      MOV      #263,-(R2)  ;MOVE TO MAILBOX # ***** 263 *****
011724 005242            INC      -(R2)    ;SET MSGTYP TO FATAL ERROR
011726 000000            HALT                    ;DEST. DATA MODIFIED
:          OR SEQUENCE ERROR

1905
1906
:*****
:TEST 140      TEST DEST. MODE 4-BYTE W/ DOP NON-MODIFYING INST.
:*****
TST140: INC      (R2)          ;UPDATE TEST NUMBER
      CMP      #140,(R2)      ;SEQUENCE ERROR?
      BNE     TST141-10      ;BR TO ERROR HALT ON SEQ ERROR
      CLR     R0              ;R0=0
      CLR     (R0)           ;LOC. 0=0
      BIS     #52652,(R0)    ;LOC. 0=52652
      BIS     #2,R0          ;R0=2
      CCC                    ;COND. CODES=C
      BITB   #201,-(R0)     ;TRY DOPNM INST W/MODE 4 ODD BYTE
      BVS     DNMB4A        ;BR TO ERROR IF V BIT SET
      BEQ     DNMB4A        ;BR TO ERROR IF Z BIT SET
      BCS     DNMB4A        ;BR TO ERROR IF C BIT SET
      BNE     DNMB4B

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 762 <====

011772      DNMB4A: MOV      #264,-(R2)  ;MOVE TO MAILBOX # ***** 264 *****
011772 012742 000264      INC      -(R2)    ;SET MSGTYP TO FATAL ERROR
011776 005242            HALT                    ;COND. CODES INCORRECT
1917 012002 022700 000001  DNMB4B: CMP      #1,R0    ;CHECK DEST. REGISTER
1918 012006 001404      BEQ      DNMB4C

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 753 <====

012010      MOV      #265,-(R2)  ;MOVE TO MAILBOX # ***** 265 *****
012014 012742 000265      INC      -(R2)    ;SET MSGTYP TO FATAL ERROR
012016 005242            HALT                    ;DEST REG. NOT DECREMENTED BY 1
1919 012020 132740 000201  DNMB4C: BITB   #201,-(R0)  ;TRY DOPNM INST. W/MODE 4 EVEN BYTE
1920 012024 001401      BEQ      DNMB4D    ;BR TO ERROR IF Z-BIT SET
1921 012026 100404      BMI      DNMB4E

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 743 <====

012030      DNMB4D: MOV      #266,-(R2)  ;MOVE TO MAILBOX # ***** 266 *****
012030 012742 000266      INC      -(R2)    ;SET MSGTYP TO FATAL ERROR
012034 005242            HALT                    ;COND. CODES INCORRECT
1922 012040 005700 000000  DNMB4E: TST      R0    ;CHECK DEST. REGISTER
1923 012042 001404      BEQ      DNMB4F

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <----
:          CONDITIONAL BRANCH INST. AND <----
:          REPLACE THE MOVE INSTRUCTION <----

```


: OR SEQUENCE ERROR

1942
 1943

 :TEST 142 TEST DEST. MODE 6 W/DOP NON-MODIFYING INST.

	012202	005212				TST142: INC	(R2)	:UPDATE TEST NUMBER
	012204	022712	000142			CMP	#142,(R2)	:SEQUENCE ERROR?
	012210	001033				BNE	TST143-10	:BR TO ERROR HALT ON SEQ ERROR
1944	012212	005000				CLR	R0	:R0=0
1945	012214	005010				CLR	(R0)	:LOC> 0=0
1946	012216	052710	000001			BIS	#1,(R0)	:LOC. 0=1
1947	012222	005100				COM	R0	:R0=-1 C-BIT=1
1948	012224	032760	000001	000001		BIT	#1,1(R0)	:TRY DOPNM W/MODE 6
1949	012232	001403				BEQ	DNM6A	:BR TO ERROR IF Z-BIT SET
1950	012234	102402				BVS	DNM6A	:BR TO ERROR IF V-BIT SET
1951	012236	103001				BCC	DNM6A	:BR TO ERROR IF C-BIT CLEAR
1952	012240	100004				BPL	DNM6B	

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <== =
 : CONDITIONAL BRANCH INST. AND <= ==
 : REPLACE THE MOVE INSTRUCTION <-===
 : WHICH FOLLOWS W/ 763 <====

	012242					DNM6A: MOV	#274,-(R2)	:MOVE TO MAILBOX # ***** 274 *****
	012242	012742	000274			INC	-(R2)	:SET MSGTYP TO FATAL ERROR
	012246	005242				HALT		:COND CODES INCORRECT
	012250	000000						:CHECK DEST. REGISTER
1953	012252	022700	177777			DNM6B: CMP	#-1,R0	
1954	012256	001404				BEQ	DNM6C	

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
 : CONDITIONAL BRANCH INST. AND <====
 : REPLACE THE MOVE INSTRUCTION <====
 : WHICH FOLLOWS W/ 754 <====

	012260	012742	000275			MOV	#275,-(R2)	:MOVE TO MAILBOX # ***** 275 *****
	012264	005242				INC	-(R2)	:SET MSGTYP TO FATAL ERROR
	012266	000000				HALT		:DEST. REGISTER MODIFIED
1955	012270	022737	000001	000000		DNM6C: CMP	#1,@#0	:CHECK DEST. DATA
1956	012276	001404				BEQ	TST143	

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
 : CONDITIONAL BRANCH INST. AND <====
 : REPLACE THE MOVE INSTRUCTION <====
 : WHICH FOLLOWS W/ 744 <====

	012300	012742	000276			MOV	#276,-(R2)	:MOVE TO MAILBOX # ***** 276 *****
	012304	005242				INC	-(R2)	:SET MSGTYP TO FATAL ERROR
	012306	000000				HALT		:DEST. DATA MODIFIED

: OR SEQUENCE ERROR

1957
 1958

 :TEST 143 TEST DEST MODE 7 W/DOP NON-MODIFYING INST.

	012310	005212				TST143: INC	(R2)	:UPDATE TEST NUMBER
	012312	022712	000143			CMP	#143,(R2)	:SEQUENCE ERROR?
	012316	001034				BNE	TST144-10	:BR TO ERROR HALT ON SEQ ERROR
1959	012320	005000				CLR	R0	:R0=0
1960	012322	005010				CLR	(R0)	:LOC. 0=0 C-BIT=0
1961	012324	052710	125125			BIS	#125125,(R0)	:LOC. 0=125125
1962	012330	052700	000001			BIS	#1,R0	:R0=1
1963	012334	132770	000125	000403		BITB	#125,@403(R0)	:TRY DOPNM W/MODE 7
1964	012342	102403				BVS	DNM7A	:BR TO ERROR IF V-BIT SET

```

1965 012344 100402      BMI      DNM7A      ;BR TO ERROR IF N-BIT SET
1966 012346 103401      BCS      DNM7A      ;BR TO ERROR IF C-BIT SET
1967 012350 001404      BEQ      DNM7B

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
;          CONDITIONAL BRANCH INST. AND <=
;          REPLACE THE MOVE INSTRUCTION <
;          WHICH FOLLOWS W/ 762 < --

```

```

      012352      DNM7A:
012352 012742 000277      MOV      #277,-(R2)      ;MOVE TO MAILBOX # ***** 277 *****
012356 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
012360 000000      HALT     ;COND. CODES INCORRECT
1968 012362 022700 000001 DNM7B:  CMP      #1,R0      ;CHECK DEST. REGISTER
1969 012366 001404      BEQ      DNM7C

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
;          CONDITIONAL BRANCH INST. AND <
;          REPLACE THE MOVE INSTRUCTION <
;          WHICH FOLLOWS W/ 753 <

```

```

      012370 012742 000300      MOV      #300,-(R2)      ;MOVE TO MAILBOX # ***** 300 *****
012374 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
012376 000000      HALT     ;DESTINATION REGISTER MODIFIED
1970 012400 022737 125125 000000 DNM7C:  CMP      #125125,@#0
1971 012406 001404      BEQ      TST144      ;CHECK DEST. DATA

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
;          CONDITIONAL BRANCH INST. AND <
;          REPLACE THE MOVE INSTRUCTION <
;          WHICH FOLLOWS W/ 743 <
; MOVE TO MAILBOX # ***** 301 *****
; SET MSGTYP TO FATAL ERROR
; DEST. DATA INCORRECT
; OR SEQUENCE ERROR

```

```

012410 012742 000301      MOV      #301,-(R2)
012414 005242      INC      -(R2)
012416 000000      HALT

```

1972
1973
1974
1975
1976
1977
1978
1979

```

*****
: THIS TEST VERIFIES THE MOV DESTINATION MODE 1 INSTRUCTION.
: DATA IS SET IN R0 USING SOP INSTRUCTIONS AND THEN MOVED TO LOC. 0
: USING MOV SRC MODE 0, DEST. MODE 1.
*****

```

```

*****
: TEST 144      TEST MOV DESTINATION MODE 1
*****

```

```

      012420 005212      INC      (R2)      ;UPDATE TEST NUMBER
012422 022712 000144      CMP      #144,(R2)      ;SEQUENCE ERROR?
012426 001016      BNE     TST145-10      ;BR TO ERROR HALT ON SEQ ERROR
1980 012430 005000      CLR     R0      ;R0=0
1981 012432 005010      CLR     (R0)      ;LOC. 0=0
1982 012434 005100      COM     R0      ;R0=-1
1983 012436 005004      CLR     R4      ;R4 POINTS TO LOC. 0
1984 012440 010014      MOV     R0,(R4)      ;TRY MOVE MODE 0,1
1985 012442 102402      BVS     MDM1A      ;BR TO ERROR IF V SET
1986 012444 001401      BEQ     MDM1A      ;BR TO ERROR IF Z SET
1987 012446 100404      BMI     MDM1B

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
;          CONDITIONAL BRANCH INST. AND <
;          REPLACE THE MOVF INSTRUCTION <
;          WHICH FOLLOWS W/ 767 <

```

```

      012450      MDM1A:
012450 012742 000302      MOV      #302,-(R2)      ;MOVE TO MAILBOX # ***** 302 *****

```

```
012454 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
012456 000000          HALT                    ;CONDITION CODE NOT CORRECT
1988 012460 005704      MDM1B: TST      R4
1989 012462 001404      BEQ      TST145
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <==
;          REPLACE THE MOVE INSTRUCTION <---
;          WHICH FOLLOWS W/ 761 <====
012464 012742 000303      MOV      #303,-(R2)      ;MOVE TO MAILBOX # ***** 303 *****
012470 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
012472 000000          HALT                    ;DESTINATION REGISTER INCORRECTLY ALTERED
; OR SEQUENCE ERROR
```

1990
1991
1992
1993
1994
1995
1996
1997

```
*****
: THIS TEST VERIFIES THE MOV DESTINATION MODE 2 INSTRUCTION.
: DATA IS SET IN R0 USING SOP INSTRUCTIONS AND THEN MOVED
: TO LOCATION 0 USING MOV SRC MODE 0, DEST. MODE 1.
*****
: TEST 145      TEST MOV DESTINATION MODE 2
*****
```

```
012474 005212          INC      (R2)          ;UPDATE TEST NUMBER
012476 022712 000145      CMP      #145,(R2)      ;SEQUENCE ERROR?
012502 001025          BNE     TST146-10      ;BR TO ERROR HALT ON SEQ ERROR
1998 012504 005000          CLR     R0             ;R0=0
1999 012506 005010          CLR     (R0)           ;LOC.0=0
2000 012510 005110          COM     (R0)           ;LOC. 0= 1
Z 2001 012512 010020          MOV     R0,(R0)+       ;TRY MOVE MODE 0,2
2002 012514 100402          BMI     MDM2A         ;BR TO ERROR IF N SET
2003 012516 102401          BVS     MDM2A         ;BR TO ERROR IF V SET
2004 012520 001404          BEQ     MDM2B
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- -
;          CONDITIONAL BRANCH INST. AND <- -
;          REPLACE THE MOVE INSTRUCTION <== -
;          WHICH FOLLOWS W/ 770 <====
```

```
012522          MDM2A: MOV      #304,-(R2)      ;MOVE TO MAILBOX # ***** 304 *****
012522 012742 000304      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
012526 005242          HALT                    ;CC'S INCORRECT
2005 012532 005300      MDM2B: DEC     R0
2006 012534 005300          DEC     R0
2007 012536 001404          BEQ     MDM2D
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <== =
;          WHICH FOLLOWS W/ 761 <====
```

```
012540          MDM2C: MOV      #305,-(R2)      ;MOVE TO MAILBOX # ***** 305 *****
012540 012742 000305      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
012544 005242          HALT                    ;DESTINATION REGISTER NOT INCREMENTED PROPERLY
2008 012550 005737 000000      MDM2D: TST     @#0
2009 012554 001404          BEQ     TST146
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 752 <====
```

```
012556 012742 000306      MOV    #306,-(R2)      ;MOVE TO MAILBOX # ***** 306 *****
012562 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
012564 000000              HALT                  ;DESTINATION DATA INCORRECT
                                           ; OR SEQUENCE ERROR
```

2010
2011
2012
2013
2014
2015
2016

```
*****
: THIS TEST VERIFIES DESTINATION MODE 2 W/MOVB INSTS. TWO DIFFERENT MOVB
: INSTRUCTIONS ARE USED TO MOVE A TEST PATTERN FIRST TO BYTE 0 THEN TO BYTE 1.
*****
: TEST 146      TEST MOV-BYTE DESTINATION MODE 2
*****
```

```
012566 005212              TST146: INC    (R2)      ;UPDATE TEST NUMBER
012570 022712 000146      CMP    #146,(R2)      ;SEQUENCE ERROR?
012574 001046              BNE    TST147-10      ;BR TO ERROR HALT ON SEQ ERROR
2017 012576 005000              CLR    R0             ;R0=0
2018 012600 005010              CLR    (R0)           ;LOC. 0=0
2019 012602 112720 000125      MOVB  #125,(R0)+      ;TRY DESTINATION MODE 2 W/EVEN BYTE
2020 012606 102402              BVS    MBDM2A         ;BR TO ERROR IF V SET
2021 012610 001401              BEQ    MBDM2A         ;BR TO ERROR IF Z SET
2022 012612 100004              BPL    MBDM2B
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <= -
: REPLACE THE MOVE INSTRUCTION <----
: WHICH FOLLOWS W/ 770 <----
```

```
012614 012742 000307      MBDM2A: MOV    #307,-(R2)      ;MOVE TO MAILBOX # ***** 307 *****
012614 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
012622 000000              HALT                  ;CC'S INCORRECT
2023 012624 022700 000001      MBDM2B: CMP    #1,R0     ;
2024 012630 001404              BEQ    MBDM2C
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <= --
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <----
: WHICH FOLLOWS W/ 761 <----
```

```
012632 012742 000310      MOV    #310,-(R2)      ;MOVE TO MAILBOX # ***** 310 *****
012636 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
012640 000000              HALT                  ;REGISTER NOT INCREMENTED BY ONE
2025 012642 112720 000252      MBDM2C: MOVB  #252,(R0)+ ;TRY DESTINATION MODE 2 W/ODD BYTE
2026 012646 102402              BVS    MBDM2D
2027 012650 001401              BEQ    MBDM2D
2028 012652 100404              BMI    MBDM2E
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <----
: WHICH FOLLOWS W/ 750 <----
```

```
012654 012742 000311      MBDM2D: MOV    #311,-(R2)      ;MOVE TO MAILBOX # ***** 311 *****
012654 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
012662 000000              HALT                  ;CC'S NOT SET CORRECT
2029 012664 022700 000002      MBDM2E: CMP    #2,R0     ;
2030 012670 001404              BEQ    MBDM2F
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <----
: REPLACE THE MOVE INSTRUCTION <----
: WHICH FOLLOWS W/ 741 <- -
```

```
012672 012742 000312      MOV      #312,-(R2)      ;MOVE TO MAILBOX # ***** 312 *****
012676 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
012700 000000      HALT
2031 012702 022737 125125 000000 MBDM2F: CMP      #125125,@#0      ;REGISTER NOT INCREMENTED BY ONE
2032 012710 001404      BEQ      TST147        ;CHECK DATA
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <==--
;          CONDITIONAL BRANCH INST. AND <====-
;          REPLACE THE MOVE INSTRUCTION <====-
;          WHICH FOLLOWS W/ 731 <====-
```

2033
2034
2035
2036
2037
2038
2039

```
*****
: THIS TEST VERIFIES MOV DESTINATION MODE 3. R0 IS USED TO PICK UP
: AN ADDRESS AT LOC. 400. LOC 400 POINTS TO LOC. 0 THE EFFECTIVE DEST. ADDR.. ALSO, MOV B
: INST. ARE USED W/ EVEN AND ODD BYTES TO CHECK MOV BYTES INST AND MODE 37 DESTINATIONS.
:*****
:TEST 147      TEST MOV(B) DESTINATION MODE 3
:*****
```

```
012722 005212      TST147: INC      (R2)          ;UPDATE TEST NUMBER
012724 022712 000147      CMP      #147,(R2)        ;SEQUENCE ERROR?
012730 001057      BNE      TST150-10       ;BR TO ERROR HALT ON SEQ ERROR
2040 012732 012700 000400      MOV      #400,R0         ;R0=400
2041 012736 005010      CLR      (R0)            ;LOC. 400 POINTS TO LOC. 0
2042 012740 005037 000000      CLR      @#0             ;LOC. 0=0
2043 012744 012730 125252      MOV      #125252,@(R0)+  ;TRY MOV DESTINATION MODE 2
2044 012750 102402      BVS      MDM3A           ;BR TO ERROR IF V SET
2045 012752 001401      BEQ      MDM3A           ;BR TO ERROR IF Z SET
2046 012754 100404      BMI      MDM3B
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <== -
;          CONDITIONAL BRANCH INST. AND <====-
;          REPLACE THE MOVE INSTRUCTION <====-
;          WHICH FOLLOWS W/ 765 <====-
```

```
012756 012742 000314      MDM3A: MOV      #314,-(R2)      ;MOVE TO MAILBOX # ***** 314 *****
012756 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
012764 000000      HALT
2047 012766 022700 000402      MDM3B: CMP      #402,R0         ;CC'S INCORRECT
2048 012772 001404      BEQ      MDM3C           ;CHECK DEST. MODE REGISTER
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====-
;          CONDITIONAL BRANCH INST. AND <====-
;          REPLACE THE MOVE INSTRUCTION <====-
;          WHICH FOLLOWS W/ 756 <====-
```

```
012774 012742 000315      MOV      #315,-(R2)      ;MOVE TO MAILBOX # ***** 315 *****
013000 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
013002 000000      HALT
2049 013004 022737 125252 000000 MDM3C: CMP      #125252,@#0      ;REGISTER NOT INCREMENTED BY 2
2050 013012 001404      BEQ      MDM3D           ;CHECK DESTINATION DATA
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====-
;          CONDITIONAL BRANCH INST. AND <====-
;          REPLACE THE MOVE INSTRUCTION <====-
;          WHICH FOLLOWS W/ 746 <====-
```

```
013014 012742 000316      MOV      #316,-(R2)      ;MOVE TO MAILBOX # ***** 316 *****
013020 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
```

```
2051 013022 000000          HALT          ;DESTINATION DATA INCORRECT
      013024 112737 000125 000000 MDM3D: MOV#125,2#0 ;TRY MOV# DESTINATION MODE 2 EVEN BYTE
2052 013032 022737 125125 000000      CMP#125125,2#0 ;CHECK DATA
2053 013040 001404          BEQ      MDM3E
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
      :          CONDITIONAL BRANCH INST. AND <=====
      :          REPLACE THE MOVE INSTRUCTION <=====
      :          WHICH FOLLOWS W/ 733 <=====
      :
      : MOVE TO MAILBOX # ***** 317 *****
      : SET MSGTYP TO FATAL ERROR
      : DESTINATION DATA INCORRECT
2054 013042 012742 000317          MOV#317,-(R2) ;MOVE TO MAILBOX # ***** 317 *****
      013046 005242          INC      -(R2) ;SET MSGTYP TO FATAL ERROR
      013050 000000          HALT
2055 013052 112737 000525 000001 MDM3E: MOV#525,2#1 ;DESTINATION DATA INCORRECT
2056 013060 022737 052525 000000      CMP#52525,2#0 ;TRY MOV# DESTINATION MODE 2 ODD BYTE
      013066 001404          BEQ      TST150 ;CHECK DATA
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
      :          CONDITIONAL BRANCH INST. AND <=====
      :          REPLACE THE MOVE INSTRUCTION <=====
      :          WHICH FOLLOWS W/ 720 <=====
      :
      : MOVE TO MAILBOX # ***** 320 *****
      : SET MSGTYP TO FATAL ERROR
      :
2057 013070 012742 000320          MOV#320,-(R2) ;MOVE TO MAILBOX # ***** 320 *****
      013074 005242          INC      -(R2) ;SET MSGTYP TO FATAL ERROR
      013076 000000          HALT
```

```
2057
2058
2059
2060
2061
2062
2063
2064
2065
:
: *****
: THIS TEST VERIFIES THE MOV DESTINATION MODE 4 INSTRUCTION.
: SOP INSTRUCTIONS ON R0 ARE USED TO CLEAR TARGET LOCATION 0.
: R4 IS USED AS THE MODE 4 ADDRESSING REGISTER, AND
: CONDITIONAL BRANCHES ARE USED TO VERIFY THE DATA.
: *****
```

```
:TEST 150 TEST MOV DESTINATION MODE 4
: *****
TST150: INC (R2) ;UPDATE TEST NUMBER
      CMP#150,(R2) ;SEQUENCE ERROR?
      BNE TST151-10 ;BR TO ERROR HALT ON SEQ ERROR
2066 013100 005212 000150          CLR R0 ;R0=0
2067 013102 022712          CLR (R0) ;LOC 0=0
2068 013106 001026          MOV#2,R4 ;R4=2
2069 013110 005000          MOV#12345,-(R4) ;TRY MOV DEST. MODE 4
2070 013112 012704 000002          BVS MDM4A ;BR TO ERROR IF V-BIT SET
2071 013114 012744 012345          BEQ MDM4A ;BR TO ERROR IF Z-BIT SET
2072 013116 001401          BPL MDM4B
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
      :          CONDITIONAL BRANCH INST. AND <=====
      :          REPLACE THE MOVE INSTRUCTION <=====
      :          WHICH FOLLOWS W/ 766 <=====
      :
```

```
MDM4A: 013132 012742 000321          MOV#321,-(R2) ;MOVE TO MAILBOX # ***** 321 *****
      013132 005242          INC      -(R2) ;SET MSGTYP TO FATAL ERROR
      013136 000000          HALT
MDM4B: 2073 013142 005704          TST R4 ;CC'S NOT CORRECT
2074 013144 001404          BEQ MDM4C ;CHECK DECREMENTING OF MODE 4 REG.
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
      :          CONDITIONAL BRANCH INST. AND <=====
      :          REPLACE THE MOVE INSTRUCTION <=====
      :          WHICH FOLLOWS W/ 760 <=====
      :
      : MOVE TO MAILBOX # ***** 322 *****
      013146 012742 000322          MOV#322,-(R2)
```

```

013152 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
013154 000000          HALT                    ;DESTINATION MODE REGISTER NOT DECREMENTED BY 2
2075 013156 022710 012345 MDM4C:  CMP      #12345,(R0)      ;CHECK DESTINATION DATA
2076 013162 001404          BEQ      TST151
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 751 <====
013164 012742 000323          MOV      #323,-(R2)      ;MOVE TO MAILBOX # ***** 323 *****
013170 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
013172 000000          HALT                    ;DESTINATION DATA INCORRECT
; OR SEQUENCE ERROR

```

2077
2078
2079
2080
2081
2082
2083
2084
2085
2086

```

*****
THIS TEST VERIFIES THE MOV B DESTINATION MODE 4 INSTRUCTION
ON BOTH ODD AND EVEN BYTES. SOP INSTRUCTIONS ON R4 ARE
USED TO CLEAR TARGET LOCATION 0. R0 IS USED AS THE MODE 4
ADDRESSING REGISTER, AND CMP AND CONDITIONAL BRANCH
INSTRUCTIONS ARE USED TO VERIFY THE DATA.
*****

```

TEST 151 TEST MOV B DESTINATION MODE 4

```

013174 005212          INC      (R2)            ;UPDATE TEST NUMBER
013176 022712 000151          CMP      #151,(R2)      ;SEQUENCE ERROR?
013202 001046          BNE     TST152-10      ;BR TO ERROR HALT ON SEQ ERROR
2087 013204 005004          CLR     R4              ;R4=0
2088 013206 005014          CLR     (R4)            ;LOC. 0=0
2089 013210 012700 000002          MOV     #2,R0           ;R0 = 2
2090 013214 112740 125125          MOV B   #125125,-(R0)   ;TRY MOV B DEST. MODE 4-ODD BYTE
2091 013220 020027 000001          CMP     R0,#1           ;CHECK THAT DEST. REG. WAS DECREMENTED
2092 013224 001404          BEQ     MBDM4A
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 766 <====
013226 012742 000324          MOV     #324,-(R2)      ;MOVE TO MAILBOX # ***** 324 *****
013232 005242          INC     -(R2)          ;SET MSGTYP TO FATAL ERROR
013234 000000          HALT                    ;DESTINATION REG. NOT DECREMENTED BY 1
2093 013236 021427 052400 MBDM4A:  CMP     (R4),#52400   ;CHECK DEST. DATA
2094 013242 001404          BEQ     MBDM4B
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 757 <====

```

```

013244 012742 000325          MOV     #325,-(R2)      ;MOVE TO MAILBOX # ***** 325 *****
013250 005242          INC     -(R2)          ;SET MSGTYP TO FATAL ERROR
013252 000000          HALT                    ;DEST. DATA NOT CORRECT
2095 013254 112740 125125 MBDM4B:  MOV B   #125125,-(R0)   ;TRY MOV B DEST. MODE 4--EVEN BYTE
2096 013260 102402          BVS    MBDM4C           ;BR. TO ERROR IF V-BIT SET
2097 013262 001401          BEQ    MBDM4C           ;BR TO ERROR IF Z-BIT SET
2098 013264 100004          BPL    MBDM4D
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 746 <====

```

```
013266 MBDM4C: MOV #326,-(R2) ;MOVE TO MAILBOX # ***** 326 *****
013266 012742 000326 INC -(R2) ;SET MSGTYP TO FATAL ERROR
013272 005242 HALT ;COND. CODES INCORRECT
2099 013276 005700 MBDM4D: TST R0 ;CHECK MODE 4 DEST. REGISTER
2100 013300 001404 BEQ MBDM4E
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 740 <====
013302 MBDM4E: MOV #327,-(R2) ;MOVE TO MAILBOX # ***** 327 *****
013306 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
013310 000000 HALT ;DESTINATION REG NOT DECREMENTED BY 1
2101 013312 021427 052525 MBDM4E: CMP (R4),#52525 ;CHECK DEST. DATA
2102 013316 001404 BEQ TST152
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 731 <====
013320 MOV #330,-(R2) ;MOVE TO MAILBOX # ***** 330 *****
013324 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
013326 000000 HALT ;DESTINATION DATA INCORRECT
; OR SEQUENCE ERROR
```

2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113

```
*****
: THIS TEST VERIFIES THE MOV DESTINATION MODE 5 AND THE MOV8
: DESTINATION MODE 5 - EVEN BYTE INSTRUCTIONS. R4 IS A
: POINTER TO TARGET LOCATION 0 AND R0 IS SETUP TO
: POINT TO LOCATION 376 FOR THE MOV, AND LOCATION 404 FOR
: THE MOV8 INSTRUCTIONS. CMP INSTRUCTIONS ARE USED TO VERIFY
: PROPER ADDRESSING AND DATA.
*****
```

TEST 152 TEST MOV DESTINATION MODE 5

```
013330 005212 TST152: INC (R2) ;UPDATE TEST NUMBER
013332 022712 000152 CMP #152,(R2) ;SEQUENCE ERROR?
013336 001051 BNE TST153-10 ;BR TO ERROR HALT ON SEQ ERROR
2114 013340 005004 CLR R4 ;R4=0
2115 013342 005014 CLR (R4) ;LOC. 0 = 0
2116 013344 012700 000400 MOV #400,R0 ;R0=400
2117 013350 012750 004321 MOV #4321,@-(R0) ;TRY MOV DEST. MODE 5
2118 013354 102402 BVS MDM5A ;BR TO ERROR IF V-BIT SET
2119 013356 001401 BEQ MDM5A ;BR TO ERROR IF Z-BIT SET
2120 013360 100004 BPL MDM5B
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 766 <====
```

```
013362 MDM5A: MOV #331,-(R2) ;MOVE TO MAILBOX # ***** 331 *****
013362 012742 000331 INC -(R2) ;SET MSGTYP TO FATAL ERROR
013366 005242 HALT ;COND. CODES INCORRECT
2121 013372 022700 000376 MDM5B: CMP #376,R0 ;CHECK MODE 5 REG. WAS DECREMENTED
2122 013376 001404 BEQ MDM5C
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
```



```

:          CONDITIONAL BRANCH INST. AND <= --
:          REPLACE THE MOVE INSTRUCTION <- --
:          WHICH FOLLOWS W/ 757 <===
:
:          :MOVE TO MAILBOX # ***** 332 *****
:          :SET MSGTYP TO FATAL ERROR
:          :MODE 5 REGISTER NOT DECREMENTED BY 2
:          :CHECK DEST. DATA
:
:          : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          :          CONDITIONAL BRANCH INST. AND <====
:          :          REPLACE THE MOVE INSTRUCTION <====
:          :          WHICH FOLLOWS W/ 750 <====
:
:          :MOVE TO MAILBOX # ***** 333 *****
:          :SET MSGTYP TO FATAL ERROR
:          :DEST. DATA INCORRECT
:          :RO=406
:          :TRY MOV DEST. MODE 5 --EVEN BYTE
:          :CHECK MODE 5 REG.
:
:          : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          :          CONDITIONAL BRANCH INST. AND <====
:          :          REPLACE THE MOVE INSTRUCTION <====
:          :          WHICH FOLLOWS W/ 735 <====
:
:          :MOVE TO MAILBOX # ***** 334 *****
:          :SET MSGTYP TO FATAL ERROR
:          :MODE 5 REGISTER NOT DECREMENTED BY 2
:          :CHECK DEST. DATA
:
:          : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          :          CONDITIONAL BRANCH INST. AND <====
:          :          REPLACE THE MOVE INSTRUCTION <====
:          :          WHICH FOLLOWS W/ 726 <====
:
:          :MOVE TO MAILBOX # ***** 335 *****
:          :SET MSGTYP TO FATAL ERROR
:          :DEST. DATA INCORRECT
:          : OR SEQUENCE ERROR
  
```

013400	012742	000332	MOV	#332,-(R2)	
013404	005242		INC	-(R2)	
013406	000000		HALT		
2123 013410	022714	004321	MDM5C:	CMP	#4321,(R4)
2124 013414	001404			BEQ	MDM5D
013416	012742	000333	MOV	#333,-(R2)	
013422	005242		INC	-(R2)	
013424	000000		HALT		
2125 013426	012700	000406	MDM5D:	MOV	#406,R0
2126 013432	112750	000377		MOVB	#377,@-(R0)
2127 013436	022700	000404		CMP	#404,R0
2128 013442	001404			BEQ	MDM5E
013444	012742	000334	MOV	#334,-(R2)	
013450	005242		INC	-(R2)	
013452	000000		HALT		
2129 013454	022714	177721	MDM5E:	CMP	#177721,(R4)
2130 013460	001404			B-Q	TST153
013462	012742	000335	MOV	#335,-(R2)	
013466	005242		INC	-(R2)	
013470	000000		HALT		

2131
 2132
 2133
 2134
 2135
 2136
 2137
 2138
 2139
 2140

```

:*****
:          THIS TEST VERIFIES THE MOV DESTINATION MODE 6 AND MOVB - EVEN BYTE
:          DESTINATION MODE 6 INSTRUCTIONS. R0 IS USED TO SETUP TARGET LOC.0
:          FOR BOTH TESTS. PATTERNS OF ONES AND ZEROES ARE MOVED INTO LOC.0
:          BY MODE 6 INSTRUCTIONS, AND CMP INSTRUCTIONS ARE USED TO VERIFY
:          PROPER ADDRESSING AND DATA.
:*****
:TEST 153      TEST MOV DESTINATION MODE 6
:*****
  
```

013472	005212		TST153:	INC	(R2)	:UPDATE TEST NUMBER
013474	022712	000153		CMP	#153,(R2)	:SEQUENCE ERROR?
013500	001054			BNF	TST154-10	:BR TO ERROR HALT ON SEQ ERROR
2141 013502	005000			CLR	R0	:R0=0
2142 013504	005010			CLR	(R0)	:LOC. 0=0
2143 013506	005200			INC	R0	:R0=1
2144 013510	012760	052525 177777		MOV	#052525,-1(R0)	:TRY MOV DEST. MODE 6
2145 013516	102402			BVS	MDM6A	:BR TO ERROR IF V-BIT SET
2146 013520	001401			BEQ	MDM6A	:BR TO ERROR IF Z-BIT SET

2147 013522 100004

BPL MDM6B

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
: CONDITIONAL BRANCH INST. AND <===
: REPLACE THE MOVE INSTRUCTION <===
: WHICH FOLLOWS W/ 766 <===

013524
013524 012742 000336
013530 005242
013532 000000
2148 013534 022700 000001

MDM6A.
MOV #336,-(R2)
INC -(R2)
HALT
MDM6B: CMP #1,R0

:MOVE TO MAILBOX # ***** 336 *****
:SET MSGTYP TO FATAL ERROR
:COND. CODES INCORRECT
:CHECK DEST. REGISTER UNALTERED

```

2150 013540 001404          BEQ      MDM6C
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
;          CONDITIONAL BRANCH INST. AND <===
;          REPLACE THE MOVE INSTRUCTION <===
;          WHICH FOLLOWS W/ 757 <===
;          MOVE TO MAILBOX # ***** 337 *****
;          SET MSGTYP TO FATAL ERROR
;          DEST. REGISTER INCORRECTLY ALTERED
;          CHECK DEST. DATA

      013542 012742 000337      MOV      #337,-(R2)
      013546 005242          INC      -(R2)
      013550 000000          HALT
2151 013552 022737 052525 000000 MDM6C:  CMP      #52525,@#0
2152 013560 001404          BEQ      MDM6D
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
;          CONDITIONAL BRANCH INST. AND <===
;          REPLACE THE MOVE INSTRUCTION <===
;          WHICH FOLLOWS W/ 747 <===
;          MOVE TO MAILBOX # ***** 340 *****
;          SET MSGTYP TO FATAL ERROR
;          DEST. DATA INCORRECT
;          R0=2
;          TRY MOV B DEST. MODE 6
;          CHECK DEST. REGISTER UNALTERED

      013562 012742 000340      MOV      #340,-(R2)
      013566 005242          INC      -(R2)
      013570 000000          HALT
2153 013572 012700 000002          MOV      #2,R0
2154 013576 112760 000377 177777 MDM6D:  MOV B   #377,-1(R0)
2155 013604 022700 000002          CMP      #2,R0
2156 013610 001404          BEQ      MDM6E
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
;          CONDITIONAL BRANCH INST. AND <===
;          REPLACE THE MOVE INSTRUCTION <===
;          WHICH FOLLOWS W/ 733 <===
;          MOVE TO MAILBOX # ***** 341 *****
;          SET MSGTYP TO FATAL ERROR
;          DEST. REGISTER INCORRECTLY ALTERED
;          CHECK DEST. DATA

      013612 012742 000341      MOV      #341,-(R2)
      013616 005242          INC      -(R2)
      013620 000000          HALT
2157 013622 022737 177525 000000 MDM6E:  CMP      #177525,@#0
2158 013630 001404          BEQ      TST154
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
;          CONDITIONAL BRANCH INST. AND <===
;          REPLACE THE MOVE INSTRUCTION <===
;          WHICH FOLLOWS W/ 723 <===
;          MOVE TO MAILBOX # ***** 342 *****
;          SET MSGTYP TO FATAL ERROR
;          DEST. DATA INCORRECT
;          OR SEQUENCE ERROR

      013632 012742 000342      MOV      #342,-(R2)
      013636 005242          INC      -(R2)
      013640 000000          HALT
  
```

2159
 2160
 2161
 2162
 2163
 2164
 2165
 2166
 2167

```

*****
; THIS TEST VERIFIES THE MOV DESTINATION MODE 7 AND MOV B - ODD BYTE
; DESTINATION MODE 7 INSTRUCTIONS. R4 POINTS TO TARGET LOC.0 AND R0
; IS USED AS THE MODE 7 ADDRESSING REGISTER. CMP INSTRUCTIONS ARE
; USED TO VERIFY PROPER ADDRESSING AND DATA.
*****
; TEST 154      TEST MOV DESTINATION MODE 7
*****
  
```

```

      013642 005212          TST154: INC      (R2)
      013644 022712 000154      CMP      #154,(R2)
      013650 001053          BNE     TST155-1C
2168 013652 005004          CLR      R4
2169 013654 005014          CLR      (R4)
2170 013656 012700 000403      MOV      #403,R0
2171 013662 012770 070707 177777      MOV      #70707,@-1(R0)
2172 013670 102402          BVS     MDM7A
; UPDATE TEST NUMBER
; SEQUENCE ERROR?
; BR TO ERROR HALT ON SEQ ERROR
; R4=0
; LOC.0=0
; R0=403
; TRY MOV W/DEST MODE 7
; BR. TO ERROR IF V-BIT SET
  
```

```

2173 013672 001401      BEQ    MDM7A      ;BR TO ERROR IF Z-BIT SET
2174 013674 100004      BPL    MDM7B
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <== ==
;          CONDITIONAL BRANCH INST. AND <-- ==
;          REPLACE THE MOVE INSTRUCTION <=== ==
;          WHICH FOLLOWS W/ 765 <--- ==

013676      MDM7A:
013676 012742 000343      MOV    #343,-(R2) ;MOVE TO MAILBOX # ***** 343 *****
013702 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
013704 000000      HALT                    ;COND. CODES INCORRECT
2175 013706 022700 000403      MDM7B: CMP    #403,R0    ;CHECK DEST. REGISTER
2176 013712 001404      BEQ    MDM7C
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=== ==
;          CONDITIONAL BRANCH INST. AND <=== ==
;          REPLACE THE MOVE INSTRUCTION <=== ==
;          WHICH FOLLOWS W/ 756 <=== ==

013714 012742 000344      MOV    #344,-(R2) ;MOVE TO MAILBOX # ***** 344 *****
013720 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
013722 000000      HALT                    ;DEST. REGISTER INCORRECTLY ALTERED
2177 013724 022737 070707 000000 MDM7C: CMP    #70707,@#0 ;CHECK DEST. DATA
2178 013732 001404      BEQ    MDM7D
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=== ==
;          CONDITIONAL BRANCH INST. AND <=== ==
;          REPLACE THE MOVE INSTRUCTION <=== ==
;          WHICH FOLLOWS W/ 746 <=== ==

013734 012742 000345      MOV    #345,-(R2) ;MOVE TO MAILBOX # ***** 345 *****
013740 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
013742 000000      HALT                    ;DEST. DATA INCORRECT
2179 013744 112770 107070 000001 MDM7D: MOVB   #107070,@1(R0) ;TRY MOVB W/DEST MODE 7--ODD BYTE
2180 013752 022700 000403      CMP    #403,R0    ;CHECK MODE 7 DEST. REG.
2181 013756 001404      BEQ    MDM7E
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=== ==
;          CONDITIONAL BRANCH INST. AND <=== ==
;          REPLACE THE MOVE INSTRUCTION <=== ==
;          WHICH FOLLOWS W/ 734 <=== ==

013760 012742 000346      MOV    #346,-(R2) ;MOVE TO MAILBOX # ***** 346 *****
013764 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
013766 000000      HALT                    ;DEST. DATA INCORRECT
2182 013770 022737 034307 000000 MDM7E: CMP    #34307,@#0 ;CHECK DEST. DATA
2183 013776 001404      BEQ    TST155
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=== ==
;          CONDITIONAL BRANCH INST. AND <=== ==
;          REPLACE THE MOVE INSTRUCTION <=== ==
;          WHICH FOLLOWS W/ 724 <=== ==

014000 012742 000347      MOV    #347,-(R2) ;MOVE TO MAILBOX # ***** 347 *****
014004 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
014006 000000      HALT                    ;DESTINATION DATA INCORRECT
; OR SEQUENCE ERROR
  
```

2184
 2185
 2186
 2187
 2188
 2189
 2190
 2191
 2192

```

:*****
:
: THIS TEST VERIFIES MODE 4 DOUBLE OPERAND INSTRUCTIONS.
: THE TEST USES MODE 4 ADDRESSING WITH REGISTER 0 TO MOVE THRU A
: TABLE OF OPERANDS. THE TABLE OF OPERANDS AND THE WORK LOCATION IS
: STORED FOLLOWING THE TEST CODE. A SERIES OF 5 DOP INSTRUCTIONS UTILIZES
: THE DATA IN THE TABLE TO CYCLE THE WORK LOCATION THRU A SET OF
: VALUE. THE DATA HAS BEEN CHOSEN TO INSURE THAT NO SINGLE ERROR WILL
  
```

2193
2194
2195
2196
2197

:GO UNDETECTED. WORD AND BYTE INSTRUCTION ACCESSING BOTH EVEN AND
:ODD ADDRESSES ARE USED IN THE TEST. THE LISTING SHOWS THE
:EXPECTED INTERMEDIATE RESULT AS EACH INSTRUCTION IS EXECUTED.

:TEST 155 TEST MODE 4 W/ DOP INSTS.

014010 005212
014012 022712 000155
014016 001015
2198 014020 012700 014072
2199 014024 014037 014072
2200 014030 064037 014072
2201 014034 144037 014072
2202 014040 154037 014073
2203 014044 024037 014072
2204 014050 001411

TST155: INC (R2) ;UPDATE TEST NUMBER
CMP #155,(R2) ;SEQUENCE ERROR?
BNE DOP4 ;BR TO ERROR HALT ON SEQ ERROR
MOV #TBL1,R0 ;INITIALIZE R0
MOV -(R0),@TBL1 ;TBL1=125252
ADD -(R0),@TBL1 ;TBL1=000377
BICB -(R0),@TBL1 ;TBL1=000252
BISB -(R0),@TBL1+1 ;TBL1=125252
CMP -(R0),@TBL1 ;CHECK RESULT
BEQ TST156

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 762 <====

014052
014052 012742 000350
014056 005242
014060 000000

DOP4: MOV #350,-(R2) ;MOVE TO MAILBOX # ***** 350 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;RESULT OF MODE 4 INSTS. INCORRECT
; OR SEQUENCE ERROR

2205
2206 014062 125252
2207 014064 052652
2208 014066 053125
2209 014070 125252
2210 014072 000000

125252
52652
53125
125252
TBL1: 0

2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221

: THIS TEST VERIFIES MODE 5 DOUBLE OPERAND INSTRUCTIONS.
: THE TEST USES AN ADDRESS TABLE STORED FOLLOWING THE TEST CODE.
: THIS TABLE IS SIMPLY A TABLE OF ADDRESS POINTERS WHICH ADDRESS
: THE DATA TABLE USED IN THE PREVIOUS TEST. THE TEST IS IDENTICAL TO
: THE PREVIOUS TEST EXCEPT THE DATA IS REFERENCED USING THIS ADDRESS
: TABLE AND MODE 5 ADDRESSING. (SEE PREVIOUS TEST).

:TEST 156 TEST MODE 5 W/ DOP INSTS.

014074 005212
014076 022712 000156
014102 001015
2222 014104 012700 014160
2223 014110 015037 014072
2224 014114 065037 014072
2225 014120 145037 014072
2226 014124 155037 014073
2227 014130 025037 014072
2228 014134 001411

TST156: INC (R2) ;UPDATE TEST NUMBER
CMP #156,(R2) ;SEQUENCE ERROR?
BNE DOP5 ;BR TO ERROR HALT ON SEQ ERROR
MOV #TBL2+2,R0 ;INITIALIZE R0
MOV @-(R0),@TBL1 ;TBL1=125252
ADD @-(R0),@TBL1 ;TBL1=000377
BICB @-(R0),@TBL1 ;TBL1=000252
BISB @-(R0),@TBL1+1 ;TBL1=125252
CMP @-(R0),@TBL1 ;CHECK RESULT
BEQ TST157

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
; CONDITIONAL BRANCH INST. AND <

```

                                :          REPLACE THE MOVE INSTRUCTION  <--=
                                :          WHICH FOLLOWS W/ 762          < --
014136                                DOP5:                                :
014136 012742 000351                MOV    #351,-(R2)                ;MOVE TO MAILBOX #, ***** 351 *****
014142 005242                        INC    -(R2)                    ;SET MSGTYP TO FATAL ERROR
014144 000000                        HALT                          ;RESULT OF MODE 5 INSTS. INCORRECT
                                           ; OR SEQUENCE ERROR

```

```

2229 014146 014062                TBL1-10
2230 014150 014064                TBL1-6
2231 014152 014065                TBL1-5
2232 014154 014066                TBL1-4
2233 014156 014070                TBL2: TBL1-2

```

```

2234
2235
2236                                :          THIS TEST VERIFIES MODE 6 DOUBLE OPERAND INSTRUCTIONS.
2237                                :          IT USES THE SAME DATA AS THAT USED IN THE MODE 4 TESTS.
2238                                :          THIS TIME THE DATA IS ACCESSED USING MODE 6.  R0 IS SET
2239                                :          TO POINT TO THE MIDDLE OF THE TABLE.  THE TABLE IS ACCESSED FROM
2240                                :          BOTTOM TO TOP BY VARYING THE OFFSET IN THE MODE 6 INSTRUCTIONS.
2241                                :          THE DATA RESULTS ARE IDENTICAL TO THOSE EXPECTED IN THE MODE 4
2242                                :          TESTS.
2243
2244

```

:TEST 157 TEST MODE 6 W/ DOP INSTS.

```

014160 005212                                TST157: INC    (R2)                ;UPDATE TEST NUMBER
014162 022712 000157                CMP    #157,(R2)                ;SEQUENCE ERROR?
014166 001022                        BNE    TST160-10                ;BR TO ERROR HALT ON SEQ ERROR
2245 014170 012700 014066                MOV    #TBL1-4,R0                ;INITIALIZE R0
2246 014174 016037 000002 014072                MOV    2(R0),@TBL1                ;TBL1=125252
2247 014202 066037 000000 014072                ADD    0(R0),@TBL1                ;TBL1=000377
2248 014210 146037 177777 014072                BICB  -1(R0),@TBL1                ;TBL1=000252
2249 014216 156037 177776 014073                BISB  -2(R0),@TBL1+1            ;TBL1=125252
2250 014224 026037 177774 014072                CMP    -4(R0),@TBL1            ;CHECK RESULT
2251 014232 001404                        BEQ    TST160

```

```

                                :          TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <
                                :          CONDITIONAL BRANCH INST. AND  <
                                :          REPLACE THE MOVE INSTRUCTION  <
                                :          WHICH FOLLOWS W/ 755          <-

```

```

014234 012742 000352                MOV    #352,-(R2)                ;MOVE TO MAILBOX # ***** 352 *****
014240 005242                        INC    -(R2)                    ;SET MSGTYP TO FATAL ERROR
014242 000000                        HALT                          ;RESULT OF MODE 6 INSTS. INCORRECT
                                           ; OR SEQUENCE ERROR

```

```

2252                                :          THIS TEST VERIFIES MODE 7 DOUBLE OPERAND INSTRUCTIONS.
2253                                :          THIS TEST USES THE SAME ADDRESS TABLE AND DATA TABLE USED BY
2254                                :          THE MODE 5 TESTS.  THIS TIME THE DATA IS ACCESSED USING MODE 7.
2255                                :          R0 IS SET TO POINT TO THE MIDDLE OF THE ADDRESS TABLE IN THE MODE 5
2256                                :          TEST.  THE TABLE IS ACCESSED FROM BOTTOM TO TOP BY VARYING THE OFFSET
2257                                :          IN THE MODE 7 INSTRUCTIONS.  THE DATA RESULTS ARE IDENTICAL TO
2258                                :          THOSE EXPECTED IN THE MODE 5 TESTS.
2259
2260
2261
2262

```

:TEST 160 TEST MODE 7 W/ DOP INSTS.

```

014244 005212                                TST160: INC    (R2)                ;UPDATE TEST NUMBER

```

```
014246 022712 000160      CMP      #160,(R2)      ;SEQUENCE ERROR?
014252 001022      BNE      TST161-10     ;BR TO ERROR HALT ON SEQ ERROR
2263 014254 012700 014152      MOV      #TBL2-4,R0    ;INITIALIZE R0
2264 014260 017037 000004 014072      MOV      @4(R0),@#TBL1 ;TBL1=125252
2265 014266 067037 000002 014072      ADD      @2(R0),@#TBL1 ;TBL1=000377
2266 014274 147037 000000 014072      BICB    @0(R0),@#TBL1 ;TBL1=000252
2267 014302 157037 177776 014073      BISB    @-2(R0),@#TBL1+1 ;TBL1=125252
2268 014310 027037 177774 014072      CMP      @-4(R0),@#TBL1 ;CHECK RESULT
2269 014316 001404      BEQ      TST161
```

```
      ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---=
      ; CONDITIONAL BRANCH INST. AND <-- -
      ; REPLACE THE MOVE INSTRUCTION <-
      ; WHICH FOLLOWS W/ 755 <----
      ; MOVE TO MAILBOX # ***** 353 *****
      ; SET MSGTYP TO FATAL ERROR
      ; RESULT OF MODE 7 INSTS INCORRECT
      ; OR SEQUENCE ERROR
```

```
014320 012742 000353
014324 005242
014326 000000
```

```
MOV      #353,-(R2)
INC      -(R2)
HALT
```

```
2270
2271
2272
2273
2274
2275
2276
2277
2278
```

```
*****
: THIS TEST VERIFIES THE ROTATE MODE 0 INSTRUCTIONS.
: R0 IS LOADED WITH A DATA PATTERN, THE C-BIT IS LOADED, AND
: AN ROL INSTRUCTION IS EXECUTED WITH MODE 0. THE OPERATION IS CHECKED
: BY TESTING THE RESULTING DATA AND THE STATE OF THE C AND V BITS.
: NEXT, THE SAME PROCEDURE IS EXECUTED TO TEST MODE 0 BYTE INSTRUCTIONS.
*****
```

```
:TEST 161 TEST ROTATE INSTRUCTIONS OF MODE 0
```

```
014330 005212
014332 022712 000161
014336 001026
2279 014340 012700 125252
2280 014344 000261
2281 014346 006100
2282 014350 102004
2283 014352 103003
2284 014354 022700 052525
2285 014360 001404
```

```
TST161: INC      (R2)      ;UPDATE TEST NUMBER
      CMP      #161,(R2)  ;SEQUENCE ERROR?
      BNE      TST162-10  ;BR TO ERROR HALT ON SEQ ERROR
      MOV      #125252,R0 ;INITIALIZE DATA
      SEC      ;SET C-BIT
      ROL      R0         ;TRY ROL W/ MODE 0
      BVC     ROT0A      ;CC=0011
      BCC     ROT0A
      CMP      #052525,R0 ;CHECK DATA
      BEQ     ROT0B
      ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-
      ; CONDITIONAL BRANCH INST. AND <
      ; REPLACE THE MOVE INSTRUCTION <
      ; WHICH FOLLOWS W/ 766 <
```

```
014362
014362 012742 000354
014366 005242
014370 000000
2286 014372 012700 125252
2287 014376 000261
2288 014400 106100
2289 014402 102004
2290 014404 103003
2291 014406 022700 125125
2292 014412 001404
```

```
ROT0A: MOV      #354,-(R2)  ;MOVE TO MAILBOX # ***** 354 *****
      INC      -(R2)
      HALT
      ; SET MSGTYP TO FATAL ERROR
      ; ROL MODE 0 FAILED
ROT0B: MOV      #125252,R0 ;INITIALIZE DATA
      SEC      ;SET C-BIT
      ROLB    R0         ;TRY ROL W/ MODE 0 EVEN BYTE
      BVC     ROT0C      ;CC=0011
      BCC     ROT0C
      CMP      #125125,R0 ;CHECK DATA
      BEQ     TST162
```

```
      ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
      ; CONDITIONAL BRANCH INST. AND <-
      ; REPLACE THE MOVE INSTRUCTION <--
      ; WHICH FOLLOWS W/ 751 <
```



```
2324 014542 102005      BVC     ROT1E      :CC=0011
2325 014544 103004      BCC     ROT1E
2326 014546 022737 052652 000000  CMP     #052652,@#0  :CHECK DATA
2327 014554 001404      BEQ     TST163
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 726 <====

014556      ROT1E:
014556 012742 000360      MOV     #360,-(R2)  :MOVE TO MAILBOX # ***** 360 *****
014562 005242      INC     -(R2)      :SET MSGTYP TO FATAL ERROR
014564 000000      HALT                    :ROLB W/ MODE 1 ODD BYTE FAILED
; OR SEQUENCE ERROR
```

2328
2329
2330
2331
2332
2333
2334
2335
2336

```
*****
: THIS TEST VERIFIES MODE 2 ROTATE INSTRUCTIONS.
: THE SAME PROCEDURE AS IN THE OTHER ROTATE TESTS ARE USED. R0
: IS USED AS THE ADDRESSING REGISTER AND IS CHECKED FOR PROPER
: INCREMENTING. BYTE INSTRUCTIONS ARE ALSO CHECKED.
*****
```

```
: TEST 163 TEST ROTATE INSTRUCTIONS W/ MODE 2
*****
```

```
014566 005212      TST163: INC     (R2)      :UPDATE TEST NUMBER
014570 022712 000163  CMP     #163,(R2)  :SEQUENCE ERROR?
014574 001057      BNE     TST164-10  :BR TO ERROR HALT ON SEQ ERROR
2337 014576 005000      CLR     R0         :POINT TO LOC 0
2338 014600 012710 173737  MOV     #173737,(R0) :INITIALIZE DATA
2339 014604 000241      CLC                    :CLEAR C-BIT
2340 014606 006120      ROL     (R0)+       :TRY ROL W/ MODE 2
2341 014610 103007      BCC     ROT2A      :CHECK C-BIT
2342 014612 022737 167676 000000  CMP     #167676,@#0 :CHECK DATA
2343 014620 001003      BNE     ROT2A      :BRANCH IF RESULT INCORRECT
2344 014622 005300      DEC     R0         :TEST R0
2345 014624 005300      DEC     R0
2346 014626 001404      BEQ     ROT2B
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 762 <====
```

```
014630      ROT2A:
014630 012742 000361      MOV     #361,-(R2)  :MOVE TO MAILBOX # ***** 361 *****
014634 005242      INC     -(R2)      :SET MSGTYP TO FATAL ERROR
014636 000000      HALT                    :ROL W/ MODE 2 FAILED
2347 014640 005000      ROT2B: CLR     R0         :POINT TO LOC 0
2348 014642 012710 004040  MOV     #4040,(R0)  :INITIALIZE DATA
2349 014646 000241      CLC                    :CLEAR C-BIT
2350 014650 106120      ROLB    (R0)+       :TRY ROLB W/ MODE 2 EVEN BYTE
2351 014652 103406      BCS     ROT2C      :CHECK C-BIT
2352 014654 022737 004100 000000  CMP     #4100,@#0  :CHECK DATA
2353 014662 001002      BNE     ROT2C      :BRANCH IF DATA INCORRECT
2354 014664 005300      DEC     R0         :CHECK R0
2355 014666 001404      BEQ     ROT2D
```

```
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <==--
```

```
014670 ROT2C: WHICH FOLLOWS W/ 742 <====
014670 012742 000362 MOV #362,-(R2) ;MOVE TO MAILBOX # ***** 362 *****
014674 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
014676 000000 HALT ;ROLB W/ MODE 2 EVEN BYTE FAILED
2356 014700 005000 ROT2D: CLR R0 ;POINT TO LOC 0
2357 014702 012710 004040 MOV #4040,(R0) ;INITIALIZE DATA
2358 014706 005200 INC R0 ;POINT TO ODD BYTE OF DATA
2359 014710 000261 SEC ;SET C-BIT
2360 014712 106120 ROLB (R0)+ ;TRY ROL W/ MODE 2 ODD BYTE
2361 014714 103407 BCS ROT2E ;CHECK C-BIT
2362 014716 022737 010440 000000 CMP #10440,@#0 ;CHECK DATA
2363 014724 001003 BNE ROT2E ;BRANCH IF DATA INCORRECT
2364 014726 005300 DEC R0 ;CHECK R0
2365 014730 005300 DEC R0
2366 014732 001404 BEQ TST164
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 720 <====
```

```
014734 ROT2E:
014734 012742 000363 MOV #363,-(R2) ;MOVE TO MAILBOX # ***** 363 *****
014740 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
014742 000000 HALT ;ROLB W/ MODE 2 ODD BYTE FAILED
; OR SEQUENCE ERROR
```

2367
2368
2369
2370
2371
2372
2373
2374
2375

```
*****
; THIS TEST VERIFIES MODE 3 ROTATE INSTRUCTIONS.
; THIS TEST USES THE SAME PROCEDURES AS IN THE OTHER ROTATE
; TESTS. THE DATA IS STORED IN LOC. 0 AND IS ADDRESSED USING
; MODE 37. BYTE ADDRESSING IS ALSO CHECKED FOR EVEN AND ODD BYTES.
*****
```

```
; TEST 164 TEST ROTATE INSTRUCTIONS /W MODE 3
```

```
014744 005212 TST164: INC (R2) ;UPDATE TEST NUMBER
014746 022712 000164 CMP #164,(R2) ;SEQUENCE ERROR?
014752 001051 BNE TST165-10 ;BR TO ERROR HALT ON SEQ ERROR
2376 014754 012737 052525 000000 MOV #52525,@#0 ;INITIALIZE DATA IN LOC 0
2377 014762 000261 SEC ;SET C-BIT
2378 014764 006137 000000 ROL @#0 ;TRO ROL W/ MODE 3
2379 014770 103404 BCS ROT3A ;CHECK C-BIT
2380 014772 022737 125253 000000 CMP #125253,@#0 ;CHECK DATA
2381 015000 001404 BEQ ROT3B
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 764 <====
```

```
015002 ROT3A:
015002 012742 000364 MOV #364,-(R2) ;MOVE TO MAILBOX # ***** 364 *****
015006 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
015010 000000 HALT ;ROL W/ MODE 3 FAILED
2382 015012 012737 125252 000000 ROT3B: MOV #125252,@#0 ;INITIALIZE DATA
2383 015020 000241 CLC ;CLEAR C-BIT
2384 015022 106137 000000 ROLB @#0 ;TRY ROL W/ MODE 3 EVEN BYTE
2385 015026 103004 BCC ROT3C ;CHECK C-BIT
```

```

2386 015030 023727 000000 125124 4S:  CMP @#0,#125124 ;CHECK DATA
2387 015036 001404  BEQ ROT3D ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 745 <====

015040 ROT3C:
015040 012742 000365  MOV #365,-(R2) ;MOVE TO MAILBOX # ***** 365 *****
015044 005242  INC -(R2) ;SET MSGTYP TO FATAL ERROR
015046 000000  HALT ;ROL W/ MODE 3 EVEN BYTE FAILED
2388 015050 012737 125252 000000 ROT3D: MOV #125252,@#0 ;INITIALIZE DATA IN LOC. 0
2389 015056 000261  SEC ;SET C-BIT
2390 015060 106137 000001  ROLB @#1 ;TRY ROL W/ MODE 3 ODD BYTE
2391 015064 103004  BCC ROT3E ;CHECK C-BIT
2392 015066 022737 052652 000000  CMP #052652,@#0 ;CHECK DATA
2393 015074 001404  BEQ TST165 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 726 <====

015076 ROT3E:
015076 012742 000366  MOV #366,-(R2) ;MOVE TO MAILBOX # ***** 366 *****
015102 005242  INC -(R2) ;SET MSGTYP TO FATAL ERROR
015104 000000  HALT ;ROL W/ MODE 3 ODD BYTE FAILED
; OR SEQUENCE ERROR

```

2394
2395
2396
2397
2398
2399
2400
2401
2402
2403

```

:*****
: THIS TEST VERIFIES MODE 4 ROTATE INSTRUCTIONS. THE DATA IS
: STORED IN LOC. 0. R0 IS SET TO 2 AND THE CARRY IS SET. AN ROL MODE 4
: IS USED TO ROTATE LOCATION 0 USING R0. THE DATA IS CHECKED
: AND THE C AND V BITS ARE TESTED. THE PROPER DECREMENTING OF
: R0 IS VERIFIED.
:*****
: TEST 165 TEST MODE 4 W/ ROTATE INSTRUCTIONS
:*****

```

```

015106 005212 000165  TST165: INC (R2) ;UPDATE TEST NUMBER
015110 022712 000165  CMP #165,(R2) ;SEQUENCE ERROR?
015114 001016  BNE TST166-10 ;BR TO ERROR HALT ON SEQ ERROR
2404 015116 012737 070707 000000  MOV #070707,@#0 ;INITIALIZE DATA IN LOC. 0
2405 015124 012700 000002  MOV #2,R0 ;INITIALIZE R0 AS POINTER
2406 015130 000261  SEC ;SET C-BIT
2407 015132 006140  ROL -(R0) ;TRY ROL W/ MODE 4
2408 015134 103406  BCS ROT4 ;CHECK C-BIT
2409 015136 022737 161617 000000  CMP #161617,@#0 ;CHECK DATA
2410 015144 001002  BNE ROT4 ;BRANCH IF DATA INCORRECT
2411 015146 005700  TST R0 ;CHECK MODE 4 REGISTER
2412 015150 001404  BEQ TST166 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 761 <====

015152 ROT4:
015152 012742 000367  MOV #367,-(R2) ;MOVE TO MAILBOX # ***** 367 *****
015156 005242  INC -(R2) ;SET MSGTYP TO FATAL ERROR
015160 000000  HALT ;ROL MODE 4 FAILED

```

; OR SEQUENCE ERROR

2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424

: THIS TEST VERIFIES MODE 5 ROTATE INSTRUCTIONS.
: THE DATA IS STORED IN A WORK LOCATION (ROTX) AT THE END OF THE
: TEST CODE. LOC. 0 IS LOADED WITH THE ADDRESS OF THE DATA (ROTX).
: R0 IS SET TO 2. THE CARRY IS CLEARED AND A MODE 5 ROL
: IS EXECUTED USING R0 AS AN ADDRESSING REGISTER. THE DATA IS
: CHECKED, THE C AND V BITS TESTED, AND R0 CHECKED FOR PROPER
: DECREMENTING.

: TEST 166 TEST MODE 5 W/ ROTATE INSTRUCTIONS

015162 005212
015164 022712 000166
015170 001021
2425 015172 012737 015244 000000
2426 015200 012700 000002
2427 015204 012767 107070 000032
2428 015212 000241
2429 015214 006150
2430 015216 103006
2431 015220 022737 016160 015244
2432 015226 001002
2433 015230 005700
2434 015232 001405

TST166: INC (R2) ;UPDATE TEST NUMBER
CMP #166,(R2) ;SEQUENCE ERROR?
BNE ROT5 ;BR TO ERROR HALT ON SEQ ERROR
MOV #ROTX,@#0 ;MOVE POINTER TO LOC. 0
MOV #2,R0 ;SET MODE 5 REG. TO LOC. 0
MOV #107070,ROTX ;INITIALIZE DATA
CLC ;CLEAR C-BIT
ROL @-(R0) ;TRY ROL W/ MODE 5
BCC ROT5 ;CHECK C-BIT
CMP #016160,@#ROTX ;CHECK DATA
BNE ROT5 ;BRANCH IF DATA INCORRECT
TST R0 ;CHECK MODE 5 REGISTER
BEQ TST167

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 756 <====

015234
015234 012742 000370
015240 005242
015242 000000

ROT5: MOV #370,-(R2) ;MOVE TO MAILBOX # ***** 370 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;ROL MODE 5 FAILED
; OR SEQUENCE ERROR

2435 015244 000000
2436
2437
2438
2439
2440
2441
2442
2443
2444

ROTX: 0

: THIS TEST VERIFIES MODE 6 ROTATE INSTRUCTIONS.
: IT USES THE SAME PROCEDURE AS THE ABOVE TEST EXCEPT THE
: ROTATE INSTRUCTION USES MODE 6 ADDRESSING WITH REGISTER 7.
: THE DATA IS STILL OPERATED ON IN LOC. ROTX (SEE PREVIOUS TEST).

: TEST 167 TEST MODE 6 W/ ROTATE INSTRUCTIONS

015246 005212
015250 022712 000167
015254 001013
2445 015256 012737 125252 015244
2446 015264 000261
2447 015266 006167 177752
2448 015272 103004
2449 015274 022737 052525 015244

TST167: INC (R2) ;UPDATE TEST NUMBER
CMP #167,(R2) ;SEQUENCE ERROR?
BNE TST170-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #125252,@#ROTX ;INITIALIZE DATA
SEC ;SET C-BIT
ROL ROTX ;TRY ROL W/ MODE 6
BCC ROT6 ;CHECK C-BIT
CMP #52525,@#ROTX ;CHECK DATA

2450 015302 001404

BEQ TST170

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 764 <====

015304
015304 012742 000371
015310 005242
015312 000000

ROT6: MOV #371,-(R2) ;MOVE TO MAILBOX # ***** 371 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;ROL W/ MODE 6 FAILED
; OR SEQUENCE ERROR

2451
2452
2453
2454
2455
2456
2457
2458
2459

: THIS TEST VERIFIES MODE 7 ROTATE INSTRUCTIONS.
: THE DATA IS SET IN LOC. ROTX, (SEE PREVIOUS TEST). THE ROL INSTRUCTION
: ADDRESSES IT INDIRECTLY USING MODE 7 AND INDIRECT ADDRESS LOCATION
: (ROTXAD) FOLLOWING THE TEST CODE.

TEST 170 TEST MODE 7 W/ ROTATE INSTRUCTIONS

015314 005212
015316 022712 000170
015322 001016
2460 015324 012737 052525 015244
2461 015332 012737 015244 015370
2462 015340 000241
2463 015342 006177 000022
2464 015346 103404
2465 015350 023727 015244 125252
2466 015356 001405

TST170: INC (R2) ;UPDATE TEST NUMBER
CMP #70,(R2) ;SEQUENCE ERROR?
BNE ROT7 ;BR TO ERROR HALT ON SEQ ERROR
MOV #52525,@ROTX ;INITIALIZE DATA
MOV #ROTX,@ROTXAD ;INITIALIZE ADDRESS POINTER
CLC ;CLEAR C-BIT
ROL @ROTXAD ;TRY ROL W/ MODE 7
BCS ROT7 ;CHECK C-BIT
CMP @ROTX,#125252 ;CHECK DATA
BEQ TST171

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 761 <====

015360
015360 012742 000372
015364 005242
015366 000000

ROT7: MOV #372,-(R2) ;MOVE TO MAILBOX # ***** 372 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;ROL W/ MODE 7 FAILED
; OR SEQUENCE ERROR

2467 015370 000000

ROTXAD: 0

2468
2469
2470
2471
2472
2473
2474
2475
2476
2477

: THIS TEST VERIFIES MODE 0 SWAB INSTRUCTION. R0 IS SET TO
: 177400. A SWAB MODE 0 IS EXECUTED AND THE CONDITIONAL BRANCH
: IS USED TO CHECK THE SIGN OF THE RESULT. ALSO, A COMPARISON
: IS MADE TO CHECK THE DATA RESULTS.

TEST 171 TEST MODE 0 W/ SWAB INST.

015372 005212
015374 022712 000171
015400 001013
2478 015402 012700 177400

TST171: INC (R2) ;UPDATE TEST NUMBER
CMP #171,(R2) ;SEQUENCE ERROR?
BNE TST172-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #177400,R0 ;MOVE TEST PATTERN TO R0

```

2479 015406 000300          SWAB  R0          ;TRY SWAB MODE 0
2480 015410 100404          BMI   SBO
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 773 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 373 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; SWAB DID NOT SET CC'S CORRECT
                                ; CHECK RESULT
                                ;
015412 012742 000373          MOV   #373,-(R2)
015416 005242              INC   -(R2)
015420 000000              HALT
2481 015422 022700 000377          SBO:  CMP   #377,R0
2482 015426 001404          BEQ   TST172
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 764 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 374 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; RESULT OF SWAB MODE 0 FAILED
                                ; OR SEQUENCE ERROR
015430 012742 000374          MOV   #374,-(R2)
015434 005242              INC   -(R2)
015436 000000              HALT

```

2483
2484
2485
2486
2487
2488
2489
2490
2491

```

:*****
: THIS TEST VERIFIES MODE 1 SWAB INSTRUCTION. THE TEST
: PATTERN IS MOVED TO LOC 0. R0 IS CLEARED AND USED AS THE ADDRESSING
: REGISTER IN THE MODE 1 SWAB. THE DATA RESULTS ARE CHECKED WITH
: A COMPARE.
:*****

```

```

:TEST 172 TEST MODE 1 W/ SWAB INST
:*****
TST172: INC   (R2)          ;UPDATE TEST NUMBER
        CMP   #172,(R2)    ;SEQUENCE ERROR?
        BNE   TST173-10    ;BR TO ERROR HALT ON SEQ ERROR
2492 015450 012737 125652 000000    MOV   #125652,@#0        ;MOVE TEST PATTERN TO LOC. 0
2493 015456 005000              CLR   R0                 ;R0=0
2494 015460 000310              SWAB (R0)                ;TRY SWAB MODE 1
2495 015462 022737 125253 000000    CMP   #125253,@#0        ;CHECK RESULT
2496 015470 001404          BEQ   *TST173
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 766 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 375 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; RESULT OF SWAB MODE 1 FAILED
                                ; OR SEQUENCE ERROR
015472 012742 000375          MOV   #375,-(R2)
015476 005242              INC   -(R2)
015500 000000              HALT

```

2497
2498
2499
2500
2501
2502
2503
2504
2505
2506

```

:*****
: THIS TEST VERIFIES MODE 2 SWAB INSTRUCTION. THE TEST
: PATTERN IS MOVED TO LOC 0. R0 IS CLEARED AND USED AS THE MODE
: 2 ADDRESSING REGISTER. THE RESULTS ARE CHECKED WITH A COMPARE.
: R0 IS CHECKED FOR PROPER DECREMENTING.
:*****

```

:TEST 173 TEST MODE 2 W/ SWAB INST

```
*****  
015502 005212 TST173: INC (R2) ;UPDATE TEST NUMBER  
015504 022712 000173 CMP #173,(R2) ;SEQUENCE ERROR?  
015510 001020 BNE TST174-10 ;BR TO ERROR HALT ON SEQ ERROR  
2507 015512 012737 125152 000000 MOV #125152,@#0 ;MOVE TEST PATTERN TO LOC. 0  
2508 015520 005000 CLR R0 ;R0=0  
2509 015522 000320 SWAB (R0)+ ;TRY SWAB MODE 2  
2510 015524 022737 065252 000000 CMP #65252,@#0 ;CHECK RESULT  
2511 015532 001404 BEQ SB2
```

```
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---=  
CONDITIONAL BRANCH INST. AND <====  
REPLACE THE MOVE INSTRUCTION <====  
WHICH FOLLOWS W/ 766 <===
```

```
015534 012742 000376 MOV #376,-(R2) ;MOVE TO MAILBOX # ***** 376 *****  
015540 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR  
015542 000000 HALT ;RESULT OF SWAB MODE 0 FAILED  
2512 015544 162700 000002 SB2: SUB #2,R0 ;CHECK EFFECT OF REG.  
2513 015550 001404 BEQ TST174
```

```
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---=  
CONDITIONAL BRANCH INST. AND <====  
REPLACE THE MOVE INSTRUCTION <====  
WHICH FOLLOWS W/ 757 <---
```

```
015552 012742 000377 MOV #377,-(R2) ;MOVE TO MAILBOX # ***** 377 *****  
015556 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR  
015560 000000 HALT ;REGISTER VALUE INCORRECT  
; OR SEQUENCE ERROR
```

2514
2515
2516
2517
2518
2519
2520
2521
2522
2523

```
*****  
: THIS TEST VERIFIES MODE 3 SWAB INSTRUCTION. THE TEST  
: PATTERN IS MOVED TO LOC 0. A MODE 3 SWAB INSTRUCTION IS EXECUTED  
: USING R7 AS THE ADDRESSING REGISTER. A COMPARE VERIFIES THE  
: DATA RESULTS.  
*****
```

```
: TEST 174 TEST MODE 3 W/SWAB INST.  
*****
```

```
015562 005212 TST174: INC (R2) ;UPDATE TEST NUMBER  
015564 022712 000174 CMP #174,(R2) ;SEQUENCE ERROR?  
015570 001011 BNE TST175-10 ;BR TO ERROR HALT ON SEQ ERROR  
2524 015572 012737 000377 000000 MOV #377,@#0 ;MOVE TEST PATTERN TO LOC. 0  
2525 015600 000337 000000 SWAB @#0 ;TRY SWAB W/ MODE 3  
2526 015604 022737 177400 000000 CMP #177400,@#0 ;CHECK RESULT  
2527 015612 001404 BEQ TST175
```

```
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
CONDITIONAL BRANCH INST. AND <====  
REPLACE THE MOVE INSTRUCTION <---=  
WHICH FOLLOWS W/ 766 <---
```

```
015614 012742 000400 MOV #400,-(R2) ;MOVE TO MAILBOX # ***** 400 *****  
015620 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR  
015622 000000 HALT ;RESULT OF SWAB INCORRECT  
; OR SEQUENCE ERROR
```

2528
2529
2530
2531

```
*****  
:
```

2532
2533
2534
2535
2536
2537

: THIS TEST VERIFIES MODE 4 SWAB INSTRUCTIONS. THE DATA
: IS MOVED TO LOC 0. R0 IS SET TO 2 AND USED AS THE MODE 4 ADDRESSING
: REGISTER. THE DATA IS CHECKED WITH A COMPARE AND R0 IS CHECKED
: FOR PROPER DECREMENTING.

: TEST 175 TEST MODE 4 W/ SWAB INST

015624 005212
015626 022712 000175
015632 001020
2538 015634 012737 125652 000000
2539 015642 012700 000002
2540 015646 000340
2541 015650 022737 125253 000000
2542 015656 001404

TST175: INC (R2) ;UPDATE TEST NUMBER
CMP #175,(R2) ;SEQUENCE ERROR?
BNE TST176-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #125652,@#0 ;MOVE TEST PATTERN TO LOC. 0
MOV #2,R0 ;SET UP REGISTER POINTER
SWAB -(R0) ;TRY SWAB MODE 4
CMP #125253,@#0 ;CHECK RESULT
BEQ SB4

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---=
: CONDITIONAL BRANCH INST. AND <---=
: REPLACE THE MOVE INSTRUCTION <---=
: WHICH FOLLOWS W/ 765 <---=

015660 012742 000401
015664 005242
015666 000000
2543 015670 005700
2544 015672 001404

MOV #401,-(R2) ;MOVE TO MAILBOX # ***** 401 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;RESULT OF SWAB INCORRECT
SB4: TST R0 ;CHECK EFFECT ON REG.
BEQ TST176

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---=
: CONDITIONAL BRANCH INST. AND <---=
: REPLACE THE MOVE INSTRUCTION <---=
: WHICH FOLLOWS W/ 757 <---=

015674 012742 000402
015700 005242
015702 000000

MOV #402,-(R2) ;MOVE TO MAILBOX # ***** 402 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;REGISTER VALUE INCORRECT
; OR SEQUENCE ERROR

2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556

: THIS TEST VERIFIES MODE 5 SWAB INSTRUCTION. THE TEST USES
: TWO LOCATIONS FOLLOWING THE TEST CODE. SB5X HOLDS THE DATA;
: SB5XAD IS A POINTER TO THE DATA LOCATION. THE DATA IS MOVED TO
: SB5X AND R0 IS SET TO TWO PLUS THE ADDRESS OF SB5XAD. FOLLOWING
: THE MODE 5 SWAB SB5X IS CHECKED FOR THE PROPER DATA. R0 IS
: CHECKED TO SEE THAT IT WAS DECREMENTED PROPERLY.

: TEST 176 TEST MODE 5 W/ SWAB INST.

015704 005212
015706 022712 000176
015712 001021
2557 015714 012700 015772
2558 015720 012767 125125 000040
2559 015726 000350
2560 015730 022767 052652 000030
2561 015736 001404

TST176: INC (R2) ;UPDATE TEST NUMBER
CMP #176,(R2) ;SEQUENCE ERROR?
BNE SB5 ;BR TO ERROR HALT ON SEQ ERROR
MOV #SB5XAD+2,R0 ;SET UP POINTER TO WORK LOCATION
MOV #125125,SB5X ;MOVE PATTERN TO WORK LOCATION
SWAB @-(R0) ;TRY SWAB MODE 5
CMP #52652,SB5X ;CHECK RESULT
BEQ SB5A

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---=
: CONDITIONAL BRANCH INST. AND <---=

```

:                                     REPLACE THE MOVE INSTRUCTION <====
:                                     WHICH FOLLOWS W/ 765 <====
:                                     MOVE TO MAILBOX # ***** 403 *****
:                                     SET MSGTYP TO FATAL ERROR
:                                     RESULT OF SWAB INCORRECT
:                                     CHECK RESULT OF REG.
015740 012742 000403      MOV      #403,-(R2)
015744 005242      INC      -(R2)
015746 000000      HALT
2562 015750 020027 015770  SB5A:  CMP      R0,#SB5XAD
2563 015754 001406      BEQ      TST177

```

```

:                                     TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:                                     CONDITIONAL BRANCH INST. AND <====
:                                     REPLACE THE MOVE INSTRUCTION <====
:                                     WHICH FOLLOWS W/ 756 <====
015756      SB5:
015756 012742 000404      MOV      #404,-(R2)
015762 005242      INC      -(R2)
015764 000000      HALT
2564 015766 000000      SB5X:  0
2565 015770 015766      SB5XAD: SB5X

```

```

:*****
:
:      THIS TEST VERIFIES MODE 6 SWAB INSTRUCTION. THIS TEST
:      USES A WORK LOCATION (SB6X) FOLLOWING THE TEST CODE. TEST DATA
:      IS LOADED INTO THE WORK LOCATION. R0, THE ADDRESSING REGISTER
:      IS LOADED WITH 6 LESS THEN THE ADDRESS OF THE WORK LOCATION.
:      THE MODE 6 SWAB IS EXECUTED WITH A +6 OFFSET. THE DATA IS
:      VERIFIED WITH A COMPARE.
:*****

```

```

:*****
:      TEST 177      TEST MODE 6 W/ SWAB INST.
:*****
TST177:  INC      (R2)          ;UPDATE TEST NUMBER
:          CMP      #177,(R2)    ;SEQUENCE ERROR?
:          BNE      SB6          ;BR TO ERROR HALT ON SEQ ERROR
2578 016002 012767 125125 000030  MOV      #125125,SB6X        ;MOVE PATTERN TO WORK LOCATION
2579 016010 012700 016032      MOV      #SB6X-6,R0         ;MOVE OFFSET POINTER TO R0
2580 016014 000360 000006      SWAB     6(R0)             ;TRY SWAB W/ MODE 6
2581 016020 022760 052652 000006  CMP      #52652,6(R0)       ;CHECK RESULT
2582 016026 001405      BEQ      TST200

```

```

:                                     TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:                                     CONDITIONAL BRANCH INST. AND <====
:                                     REPLACE THE MOVE INSTRUCTION <====
:                                     WHICH FOLLOWS W/ 764 <====
016030      SB6:
016030 012742 000405      MOV      #405,-(R2)
016034 005242      INC      -(R2)
016036 000000      HALT
2583 016040 000000      SB6X:  0

```

```

:*****
:
:      THIS TEST VERIFIES MODE 7 SWAB INSTRUCTION. THIS TEST
:      USES TWO LOCATIONS FOLLOWING THE TEST CODE: A WORK LOCATION
:      (SB7X) AND A POINTER TO THE WORK LOCATION (SB7XAD). DATA IS MOVED
:*****

```

2591
 2592
 2593
 2594
 2595
 2596

016042 005212
 016044 022712 000200
 016050 001013
 2597 016052 012767 177400 000030
 2598 016060 012700 016020
 2599 016064 000370 000072
 2600 016070 027027 000072 000377
 2601 016076 001406

:TO THE WORK LOCATION. R0 IS LOADED WITH 72 LESS THAN THE ADDRESS
 :OF THE ADDRESS POINTER. THE DATA IS SWAB'ED USING A MODE 7
 :INSTRUCTION WITH AN OFFSET OF +72. THE DATA IS VERIFIED WITH A
 :COMPARE.

 :TEST 200 TEST MODE 7 w/ SWAB INST.

TST200: INC (R2) ;UPDATE TEST NUMBER
 CMP #200,(R2) ;SEQUENCE ERROR?
 BNE SB7 ;BR TO ERROR HALT ON SEQ ERROR
 MOV #177400,SB7X ;MOVE PATTERN TO WORK LOCATION
 MOV #SB7XAD-72,R0 ;MOVE OFFSET POINTER TO R0
 SWAB @72(R0) ;TRY SWAB MODE 7
 CMP @72(R0),#377 ;CHECK RESULTS
 BEQ TST201

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
 : CONDITIONAL BRANCH INST. AND <====
 : REPLACE THE MOVE INSTRUCTION <====
 : WHICH FOLLOWS W/ 764 <====

016100
 016100 012742 000406
 016104 005242
 016106 000000

2602 016110 000000
 2603 016112 016110
 2604
 2605
 2606
 2607
 2608
 2609
 2610
 2611
 2612
 2613
 2614
 2615
 2616
 2617
 2618
 2619
 2620
 2621
 2622
 2623
 2624
 2625
 2626
 2627
 2628
 2629
 2630
 2631
 2632
 2633

SB7: MOV #406,-(R2) ;MOVE TO MAILBOX # ***** 406 *****
 INC -(R2) ;SET MSGTYP TO FATAL ERROR
 HALT ;RESULT OF SWAB INCORRECT
 ; OR SEQUENCE ERROR
 ;WORK LOCATION
 SB7X: 0 ;
 SB7XAD: SB7X ;POINTER TO WORK LOCATION

: THIS TEST VERIFIES ALL LEGAL MODES OF THE JMP INSTRUCTION.
 : BECAUSE OF THE NATURE OF THE INSTRUCTION UNDER TEST, THIS TEST
 : UTILIZES SEVERAL DIFFERENT TECHNIQUES. THE CODE IS NOT EXECUTED
 : IN A LINEAR FASHION. THE DIFFERENT MODES ARE EXECUTED IN ORDER
 : FROM 1-7; HOWEVER, THE CODE IS ARRANGED SO THAT CONTROL LEAP
 : FROGS THRU THE TEST CODE. THE ORDER OF APPEARANCE OF THE CODE
 : IS:

- JMP MODE 1
- JMP MODE 3
- JMP MODE 2
- JMP MODE 4
- JMP MODE 6
- JMP MODE 5
- JMP MODE 7

: AN INTERNAL SEQUENCE TEST (JMPSEQ) IS USED TO INSURE THAT THE
 : JUMPS ARE OCCURRING IN THE PROGRAMMED SEQUENCE.
 : THE TEST IS MADE UP OF SEVERAL BLOCKS OF CODE. EACH CODE
 : BEGINS WITH A LABEL WHICH INDICATES THE MODE BEING EXECUTED IN
 : THAT BLOCK. A SIMPLE PROCEDURE IS FOLLOWED IN EACH BLOCK. FOR
 : EXAMPLE THE CODE BEGINNING AT JMP3 WILL FIRST COMPARE THE RESULTS
 : OF THE PREVIOUS MODE 2 JUMP. (ANY REGISTER CHANGES ARE VERIFIED
 : AND THE SEQUENCE CHECK IS MADE). THEN THE REGISTERS ARE SETUP
 : FOR A MODE 3 JUMP TO THE NEXT TEST BLOCK (HERE, JMP4), THE SEQUENCE
 : CHECKER IS UPDATED AND THE JUMP IS EXECUTED.
 : IF A FAILURE OCCURS, THE SEQUENCE CHECKER WILL ASSIST IN
 : DETERMINING JUST WHICH MODE FAILED. IF THE SEQUENCE IS CORRECT

```

2634 ; THEN THE ERROR DETECTED WAS A MODE FAILURE (E.G. FAILURE OF THE
2635 ; REGISTER TO BE INCREMENTED IN MODE 2 JUMP.)
2636 ;
2637 ;*****
;TEST 201 TEST THE JMP INSTRUCTION IN ALL MODES
;*****
016114 005212 TST201: INC (R2) ;UPDATE TEST NUMBER
016116 022712 000201 CMP #201,(R2) ;SEQUENCE ERROR?
016122 001150 BNE JMPCK+6 ;BR TO ERROR HALT ON SEQ ERROR
2638 016124 005067 000326 CLR JMPSEQ ;ESTABLISH A SEQUENCE CHECKER
2639 016130 012700 016210 MOV #JMP2,R0 ;SET R0=JUMP TARGET
2640 016134 000110 JMP (R0) ;TRY JMP MODE 1
2641 016136 022700 016140 JMP3: CMP #.+2,R0 ;CHECK RESULT OF MODE 2 JUMP
2642 016142 001404 BEQ JMP3A

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 767 <====

016144 012742 000407 MOV #407,-(R2) ;MOVE TO MAILBOX # ***** 407 *****
016150 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
016152 000000 HALT ;REGISTER VALUE AFTER JMP MODE 2 INCORRECT
2643 016154 026727 000276 000001 JMP3A: CMP JMPSEQ,#1 ;MAKE SURE JMPS ARE IN SEQUENCE: JMPSEQ=1?
2644 016162 001404 BEQ JMP3B

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 757 <====

016164 012742 000410 MOV #410,-(R2) ;MOVE TO MAILBOX # ***** 410 *****
016170 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
016172 000000 HALT ;SHOULD BE HERE FROM JMP MODE 2 ONLY
2645 016174 012700 016206 JMP3B: MOV #IJMP4,R0 ;POINT R0 TO INDIRECT JMP ADDR.
2646 016200 005267 000252 INC JMPSEQ ;UPDATE SEQUENCE CHECKER
2647 016204 000130 JMP @ (R0)+ ;TRY JMP MODE 3
2648 016206 016240 IJMP4: JMP4 ;ADDRESS INDIRECT JUMP
2649
2650 016210 005767 000242 JMP2: TST JMPSEQ ;CHECK THAT JMPS ARE IN SEQUENCE: JMPSEQ=0?
2651 016214 001404 BEQ JMP2A

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 742 <====

016216 012742 000411 MOV #411,-(R2) ;MOVE TO MAILBOX # ***** 411 *****
016222 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
016224 000000 HALT ;SHOULD BE HERE FROM JMP MODE 1 ONLY
2652 016226 005267 000224 JMP2A: INC JMPSEQ ;UPDATE SEQUENCE CHECKER
2653 016232 012700 016136 MOV #JMP3,R0 ;SET R0=JUMP TARGET
Z 2654 016236 000120 JMP (R0)+ ;TRY A JUMP MODE 2 TO 'JMP3'
2655 016240 022700 016210 JMP4: CMP #IJMP4+2,R0 ;CHECK RESULT OF REGISTER IN MODE 3 JUMP
2656 016244 001404 BEQ JMP4A

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 726 <====

016246 012742 000412 MOV #412,-(R2) ;MOVE TO MAILBOX # ***** 412 *****
016252 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
016254 000000 HALT ;REGISTER VALUE AFTER MODE 3 JUMP INCORRECT
2657 016256 022767 000002 000172 JMP4A: CMP #2,JMPSEQ ;CHECK JUMP SEQUENCE: JMPSEQ=2?

```

```

2658 016264 001404          BEQ      JMP4B          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 716 <====
      016266 012742 000413      MOV      #413,-(R2)      ;MOVE TO MAILBOX # ***** 413 *****
      016272 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
      016274 000000          HALT                    ;SHOULD BE ONLY FROM MODE 3 JUMP
2659 016276 012700 016346      JMP4B:  MOV      #JMP5+2,R0 ;SET UP POINTER TO JUMP TARGET
2660 016302 005267 000150          INC      JMPSEQ        ;UPDATE SEQUENCE CHECKER
2661 016306 000140          JMP      -(R0)         ;TRY JUMP MODE 4 TO 'JMP4'
2662
2663 016310 022767 000004 000140  JMP6:   CMP      #4,JMPSEQ  ;CHECK THAT JUMPS ARE IN SEQUENCE: JMPSEQ=4?
2664 016316 001404          BEQ      JMP6A
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 701 <====
      016320 012742 000414      MOV      #414,-(R2)      ;MOVE TO MAILBOX # ***** 414 *****
      016324 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
      016326 000000          HALT                    ;SHOULD BE HERE ONLY FROM MODE 5 JUMP
2665 016330 012700 016776      JMP6A:  MOV      #JMP7+376,R0 ;SET UP OFFSET POINTER TO JUMP TARGET
2666 016334 005267 000116          INC      JMPSEQ        ;UPDATE JUMP SEQUENCE
2667 016340 000160 177402          JMP      -376(R0)       ;TRY MODE 6 JUMP
2668
2669 016344 022767 000003 000104  JMP5:   CMP      #3,JMPSEQ  ;CHECK THAT JUMPS ARE IN SEQUENCE: JMPSEQ=3?
2670 016352 001404          BEQ      JMP5A
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 663 <====
      016354 012742 000415      MOV      #415,-(R2)      ;MOVE TO MAILBOX # ***** 415 *****
      016360 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
      016362 000000          HALT                    ;SHOULD ONLY BE HERE FROM MODE 4 JUMP
2671 016364 012700 016400      JMP5A:  MOV      #I JMP5+2,R0 ;SET UP POINTER TO INDIRECT JUMP ADDR.
2672 016370 005267 000062          INC      JMPSEQ        ;UPDATE JUMP SEQUENCE
2673 016374 000150          JMP      @-(R0)         ;TRY JUMP MODE 5 TO 'JMP6'
2674 016376 016310          I JMP5:  JMP6
2675
2676 016400 022767 000005 000050  JMP7:   CMP      #5,JMPSEQ  ;CHECK JUMPS IN SEQUENCE: JMPSEQ=5?
2677 016406 001404          BEQ      JMP7A
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 645 <====
      016410 012742 000416      MOV      #416,-(R2)      ;MOVE TO MAILBOX # ***** 416 *****
      016414 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
      016416 000000          HALT                    ;SHOULD ONLY BE HERE FROM MODE 6 JUMP
2678 016420 012700 016444      JMP7A:  MOV      #I JMP+10,R0 ;SET UP OFFSET POINTER TO INDIRECT ADDR.
2679 016424 005267 000026          INC      JMPSEQ        ;UPDATE JUMP SEQUENCE
2680 016430 000170 177770          JMP      @-10(R0)       ;TRY MODE 7 JUMP
2681 016434 016436          I JMP:   JMPCK
2682
2683 016436 026727 000014 000006  JMPCK:  CMP      JMPSEQ,#6     ;CHECK JUMPS IN SEQUENCE: JMPSEQ
2684 016444 001405          BEQ      TST202
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====

```

```

:                                     REPLACE THE MOVE INSTRUCTION <===-
:                                     WHICH FOLLOWS W/ 626          <===-
016446 012742 000417      MOV      #417,-(R2)      ;MOVE TO MAILBOX # ***** 417 *****
016452 005242            INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
016454 000000            HALT          ;SHOULD ONLY BE HERE FROM MODE 6 JUMP
:                                     ; OR SEQUENCE ERROR

```

2685 016456 000000 JMPSEQ: 0

2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703

```

: THIS TEST VERIFIES ALL LEGAL MODES OF THE JSR INSTRUCTION.
: THE CONCEPT OF LEAP FROGGING AND SEQUENCE CHECKING (JSRSEQ) IS
: IDENTICAL TO THAT USED IN JMP TEST (SEE PREVIOUS TEST). EACH
: BLOCK OF CODE VERIFIES THE PREVIOUS JSR BY CHECKING THE SEQUENCE,
: CHECKING THAT THE PC WAS SAVED IN THE SPECIFIED REGISTER, CHECKING
: THAT THE SP WAS DECREMENTED, CHECKING THAT THE REGISTER WAS
: SAVED ON THE STACK, AND FINALLY CHECKING THAT ANY MODE ADDRESS
: REGISTER ALTERATIONS (E.G. INCREMENT REGISTER IN MODE 2) WERE
: SUCCESSFUL. R1 IS USED AS THE REGISTER IN ALL JSR INSTRUCTIONS.
: IF A FAILURE OCCURS, THE SEQUENCE CHECKER WILL ASSIST IN
: DETERMINING JUST WHICH MODE FAILED. IF THE SEQUENCE IS CORRECT
: THEN THE ERROR DETECTED WAS A FUNCTIONAL FAILURE (E.G., INCORRECT
: REGISTER SAVED).

```

: TEST 202 TEST JSR INSTRUCTION W/ ALL MODES

```

016460 005212            TST202: INC      (R2)      ;UPDATE TEST NUMBER
016462 022712 000202    CMP      #202,(R2)      ;SEQUENCE ERROR?
016466 001001            BNE     JSR0           ;BR TO ERROR HALT ON SEQ ERROR
2704 016470 000402      BR      JSR1
2705 016472 000137 017126 JSR0:  JMP      @#JSRCK1
2707 016476 012706 001000 JSR1:  MOV      #STBOT,R6   ;SET STACK POINTER
2708 016502 012700 016610 MOV      #JSR2,R0       ;SET TARGET ADDRESS
2709 016506 005037 017106 CLR      @#JSRSEQ       ;INITIALIZE SEQUENCE CHECKER
2710 016512 005001      CLR      R1           ;INITIALIZE R1
2711 016514 005101      COM     R1
2712 016516 004110      JSR     R1,(R0)       ;TRY JSR MODE 1
2713                                     ; TO SCOPE: REPLACE THE MOVE INSTRUCTION <
2714                                     ; FOLLOWING W/ 774          <
2715 JSR1A:
016520 012742 000420      MOV      #420,-(R2)     ;MOVE TO MAILBOX # ***** 420 *****
016524 005242            INC      -(R2)     ;SET MSGTYP TO FATAL ERROR
016526 000000            HALT          ;JSR MODE 1 FAILED
2716
2717 016530 022737 000001 017106 JSR3:  CMP      #1,@#JSRSEQ   ;CHECK SEQUENCE: JSRSEQ=1?
2718 016536 001014      BNE     JSR3A         ;BRANCH IF OUT OF SEQUENCE
2719 016540 020127 016672      CMP     R1,#JSR4      ;PROPER PC SAVED?
2720 016544 001011      BNE     JSR3A         ;BRANCH IF PC WRONG
2721 016546 022706 000776      CMP     #STBOT-2,R6   ;STACK POINTER DECREMENTED?
2722 016552 001006      BNE     JSR3A         ;BRANCH IF SP WRONG
2723 016554 022716 125252      CMP     #125252,(R6)  ;REG SAVED ON STACK?
2724 016560 001003      BNE     JSR3A         ;BRANCH IF REG. NOT SAVED
2725 016562 022700 016532      CMP     #JSR3+2,R0   ;MODE 2 INCREMENT CORRECT?
2726 016566 001404      BEQ    JSR3B
:                                     ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- -

```

```

:                                     :
:                                     :   CONDITIONAL BRANCH INST. AND   <====
:                                     :   REPLACE THE MOVE INSTRUCTION   <----
:                                     :   WHICH FOLLOWS W/ 737          <----
:
016570                               JSR3A:  MOV    #421,-(R2)   ;MOVE TO MAILBOX # ***** 421 *****
016570 012742 000421                INC    -(R2)       ;SET MSGTYP TO FATAL ERROR
016574 005242                       HALT                    ;JSR MODE 3 MALFUNCTIONED
016576 000000                       ;JSR MODE 3 MALFUNCTIONED
2727 016600 005237 017106           JSR3B:  INC    @#JSRSEQ   ;UPDATE SEQUENCE CHECKER
2728 016604 004137 016672           JSR    R1,@#JSR4       ;TRY JSR MODE 4
2729
2730 016610 005737 017106           JSR2:   TST    @#JSRSEQ   ;CHECK SEQUENCE: JSRSEQ=0?
2731 016614 001011                   BNE    JSR2A           ;BRANCH IF OUT OF SEQUENCE
2732 016616 020127 016520           CMP    R1,#JSR1A      ;PROPER PC SAVED?
2733 016622 001006                   BNE    JSR2A           ;BRANCH IF PC WRONG
2734 016624 022706 000776           CMP    #STBOT-2,R6    ;R6 DECREMENT?
2735 016630 001003                   BNE    JSR2A           ;BRANCH IF R6 IS INCORRECT
2736 016632 021627 177777           CMP    (R6),#-1      ;REGISTER SAVED?
2737 016636 001404                   BEQ    JSR2B

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
;                                     < ==
;                                     < -
;                                     < ==
;                                     WHICH FOLLOWS W/ 713

016640                               JSR2A:  MOV    #422,-(R2)   ;MOVE TO MAILBOX # ***** 422 *****
016640 012742 000422                INC    -(R2)       ;SET MSGTYP TO FATAL ERROR
016644 005242                       HALT                    ;JSR MODE 1 MALFUNCTIONED
016646 000000                       ;JSR MODE 1 MALFUNCTIONED
2738 016650 012706 001000           JSR2B:  MOV    #STBOT,R6 ;INITIALIZE R6
2739 016654 012701 125252           MOV    #125252,R1    ;INITIALIZE R1
2740 016660 005237 017106           INC    @#JSRSEQ       ;UPDATE SEQUENCE CHECKER
2741 016664 012700 016530           MOV    #JSR3,R0      ;SET TARGET ADDRESS
Z 2742 016670 004120                JSR    R1,(R0)+       ;TRY JSR MODE 2
2743
2744 016672 022737 000002 017106   JSR4:   CMP    #2,@#JSRSEQ ;CHECK SEQUENCE: JSRSEQ=2?
2745 016700 001003                   BNE    JSR4A           ;BRANCH IF OUT OF SEQUENCE
2746 016702 022701 016610           CMP    #JSR2,R1      ;PROPER PC SAVED?
2747 016706 001404                   BEQ    JSR4B

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     <====
;                                     <====
;                                     <====
;                                     WHICH FOLLOWS W/ 667

016710                               JSR4A:  MOV    #423,-(R2)   ;MOVE TO MAILBOX # ***** 423 *****
016710 012742 000423                INC    -(R2)       ;SET MSGTYP TO FATAL ERROR
016714 005242                       HALT                    ;JSR MODE 3 MALFUNCTIONED
016716 000000                       ;JSR MODE 3 MALFUNCTIONED
2748 016720 005237 017106           JSR4B:  INC    @#JSRSEQ   ;UPDATE SEQUENCE CHECKER
2749 016724 012700 017000           MOV    #JSR5+2,R0    ;SET TARGET ADDRESS
2750 016730 004140                JSR    R1,-(R0)      ;TRY JSR MODE 4
2751
2752 016732 022767 000004 000146   JSR6:   CMP    #4,JSRSEQ  ;CHECK SEQUENCE: JSRSEQ=4?
2753 016740 001006                   BNE    JSR6A           ;BRANCH IF OUT OF SEQUENCE
2754 016742 022701 017044           CMP    #JSR7,R1      ;PROPER PC SAVED?
2755 016746 001003                   BNE    JSR6A           ;BRANCH IF PC WRONG
2756 016750 022700 017102           CMP    #JSR6AD,R0    ;MODE 5 REGISTER CORRECT?
2757 016754 001404                   BEQ    JSR6B

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <----
;                                     <----
;                                     <----
;                                     <--

```

```

;          WHICH FOLLOWS W/ 644          <====
016756      JSR6A:
016756      MOV      #424,-(R2)          ;MOVE TO MAILBOX # ***** 424 *****
016762      INC      -(R2)              ;SET MSGTYP TO FATAL ERROR
016764      HALT
016764      000000
2758 016766 005237 017106      JSR6B: INC      @#JSRSEQ          ;UPDATE SEQUENCE CHECKER
2759 016772 004167 000046      JSR      R1,JSR7          ;TRY JSR MODE 6
2760 016776 022767 000003 000102 JSR5:  CMP      #3,JSRSEQ          ;CHECK SEQUENCE: JSRSEQ=3?
2761 017004 001006              BNE      JSR5A          ;BRANCH IF OUT OF SEQUENCE
2762 017006 022701 016732      CMP      #JSR6,R1       ;PROPER PC SAVED?
2763 017012 001003              BNE      JSR5A          ;BRANCH IF PC WRONG
2764 017014 022700 016776      CMP      #JSR5,R0       ;CHECK MODE 4 REGISTER
2765 017020 001404              BEQ      JSR5B

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
;          CONDITIONAL BRANCH INST. AND <
;          REPLACE THE MOVE INSTRUCTION <
;          WHICH FOLLOWS W/ 622          <

017022      JSR5A:
017022      MOV      #425,-(R2)          ;MOVE TO MAILBOX # ***** 425 *****
017026      INC      -(R2)              ;SET MSGTYP TO FATAL ERROR
017030      HALT
017030      000000
2766 017032 005237 017106      JSR5B: INC      @#JSRSEQ          ;UPDATE SEQUENCE CHECKER
2767 017036 012700 017104      MOV      #JSR6AD+2,R0    ;POINT R0 TO TARGET ADDRESS
2768 017042 004150              JSR      R1,@-(R0)       ;TRY JSR MODE 5
2769
2770 017044 022737 000005 017106 JSR7:  CMP      #5,@#JSRSEQ          ;CHECK SEQUENCE: JSRSEQ=5?
2771 017052 001003              BNE      JSR7A          ;BRANCH IF OUT OF SEQUENCE
2772 017054 022701 016776      CMP      #JSR5,R1       ;PROPER PC SAVED?
2773 017060 001404              BEQ      JSR7B

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 602          <====

017062      JSR7A:
017062      MOV      #426,-(R2)          ;MOVE TO MAILBOX # ***** 426 *****
017066      INC      -(R2)              ;SET MSGTYP TO FATAL ERROR
017070      HALT
017070      000000
2774 017072 005237 017106      JSR7B: INC      @#JSRSEQ          ;UPDATE SEQUENCE CHECKER
2775 017076 004177 000002      JSR      R1,@JSRCKAD    ;TRY JSR MODE 7
2776
2777 017102 016732      JSR6AD: JSR6          ;MODE 5 TARGET ADDRESS
2778 017104 017110      JSRCKAD:JSRCK        ;MODE 7 TARGET ADDRESS
2779 017106 000000      JSRSEQ: 0            ;SEQUENCE CHECKER
2780
2781 017110 022767 000006 177770 JSRCK: CMP      #6,JSRSEQ          ;CHECK SEQUENCE: JSRSEQ=6?
2782 017116 001003              BNE      JSRCK1        ;BRANCH IF OUT OF SEQUENCE
2783 017120 022701 017102      CMP      #JSR6AD,R1     ;PROPER PC SAVED?
2784 017124 001404              BEQ      TST203

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 560          <====

017126      JSRCK1:
017126      MOV      #427,-(R2)          ;MOVE TO MAILBOX # ***** 427 *****
017132      INC      -(R2)              ;SET MSGTYP TO FATAL ERROR
017134      HALT
017134      000000
;          OR SEQUENCE ERROR

```

2785
2786
2787
2788
2789
2790
2791
2792
2793
2794

: THIS TEST VERIFIES THE RTS INSTRUCTION. THE STACK POINTER
: IS INITIALIZED AND A TEST PATTERN STORED ON STACK. R0 IS LOADED
: WITH RETURN ADDRESS. AN RTS IS EXECUTED, AND, AT THE TARGET
: ADDRESS, A CHECK IS MADE THAT R0 WAS PROPERLY RESTORED FROM THE
: STACK.

: TEST 203 TEST RTS INSTRUCTION

017136 005212
017140 022712 000203
017144 001016
2795 017146 012706 001000
2796 017152 012746 052525
2797 017156 012700 017174
2798 017162 000200
2799
2800
2801 017164 012742 000430
017170 005242
017172 000000
2802 017174 022700 052525
2803 017200 001404

TST203: INC (R2) ;UPDATE TEST NUMBER
CMP #203,(R2) ;SEQUENCE ERROR?
BNE TST204-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #STBOT,R6 ;INITIALIZE STACK POINTER
MOV #52525,-(R6) ;INITIALIZE TOP OF STACK
MOV #RTS1,R0 ;INITIALIZE RETURN REGISTER
RTS R0 ;TRY RTS THROUGH R0
: TO SCOPE: REPLACE THE MOVE INSTRUCTION <=====
: FOLLOWING W/ 770 <=====
MOV #430,-(R2) ;MOVE TO MAILBOX # ***** 430 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;RTS FAILED
RTS1: CMP #52525,R0 ;CHECK THAT R0 RESTORED FROM STACK
BEQ TST204
: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
: CONDITIONAL BRANCH INST. AND <=====
: REPLACE THE MOVE INSTRUCTION <=====
: WHICH FOLLOWS W/ 761 <=====
MOV #431,-(R2) ;MOVE TO MAILBOX # ***** 431 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;RTS MALFUNCTIONED
: OR SEQUENCE ERROR

017202 012742 000431
017206 005242
017210 000000

2804

2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819

```

: THESE NEXT FOUR TESTS VERIFY THE FUNCTIONING OF A GROUP
: OF FOUR INSTRUCTIONS. THE GROUP CONSISTS OF THE INSTRUCTIONS:
: MOV, BIC, BIT, AND BIS. THESE INSTRUCTIONS ARE SIMILAR IN THE
: WAY THEY EFFECT THE C AND V BITS. THEY ALL LEAVE THE V-BIT
: CLEAR AND THE C-BIT UNAFFECTED.
: THE TEST PROCEDURE IS AS FOLLOWS: THE N, Z, AND V BITS
: ARE LOADED WITH THE COMPLEMENT OF THE EXPECTED RESULTS, THE C-BIT
: IS LOADED WITH THE DESIRED RESULT. THE INSTRUCTION IS EXECUTED
: WITH DIFFERENT DATA PATTERNS AND THE RESULTS ARE VERIFIED WITH
: A SERIES OF CONDITIONAL BRANCH INSTRUCTIONS. THE DATA IS CHOSEN
: TO PRODUCT ALL POSSIBLE COMBINATIONS OF THE C AND V BITS.

```

```

*****
: TEST 204 TEST MOV INSTRUCTION
*****

```

017212 005212
017214 022712 000204
017220 001022
2820 017222 000277
2821 017224 000251
2822 017226 012700 100000
2823 017232 101402
2824 017234 102401
2825 017236 100404

```

TST204: INC (R2) ;UPDATE TEST NUMBER
CMP #204,(R2) ;SEQUENCE ERROR?
BNE TST205-10 ;BR TO ERROR HALT ON SEQ ERROR
SCC ;CC=0110
+CLN!CLC
MOV #100000,R0 ;CC=1000
BLOS MOV1
BVS MOV1
BMI MOV2
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===
; CONDITIONAL BRANCH INST. AND <===
; REPLACE THE MOVE INSTRUCTION <===
; WHICH FOLLOWS W/ 770 <===

```

017240
017240 012742 000432
017244 005242
017246 000000
2826
2827 017250 000277
2828 017252 000244
2829 017254 012700 000000
2830 017260 101002
2831 017262 102401
2832 017264 100004

```

MOV1: MOV #432,-(R2) ;MOVE TO MAILBOX # ***** 432 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;MOV DID NOT SET CC'S CORRECTLY

MOV2: SCC ;CC=1011
CLZ
MOV #0,R0 ;CC=0101
BHI MOV3 ;C OR Z = 0?
BVS MOV3 ;V=1?
BPL TST205
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <== -
; CONDITIONAL BRANCH INST. AND <= - =
; REPLACE THE MOVE INSTRUCTION <-- -
; WHICH FOLLOWS W/ 755 <- - =

```

017266
017266 012742 000433
017272 005242
017274 000000

```

MOV3: MOV #433,-(R2) ;MOVE TO MAILBOX # ***** 433 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;MOV DID NOT SET CC'S CORRECTLY
; OR SEQUENCE ERROR

```

2833

```

*****
: TEST 205 TEST BIT INSTRUCTION
*****

```

017276 005212
017300 022712 000205
017304 001024
2834 017306 012700 100001
2835 017312 000277

```

TST205: INC (R2) ;UPDATE TEST NUMBER
CMP #205,(R2) ;SEQUENCE ERROR?
BNE TST206-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #100001,R0
SCC ;CC=0110

```

```

2836 017314 000251          +CLN!CLC
2837 017316 032700 100000  BIT      #100000,R0      ;CC=1000
2838 017322 101402          BLOS    XBIT1
2839 017324 102401          BVS     XBIT1
2840 017326 100404          BMI     XBIT2
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 766 <====

017330          XBIT1:
017330 012742 000434      MOV     #434,-(R2)      ;MOVE TO MAILBOX # ***** 434 *****
017334 005242          INC     -(R2)          ;SET MSGTYP TO FATAL ERROR
017336 000000          HALT                    ;BIT DID NOT SET CC'S CORRECTLY

2841
2842 017340 000277          XBIT2:  SCC                    ;CC=1011
2843 017342 000244          CLZ
2844 017344 032700 077776  BIT     #77776,R0      ;CC=0101
2845 017350 101002          BHI    XBIT3
2846 017352 102401          BVS    XBIT3
2847 017354 100004          BPL    TST206
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 753 <====

017356          XBIT3:
017356 012742 000435      MOV     #435,-(R2)      ;MOVE TO MAILBOX # ***** 435 *****
017362 005242          INC     -(R2)          ;SET MSGTYP TO FATAL ERROR
017364 000000          HALT                    ;BIT DID NOT SET CC'S CORRECTLY
; OR SEQUENCE ERROR

2848
;*****
;TEST 206 TEST BIC INSTRUCTION
;*****
017366 005212          TST206: INC    (R2)          ;UPDATE TEST NUMBER
017370 022712 000206      CMP     #206,(R2)      ;SEQUENCE ERROR?
017374 001024          BNE    TST207-10      ;BR TO ERROR HALT ON SEQ ERROR
2849 017376 012700 177777  MOV     #177777,R0
2850 017402 000277          SCC                    ;CC=0110
2851 017404 000251          +CLN!CLC
2852 017406 042700 077777  BIC     #77777,R0      ;CC=1000
2853 017412 101402          BLOS   BIC1
2854 017414 102401          BVS    BIC1
2855 017416 100404          BMI    BIC2
; TO SCOPE: CLFAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 766 <====

017420          BIC1:
017420 012742 000436      MOV     #436,-(R2)      ;MOVE TO MAILBOX # ***** 436 *****
017424 005242          INC     -(R2)          ;SET MSGTYP TO FATAL ERROR
017426 000000          HALT                    ;BIC DID NOT SET CC'S CORRECTLY

2856 017430 000277          BIC2:  SCC                    ;CC=1011
2857 017432 000244          CLZ
2858 017434 042700 100000  BIC     #100000,R0      ;CC=0101
2859 017440 101002          BHI    BIC3
2860 017442 102401          BVS    BIC3
2861 017444 100004          BPL    TST207
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====

```

```

:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <====
:          WHICH FOLLOWS W/ 753          <====
017446      BIC3:
017446      012742 000437      MOV      #437,-(R2)      ;MOVE TO MAILBOX # ***** 437 *****
017452      005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
017454      000000      HALT          ;BIC DID NOT SET CC'S CORRECTLY
:          ; OR SEQUENCE ERROR
2862
:*****
:TEST 207      TEST BIS INSTRUCTION
:*****
TST207:      INC      (R2)          ;UPDATE TEST NUMBER
017456      005212      000207      CMP      #207,(R2)      ;SEQUENCE ERROR?
017460      022712      000207      BNE     TST210-10      ;BR TO ERROR HALT ON SEQ ERROR
017464      001025      000207      CLR     R0              ;R0=0
2863 017466      005000      000207      SCC     ;CC=1010
2864 017470      000277      000207      +CLN.CLC
2865 017472      000251      000207      BIS     #0,R0          ;CC=0100 R0=0
2866 017474      052700      000000      BCS     BIS1
2867 017500      103403      000000      BVS     BIS1
2868 017502      102402      000000      BMI     BIS1
2869 017504      100401      000000      BEQ     BIS2
2870 017506      001404      000000      BEQ     BIS2
:          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          ;          CONDITIONAL BRANCH INST. AND <====
:          ;          REPLACE THE MOVE INSTRUCTION <====
:          ;          WHICH FOLLOWS W/ 766 <====
017510      BIS1:
017510      012742 000440      MOV      #440,-(R2)      ;MOVE TO MAILBOX # ***** 440 *****
017514      005242      000440      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
017516      000000      000440      HALT          ;BIS DID NOT SET CC'S CORRECTLY
2871 017520      000277      000440      BIS2:      SCC     ;CC=0111
2872 017522      000250      000440      CLN
2873 017524      052700      177777      BIS     #177777,R0      ;CC=1001
2874 017530      103003      177777      BCC     BIS3
2875 017532      102402      177777      BVS     BIS3
2876 017534      001401      177777      BEQ     BIS3
2877 017536      100404      177777      BMI     TST210
:          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          ;          CONDITIONAL BRANCH INST. AND <====
:          ;          REPLACE THE MOVE INSTRUCTION <====
:          ;          WHICH FOLLOWS W/ 752 <====
017540      BIS3:
017540      012742 000441      MOV      #441,-(R2)      ;MOVE TO MAILBOX # ***** 441 *****
017544      005242      000441      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
017546      000000      000441      HALT          ;BIS DID NOT SET CC'S CORRECTLY
:          ; OR SEQUENCE ERROR

```

2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888

```

:*****
:
:          THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE INC AND
:DEC INSTRUCTIONS. THESE INSTRUCTIONS BOTH EFFECT THE C AND V
:BITS THE SAME; THE C-BIT IS LEFT UNCHANGED AND THE V-BIT IS DEPENDENT
:UPON THE DATA RESULTS. THE SAME PROCEDURE IS USED. THE CONDITION
:CODE BITS ARE INITIALIZED, THE INSTRUCTION IS EXECUTED AND THE
:RESULTS ARE VERIFIED WITH A SERIES OF CONDITIONAL BRANCH INSTRUCTIONS.
:THIS PROCEDURE IS REPEATED WITH SEVERAL DATA PATTERNS TO PRODUCE
:DIFFERENT COMBINATIONS OF THE C AND V BITS.

```

2889
2890

:TEST 210 TEST INC INSTRUCTION

017550 005212
017552 022712 000210
017556 001037
2891 017560 012700 077777
2892 017564 000257
2893 017566 000264
2894 017570 005200
2895 017572 101402
2896 017574 100001
2897 017576 102404

TST210: INC (R2) :UPDATE TEST NUMBER
CMP #210,(R2) :SEQUENCE ERROR?
BNE TST211-10 :BR TO ERROR HALT ON SEQ ERROR
MOV #077777,R0 :RO=077777
CCC :CC=0100
SEZ
INC R0 :CC=1010 RO-10000
BLOS INC1
BPL INC1
BVS INC2

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 767 <====

017600
017600 012742 000442
017604 005242
017606 000000
2898 017610 052700 077777
2899 017614 000261
2900 017616 000244
2901 017620 005200
2902 017622 100403
2903 017624 102402
2904 017626 103001
2905 017630 001404

INC1: MOV #442,-(R2) :MOVE TO MAILBOX # ***** 442 *****
INC :-(R2) :SET MSGTYP TO FATAL ERROR
HALT :INC DID NOT SET CC'S CORRECTLY
INC2: BIS #77777,R0 :RO=177777
SEC :CC=1011
CLZ
INC R0 :CC=0101 RO 0
BMI INC3
BVS INC3
BCC INC3
BEQ INC4

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 752 <====

017632
017632 012742 000443
017636 005242
017640 000000
2906
2907 017642 000277
2908 017644 000241
2909 017646 005200
2910 017650 101402
2911 017652 100401
2912 017654 100004

INC3: MOV #443,-(R2) :MOVE TO MAILBOX # ***** 443 *****
INC :-(R2) :SET MSGTYP TO FATAL ERROR
HALT :INC DID NOT SET CC'S CORRECTLY
INC4: SCC :CC=1110
CLC
INC R0 :CC=0000 RO-1
BLOS INC5
BMI INC5
BPL TST211

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 740 <====

017656
017656 012742 000444
017662 005242
017664 000000

INC5: MOV #444,-(R2) :MOVE TO MAILBOX # ***** 444 *****
INC :-(R2) :SET MSGTYP TO FATAL ERROR
HALT :INC DID NOT SET CC'S CORRECTLY
: OR SEQUENCE ERROR

2913
2914

:TEST 211 TEST DEC INSTRUCTION

```

*****
017666 005212          TST211: INC      (R2)          ;UPDATE TEST NUMBER
017670 022712 000211  CMP      #211,(R2)        ;SEQUENCE ERROR?
017674 001051          BNE      TST212-10       ;BR TO ERROR HALT ON SEQ ERROR
2915 017676 012700 000002 MOV      #2,R0          ;R0=2
2916 017702 000277          SCC                      ;CC=1111
2917 017704 005300          DEC      R0              ;CC=0001  R0-1
2918 017706 100403          BMI      DEC1
2919 017710 001402          BEQ      DEC1
2920 017712 102401          BVS      DEC1
2921 017714 103404          BCS      DEC2

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 767 <====

017716          DEC1:
017716 012742 000445  MOV      #445,-(R2)      ;MOVE TO MAILBOX # ***** 445 *****
017722 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
017724 000000          HALT
2922 017726 000261  DEC2:  SEC          ;DEC DID NOT SET CC'S CORRECTLY
2923 017730 000244          CLZ
2924 017732 005300          DEC      R0              ;CC=0101  R0-0
2925 017734 101002          BHI      DEC3
2926 017736 100401          BMI      DEC3
2927 017740 102004          BVC      DEC4

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====

017742          DEC3:
017742 012742 000446  MOV      #446,-(R2)      ;MOVE TO MAILBOX # ***** 446 *****
017746 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
017750 000000          HALT
2928 017752 000277  DEC4:  SCC          ;DEC DID NOT SET CC'S CORRECTLY
2929 017754 000251          +CLN,CLL
2930 017756 005300          DEC      R0              ;CC=1000  R0=177777
2931 017760 101402          BLOS     DEC5
2932 017762 102401          BVS      DEC5
2933 017764 100404          BMI      DEC6

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 743 <====

017766          DEC5:
017766 012742 000447  MOV      #447,-(R2)      ;MOVE TO MAILBOX # ***** 447 *****
017772 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
017774 000000          HALT
2934 017776 042700 077777  DEC6:  BIC      #77777,R0      ;DEC DID NOT SET CC'S CORRECTLY
2935 020002 000277          SCC          ;R0=100000
2936 020004 000252          +CLN,CLV          ;CC=0101
2937 020006 005300          DEC      R0              ;CC=1011  R0=77777
2938 020010 100403          BMI      DEC7          ;CC=0011
2939 020012 001402          BEQ      DEC7
2940 020014 102001          BVC      DEC7
2941 020016 103404          BCS      TST212

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====

```

```

:                                     REPLACE THE MOVE INSTRUCTION <====
:                                     WHICH FOLLOWS W/ 726 <====
020020                                DEC7:
020020 012742 000450                   MOV #450,-(R2) ;MOVE TO MAILBOX # ***** 450 *****
020024 005242                               INC -(R2) ;SET MSGTYP TO FATAL ERROR
020026 000000                               HALT ;DEC DID NOT SET CC'S CORRECTLY
:                                     ; OR SEQUENCE ERROR

```

2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953

```

:*****
:      THESE NEXT THREE TESTS VERIFY THE FUNCTIONING OF THE CLR,
:      TST, AND SWAB INSTRUCTIONS. THESE THREE INSTRUCTIONS ALL LEAVE
:      THE C AND V BITS CLEARED. AGAIN, THE CONDITION CODES ARE PRESET,
:      THE INSTRUCTION EXECUTED AND THE RESULTS CHECKED WITH CONDITIONAL
:      BRANCH INSTRUCTIONS. THE PROCEDURE IS REPEATED TO PRODUCE OTHER
:      COMBINATIONS OF CONDITION CODES.
:*****

```

```

:*****
:TEST 212      TEST CLR INSTRUCTION
:*****

```

```

020030 005212 000212  TST212: INC (R2) ;UPDATE TEST NUMBER
020032 022712                               CMP #212,(R2) ;SEQUENCE ERROR?
020036 001007                               BNE TST213-10 ;BR TO ERROR HALT ON SEQ ERROR
2954 020040 000277                               SCC ;CC=1011
2955 020042 000244                               CLZ
2956 020044 005000                               CLR R0 ;CC=0100 R0=0
2957 020046 100403                               BMI CLR1
2958 020050 102402                               BVS CLR1
2959 020052 103401                               BCS CLR1
2960 020054 001404                               BEQ TST213

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 770 <====

```

```

020056                                CLR1:
020056 012742 000451                   MOV #451,-(R2) ;MOVE TO MAILBOX # ***** 451 *****
020062 005242                               INC -(R2) ;SET MSGTYP TO FATAL ERROR
020064 000000                               HALT ;CLR DID NOT SET CC'S CORRECTLY
:                                     ; OR SEQUENCE ERROR

```

2961
2962

```

:*****
:TEST 213      TEST TST INSTRUCTION
:*****

```

```

020066 005212 000213  TST213: INC (R2) ;UPDATE TEST NUMBER
020070 022712                               CMP #213,(R2) ;SEQUENCE ERROR?
020074 001022                               BNE TST214-10 ;BR TO ERROR HALT ON SEQ ERROR
2963 020076 000277                               SCC ;CC=1011
2964 020100 000244                               CLZ
2965 020102 005700                               TST R0 ;CC=0100
2966 020104 100403                               BMI TEST1
2967 020106 102402                               BVS TEST1
2968 020110 103401                               BCS TEST1
2969 020112 001404                               BEQ TEST2

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====

```

```
020114 TEST1: MOV #452,-(R2) ; MOVE TO MAILBOX # ***** 452 *****  
020114 012742 000452 INC -(R2) ; SET MSGTYP TO FATAL ERROR  
020120 005242 HALT ; TEST DID NOT SET CC'S CORRECTLY  
020122 000000 ; MAKE R0 NEGATIVE  
2970 020124 005300 TEST2: DEC R0 ; CC=0111  
2971 020126 000277 SCC ;  
2972 020130 000250 CLN ;  
2973 020132 005700 TST R0 ; CC=1000  
2974 020134 101402 BLOS TEST3  
2975 020136 102401 BVS TEST3  
2976 020140 100404 BMI TST214
```

```
; WHICH FOLLOWS W/ 770 <===-  
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 755 <====
```

```
020142 TEST3: MOV #453,-(R2) ; MOVE TO MAILBOX # ***** 453 *****  
020142 012742 000453 INC -(R2) ; SET MSGTYP TO FATAL ERROR  
020146 005242 HALT ; TEST DID NOT SET CC'S CORRECTLY  
020150 000000 ; OR SEQUENCE ERROR
```

2977

```
*****  
:TEST 214 TEST SWAB INSTRUCTION  
*****
```

```
020152 TST214: INC (R2) ; UPDATE TEST NUMBER  
020154 022712 000214 CMP #214,(R2) ; SEQUENCE ERROR?  
020160 001023 BNE TST215-10 ; BR TO ERROR HALT ON SEQ ERROR  
2978 020162 012700 170000 MOV #170000,R0 ; R0=170000  
2979 020166 000277 SCC ; CC=0111  
2980 020170 000250 CLN ;  
2981 020172 000300 SWAB R0 ; CC=1000 R0-360  
2982 020174 101402 BLOS SWB1  
2983 020176 102401 BVS SWB1  
2984 020200 100404 BMI SWB2
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <===-  
; CONDITIONAL BRANCH INST. AND <===-  
; REPLACE THE MOVE INSTRUCTION <===-  
; WHICH FOLLOWS W/ 767 <====
```

```
020202 SWB1: MOV #454,-(R2) ; MOVE TO MAILBOX # ***** 454 *****  
020202 012742 000454 INC -(R2) ; SET MSGTYP TO FATAL ERROR  
020206 005242 HALT ; SWAB DID NOT SET CC'S CORRECTLY  
020210 000000 ; CC=1011  
2985 020212 000277 SWB2: SCC ;  
2986 020214 000244 CLZ ;  
2987 020216 000300 SWAB R0 ; CC=0100 R0-170000  
2988 020220 102403 BVS SWB3  
2989 020222 103402 BCS SWB3  
2990 020224 100401 BMI SWB3  
2991 020226 001404 BEQ TST215
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 754 <====
```

```
020230 SWB3: MOV #455,-(R2) ; MOVE TO MAILBOX # ***** 455 *****  
020230 012742 000455 INC -(R2) ; SET MSGTYP TO FATAL ERROR  
020234 005242 HALT ;  
020236 000000 ;
```

2992
 2993
 2994
 2995
 2996
 2997
 2998
 2999
 3000
 3001
 3002
 3003

```

:*****
:      THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE ADD AND
:ADC INSTRUCTIONS.  BOTH OF THESE INSTRUCTIONS HANDLE THE C AND
:V BITS IDENTICALLY.  THE PROCEDURE IS TO PRESET THE CONDITION
:CODES, EXECUTE THE INSTRUCTION WITH A PARTICULAR SET OF DATA, AND
:THEN CHECK THE RESULTS BY EXECUTING A SERIES OF CONDITIONAL
:BRANCHES.  THIS PROCEDURE IS REPEATED SEVERAL TIMES WITH DIFFERENT
:DATA TO PRODUCE EVERY COMBINATION OF C AND V BITS.
:*****
  
```

TEST 215 TEST ADD INSTRUCTION

020240 005212
 020242 022712 000215
 020246 001062
 3004 020250 012700 040000
 3005 020254 000277
 3006 020256 062700 030000
 3007 020262 101402
 3008 020264 102401
 3009 020266 100004

```

TST215: INC      (R2)          ;UPDATE TEST NUMBER
        CMP      #215,(R2)    ;SEQUENCE ERROR?
        BNE     TST216-10    ;BR TO ERROR HALT ON SEQ ERROR
        MOV     #40000,R0     ;R0=40000
        SCC     ;CC=1111
        ADD     #30000,R0     ;CC=0000 R0=70000
        BLOS   ADD1
        BVS    ADD1
        BPL    ADD2
  
```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 767 <====
  
```

020270
 020270 012742 000456
 020274 005242
 020276 000000
 3010 020300 000264
 3011
 3012 020302 062700 010000
 3013 020306 101402
 3014 020310 102001
 3015 020312 100404

```

ADD1:  MOV     #456,-(R2)    ;MOVE TO MAILBOX # ***** 456 *****
        INC     -(R2)        ;SET MSGTYP TO FATAL ERROR
        HALT
ADD2:  SEZ
        ADD     #10000,R0    ;CC=1010 40-100000
        BLOS   ADD3
        BVC    ADD3
        BMI    ADD4
  
```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 755 <====
  
```

020314
 020314 012742 000457
 020320 005242
 020322 000000
 3016 020324 000257
 3017 020326 000270
 3018 020330 062700 100000
 3019 020334 101002
 3020 020336 102001
 3021 020340 100004

```

ADD3:  MOV     #457,-(R2)    ;MOVE TO MAILBOX # ***** 457 *****
        INC     -(R2)        ;SET MSGTYP TO FATAL ERROR
        HALT
ADD4:  CCC
        SEN
        ADD     #100000,R0   ;CC=0111 R0=0
        BHI    ADD5
        BVC    ADD5
        BPL    ADD6
  
```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 742 <====
  
```

020342
 020342 012742 000460

```

ADD5:  MOV     #460,-(R2)    ;MOVE TO MAILBOX # ***** 460 *****
  
```



```

020346 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
020350 000000      HALT                    ;ADD DID NOT SET CC'S CORRECTLY
3022 020352 062700 177777  ADD6:  ADD      #177777,R0 ;CC=1000  R0=177777
3023 020356 101402      BLOS     ADD7
3024 020360 102401      BVS      ADD7
3025 020362 100404      BMI      ADD8

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 731 <====

020364 012742 000461  ADD7:  MOV      #461,-(R2) ;MOVE TO MAILBOX # ***** 461 *****
020370 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
020372 000000      HALT                    ;ADD DID NOT SET CC'S CORRECTLY
3026 020374 000277  ADD8:  SCC
3027 020376 000245      +CLC!CLZ ;CC=1010  R=0
3028 020400 062700 000001  ADD      #1,R0
3029 020404 102403      BVS      ADD9
3030 020406 103002      BCC      ADD9
3031 020410 100401      BMI      ADD9
3032 020412 001404      BEQ      TST216

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 715 <====

020414 012742 000462  ADD9:  MOV      #462,-(R2) ;MOVE TO MAILBOX # ***** 462 *****
020420 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
020422 000000      HALT                    ;ADD DID NOT SET CC'S CORRECTLY
; OR SEQUENCE ERROR

3033
3034
;*****
;TEST 216 TEST ADC INSTRUCTION
;*****
TST216: INC      (R2)      ;UPDATE TEST NUMBER
CMP      #216,(R2) ;SEQUENCE ERROR?
BNE      TST217-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV      #077777,R0
SCC
;CC=0101
+CLN!CLV ;CC=1010
ADC      R0
BLOS     ADC1
BVC      ADC1
BMI      ADC2

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
; CONDITIONAL BRANCH INST. AND <
; REPLACE THE MOVE INSTRUCTION <
; WHICH FOLLOWS W/ 767 <

020454 012742 000463  ADC1:  MOV      #463,-(R2) ;MOVE TO MAILBOX # ***** 463 *****
020460 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
020462 000000      HALT                    ;ADC DID NOT SET CC'S CORRECTLY
3042 020464 052700 077777  ADC2:  BIS      #77777,R0 ;CC=1011
3043 020470 000277      SCC
3044 020472 000244      CLZ
3045 020474 005500      ADC      R0 ;CC=0101  R0 0
3046 020476 101002      BMI      ADC3
  
```

3047 020500 102401
3048 020502 100004

BVS ADC3
BPL ADC4

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- - -
: CONDITIONAL BRANCH INST. AND <= - -
: REPLACE THE MOVE INSTRUCTION <- -
: WHICH FOLLOWS W/ 753 <- -

020504
020504 012742 000464

ADC3: MOV #464,-(R2)
INC -(R2)

:MOVE TO MAILBOX # ***** 464 *****
:SET MSGTYP TO FATAL ERROR
:ADC DID NOT SET CC'S CORRECTLY

3049 020514 000277

ADC4: SCC
+CLZ,CLC

:CC=1010
:CC=0100

3050 020516 000245
3051 020520 005500
3052 020522 102403
3053 020524 103402
3054 020526 100401
3055 020530 001404

ADC: RO
BVS ADC5
BCS ADC5
BMI ADC5
BEQ TST217

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
: CONDITIONAL BRANCH INST. AND <
: REPLACE THE MOVE INSTRUCTION <
: WHICH FOLLOWS W/ 740 <

020532
020532 012742 000465
020536 005242
020540 000000

ADC5: MOV #465,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 465 *****
:SET MSGTYP TO FATAL ERROR
:ADC DID NOT SET CC'S CORRECTLY
: OR SEQUENCE ERROR

3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067

: THESE NEXT THREE TESTS VERIFY THE FUNCTIONING OF THE NEG,
: CMP, AND COM INSTRUCTIONS. EACH OF THESE INSTRUCTIONS GENERATE
: THE C AND V BITS IDENTICALLY. THE CONDITION CODES ARE PRESET,
: THE INSTRUCTIONS EXECUTED, AND THE RESULTS CHECKED WITH A SERIES
: OF CONDITIONAL BRANCH INSTRUCTIONS. THIS PROCEDURE IS REPEATED
: SEVERAL TIMES WITH DIFFERENT DATA IN ORDER TO GENERATE DIFFERENT
: COMBINATIONS OF THE C AND V BITS.

: TEST 217 TEST NEG INSTRUCTION

020542 005212
020544 022712 000217
020550 001042
3068 020552 012700 000001
3069 020556 000277
3070 020560 000251
3071 020562 005400
3072 020564 103003
3073 020566 102402
3074 020570 001401
3075 020572 100404

TST217: INC (R2) :UPDATE TEST NUMBER
CMP #217,(R2) :SEQUENCE ERROR?
BNE TST220-10 :BR TO ERROR HALT ON SEQ ERROR
MOV #1,R0
SCC :CC=0110
+CLN,CLC :CC=1001 R0=177777
NEG RO
BCC NEG1
BVS NEG1
BEQ NEG1
BMI NEG2

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=
: CONDITIONAL BRANCH INST. AND <- -
: REPLACE THE MOVE INSTRUCTION <- -
: WHICH FOLLOWS W/ 766 <- -

020574
020574 012742 000466

NEG1: MOV #466,-(R2)

:MOVE TO MAILBOX # ***** 466 *****

```
020600 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
020602 000000 HALT ;NEG DID NOT SET CC'S CORRECTLY
3076 020604 042700 077777 NEG2: BIC #77777,R0
3077 020610 000257 CCC ;CC=0100
3078 020612 000264 SEZ
3079 020614 005400 NEG R0 ;CC=1011 R0=100000
3080 020616 102003 BVC NEG3
3081 020620 103002 BCC NEG3
3082 020622 001401 BEQ NEG3
3083 020624 100404 BMI NEG4
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <--==
; CONDITIONAL BRANCH INST. AND <---==
; REPLACE THE MOVE INSTRUCTION <- --
; WHICH FOLLOWS W/ 751 <---==
```

```
020626 012742 000467 NEG3: MOV #467,-(R2) ;MOVE TO MAILBOX # ***** 467 *****
020632 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
020634 000000 HALT ;NEG DID NOT SET CC'S CORRECTLY
3084 020636 005000 NEG4: CLR R0
3085 020640 000277 SCC ;CC=1011
3086 020642 000244 CLZ
3087 020644 005400 NEG R0 ;CC=0100 R0=0
3088 020646 102403 BVS NEG5
3089 020650 103402 BCS NEG5
3090 020652 001001 BNE NEG5
3091 020654 100004 BPL TST220
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 735 <====
```

```
020656 012742 000470 NEG5: MOV #470,-(R2) ;MOVE TO MAILBOX # ***** 470 *****
020662 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
020664 000000 HALT ;NEG DID NOT SET CC'S CORRECTLY
; OR SEQUENCE ERROR
```

3092
3093

```
*****  
;TEST 220 TEST CMP INSTRUCTION  
*****
```

```
TST220: INC (R2) ;UPDATE TEST NUMBER
020666 005212 000220 CMP #220,(R2) ;SEQUENCE ERROR?
020670 022712 001060 BNE TST221-10 ;BR TO ERROR HALT ON SEQ ERROR
3094 020676 012700 000005 MOV #5,R0
3095 020702 000257 CCC ;CC=1010
3096 020704 000271 +SEN!SEC
3097 020706 022700 000005 CMP #5,R0 ;CC=0101
3098 020712 101002 BHI CMP1
3099 020714 102401 BVS CMP1
3100 020716 100004 BPL CMP2
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
; CONDITIONAL BRANCH INST. AND < --=
; REPLACE THE MOVE INSTRUCTION <- -
; WHICH FOLLOWS W/ 766 <- -
```

```
020720 012742 000471 CMP1: MOV #471,-(R2) ;MOVE TO MAILBOX # ***** 471 *****
020720 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
020726 000000 HALT ;CMP DID NOT SET CC'S CORRECTLY
```

```

3101 020730 012700 100000      CMP2:  MOV      #100000,R0
3102 020734 000277              SCC
3103 020736 000242              CLV
3104 020740 020027 077777      CMP      R0,#77777      :CC=0010
3105 020744 101402              BLCs    CMP3
3106 020746 102001              BVC    CMP3
3107 020750 100004              BPL     CMP4

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 751 <====

      020752
      020752 012742 000472      CMP3:  MOV      #472,-(R2)      :MOVE TO MAILBOX # ***** 472 *****
      020756 005242              INC      -(R2)
      020760 000000              HALT
3108 020762 052700 040000      CMP4:  BIS      #40000,R0      :MOVE TO MAILBOX # ***** 472 *****
3109 020766 000257              CCC
3110 020770 000264              SEZ
3111 020772 022700 040000      CMP      #40000,R0      :SET MSGTYP TO FATAL ERROR
3112 020776 102003              BVC    CMP5              :CMP DID NOT SET CC'S CORRECTLY
3113 021000 103002              BCC    CMP5              :R0=140000
3114 021002 001401              BEQ    CMP5              :CC=0100
3115 021004 100404              BMT     CMP6

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 733 <====

      021006
      021006 012742 000473      CMP5:  MOV      #473,-(R2)      :MOVE TO MAILBOX # ***** 473 *****
      021012 005242              INC      -(R2)
      021014 000000              HALT
3116 021016 042700 040000      CMP6:  BIC      #40000,R0      :MOVE TO MAILBOX # ***** 473 *****
3117 021022 000277              SCC
3118 021024 022700 177777      CMP      #-1,R0          :SET MSGTYP TO FATAL ERROR
3119 021030 101402              BLOS   CMP7              :CMP DID NOT SET CC'S CORRECTLY
3120 021032 102401              BVS    CMP7              :CC=1111
3121 021034 100004              BPL     TST221           :CC=0000

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 717 <====

      021036
      021036 012742 000474      CMP7:  MOV      #474,-(R2)      :MOVE TO MAILBOX # ***** 474 *****
      021042 005242              INC      -(R2)
      021044 000000              HALT
; OR SEQUENCE ERROR
  
```

3122
3123

```

*****
;TEST 221 TEST COM INSTRUCTION
*****
TST221: INC      (R2)          :UPDATE TEST NUMBER
          CMP      #221,(R2)   :SEQUENCE ERROR?
          BNE     TST222-10    :BR TO ERROR HALT ON SEQ ERROR
          MOV     #-1,R0
          CCC
          +SEC!SEZ
          COM     R0           :CC=1010
          :CC=0101
  
```

```

      021046 005212
      021050 022712 000221
      021054 001010
3124 021056 012700 177777
3125 021062 000257
3126 021064 000265
3127 021066 005100
  
```

3128 021070 101002
3129 021072 102401
3130 021074 100004

BHI COM1
BVS COM1
BPL TST222

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 767 <====

021076
021076 012742 000475
021102 005242
021104 000000

COM1: MOV #475,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 475 *****
:SET MSGTYP TO FATAL ERROR
:COM DID NOT SET CC'S CORRECTLY
: OR SEQUENCE ERROR

3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143

: THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE SUB
: AND SBC INSTRUCTIONS. BOTH OF THESE INSTRUCTIONS HANDLE THE
: C AND V BITS IDENTICALLY. THE PROCEDURE IS TO PRESET THE CONDITION
: CODES, EXECUTE THE INSTRUCTION WITH A PARTICULAR SET OF DATA, AND
: THEN CHECK THE RESULTS BY EXECUTING A SERIES OF CONDITIONAL
: BRANCHES. THIS PROCEDURE IS REPEATED SEVERAL TIMES WITH DIFFERENT
: DATA PATTERNS TO PROVIDE EVERY COMBINATION OF THE C AND V BITS.

TEST 222 TEST SUB INSTRUCTION

021106 005212
021110 022712 000222
021114 001055
3144 021116 012700 125252
3145 021122 000257
3146 021124 000271
3147 021126 162700 125252
3148 021132 101002
3149 021134 102401
3150 021136 100004

TST222: INC (R2) ;UPDATE TEST NUMBER
CMP #222,(R2) ;SEQUENCE ERROR?
BNE TST223-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #125252,R0
CCC ;CC=1010
+SEN!SEC
SUB #125252,R0 ;CC=0101 R0=0
BHI SUB1
BVS SUB1
BPL SUB2

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 766 <====

021140
021140 012742 000476
021144 005242
021146 000000
3151 021150 052700 100000
3152 021154 000277
3153 021156 000242
3154 021160 162700 077777
3155 021164 101402
3156 021166 102001
3157 021170 100004

SUB1: MOV #476,-(R2)
INC -(R2)
HALT
SUB2: BIS #100000,R0
SCC
CLV
SUB #77777,R0
BLOS SUB3
BVC SUB3
BPL SUB4

:MOVE TO MAILBOX # ***** 476 *****
:SET MSGTYP TO FATAL ERROR
:SUB DID NOT SET CC'S CORRECTLY
:CC=1101
:CC=0010 R0=1

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 751 <====

021172

SUB3:

```

021172 012742 000477      MOV      #477,-(R2)      ;MOVE TO MAILBOX # ***** 477 *****
021176 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
021200 000000              HALT
3158 021202 005100      SUB4:  COM      R0          ;R0=177777
3159 021204 000277      SCC
3160
3161 021206 162700 100000  SUB      #100000,R0      ;CC=0000 R0=77777
3162 021212 101402      BLOS    SUB5
3163 021214 102401      BVS     SUB5
3164 021216 100004      BPL     SUB6

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 736 <====

```

```

021220 012742 000500      SUB5:  MOV      #500,-(R2)  ;MOVE TO MAILBOX # ***** 500 *****
021220 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
021226 000000              HALT
3165 021230 000257      SUB6:  CCC
3166 021232 000264      SEZ
3167 021234 162700 140000  SUB      #140000,R0      ;CC=1011
3168 021240 102003      BVC    SUB7
3169 021242 103002      BCC    SUB7
3170 021244 001401      BEQ    SUB7
3171 021246 100404      BMI    TST223

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <==
; CONDITIONAL BRANCH INST. AND <- -
; REPLACE THE MOVE INSTRUCTION <
; WHICH FOLLOWS W/ 722 <

```

```

021250 012742 000501      SUB7:  MOV      #501,-(R2)  ;MOVE TO MAILBOX # ***** 501 *****
021250 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
021256 000000              HALT

```

3172
3173

```

;*****
;TEST 223 TEST SBC INSTRUCTION
;*****

```

```

021260 005212 000223      TST223: INC      (R2)          ;UPDATE TEST NUMBER
021262 022712              CMP      #223,(R2)      ;SEQUENCE ERROR?
021266 001053              BNE     TST224-10      ;BR TO ERROR HALT ON SEQ ERROR
3174 021270 012700 000001  MOV      #1,R0
3175 021274 000277      SCC
3176 021276 000244      CLZ
3177 021300 005600      SBC     R0              ;CC=0100 R=0
3178 021302 103403      BCS    SBC1
3179 021304 102402      BVS    SBC1
3180 021306 100401      BMI    SBC1
3181 021310 001404      BEQ    SBC2

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 766 <====

```

```

021312 012742 000502      SBC1:  MOV      #502,-(R2)  ;MOVE TO MAILBOX # ***** 502 *****
021312 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
021316 000000              HALT
3182 021322 000277      SBC2:  SCC

```

```

3183 021324 000245      +CLZ.CLC
3184 021326 005600      SBC      R0          ;CC=010C  R=0
3185 021330 103403      BCS      SBC3
3186 021332 102402      BVS      SBC3
3187 021334 100401      BMI      SBC3
3188 021336 001404      BEQ      SBC4

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
; CONDITIONAL BRANCH INST. AND           <====
; REPLACE THE MOVE INSTRUCTION          <====
; WHICH FOLLOWS W/ 753                   <====

      021340
      021340 012742 000503  SBC3:  MOV      #503,-(R2)  ;MOVE TO MAILBOX # ***** 503 *****
      021344 005242          INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
      021346 000000          HALT
3189 021350 000277      SBC4:  SCC
3190 021352 000250      CLN
3191 021354 005600      SBC      R0          ;CC=1001  R0=177777
3192 021356 103003      BCC      SBC5
3193 021360 102402      BVS      SBC5
3194 021362 001401      BEQ      SBC5
3195 021364 100404      BMI      SBC6

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
; CONDITIONAL BRANCH INST. AND           <====
; REPLACE THE MOVE INSTRUCTION          <====
; WHICH FOLLOWS W/ 740                   <====

      021366
      021366 012742 000504  SBC5:  MOV      #504,-(R2)  ;MOVE TO MAILBOX # ***** 504 *****
      021372 005242          INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
      021374 000000          HALT
3196 021376 042700 077777  SBC6:  BIC      #77777,R0  ;RO=100000
3197 021402 000277      SCC
3198 021404 000242      CLV
3199 021406 005600      SBC      R0          ;CC=0010
3200 021410 101402      BLOS     SBC7
3201 021412 102001      BVC      SBC7
3202 021414 100004      BPL      TST224

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS  <====
; CONDITIONAL BRANCH INST. AND           <====
; REPLACE THE MOVE INSTRUCTION          <====
; WHICH FOLLOWS W/ 724                   <====

      021416
      021416 012742 000505  SBC7:  MOV      #505,-(R2)  ;MOVE TO MAILBOX # ***** 505 *****
      021422 005242          INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
      021424 000000          HALT
; OR SEQUENCE ERROR

```

3203
 3204
 3205
 3206
 3207
 3208
 3209
 3210
 3211
 3212
 3213
 3214

```

;*****
;
; THESE NEXT FOUR TESTS VERIFY THE FUNCTIONING OF THE ROL,
; ROR, ASL AND ASR INSTRUCTIONS. SPECIAL DATA PATTERNS ARE LOADED
; AND ROTATED SEVERAL TIMES FOR EACH TEST. THE CONDITION CODES
; ARE PRESET BEFORE EACH ROTATION AND THE CONDITION CODES ARE
; CHECKED AFTER EACH ROTATION. THE FINAL CHECK IN EACH TEST IS
; TO VERIFY THE CUMULATIVE DATA RESULT. THE DATA PATTERNS HAVE
; BEEN SELECTED TO PRODUCE ALL COMBINATIONS OF THE C AND V BITS.
;*****

```

:TEST 224 TEST ROL INSTRUCTION

021426 005212
021430 022712 000224
021434 001053
3215 021436 012700 144000
3216 021442 000257
3217 021444 000266
3218 021446 006100
3219 021450 103003
3220 021452 102402
3221 021454 001401
3222 021456 100404

TST224: INC (R2)
CMP #224,(R2)
BNE TST225-10
MOV #144000,R0
CCC
+SEZ!SEV
ROL R0
BCC ROL1
BVS ROL1
BEQ ROL1
BMI ROL2

;UPDATE TEST NUMBER
;SEQUENCE ERROR?
;BR TO ERROR HALT ON SEQ ERROR
;R0=144000
;CC=0110
;CC=1001 R0=110000

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 766 <====

021460
021460 012742 000506
021464 005242
021466 000000
3223 021470 000277
3224 021472 000243
3225 021474 006100
3226 021476 103003
3227 021500 102002
3228 021502 001401
3229 021504 100004

ROL1: MOV #506,-(R2)
INC -(R2)
HALT
ROL2: SCC
+CLV!CLC
ROL R0
BCC ROL3
BVC ROL3
BEQ ROL3
BPL ROL4

;MOVE TO MAILBOX # ***** 506 *****
;SET MSGTYP TO FATAL ERROR
;CC=1100
;CC=0011 R0=020000

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 753 <====

021506
021506 012742 000507
021512 005242
021514 000000
3230 021516 000277
3231 021520 000250
3232 021522 006100
3233 021524 101402
3234 021526 102401
3235 021530 100004

ROL3: MOV #507,-(R2)
INC -(R2)
HALT
ROL4: SCC
CLN
ROL R0
BLOS ROL5
BVS ROL5
BPL ROL6

;MOVE TO MAILBOX # ***** 507 *****
;SET MSGTYP TO FATAL ERROR
;ROL DID NOT SET CC'S CORRECTLY
;CC=0111
;CC=0000 R0=040001

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 741 <====

021532
021532 012742 000510
021536 005242
021540 000000
3236 021542 000257
3237 021544 000265
3238 021546 006100
3239 021550 101405
3240 021552 102004
3241 021554 100003
3242 021556 022700 100003

ROL5: MOV #510,-(R2)
INC -(R2)
HALT
ROL6: CCC
+SEZ!SEC
ROL R0
BLOS ROL7
BVC ROL7
BPL ROL7
CMP #100003,R0

;MOVE TO MAILBOX # ***** 510 *****
;SET MSGTYP TO FATAL ERROR
;ROL DID NOT SET CC'S CORRECTLY
;CC=0101
;CC=1010 R0=100003


```

3243 021562 001404          BEQ      TST225          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 724 <----

      021564          ROL7:
021564 012742 000511      MOV      #511,-(R2)      ;MOVE TO MAILBOX # ***** 511 *****
021570 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
021572 000000          HALT                    ;ROL MALFUNCTIONED
;                                     OR SEQUENCE ERROR

3244          ;*****
;TEST 225      TEST ROR INSTRUCTION
;*****
021574 005212          TST225: INC      (R2)          ;UPDATE TEST NUMBER
021576 022712 000225      CMP      #225,(R2)      ;SEQUENCE ERROR?
021602 001051          BNE      TST226-10      ;BR TO ERROR HALT ON SEQ ERROR
3245 021604 012700 000023  MOV      #23,R0          ;R0=23
3246 021610 000277          SCC                    ;CC=0111
3247 021612 000250          CLN                    ;
3248 021614 006000          ROR      R0          ;CC=1001 R0-100011
3249 021616 102403          BVS      ROR1          ;
3250 021620 103002          BCC      ROR1          ;
3251 021622 001401          BEQ      ROR1          ;
3252 021624 100404          BMI      ROR2          ;

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <----
;                                     CONDITIONAL BRANCH INST. AND <----
;                                     REPLACE THE MOVE INSTRUCTION <----
;                                     WHICH FOLLOWS W/ 766 <----

      021626          ROR1:
021626 012742 000512      MOV      #512,-(R2)      ;MOVE TO MAILBOX # ***** 512 *****
021632 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
021634 000000          HALT                    ;ROR DID NOT SET CC'S CORRECTLY
3253 021636 000257          ROR2: CCC          ;CC=1100
3254 021640 000274          +SEN!SEZ
3255 021642 006000          ROR      R0          ;CC=0011 R0 040004
3256 021644 102003          BVC      ROR3          ;
3257 021646 105002          BCC      ROR3          ;
3258 021650 001401          BEQ      ROR3          ;
3259 021652 100004          BPL      ROR4          ;

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;                                     CONDITIONAL BRANCH INST. AND <====
;                                     REPLACE THE MOVE INSTRUCTION <====
;                                     WHICH FOLLOWS W/ 753 <====

      021654          ROR3:
021654 012742 000513      MOV      #513,-(R2)      ;MOVE TO MAILBOX # ***** 513 *****
021660 005242          INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
021662 000000          HALT                    ;ROR DID NOT SET CC'S CORRECTLY
3260 021664 000277          ROR4: SCC          ;CC=1110
3261 021666 000241          CLC                    ;
3262 021670 006000          ROR      R0          ;CC=0000 R0-020002
3263 021672 101403          BLOS     ROR5          ;
3264 021674 102402          BVS      ROR5          ;
3265 021676 001401          BEQ      ROR5          ;
3266 021700 100004          BPL      ROR6          ;

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <----
;                                     CONDITIONAL BRANCH INST. AND <----
;                                     REPLACE THE MOVE INSTRUCTION <----
  
```

```

021702 ROR5: MOV #514,-(R2) ;MOVE TO MAILBOX # ***** 514 *****
021702 012742 000514 INC -(R2) ;SET MSGTYP TO FATAL ERROR
021706 005242 HALT ;ROR DID NOT SET CC'S CORRECTLY
3267 021712 000000 ROR6: CCC ;CC=0101
3268 021714 000257 +SEC.SEZ ;CC=1010 R0=110001
3269 021716 006000 ROR RO
3270 021720 101402 BLOS ROR/
3271 021722 102001 BVC ROR7
3272 021724 100404 BMI TST226

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
 : CONDITIONAL BRANCH INST. AND
 : REPLACE THE MOVE INSTRUCTION
 : WHICH FOLLOWS W/ 740 <=====
 <=====
 <=====
 <=====

```

021726 ROR7: MOV #515,-(R2) ;MOVE TO MAILBOX # ***** 515 *****
021726 012742 000515 INC -(R2) ;SET MSGTYP TO FATAL ERROR
021732 005242 HALT ;ROR DID NOT PRODUCE CORRECT RESULTS
021734 000000 ; OR SEQUENCE ERROR

```

3273 :*****
 :TEST 226 TEST ASL INSTRUCTION
 :*****

```

021736 TST226: INC (R2) ;UPDATE TEST NUMBER
021740 CMP #226,(R2) ;SEQUENCE ERROR?
021744 BNE TST227-10 ;BR TO ERROR HALT ON SEQ ERROR
3274 021746 012700 144000 MOV #144000,R0 ;R0=14000
3275 021752 000257 CCC ;CC=0110
3276 021754 000271 +SEN!SEC ;CC=1001 R0=110000
3277 021756 006300 ASL RO
3278 021760 103003 BCC ASL1
3279 021762 102402 BVS ASL1
3280 021764 001401 BEQ ASL1
3281 021766 100404 BMI ASL2

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
 : CONDITIONAL BRANCH INST. AND
 : REPLACE THE MOVE INSTRUCTION
 : WHICH FOLLOWS W/ 726 <=====
 <=====
 <=====
 <=====

```

021770 ASL1: MOV #516,-(R2) ;MOVE TO MAILBOX # ***** 516 *****
021770 012742 000516 INC -(R2) ;SET MSGTYP TO FATAL ERROR
021774 005242 HALT
3282 022000 000277 ASL2: SCC ;CC=1100
3283 022002 000243 +CLV!CLC ;CC=0011 R0=020000
3284 022004 006300 ASL RO
3285 022006 103003 BCC ASL3
3286 022010 102002 BVC ASL3
3287 022012 001401 BEQ ASL3
3288 022014 100004 BPL ASL4

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
 : CONDITIONAL BRANCH INST. AND
 : REPLACE THE MOVE INSTRUCTION
 : WHICH FOLLOWS W/ 766 <=====
 <=====
 <=====
 <=====

```

022016 ASL3: MOV #517,-(R2) ;MOVE TO MAILBOX # ***** 517 *****
022016 012742 000517 INC -(R2) ;SET MSGTYP TO FATAL ERROR
022022 005242 HALT ;ASL DID NOT SET CC'S CORRECTLY
022024 000000

```

```
3289 022026 000277      ASL4:  SCC                ;CC=0111
3290 022030 000250      CLN
3291 022032 006300      ASL    R0                ;CC=0000  R0=040000
3292 022034 101402      BLOS   ASL5
3293 022036 102401      BVS    ASL5
3294 022040 100004      BPL    ASL6
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 741 <====

      022042
      022042 012742 000520      ASL5:  MOV    #520,-(R2)    ;MOVE TO MAILBOX # ***** 520 *****
      022046 005242            INC    -(R2)            ;SET MSGTYP TO FATAL ERROR
      022050 000000            HALT                   ;ASL DID NOT SET CC'S CORRECTLY
3295 022052 000257      ASL6:  CCC                ;CC=0101
3296 022054 000265      +SEZ!SEC
3297 022056 006300      ASL    R0                ;CC=1010  R0=100000
3298 022060 103406      BCS    ASL7
3299 022062 001405      BEQ    ASL7
3300 022064 102004      BVC    ASL7
3301 022066 100003      BPL    ASL7
3302 022070 022700 100000      CMP    #100000,R0
3303 022074 001404      BEQ    TST227
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 723 <====

      022076
      022076 012742 000521      ASL7:  MOV    #521,-(R2)    ;MOVE TO MAILBOX # ***** 521 *****
      022102 005242            INC    -(R2)            ;SET MSGTYP TO FATAL ERROR
      022104 000000            HALT                   ;ASL MALFUNCTIONED
; OR SEQUENCE ERROR
3304
;*****
;TEST 227      TEST ASR INSTRUCTION
;*****
      022106 005212            TST227: INC    (R2)            ;UPDATE TEST NUMBER
      022110 022712 000227      CMP    #227,(R2)        ;SEQUENCE ERROR?
      022114 001060            BNE    TST230-10        ;BR TO ERROR HALT ON SEQ ERROR
3305 022116 012700 100023      MOV    #100023,R0      ;R0=100023
3306 022122 000277            SCC                ;CC=0110
3307 022124 000250            CLN
3308 022126 006200            ASR    R0                ;CC=1001  RP=140011
3309 022130 102403            BVS    ASR1
3310 022132 103002            BCC    ASR1
3311 022134 001401            BEQ    ASR1
3312 022136 100404            BMI    ASR2
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 766 <====

      022140
      022140 012742 000522      ASR1:  MOV    #522,-(R2)    ;MOVE TO MAILBOX # ***** 522 *****
      022144 005242            INC    -(R2)            ;SET MSGTYP TO FATAL ERROR
      022146 000000            HALT                   ;ASR DID NOT SET CC'S CORRECTLY
3313 022150 042700 100000      ASR2:  BIC    #100000,R0  ;R0=40011
3314 022154 000277            SCC                ;CC=1100
3315 022156 000243            +CLV.CLC
```

```

3316 022160 006200      ASR      R0      ;CC=0011  R0=020004
3317 022162 102003      BVC      ASR3
3318 022164 103002      BCC      ASR3
3319 022166 001401      BEQ      ASR3
3320 022170 100C04      BPL      ASR4

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 751 <====

      022172      ASR3:
022172 012742 000523      MOV      #523,-(R2) ;MOVE TO MAILBOX # ***** 523 *****
022176 005242      INC      -(R2)   ;SET MSGTYP TO FATAL ERROR
022200 000000      HALT
3321 022202 000277      ASR4:  SCC      ;ASR DID NOT SET CC'S CORRECTLY
3322                                ;CC=1111
3323 022204 006200      ASR      R0      ;CC=0000  R0=010002
3324 022206 101403      BLOS     ASR5
3325 022210 102402      BVS      ASR5
3326 022212 001401      BEQ      ASR5
3327 022214 100004      BPL      ASR6

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 737 <====

      022216      ASR5:
022216 012742 000524      MOV      #524,-(R2) ;MOVE TO MAILBOX # ***** 524 *****
022222 005242      INC      -(R2)   ;SET MSGTYP TO FATAL ERROR
022224 000000      HALT
3328 022226 052700 100000 ASR6:  BIS      #100000,R0 ;ASR DID NOT SET CC'S CORRECTLY
3329 022232 000257      CCC
3330 022234 000265      +SEZ,SEC ;R0=110002
3331 022236 006200      ASR      R0      ;CC=0101
3332 022240 101406      BLOS     ASR7      ;C=1010  R0=144001
3333 022242 102005      SVC      ASR7
3334 022244 100004      BPL      ASR7
3335 022246 001403      BEQ      ASR7
3336 022250 022700 144001 CMP      #144001,R0 ;CHECK RESULT OF ASR'S
3337 022254 001404      BEQ      TST230

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 717 <====

      022256      ASR7:
022256 012742 000525      MOV      #525,-(R2) ;MOVE TO MAILBOX # ***** 525 *****
022262 005242      INC      -(R2)   ;SET MSGTYP TO FATAL ERROR
022264 000000      HALT      ;ASR DID NOT FUNCTION CORRECTLY
; OR SEQUENCE ERROR

```

3344
 3345
 3346
 3347
 3348
 3349
 3350
 3351
 3352
 3353

```

:*****
:
: THIS TEST VERIFIES THE SXT INSTRUCTION. CONDITION CODES
: ARE PRESET IN EACH OF THE TWO POSSIBLE CASES. WITH THE N-BIT SET,
: THE TEST CHECKS FOR ALL ONES IN THE DESTINATION. WITH THE N-BIT
: CLEAR, THE DESTINATION SHOULD CONTAIN ALL ZEROES. THE DATA
: IS VERIFIED BY CONDITIONAL BRANCHES.

```

3354
 3355

```

:*****
:TEST 230      TEST THE SXT INSTRUCTION
:*****
  
```

022266 005212
 022270 022712 000230
 022274 001033
 3356 022276 005000
 3357 022300 000277
 3358 022302 000244
 3359 022304 006700
 3360 022306 100006
 3361 022310 001405
 3362 022312 102404
 3363 022314 103003
 3364 022316 022700 177777
 3365 022322 001404

```

TST230: INC      (R2)           ;UPDATE TEST NUMBER
        CMP      #230,(R2)      ;SEQUENCE ERROR?
        BNE     TST231-10      ;BR TO ERROR HALT ON SEQ ERROR
        CLR     R0
        SCC     ;SET CC-1011
        CLZ
        SXT     R0             ;TRY SXT
        BPL     SXT0           ;TEST CC-1001
        BEQ     SXT0
        BVS     SXT0
        BCC     SXT0
        CMP     #-1,R0         ;CHECK DATA RESULT
        BEQ     SXT1
        ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
        ;          CONDITIONAL BRANCH INST. AND <====
        ;          REPLACE THE MOVE INSTRUCTION <====
        ;          WHICH FOLLOWS W/ 764 <====
  
```

022324
 022324 012742 000526
 022330 005242
 022332 000000
 3366 022334 005000
 3367 022336 005010
 3368 022340 005110
 3369 022342 000257
 3370 022344 000266
 3371 022346 006710
 3372 022350 001005
 3373 022352 103404
 3374 022354 102403
 3375 022356 100402
 3376 022360 005710
 3377 022362 001404

```

SXT0:   MOV     #526,-(R2)      ;MOVE TO MAILBOX # ***** 526 *****
        INC     -(R2)
        HALT
SXT1:   CLR     R0              ;RO=0
        CLR     (R0)           ;LOC. 0=0
        COM     (R0)           ;LOC. 0=177777
        CCC
        +SEZ!SEV
        SXT     (R0)
        BNE     SXT2           ;TEST CC=0100
        BCS     SXT2
        BVS     SXT2
        BMI     SXT2
        TST     (R0)
        BEQ     TST231
        ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
        ;          CONDITIONAL BRANCH INST. AND <====
        ;          REPLACE THE MOVE INSTRUCTION <====
        ;          WHICH FOLLOWS W/ 744 <====
  
```

022364
 022364 012742 000527
 022370 005242
 022372 000000

```

SXT2:   MOV     #527,-(R2)      ;MOVE TO MAILBOX # ***** 527 *****
        INC     -(R2)
        HALT
        ; SET MSGTYP TO FATAL ERROR
        ; RESULTS OF SXT INCORRECT
        ; OR SEQUENCE ERROR
  
```

3378
 3379
 3380
 3381
 3382
 3383
 3384
 3385
 3386

```

:*****
:      THIS TEST VERIFIES THE XOR INSTRUCTION. UNIQUE PATTERNS
:OF ONES AND ZEROES ARE MOVED TO DATA REGISTERS R0 AND R1.
: AFTER THE FIRST XOR INSTRUCTION R0=36146. AN XOR IS THEN
: EXECUTED WITH THIS NEW VALUE AND THE CONTENTS OF R1 TO
: REPRODUCE THE ORIGINAL VALUE IF R0=31525.
:*****
:TEST 231      TEST THE XOR INSTRUCTION
:*****
  
```

```

022374 005212          TST231: INC      (R2)          ;UPDATE TEST NUMBER
022376 022712 000231  CMP      #231,(R2)       ;SEQUENCE ERROR?
022402 001035          BNE     TST232-10       ;BR TO ERROR HALT ON SEQ ERROR
3387 022404 012700 007463  MOV     #7463,R0        ;SET UP R0
3388 022410 012701 031525  MOV     #31525,R1       ;SET UP R1
3389 022414 000277          SCC                      ;SET CC-1110
3390 022416 000241          CLC
3391 022420 074100          XOR     R1,R0           ;TRY XOR
3392 022422 101406          BLOS   XOR1            ;CC=0000?
3393 022424 102405          BVS   XOR1
3394 022426 001404          BEQ   XOR1
3395 022430 100403          BMI   XOR1
3396 022432 022700 036146  CMP     #36146,R0       ;DATA RESULT CORRECT?
3397 022436 001404          BEQ   XOR2

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 761 <====

```

```

022440          XOR1:  MOV     #530,-(R2)       ;MOVE TO MAILBOX # ***** 530 *****
022440 012742 000530  INC     -(R2)         ;SET MSGTYP TO FATAL ERROR
022444 005242          HALT
022446 000000          XOR2:  MOV     R1,R4
3398 022450 010104          SEC
3399 022452 000261          CLC
3400 022454 000241          XOR     R4,R0           ;TRY XOR MODE 0,0
3401 022456 074400          BLOS   XOR3            ;CC=0000?
3402 022460 101406          BVS   XOR3
3403 022462 102405          BEQ   XOR3
3404 022464 001404          BMI   XOR3
3405 022466 100403          CMP     #7463,R0
3406 022470 022700 007463  BEQ     TST232
3407 022474 001404

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 742 <====

```

```

022476          XOR3:  MOV     #531,-(R2)       ;MOVE TO MAILBOX # ***** 531 *****
022476 012742 000531  INC     -(R2)         ;SET MSGTYP TO FATAL ERROR
022502 005242          HALT
022504 000000          ;RESULT OF XOR INCORRECT
; OR SEQUENCE ERROR

```

3408
3409
3410
3411
3412
3413
3414
3415

```

:*****
: THIS TEST VERIFIES THE SOB INSTRUCTION. R4 IS USED AS A
: COUNTER WHILE R0 IS THE ADDRESS REGISTER. CONDITIONAL
: BRANCHES ARE USED TO VERIFY PROPER TRANSFER OF CONTROL
: WHILE R4 IS CHECKED TO INSURE PROPER DECREMENTING OF R0.
:*****
:TEST 232 TEST SOB INSTRUCTION
:*****

```

```

022506 005212          TST232: INC      (R2)          ;UPDATE TEST NUMBER
022510 022712 000232  CMP     #232,(R2)       ;SEQUENCE ERROR?
022514 001023          BNE     TST233-10       ;BR TO ERROR HALT ON SEQ ERROR
3416 022516 012700 000525  MOV     #525,R0
3417 022522 010004          MOV     R0,R4
3418 022524 000277          SCC                      ;SET CC=1111

```

```

3419 022526 101002 SOB1: BHI SOB2 ;CC=1111?
3420 022530 100001 BPL SOB2
3421 022532 102404 BVS SOB3
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 770 <====

022534 SOB2:
022534 012742 000532 MOV #532,-(R2) ;MOVE TO MAILBOX # ***** 532 *****
022540 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
022542 000000 HALT
3422 022544 005304 SOB3: DEC R4 ;COUNT ITERATIONS
3423 022546 000277 SCC ;CC=1111
3424 022550 077012 SOB RO,SOB1 ;DO SOB W/ RO
3425 022552 101004 BHI SOB4 ;CHECK CC=1111
3426 022554 100003 BPL SOB4
3427 022556 102002 BVC SOB4
3428 022560 005704 TST R4 ;ITERATION COUNT OK?
3429 022562 001404 BEQ TST233
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 754 <====

022564 SOB4:
022564 012742 000533 MOV #533,-(R2) ;MOVE TO MAILBOX # ***** 533 *****
022570 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
022572 000000 HALT ;INCORRECT # OF BRANCHES OR CC'S CHANGED
; OR SEQUENCE ERROR

3430 :*****
3431 :
3432 : THIS TEST VERIFIES THE MARK INSTRUCTION. THE EFFECTS
3433 : OF THE MARK INSTRUCTION ARE SIMULATED BY THE PROGRAM INSTRUCTIONS.
3434 : THE CONTENTS OF R5 AND THE STACK POINTER ARE CHECKED AFTER EACH
3435 : OF THE TWO ROUTINES IN THE TEST.
3436 :
3437 :*****
;TEST 233 TEST MARK INSTRUCTION
;*****
TST233: INC (R2) ;UPDATE TEST NUMBER
022576 022712 000233 CMP #233,(R2) ;SEQUENCE ERROR?
022602 001062 BNE TST234-10 ;BR TO ERROR HALT ON SEQ ERROR
3438 022604 012706 001000 MOV #STBOT,SP
3439 022610 012746 125252 MOV #125252,-(SP) ;PUT R5 VALJE ON STACK
3440 022614 162706 000074 SUB #74,SP ;EFFECTIVELY PUT 36 ARGUMENTS ON STACK
3441 022620 012705 022646 MOV #MRK1,R5 ;SET NEW PC IN R5
3442 022624 012746 006436 MOV #6436,-(SP) ;PUT MARK 36 INST. ON STACK
3443 022630 000277 SCC ;SET CC=1111
3444 022632 000137 000700 JMP @#700 ;XFER CONTL TO MARK 36 INST. ON STACK
3445 022636 012742 000534 MOV #534,-(R2) ;MOVE TO MAILBOX # ***** 534 *****
022642 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
022644 000000 HALT ;MARK INST. SHOULD HAVE JUMPED TO MRK1
3446 022646 101010 MRK1: BHI MRK2 ;TEST CC UNAFFECTED
3447 022650 100007 BPL MRK2 ;IE. CC=1111
3448 022652 102006 BVC MRK2
3449 022654 020527 125252 CMP R5,#125252 ;CHECK R5 RESTORED FROM STACK
3450 022660 001003 BNE MRK2
3451 022662 022706 001000 CMP #STBOT,R6 ;CHECK STACK POINTER READJUSTED CORRECTL'.
```

3452 022666 001404

BEQ MRK3

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 745 <====

022670

MRK2:

022670 012742 000535

MOV #535,-(R2)

:MOVE TO MAILBOX # ***** 535 *****

022674 005242

INC -(R2)

:SET MSGTYP TO FATAL ERROR

022676 000000

HALT

:RESULTS OF MARK INCORRECT

3453 022700 012746 052525

MRK3:

MOV #52525,-(SP)

:PUT MARK 0 INST. ON STACK

3454 022704 012746 006400

MOV #6400,-(SP)

:SET ADDR. OF MARK INST. IN R5

3455 022710 010605

MOV SP,R5

:DO JSR

3456 022712 004737 022722

JSR PC,@MRK4

:DO JSR

3457 022716 000137 022734

JMP @MRK5

:

3458 022722 000205

MRK4:

RTS R5

:DO RTS WITH R5 TO MARK INST ON STACK

3459 022724 012742 000536

MOV #536,-(R2)

:MOVE TO MAILBOX # ***** 536 *****

022730 005242

INC -(R2)

:SET MSGTYP TO FATAL ERROR

022732 000000

HALT

:RTS,MARK SEQUENCE FAILED

3460 022734 022706 001000

MRK5:

CMP #STBOT,R6

:STACK ADJUSTED CORRECTLY

3462 022740 001003
3463 022742 022705 052525
3464 022746 001404

BNE MRK6
CMP #52525,R5
BEQ TST234

:IF NOT: BR
:CHECK IF R5 RESTORED FROM STACK

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 715 <====

022750
022750 012742 000537
022754 005242
022756 000000

MRK6:

MOV #537,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 537 *****
:SET MSGTYP TO FATAL ERROR
:RESULTS OF MARK INCORRECT
: OR SEQUENCE ERROR

3465
3466
3467
3468
3469
3470
3471
3472
3473

: THIS TEST VERIFIES THAT RESET DOES NOT CLEAR THE PSW.
: THE PSW IS LOADED WITH ONES, A RESET IS ISSUED, AND THE
: CONTENTS OF THE PSW ARE CHECKED TO VERIFY THAT THEY HAVE NOT
: CHANGED. THIS TEST IS EXECUTED ONLY ONCE EVERY 256 (DECIMAL)
: PASSES.

: TEST 234 TEST THAT RESET DOES NOT CLEAR PSW

022760 005212
022762 022712 000234
022766 001014
3474 022770 123727 036066 000377
3475 022776 001014
3476 023000 012737 000357 177776
3477 023006 000005
3478 023010 022737 000357 177776
3479 023016 001404

TST234: INC (R2) ;UPDATE TEST NUMBER
CMP #234,(R2) ;SEQUENCE ERROR?
BNE TST235-10 ;BR TO ERROR HALT ON SEQ ERROR
CMPB @PASSPT,#377 ;ONLY DUE RESET EVERY 256. PASSES
BNE REST ;BR IF TO SKIP TEST
MOV #357,@PS ;MOV ONES TO PSW
RESET
CMP #357,@PS ;PSW CORRECT'
BEQ TST235

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 763 <====

023020 012742 000540
023024 005242
023026 000000

MOV #540,-(R2)
INC -(R2)
HALT

:MOVE TO MAILBOX # ***** 540 *****
:SET MSGTYP TO FATAL ERROR
:RESET ALTERED PSW
: OR SEQUENCE ERROR

3480 023030
3481
3482
3483
3484
3485
3486
3487

REST:

: THE FOLLOWING TEST CHECKS THE INDEPENDENT FUNCTIONING OF BASIC
: DATA PATH COMPONENTS WITH USER/SUPERVISOR MODE SET.

: TEST 235 TEST USER/SUPER S.P. CAN HOLD A 1 IN EVERY BIT

023030 005212
023032 022712 000235
023036 001034
3488 023040 052767 140000 154730
3489 023046 012706 000001
3490 023052 000241
3491 023054 006106

TST235: INC (R2) ;UPDATE TEST NUMBER
CMP #235,(R2) ;SEQUENCE ERROR?
BNE TST236-10 ;BR TO ERROR HALT ON SEQ ERROR
BIS #USR,PS ;SET USER MODE
MOV #1,R6 ;SET BIT 0
CLC ;CLEAR C-BIT
R6 ;ROTATE 1 POSITION

```

3492 023056 103376          BCC  USP1          ;BR IF NOT ALL DONE
3493 023060 042767 140000 154710 BIC  #USRM,PS      ;CLEAR USER MODE
3494 023066 001404          BEQ  SSP1A
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 763 <====
; MOVE TO MAILBOX # ***** 541 *****
; SET MSGTYP TO FATAL ERROR
; USER MODE R6 PICKED A BIT
; SET SUPERVISOR MODE
; SET BIT 0
; CLEAR C-BIT
; ROTATE 1 POSITION
; BR IF NOT ALL DONE
; CLEAR SUPERVISOR MODE

023070 012742 000541      MOV  #541,-(R2)
023074 005242          INC  -(R2)
023076 000000          HALT
3495 023100 042767 100000 154670 SSP1A: BIC  #100000,PS
3496 023106 012706 000001      MOV  #1,R6
3497 023112 000241          CLC
3498 023114 006106          SSP2: ROL  R6
3499 023116 103376          BCC  SSP2
3500 023120 042767 140000 154650 BIC  #140000,PS
3501 023126 001404          BEQ  TST236
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 743 <====
; MOVE TO MAILBOX # ***** 542 *****
; SET MSGTYP TO FATAL ERROR
; SUPER MODE R6 PICKED A BIT
; OR SEQUENCE ERROR

023130 012742 000542      MOV  #542,-(R2)
023134 005242          INC  -(R2)
023136 000000          HALT

3502 023140          USP2:
3503
3504
3505
3506
3507
3508
3509
3510
3511
;*****
; THIS TEST CHECKS THE INDEPENDENT FUNCTIONING OF THE USER
; SUPERVISOR AND KERNEL MODE R6'S. R6 IS SETUP AND ADDRESSED IN EACH
; OF THE TWO MODES TO VERIFY THAT THE TWO R6'S ARE INDEPENDENT
; OF EACH OTHER.
;*****
; TEST 236 TEST INDEPENDENCE OF USER/SUPER/KERNEL MODE R6,R6
;*****
TST236: INC  (R2)          ;UPDATE TEST NUMBER
        CMP  #236,(R2)   ;SEQUENCE ERROR?
        BNE  TST237-10   ;BR TO ERROR HALT ON SEQ ERROR
        BIS  #USRM,PS    ;SET USER MODE
        MOV  #011111,R6  ;SET USER R6 TO #011111
        BIC  #100000,PS  ;SET SUPERVISOR MODE
        MOV  #022222,R6  ;SET SUPER R6 TO #022222
        BIC  #140000,PS  ;SET KERNEL MODE
        MOV  #033333,R6  ;SET KERNEL R6 TO #033333
        CMP  #033333,R6  ;VERIFY R6 WITH KERNEL
        BEQ  USP2A
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====
; MOVE TO MAILBOX # ***** 543 *****
; SET MSGTYP TO FATAL ERROR
; DUAL ADDRESSING SEQUENCE ERROR
; SET SUPER MODE
; VERIFY R6 WITH SUPER

023214 012742 000543      MOV  #543,-(R2)
023220 005242          INC  -(R2)
023222 000000          HALT
3520 023224 052767 040000 154544 USP2A: BIS  #040000,PS
3521 023232 022706 022222      CMP  #022222,R6

```

```
3522 023236 001404          BEQ      USP3
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 743 <====
; MOVE TO MAILBOX # ***** 544 *****
; SET MSGTYP TO FATAL ERROR
; DUAL ADDRESSING SEQUENCE ERROR
; SET USEF. MODE
; VERIFY R6 WITH USER

      023240 012742 000544      MOV      #544,-(R2)
      023244 005242          INC      -(R2)
      023246 000000          HALT
3523 023250 052767 140000 154520 USP3: BIC      #USRM,PS
3524 023256 022706 011111          CMP      #011111,R6
3525 023262 001404          BEQ      USP4
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 731 <====
; MOVE TO MAILBOX # ***** 545 *****
; SET MSGTYP TO FATAL ERROR
; DUAL ADDRESSING SEQUENCE ERROR
; SET KERNEL MODE
; SET R6 TO #044444
; SET SUPER MODE
; SET R6 TO #055555
; SET USER MODE
; SET R6 TO #066666
; CHECK R6 TO #066666

      023264 012742 000545      MOV      #545,-(R2)
      023270 005242          INC      -(R2)
      023272 000000          HALT
3526 023274 042767 140000 154474 USP4: BIC      #140000,PS
3527 023302 012706 044444          MOV      #044444,R6
3528 023306 052767 040000 154462          BIC      #040000,PS
3529 023314 012706 055555          MOV      #055555,R6
3530 023320 052767 140000 154450          BIC      #140000,PS
3531 023326 012706 066666          MOV      #066666,R6
3532 023332 022706 066666          CMP      #066666,R6
3533 023336 001404          BEQ      USP5
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 703 <====
; MOVE TO MAILBOX # ***** 546 *****
; SET MSGTYP TO FATAL ERROR
; DUAL ADDRESSING SEQUENCE ERROR
; SET SUPER MODE
; CHECK R6 #055555

      023340 012742 000546      MOV      #546,-(R2)
      023344 005242          INC      -(R2)
      023346 000000          HALT
3534 023350 042767 100000 154420 USP5: BIC      #100000,PS
3535 023356 022706 055555          CMP      #055555,R6
3536 023362 001404          BEQ      USP6
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 671 <====
; MOVE TO MAILBOX # ***** 547 *****
; SET MSGTYP TO FATAL ERROR
; DUAL ADDRESSING SEQUENCE ERROR
; SET KERNEL MODE
; CHECK R6 FOR #055555

      023364 012742 000547      MOV      #547,-(R2)
      023370 005242          INC      -(R2)
      023372 000000          HALT
3537 023374 042767 140000 154374 USP6: BIC      #140000,PS
3538 023402 022706 044444          CMP      #044444,R6
3539 023406 001404          BEQ      TST237
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 657 <====
; MOVE TO MAILBOX # ***** 550 *****
; SET MSGTYP TO FATAL ERROR
; DUAL ADDRESSING SEQUENCE ERROR
; OR SEQUENCE ERROR

      023410 012742 000550      MOV      #550,-(R2)
      023414 005242          INC      -(R2)
      023416 000000          HALT
```

3540
3541
3542

;

3543
3544
3545
3546

: THESE NEXT TWO TESTS VERIFY MFPI AND MTPI INSTRUCTIONS
: WITH R6 IN MODE 0.

: TEST 237 TEST MFPI WITH R6 IN MODE 0

023420 005212
023422 022712 000237
023426 001033
3547 023430 012706 001000
3548 023434 012767 140000 154334
3549 023442 012706 036370
3550 023446 006506
3551 023450 022767 140000 154320
3552 023456 001410
3553 023460 042767 140000 154310
3554 023466 001417

TST237: INC (R2) ;UPDATE TEST NUMBER
CMP #237,(R2) ;SEQUENCE ERROR?
BNE TST240-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #STBOT,R6 ;INITIALIZE KERNEL STACK POINTER
MOV #USRM,PS ;SET USER MODE.PREVIOUS KERNEL
MOV #USTBOT,R6 ;INITIALIZE USER STACK POINTER
MFPI R6 ;TRY MFPI WITH MODE 0
CMP #140000,PS ;CHECK PSW
BEQ MFPI0 ;BR IF NO ERROR
BIC #USRM,PS ;CLEAR USER MODE
BEQ TST240

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 757 <====

023470 012742 000551
023474 005242
023476 000000
3555 023500 022767 001000 012660 MFPI0:
3556 023506 001407
3557 023510 042767 140000 154260
3558 023516 012742 000552
023522 005242
023524 000000
3559 023526 MFPI0A:
3560
3561

MOV #551,-(R2) ;MOVE TO MAILBOX # ***** 551 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;INCORRECT PSW FROM MFPI
OR SEQUENCE ERROR
CMP #STBOT,USTBOT-2 ;CHECK DATA ON STACK
BEQ MFPI0A ;BR IF NO ERROR
BIC #USRM,PS ;CLEAR USER MODE
MOV #552,-(R2) ;MOVE TO MAILBOX # ***** 552 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;INCORRECT DATA FROM MFPI

: TEST 240 TEST MTPI WITH R6 IN MODE 0

023526 005212
023530 022712 000240
023534 001033
3562 023536 005067 154234
3563 023542 005006
3564 023544 012767 140000 154224
3565 023552 012706 036370
3566 023556 012746 001000
3567 023562 006606
3568 023564 022767 140000 154204
3569 023572 001407
3570 023574 042767 140000 154174
3571 023602 012742 000553
023606 005242
023610 000000
3572 023612 005067 154160 MTP10:
3573 023616 020627 001000
3574 023622 001404

TST240: INC (R2) ;UPDATE TEST NUMBER
CMP #240,(R2) ;SEQUENCE ERROR?
BNE TST241-10 ;BR TO ERROR HALT ON SEQ ERROR
CLR PS ;SET KERNEL MODE
CLR R6 ;INITIALIZE KERNEL R6
MOV #USRM,PS ;SET USER MODE/PREVIOUS KERNEL
MOV #USTBOT,R6 ;INITIALIZE USER STACK POINTER
MOV #STBOT,-(R6) ;SET UP TARGET DATA
MTPI R6 ;TRY MODE 0 MTPI
CMP #USRM,PS ;CHECK PSW
BEQ MTP10 ;BR IF NO ERROR
BIC #USRM,PS ;CLEAR USER MODE
MOV #553,-(R2) ;MOVE TO MAILBOX # ***** 553 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;PS INCORRECT FOLLOWING MTP1
CLR PS ;SET KERNEL MODE
CMP R6,#STBOT ;CHECK TARGET DATA
BEQ TST241

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=
: CONDITIONAL BRANCH INST. AND <=
: REPLACE THE MOVE INSTRUCTION <=

```
023624 012742 000554      MOV    #554,-(R2)      ; WHICH FOLLOWS W/ 744 <====  
023630 005242      INC    -(R2)          ; MOVE TO MAILBOX # ***** 554 *****  
023632 000000      HALT                   ; SET MSGTYP TO FATAL ERROR  
                                ; DATA INCORRECT FOLLOWING MTP1  
                                ; OR SEQUENCE ERROR
```

3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600

```
.....  
: THIS TEST VERIFIES THE CONTENTS OF THE BRANCH ROM. THE 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 241 TEST THE BRANCH ROM

```
023634 005212      TST241: INC    (R2)          ; UPDATE TEST NUMBER  
023636 022712 000241  CMP    #241,(R2)      ; SEQUENCE ERROR?  
023642 001062      BNE    ER            ; BR TO ERROR HALT ON SEQ ERROR  
3601 023644 012700 036142  SETUP: MOV    #BRTAB,R0    ; INITIALIZE BRANCH TABLE POINTER  
3602 023650 012704 036216  MOV    #YNTAB,R4      ; INITIALIZE YES/NO BRANCH MAP POINTER  
3603 023654 012767 000017 000142  MOV    #15.,BRCT      ; INITIALIZE BRANCH TABLE COUNT  
3604 023662 012067 000110  SETBR: MOV    (R0)+,BRH    ; GET NEXT BRANCH INST.  
3605 023666 012401      MOV    (R4)+,R1      ; GET NEXT BRANCH MAP  
3606 023670 012767 177777 000074  MOV    #-1,CC        ; INITIALIZE CONDITION CODE VALUE  
3607 023676 012703 000020  MOV    #16.,R3        ; INITIALIZE CONDITION CODE COUNT  
3608 023702 005267 000064  SETCC: INC    CC        ; SET FOR NEXT CC VALUE  
3609 023706 032701 100000  BIT    #100000,R1     ; SEE IF SHOULD BR W/ THESE CC'S  
3610 023712 013705 177776  MOV    @#177776,R5    ; SIMULATE A JNE  
023716 042705 177773  BIC    #177773,R5     ; (JUMP NOT EQUAL)  
023722 000165 023726  JMP    .+4(R5)        ; TO SET2BR  
023726 000167 000020  JMP    SET2BR  
3611 023732 012767 024026 000042  MOV    #CONT,NBR      ; SET TO CONTINUE IF NO BRANCH  
3612 023740 012767 024010 000040  MOV    #ER,YBR        ; SET TO REPORT ERROR IF BRANCH  
3613 023746 000167 000014  JMP    AROUND         ; GO AROUND OPPOSITE CONDITION  
3614 023752 012767 024010 000022  SET2BR: MOV    #ER,NBR   ; SET TO REPORT ERROR IF NO BRANCH  
3615 023760 012767 024026 000020  MOV    #CONT,YBR     ; SET TO CONTINUE IF BRANCH  
3616 023766 006101  AROUND: ROL    R1      ; UPDATE BIT MAP  
3617  
3618 023770 012737      MOV    (PC)+,@(PC)+  ; SET CONDITION CODE
```

```

3619 023772 000000      CC:      0          ;NEW CC VALUE GOES HERE
3620 023774 177776      BRH:      0          ;BRANCH INST. GOES HERE
3621 023776 000000      JMP      @PC+      ;THIS JUMP IF NO BRANCH
3622 024000 000137      NBR:      0          ;WHERE TO GO IF NO BRANCH OCCURS
3623 024002 000000      JMP      @PC+      ;THIS JUMP IF BRANCH OCCURS
3624 024004 000137      YBR:      0          ;WHERE TO GO IF BRANCH OCCURS
3625 024006 000000      ER:      MOV      #STESTN,R2 ;RESTORE POINTER
3626 024010 012702 000304      MOV      #555,-(R2) ;MOVE TO MAILBOX # ***** 555 *****
3627 024014 012742 000555      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
          024020 005242      HALT
          024022 000000
3628 024024 000000      BRCT:     0          ;CC'S DONE?
3629 024026 005303      CONT:     DEC      R3          ;SIMULATE A JNE
3630 024030 013705 177776      MOV      @#177776,R5 ;
          024034 042705 177773      BIC      #177773,R5 ; (JUMP NOT EQUAL)
          024040 000165 024044      JMP      .+4(R5)      ; TO SETCC
          024044 000167 177632      JMP      SFTCC
3631 024050 005367 177750      DEC      BRCT          ;BR'S DONE?
3632 024054 013705 177776      MOV      @#177776,R5 ;SIMULATE A JNE
          024060 042705 177773      BIC      #177773,R5 ; (JUMP NOT EQUAL)
          024064 000165 024070      JMP      .+4(R5)      ; TO SETBR
          024070 000167 177566      JMP      SETBR
    
```

3633
 3634
 3635
 3636
 3637
 3638
 3639
 3640
 3641

```

:*****
:THE FOLLOWING TEST VERIFIES THAT NO DUAL ADDRESSING OF THE GENERAL
:REGISTERS OCCURS. ALL REGISTERS ARE CLEARED, AND A UNIQUE BIT IS SET
:IN EACH. CMP INSTRUCTIONS CHECK THAT ONLY ONE BIT IS SET IN EACH
:REGISTER.
:*****
    
```

TEST 242 DUAL REGISTER ADDRESSING TEST

```

          024074 005212      TST242: INC      (R2)          ;UPDATE TEST NUMBER
          024076 022712 000242      CMP      #242,(R2)      ;SEQUENCE ERROR?
          024102 001052      BNE      DAERR          ;BR TO ERROR HALT ON SEQ ERROR
3642 024104 005000      BITCLR: CLR      R0          ;INITIALIZE ALL REGISTERS
3643 024106 005001      CLR      R1
3644 024110 005002      CLR      R2
3645 024112 005003      CLR      R3
3646 024114 005004      CLR      R4
3647 024116 005005      CLR      R5
3648 024120 005006      CLR      R6
3649 024122 052700 000001      BITSET: BIS      #1,R0      ;SET R0=1
3650 024126 052701 000002      BIS      #2,R1          ;R1=2
3651 024132 052702 000004      BIS      #4,R2          ;R2=4
3652 024136 052703 000010      BIS      #10,R3         ;R3=10
3653 024142 052704 000020      BIS      #20,R4         ;R4=20
3654 024146 052705 000040      BIS      #40,R5         ;R5=40
3655 024152 052706 000100      BIS      #100,R6        ;R6=100
3656 024156 022706 000100      BITCHK: CMP      #100,R6   ;TEST THAT NO DUAL ADDRESSING OCCURRED
3657 024162 001022      BNE      DAERR          ;BR TO ERROR HALT IF ANY OTHER BITS ARE SET
3658 024164 022705 000040      CMP      #40,R5
3659 024170 001017      BNE      DAERR
3660 024172 022704 000020      CMP      #20,R4
3661 024176 001014      BNE      DAERR
3662 024200 022703 000010      CMP      #10,R3
    
```

```

3663 024204 001011
3664 024206 022702 000004
3665 024212 001006
3666 024214 022701 000002
3667 024220 001003
3668 024222 022700 000001
3669 024226 001404

```

```

BNE DAERR
CMP #4,R2
BNE DAERR
CMP #2,R1
BNE DAERR
CMP #1,R0
BEQ BITCON

```

```

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <== =
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 725 <====

```

```

024230
024230 012742 000556
024234 005242
024236 000000
3670 024240 012702 000304
3671
3672
3673
3674
3675
3676
3677
3678

```

```

DAERR: MOV #556,-(R2) ;MOVE TO MAILBOX # ***** 556 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;DUAL ADDRESSING ERROR
BITCON: MOV #STESTN,R2 ;RESTORE POINTER

```

```

:*****
: THIS TEST VERIFIES THAT THE UPPER BYTE OF THE PSW IS NOT AFFECTED
: WHEN THE PRIORITY LEVEL OR CC'S ARE CHANGED. ALL BITS ARE
: INITIALLY SET IN THE PSW, AND THE LOW BYTE IS CLEARED. A BIT
: INSTRUCTION VERIFIES THE DATA.
:*****

```

```

:TEST 243 TEST BYTE INSTRUCTION ON PSW
:*****

```

```

024244 005212
024246 022712 000243
024252 001012
3679 024254 052737 170357 177776
3680 024262 105037 177776
3681 024266 013700 177776
3682 024272 032700 170000
3683 024276 001006
3684 024300 005037 177776
3685 024304 012742 000557
024310 005242
024312 000000
3686 024314 005037 177776
3687
3688
3689
3690
3691
3692
3693
3694

```

```

TST243: INC (R2) ;UPDATE TEST NUMBER
CMP #243,(R2) ;SEQUENCE ERROR?
BNE BTERR ;BR TO ERROR HALT ON SEQ ERROR
BIS #170357,@PS ;SET ALL POSSIBLE BITS IN PSW
CLRB @PS ;CLR PR LEVEL AND CC'S
MOV @PS,R0 ;COPY CONTENTS OF PSW
BIT #170000,R0 ;TEST THAT UPPER BYTE IS UNAFFECTED
BNE BTCON ;CONTINUE IF OK
BTERR: CLR @PS ;RETURN TO KERNEL MODE
MOV #557,-(R2) ;MOVE TO MAILBOX # ***** 557 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;BYTE INSTRUCTION ALTERED PSW
BTCON: CLR @PS ;RETURN TO KERNEL MODE

```

```

:*****
: THIS TEST VERIFIES THAT A JMP INSTRUCTION DOES NOT ALTER THE
: CONDITION CODES IN THE PSW. THE CC'S ARE PRESET,THE JMP IS
: EXECUTED, AND CONDITIONAL BRANCHES VERIFY THE STATE OF THE CC'S.
:*****

```

```

:TEST 244 TEST THAT JMP OPCODE DOES NOT AFFECT C.C.'S
:*****

```

```

024320 005212
024322 022712 000244
024326 001010
3695 024330 000277
3696 024332 000252
3697 024334 000167 000000
3698 024340 100403
3699 024342 001002

```

```

TST244: INC (R2) ;UPDATE TEST NUMBER
CMP #244,(R2) ;SEQUENCE ERROR?
BNE TST245-10 ;BR TO ERROR HALT ON SEQ ERROR
SCC ;CC=0101
+CLN!CLV ;JUMP TO TEST PSW
JMPT: BMI JMPERR ;BR TO ERROR HALT IF N-BIT IS SET
BNE JMPERR ;BR TO ERROR HALT IF Z-BIT IS CLEAR

```

```

3700 024344 102401      BVS      JMPERR      ;BR TO ERROR HALT IF V-BIT IF SET
3701 024346 103404      BCS      TST245
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <==--
;          CONDITIONAL BRANCH INST. AND <==--
;          REPLACE THE MOVE INSTRUCTION <==--
;          WHICH FOLLOWS W/ 767 <==--

```

```

024350      JMPERR:
024350 012742 000560      MOV      #560,-(R2)      ;MOVE TO MAILBOX # ***** 560 *****
024354 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
024356 000000      HALT          ;JMP INSTRUCTION AFFECTED CC'S
; OR SEQUENCE ERROR

```

```

3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
:*****
: THIS TEST VERIFIES THE SET AND CLEAR CONDITION CODE INSTRUCTIONS.
: THE TEST CONSISTS OF TWO ROUTINES, ONE TO TEST ALL CLEAR CC
: INSTRUCTIONS, AND THE SECOND TO TEST ALL SET CC INSTRUCTIONS. ALL
: POSSIBLE COMBINATIONS OF CONDITION CODES ARE TESTED, INCLUDING NOP'S.
: TO TEST THE CLEAR CC INSTRUCTIONS, ALL CONDITION CODES ARE
: INITIALLY SET. THE INSTRUCTION IS EXECUTED, AND THE PSW IS CHECKED
: TO VERIFY THE PROPER COMBINATION OF CONDITION CODES.
: TO TEST THE SET CC INSTRUCTIONS, THE CONDITION CODES ARE
: INITIALLY CLEARED, AND ONLY THE REQUIRED BITS ARE SET BY THE SET CC
: INSTRUCTION. THE CONTENTS OF THE PSW ARE CHECKED TO VERIFY THAT
: ONLY THE REQUIRED BITS WERE SET.
:*****

```

:TEST 245 TEST SET CC AND CLEAR CC INSTRUCTIONS

```

024360 005212      TST245: INC      (R2)          ;UPDATE TEST NUMBER
024362 022712 000245      CMP      #245,(R2)      ;SEQUENCE ERROR?
024366 001062      BNE      CCERR          ;BR TO ERROR HALT ON SEQ ERROR
3717 024370 012767 000240 000024      MOV      #240,CC1       ;INITIALIZE CLR CC INSTRUCTION CODES
3718 024376 012767 000017 000032      MOV      #17,CC2        ;INITIALIZE OCTAL MAP
3719 024404 012767 000261 000102      MOV      #261,SC3       ;INITIALIZE SET CC INSTRUCTION CODES
3720 024412 012767 000001 000110      MOV      #1,SC4         ;INITIALIZE OCTAL MAP
3721 024420 000277      CLRCD: SCC          ;SET ALL CONDITION CODES
3722 024422 000000      CC1: 0              ;CONDITION CODE INSTRUCTION
3723 024424 013704 177776      MOV      @RPS,R4        ;COPY THE PSW
3724 024430 042704 177760      BIC      #177760,R4     ;ISOLATE CONDITION CODES
3725 024434 022704      CMP      (PC)+,R4       ;CHECK THAT PROPER CC'S WERE CLEARED
3726 024436 000000      CC2: 0              ;OCTAL REPRESENTATION OF CC'S
3727 024440 001404      REQ      CON1

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 752 <====

```

```

024442 012742 000561      MOV      #561,-(R2)     ;MOVE TO MAILBOX # ***** 561 *****
024446 005242      INC      -(R2)         ;SET MSGTYP TO FATAL ERROR
024450 000000      HALT          ;CLEAR CC INSTRUCTION FAILED
3728 024452 005367 177760      CON1: DEC      CC2      ;SET NEXT OCTAL MAP OF CC'S
3729 024456 005267 177740      INC      CC1          ;GET NEXT CLEAR CC INSTRUCTION
3730 024462 026727 177734 000257      CMP      CC1,#257      ;TEST FOR CCC INSTRUCTION
3731 024470 003753      BLE      CLRCD        ;GO TEST NEXT INSTRUCTION IF NOT FOUND
3732 024472 026727 177724 000260      CMP      CC1,#260      ;CHECK FOR NOP=260
3733 024500 001004      BNE      SETCD        ;GO TEST SET CC INSTRUCTIONS
3734 024502 012767 000017 177726      MOV      #17,CC2       ;SET OCTAL MAP TO TEST NOP
3735 024510 000743      BR       CLRCD        ;GO TEST NOP

```



```

3736 024512 000257      SETCD:  CCC      ;CLEAR ALL CONDITION CODES
3737 024514 000000      SC3:    0        ;CONDITION CODE INSTRUCTION
3738 024516 013704 177776  MOV     @PSW,R4  ;COPY PSW
3739 024522 042704 177760  BIC     #177760,R4 ;CLEAR AWAY UNWANTED BITS
3740 024526 022704      CMP     (PC)+,R4 ;CHECK THAT PROPER CC'S WERE SET
3741 024530 000000      SC4:    0        ;OCTAL REPRESENTATION OF CC'S
3742 024532 001404      BEQ     CON2

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 715 <====

```

```

024534      CCERR:
024534 012742 000562      MOV     #562,-(R2) ;MOVE TO MAILBOX # ***** 562 *****
024540 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
024542 000000      HALT                    ;SET CC FAILED OR SEQUENCE ERROR
3743 024544 005267 177760  CON2:  INC     SC4      ;SET NEXT OCTAL MAP
3744 024550 005267 177740      INC     SC3          ;PREPARE NEXT SET CC INSTRUCTION
3745 024554 026727 177734 000277  CMP     SC3,#277    ;FINISHED?
3746 024562 003753      BLF     SETCD       ;BR IF NO

```

```

3747
3748
3749
3750
3751
3752
3753

```

```

:*****
:
: THESE NEXT TWO TEST VERIFY MFPD AND MTPD INSTRUCTIONS
: WITH R6 IN MODE 0.
:
:*****

```

```

:TEST 246 TEST MFPD WITH R6 IN MODE 0
:*****

```

```

024564 005212      TST246: INC     (R2)      ;UPDATE TEST NUMBER
024566 022712 000246      CMP     #246,(R2)  ;SEQUENCE ERROR?
024572 001032      BNE     TST247-10  ;BR TO ERROR HALT ON SEQ ERROR
3754 024574 012706 001000      MOV     #STBOT,R6  ;INITIALIZE KERNEL STACK POINTER
3755 024600 012767 140000 153170  MOV     #USRM,PS   ;SETUP USER MODE .PREVIOUS KERNEL
3756 024606 012706 036370      MOV     #USTBOT,R6 ;INITIALIZE USER STACK POINTER
3757 024612 106506      MFPD    R6         ;TRY MFPD WITH MODE 0
3758 024614 022767 140000 153154  CMP     #140000,PS ;CHECK PSW
3759 024622 001407      BEQ     MFPD0      ;BR IF NO ERROR
3760 024624 042767 140000 153144  BIC     #USRM,PS   ;CLEAR USER MODE
3761 024632 012742 000563      MOV     #563,-(R2) ;MOVE TO MAILBOX # ***** 563 *****
024636 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
024640 000000      HALT                    ;INCORRECT PSW FROM MFPD
3762 024642 022767 001000 011516  MFPD0: CMP     #STBOT,USTBOT-2 ;CHECK DATA ON STACK
3763 024650 001407      BEQ     MFPD0A      ;BR IF NO ERROR
3764 024652 042767 140000 153116  BIC     #USRM,PS   ;CLEAR USER MODE
3765 024660 012742 000564      MOV     #564,-(R2) ;MOVE TO MAILBOX # ***** 564 *****
024664 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
024666 000000      HALT                    ;INCORRECT DATA FROM MFPD

```

```

3766 024670
3767
3768

```

```

:*****
:TEST 247 TEST MTPD WITH R6 IN MODE 0
:*****

```

```

024670 005212      TST247: INC     (R2)      ;UPDATE TEST NUMBER
024672 022712 000247      CMP     #247,(R2)  ;SEQUENCE ERROR?
024676 001031      BNE     TST250-10  ;BR TO ERROR HALT ON SEQ ERROR
3769 024700 005067 153072      CLR     PS         ;SET KERNEL MODE
3770 024704 005006      CLR     R6         ;INITIALIZE KERNEL R6

```

```
3771 024706 012767 140000 153062      MOV      #USRM,PS          ;SET USER MODE/PREVIOUS KERNEL
3772 024714 012746 001000              MOV      #STBOT,-(R6)     ;SET UP TARGET DATA
3773 024720 106606              MTPD      R6              ;TRY MODE 0 MTPD
3774 024722 022767 140000 153046      CMP      #USRM,PS        ;CHECK PSW
3775 024730 001407              BEQ      MTPD0           ;BR IF NO ERROR
3776 024732 042767 140000 153036      BIC      #USRM,PS        ;CLEAR USER MODE
3777 024740 012742 000565              MOV      #565,-(R2)      ;MOVE TO MAILBOX # ***** 565 *****
                                INC      -(R2)              ;SET MSGTYP TO FATAL ERROR
                                HALT                    ;PS INCORRECT FOLLOWING MTPD
MTPD0: 024744 005242              CLR      PS              ;SET KERNEL MODE
                                024746 000000              CMP      R6,#STBOT      ;CHECK TARGET DATA
                                3778 024750 005067 153022              BEQ      TST250
                                3779 024754 020627 001000
                                3780 024760 001404
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 746 <====
; MOVE TO MAILBOX # ***** 566 *****
; SET MSGTYP TO FATAL ERROR
; DATA INCORRECT FOLLOWING MTPD
; OR SEQUENCE ERROR
```

```
024762 012742 000566      MOV      #566,-(R2)
024766 005242      INC      -(R2)
024770 000000      HALT
```

3781
3782
3783

```
*****
:TEST 250      TEST MFPT INSTRUCTION
*****
```

```
024772 005212              TST250: INC      (R2)          ;UPDATE TEST NUMBER
024774 022712 000250      CMP      #250,(R2)       ;SEQUENCE ERROR?
025000 001005              BNE      TST251-10       ;BR TO ERROR HALT ON SEQ ERROR
3784 025002 005000              CLR      R0              ;SET UP R0
3785 025004 000007              MFPT                    ;MOVE FROM PROCCSOR TYPE
3786 025006 122700 000001      CMPSB   #1,R0           ;CHECK FOR A ONE IN R0
3787 025012 001404              BEQ      TST251
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 772 <====
; MOVE TO MAILBOX # ***** 567 *****
; SET MSGTYP TO FATAL ERROR
; MFPT FAILED TO RETURN A 1,IN R0
; OR SEQUENCE ERROR
```

```
025014 012742 000567      MOV      #567,-(R2)
025020 005242      INC      -(R2)
025022 000000      HALT
```

3788
3789
3790
3791
3792
3793
3794
3795
3796

```
*****
; .SBTTL BIT TEST OF PIRQ REGISTER
; A COUNT PATTERN IS THEN RUN THRU THE REGISTER TO ENSURE THAT
; THE ENCODER FUNCTIONS PROPERLY.
*****
```

```
:TEST 251      BIT TEST OF PIRQ REGISTER
*****
025024 005212              TST251: INC      (R2)          ;UPDATE TEST NUMBER
025026 022712 000251      CMP      #251,(R2)       ;SEQUENCE ERROR?
025032 001064              BNE      TST252-10       ;BR TO ERROR HALT ON SEQ ERROR
3797 025034 012706 001000      MOV      #STBOT,SP      ;INITIALIZE THE SP
3798 025040 012767 025054 011144      MOV      #4$, $LPADR     ;SETUP LOOP ADR
3799 025046 012767 025054 011140      MOV      #4$, $LPERR     ;SETUP ERROR LOOP
```

3800	025054	052737	000340	177776	4\$:	BIS	#340,@#177776		;SET THE CPU PRIORITY AT 7.
3801	025062	005067	011112			CLR	\$TMP0		;SETUP COMPARISON LOCATION
3802	025066	005037	177772			CLR	@#PIRQ		;CLEAR PIRQ REGISTER
3803	025072	026737	011102	177772		CMP	\$TMP0,@#PIRQ		;DID PIRQ CLEAR
3804	025100	001035				BNE	1\$;BRANCH IF NO
3805	025102	012700	000177			MOV	#177,R0		;SETUP ITERATION COUNT
3806	025106	012767	001042	011064		MOV	#1042,\$TMP0		;SETUP COMPARISON LOCATION
3807	025114	012701	000002			MOV	#2,R1		;SETUP R1
3808	025120	062737	001000	177772	2\$:	ADD	#1000,@#PIRQ		;START COUNT PATTERN
3809	025126	026737	011046	177772		CMP	\$TMP0,@#PIRQ		;DID REGISTER SET CORRECT?
3810	025134	001017				BNE	1\$;BRANCH IF NO
3811	025136	120137	177773			CMPS	R1,@#PIRQ+1		;IS PIRQ READY TO GO TO NEXT LEVEL?
3812	025142	001005				BNE	3\$;BRANCH IF NO
3813	025144	062767	000042	011026		ADD	#42,\$TMP0		;INCREMENT ENCODED VALUE IN TEST LOC.
3814	025152	005201				INC	R1		;SETUP R1 FOR ROTATE
3815	025154	006101				ROL	R1		;SET R1 TO NEXT CHECK LEVEL
3816	025156	062767	001000	011014	3\$:	ADD	#1000,\$TMP0		;INC. PIRQ LEVEL IN TEST LOCATION
3817	025164	077023				SOS	R0,2\$;CONTINUE COUNT
3818	025166	005037	177772			CLR	@#PIRQ		;ENSURE PIRQ CLEAR
3819	025172	000410				BR	EIS		;GO TO NEXT TEST
3820	025174	013767	177772	011006	1\$:	MOV	@#PIRQ,\$EPIRQ		;SAVE PIRQ FOR ERRC } CHECKING
3821	025202	001404				BEQ	TST252		

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <--
; CONDITIONAL BRANCH INST. AND <-
; REPLACE THE MOVE INSTRUCTION <-
; WHICH FOLLOWS W/ 713 <-
; MOVE TO MAILBOX # ***** 570 *****
; SET MSGTYP TO FATAL ERROR
; PIRQ REG. FAILED
; OR SEQUENCE ERROR
  
```

025204	012742	000570				MOV	#570,-(R2)
025210	005242					INC	-(R2)
025212	000000					HALT	

3822 025214 EIS:

3824
 3825
 3826
 3827
 3828
 3829

.SBTTL EIS ASH/ASCH/MUL/DIV TESTS
 .SBTTL ASH SHIFTING RIGHT USING DM2 REG 7
 ;SHIFT RIGHT CLEAR N-BIT AND C-BIT
 ;MODE 2-REG 7

 ;TEST 252 ASH 40000 SHIFTED BY 177765=10 PS=0

025214 005212
 025216 022712 000252
 025222 001026
 3830 025224 012706 001000
 3831 025230 005037 177776
 3832 025234 012700 040000
 3833 025240 072027 177765
 3834 025244 013767 177776 011216
 3835 025252 122767 000000 011210
 3836 025260 001404

TST252: INC (R2) ;UPDATE TEST NUMBER
 CMP #252,(R2) ;SEQUENCE ERROR?
 BNE TST253-10 ;BR TO ERROR HALT ON SEQ ERROR
 MOV #STBOT,R6
 CLR @#PS
 MOV #40000,R0 ;LOAD R0 WITH 40000
 ASH #177765,R0 ;SHIFT R0 BY 177765
 MOV @#PS,SPSW ;SAVE PS
 CMPB #0,SPSW ;IS THE PS 0?
 BEQ 1\$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- -
 ; CONDITIONAL BRANCH INST. AND < ---
 ; REPLACE THE MOVE INSTRUCTION <=====
 ; WHICH FOLLOWS W/ 760 <=====
 ;

025262 012742 000571
 025266 005242
 025270 000000
 3837 025272 022700 000010
 3838 025276 001404

MOV #571,-(R2) ;MOVE TO MAILBOX # ***** 571 *****
 INC -(R2) ;SET MSGTYP TO FATAL ERROR
 HALT ;THE PS IS NOT EQUAL TO 0
 1\$: CMP #10,R0 ;IS THE RESULT 10?
 BEQ TST253

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
 ; CONDITIONAL BRANCH INST. AND <=====
 ; REPLACE THE MOVE INSTRUCTION <=====
 ; WHICH FOLLOWS W/ 751 <=====
 ;

025300 012742 000572
 025304 005242
 025306 000000

MOV #572,-(R2) ;MOVE TO MAILBOX # ***** 572 *****
 INC -(R2) ;SET MSGTYP TO FATAL ERROR
 HALT ;R0 IS NOT EQUAL TO 10
 ; OR SEQUENCE ERROR

3839
 3840
 3841
 3842

;SHIFT RIGHT SET N-BIT CLEAR C-BIT
 ;MODE 2 /REG 7

 ;TEST 253 ASH 125252 SHIFTED BY -2=165252 PS=11

025310 005212
 025312 022712 000253
 025316 001022
 3843 025320 012700 125252
 3844 025324 072027 177776
 3845 025330 013767 177776 011132
 3846 025336 122767 000011 011124
 3847 025344 001404

TST253: INC (R2) ;UPDATE TEST NUMBER
 CMP #253,(R2) ;SEQUENCE ERROR?
 BNE TST254-10 ;BR TO ERROR HALT ON SEQ ERROR
 MOV #125252,R0 ;LOAD R0 WITH 125252
 ASH #-2,R0 ;SHIFT R0 BY -2
 MOV @#PS,SPSW ;SAVE PS
 CMPB #11,SPSW ;IS THE PS 11?
 BEQ 1\$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- -
 ; CONDITIONAL BRANCH INST. AND <---
 ; REPLACE THE MOVE INSTRUCTION < ---
 ; WHICH FOLLOWS W/ 764 < -

025346 012742 000573
 025352 005242
 025354 000000
 3848 025356 022700 165252

MOV #573,-(R2) ;MOVE TO MAILBOX # ***** 573 *****
 INC -(R2) ;SET MSGTYP TO FATAL ERROR
 HALT ;THE PS IS NOT EQUAL TO 11
 1\$: CMP #165252,R0 ;IS THE RESULT 165252?

3849 025362 001404 BEQ TST254
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====
025364 012742 000574 MOV #574,-(R2) ;MOVE TO MAILBOX # ***** 574 *****
025370 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
025372 000000 HALT ;R0 IS NOT EQUAL TO 165252
; OR SEQUENCE ERROR

3850
3851 ;SHIFT RIGHT CLEAR N-BIT SET Z-BIT
3852 ;MODE 2 /REG 7
3853

;TEST 254 ASH 0 SHIFTED BY -16. =0 PS=4

TST254: INC (R2) ;UPDATE TEST NUMBER
025374 005212 CMP #254,(R2) ;SEQUENCE ERROR?
025376 022712 000254 BNE TST255-10 ;BR TO ERROR HALT ON SEQ ERROR
025402 001022 MOV #0,R0 ;LOAD R0 WITH 0
3854 025404 012700 000000 ASH #-16.,R0 ;SHIFT R0 BY -16.
3855 025410 072027 177760 MOV @WPS,SPSW ;SAVE PS
3856 025414 013767 177776 011046 CMPB #4,SPSW ;IS THE PS 4?
3857 025422 122767 000004 011040 BEQ 1\$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 764 <====
025432 012742 000575 MOV #575,-(R2) ;MOVE TO MAILBOX # ***** 575 *****
025436 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
025440 000000 HALT ;THE PS IS NOT EQUAL TO 4
3859 025442 022700 000000 1\$: CMP #0,R0 ;IS THE RESULT 0?
3860 025446 001404 BEQ TST255

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====
025450 012742 000576 MOV #576,-(R2) ;MOVE TO MAILBOX # ***** 576 *****
025454 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
025456 000000 HALT ;R0 IS NOT EQUAL TO 0
; OR SEQUENCE ERROR

3861
3862 .SBTTL ASH SHIFTING LEFT STORE ASH DM2 REG 7
3863 ;SHIFT LEFT SET C-BIT=0 STORE ASH
3864 ;MODE 2 /REG 7
3865

;TEST 255 ASH 0 SHIFTED BY 0=0 PS=4

TST255: INC (R2) ;UPDATE TEST NUMBER
025460 005212 CMP #255,(R2) ;SEQUENCE ERROR?
025462 022712 000255 BNE TST256-10 ;BR TO ERROR HALT ON SEQ ERROR
025466 001022 MOV #0,R0 ;LOAD R0 WITH 0
3866 025470 012700 000000 ASH #0,R0 ;SHIFT R0 BY 0
3867 025474 072027 000000 MOV @WPS,SPSW ;SAVE PS
3868 025500 013767 177776 010762 CMPB #4,SPSW ;IS THE PS 4?
3869 025506 122767 000004 010754 BEQ 1\$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====

```
025516 012742 000577      MOV      #577,-(R2)      ;REPLACE THE MOVE INSTRUCTION <====
025522 005242              INC      -(R2)          ;WHICH FOLLOWS W/ 764 <====
025524 000000              HALT                    ;MOVE TO MAILBOX # ***** 577 *****
3871 025526 022700 000000 1$: CMP      #0,R0        ;SET MSGTYP TO FATAL ERROR
3872 025532 001404          BEQ      TST256         ;THE PS IS NOT EQUAL TO 4
;IS THE RESULT 0?
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====
025534 012742 000600      MOV      #600,-(R2)     ;MOVE TO MAILBOX # ***** 600 *****
025540 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
025542 000000              HALT                    ;R0 IS NOT EQUAL TO 0
; OR SEQUENCE ERROR
```

3873
3874
3875
3876
3877
3878

.SBTTL ASH SHIFTING LEFT DM6 REG 7
;SHIFT LEFT SET C-BIT=1
;MODE 6 /REG 7

:TEST 256 ASH 125252 SHIFTED BY S1=125250 PS-12

```
025544 005212              TST256: INC      (R2)      ;UPDATE TEST NUMBER
025546 022712 000256      CMP      #256,(R2)     ;SEQUENCE ERROR?
025552 001024              BNE      TST257-10     ;BR TO ERROR HALT ON SEQ ERROR
3879 025554 012700 125252  MOV      #125252,R0    ;LOAD R0 WITH 125252
3880 025560 012704 036476  MOV      #S1,R4        ;SET UP R4
3881 025564 072067 010706  ASH      S1,R0        ;SHIFT R0 BY S1
3882 025570 013767 177776 010672  MOV      @R4,SPSW      ;SAVE PS
3883 025576 122767 000012 010664  CMPB    #12,SPSW      ;IS THE PS 12?
3884 025604 001404          BEQ      1$
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 762 <====
```

```
025606 012742 000601      MOV      #601,-(R2)     ;MOVE TO MAILBOX # ***** 601 *****
025612 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
025614 000000              HALT                    ;THE PS IS NOT EQUAL TO 12
3885 025616 022700 125250 1$: CMP      #125250,R0    ;IS THE RESULT 125250?
3886 025622 001404          BEQ      TST257
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
; CONDITIONAL BRANCH INST. AND <
; REPLACE THE MOVE INSTRUCTION <
; WHICH FOLLOWS W/ 753 <
```

```
025624 012742 000602      MOV      #602,-(R2)     ;MOVE TO MAILBOX # ***** 602 *****
025630 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
025632 000000              HALT                    ;R0 IS NOT EQUAL TO 177525
; OR SEQUENCE ERROR
```

3887
3888
3889
3890
3891

.SBTTL ASH SHIFTING LEFT TEST SIGN Z-BIT DM7 REG 7
;SHIFT LEFT COUNT DOWN TO ZERO TEST SIGN Z-BIT
;MODE 7 /REG 7

:TEST 257 ASH 125252 SHIFTED BY @S2= 177525 PS=10

```
025634 005212              TST257: INC      (R2)      ;UPDATE TEST NUMBER
```

```

025636 022712 000257      CMP      #257,(R2)      ;SEQUENCE ERROR?
025642 001024      BNE      TST260-10    ;BR TO ERROR HALT ON SEQ ERROR
3892 025644 012700 125252    MOV      #125252,R0   ;LOAD R0 WITH 125252
3893 025650 012703 036504    MOV      #S4,R3      ;SET UP R3
3894 025654 072077 010624    ASH      @S4,R0      ;SHIFT R0 BY @S2
3895 025660 013767 177776 010602  MOV      @WPS,SPSW    ;SAVE PS
3896 025666 122767 000010 010574  CMPB     #10,SPSW    ;IS THE PS 10?
3897 025674 001404      BEQ      1$          ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 762 <====
025676 012742 000603      MOV      #603,-(R2)   ;MOVE TO MAILBOX # ***** 603 *****
025702 005242      INC      -(R2)       ;SET MSGTYP TO FATAL ERROR
025704 000000      HALT     ;THE PS IS NOT EQUAL TO 10
3898 025706 022700 177525 1$:  CMP      #177525,R0   ;IS RESULT 177525?
3899 025712 001404      BEQ      TST260     ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 753 <====
025714 012742 000604      MOV      #604,-(R2)   ;MOVE TO MAILBOX # ***** 604 *****
025720 005242      INC      -(R2)       ;SET MSGTYP TO FATAL ERROR
025722 000000      HALT     ;R0 IS NOT EQUAL TO 177525
; OR SEQUENCE ERROR
3900
3901      .SBTTL ASH SHIFTING LEFT TESTING IR9 FOR ASH/ASHC DM3 REG 7
3902      ;SHIFT LEFT TEST IR9 TO DETERMINE ASH/ASHC
3903      ;MODE 3 /REG 7
3904      ;*****
;TEST 260      ASH 125252 SHIFTED BY @S1=177525 PS=10
;*****
025724 005212      TST260: INC      (R2)      ;UPDATE TEST NUMBER
025726 022712 000260      CMP      #260,(R2)   ;SEQUENCE ERROR?
025732 001024      BNE      TST261-10    ;BR TO ERROR HALT ON SEQ ERROR
3905 025734 012700 125252    MOV      #125252,R0   ;LOAD R0 WITH 125252
3906 025740 012704 036502    MOV      #S3,R4      ;SET UP R4
3907 025744 072037 036502    ASH      @W$3,R0     ;SHIFT R0 BY @W$1
3908 025750 013767 177776 010512  MOV      @WPS,SPSW    ;SAVE PS
3909 025756 122767 000010 010504  CMPB     #10,SPSW    ;IS THE PS 10?
3910 025764 001404      BEQ      1$          ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 762 <====
025766 012742 000605      MOV      #605,-(R2)   ;MOVE TO MAILBOX # ***** 605 *****
025772 005242      INC      -(R2)       ;SET MSGTYP TO FATAL ERROR
025774 000000      HALT     ;THE PS IS NOT EQUAL TO 10
3911 025776 022700 177525 1$:  CMP      #177525,R0   ;IS THE RESULT 177525?
3912 026002 001404      BEQ      TST261     ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 753 <====
026004 012742 000606      MOV      #606,-(R2)   ;MOVE TO MAILBOX # ***** 606 *****
026010 005242      INC      -(R2)       ;SET MSGTYP TO FATAL ERROR
026012 000000      HALT     ;R0 IS NOT EQUAL TO 177525

```

; OR SEQUENCE ERROR

3913
3914
3915
3916
3917
3918
3919

.SBTTL ASH SHIFTING LEFT USING DM6 REG 7
;SHIFT LEFT TEST R17 TO DETERMINE C-BIT
;CLEAR BX TO INDICATE C-BIT=0
;MODE 6 /REG 7

;TEST 261 ASH 025252 SHIFTED S1=125250 PS=12

026014 005212
026016 022712 000261
026022 001024
3920 026024 012700 025252
3921 026030 012704 036476
3922 026034 072067 010436
3923 026040 013767 177776 010422
3924 026046 122767 000012 010414
3925 026054 001404

TST261: INC (R2) ;UPDATE TEST NUMBER
CMP #261,(R2) ;SEQUENCE ERROR?
BNE TST262-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #025252,R0 ;LOAD R0 WITH 025252
MOV #S1,R4 ;SET UP R4
ASH S1,R0 ;SHIFT R0 BY S1
MOV @#PS,SPSW ;SAVE PS
CMPB #12,SPSW ;IS THE PS 10?
BEQ 1\$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 762 <====

026056 012742 000607
026062 005242
026064 000000
3926 026066 022700 125250
3927 026072 001404

1\$: MOV #607,-(R2) ;MOVE TO MAILBOX # ***** 607 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;THE PS IS NOT EQUAL TO 10
CMP #125250,R0 ;IS THE RESULT 125250
BEQ TST262

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 753 <====

026074 012742 000610
026100 005242
026102 000000

MOV #610,-(R2) ;MOVE TO MAILBOX # ***** 610 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;R0 IS NOT EQUAL TO 077525
; OR SEQUENCE ERROR

3928
3929
3930
3931
3932
3933

.SBTTL ASHC SHIFTING POS.USING DM0 REG 2
;MODE 0 REG 4

;TEST 262 ASHC 125252,125252,SHIF BY R4=177525 52525 PS=10

026104 005212
026106 022712 000262
026112 001035
3934 026114 012700 125252
3935 026120 012701 125252
3936 026124 000241
3937 026126 012704 177771
3938 026132 073004
3939 026134 013767 177776 010326
3940 026142 122767 000010 010320
3941 026150 001404

TST262: INC (R2) ;UPDATE TEST NUMBER
CMP #262,(R2) ;SEQUENCE ERROR?
BNE TST263-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #125252,R0 ;LOAD R0 WITH 125252
MOV #125252,R1 ;LOAD R0!1 WITH 125252
CLC
MOV #-7,R4 ;SET UP R4
ASHC R4,R0 ;SHIFT R0,R0!1 BY R4
MOV @#PS,SPSW ;SAVE PS
CMPB #10,SPSW ;IS THE PS 10?
BEQ 1\$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====


```
026152 012742 000611      MOV      #611,-(R2)      ; REPLACE THE MOVE INSTRUCTION <====
026156 005242              INC      -(R2)          ; WHICH FOLLOWS W/ 760 <====
026160 000000              HALT                    ; MOVE TO MAILBOX # ***** 611 *****
3942 026162 022700 177525 1$: CMP      #177525,R0      ; SET MSGTYP TO FATAL ERROR
3943 026166 001404              BEQ      2$            ; THE PS IS NOT EQUAL TO 10
; IS RESULT 177525?
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 751 <====
026170 012742 000612      MOV      #612,-(R2)      ; MOVE TO MAILBOX # ***** 612 *****
026174 005242              INC      -(R2)          ; SET MSGTYP TO FATAL ERROR
026176 000000              HALT                    ; R0 IS NOT EQUAL TO 177525
3944 026200 022701 052525 2$: CMP      #52525,R1      ; IS THE RESULT 52525?
3945 026204 001404              BEQ      TST263
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 742 <====
026206 012742 000613      MOV      #613,-(R2)      ; MOVE TO MAILBOX # ***** 613 *****
026212 005242              INC      -(R2)          ; SET MSGTYP TO FATAL ERROR
026214 000000              HALT                    ; R1 IS NOT EQUAL TO 52525
; OR SEQUENCE ERROR
3946 .SBTTL ASHC SHIFTING NEG. USING DM4 REG 3
3947
3948
3949
```

:MODE 4 REG 3

:TEST 263 ASHC 125252,125252,SHIF,-(3)=177525,52525 PS=10

```
TST263: 026216 005212              INC      (R2)          ; UPDATE TEST NUMBER
026220 022712 000263      CMP      #263,(R2)      ; SEQUENCE ERROR?
026224 001035              BNE     TST264-10      ; BR TO ERROR HALT ON SEQ ERROR
3950 026226 012700 125252      MOV      #125252,R0     ; LOAD R0 WITH 125252
3951 026232 012701 125252      MOV      #125252,R1     ; LOAD R0!1 WITH 125252
3952 026236 000241              CLC
3953 026240 012703 036504      MOV      #S3+2,R3      ; SET UP R3
3954 026244 073043      ASHC     -(R3),R0      ; SHIFT R0,R0!1 BY -(3)
3955 026246 013767 177776 010214      MOV      @RPS,SPSW     ; SAVE PS
3956 026254 122767 000010 010206      CMPB    #10,SPSW      ; IS THE PS 10?
3957 026262 001404              BEQ      1$
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 760 <====
026264 012742 000614      MOV      #614,-(R2)      ; MOVE TO MAILBOX # ***** 614 *****
026270 005242              INC      -(R2)          ; SET MSGTYP TO FATAL ERROR
026272 000000              HALT                    ; THE PS IS NOT EQUAL TO 10
3958 026274 022700 177525 1$: CMP      #177525,R0      ; IS THE RESULT 177525?
3959 026300 001404              BEQ      2$
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 751 <====
026302 012742 000615      MOV      #615,-(R2)      ; MOVE TO MAILBOX # ***** 615 *****
026306 005242              INC      -(R2)          ; SET MSGTYP TO FATAL ERROR
026310 000000              HALT                    ; R0 IS NOT EQUAL TO 177252
```

```

3960 026312 022701 052525      2$:  CMP      #52525,R1      ;IS THE RESULT 52525?
3961 026316 001404              BEQ      TST264
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---=
;          CONDITIONAL BRANCH INST. AND <---=
;          REPLACE THE MOVE INSTRUCTION <---=
;          WHICH FOLLOWS W/ 742 <---=
026320 012742 000616          MOV      #616,-(R2) ;MOVE TO MAILBOX # ***** 616 *****
026324 005242              INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
026326 000000              HALT     ;R0:1 IS NOT EQUAL TO 52525
; OR SEQUENCE ERROR

```

3962
3963
3964
3965

.SBTTL ASHC SHIFTED BY @-(4) DM5 REG 4
;MODE 5 REG 4

:TEST 264 ASHC 125252 SHIFTED BY @-(4)=177525 52525 PS=10

```

026330 005212              TST264: INC      (R2)      ;UPDATE TEST NUMBER
026332 022712 000264          CMP      #264,(R2)    ;SEQUENCE ERROR?
026336 001035          BNE      TST265-10  ;BR TO ERROR HALT ON SEQ ERROR
3966 026340 012700 125252      MOV      #125252,R0   ;LOAD R0 WITH 125252
3967 026344 012701 125252      MOV      #125252,R1   ;LOAD R0:1 WITH 125252
3968 026350 000241          CLC
3969 026352 012704 036506      MOV      #54+2,R4     ;SET UP R4
3970 026356 073054          ASHC     @-(R4),R0    ;SHIFT R0:1 BY @-(4)
3971 026360 013767 177776 010102  MOV      @#PS,SPSW    ;SAVE PS
3972 026366 122767 000010 010074  CMPB    #10,SPSW     ;IS THE PS 10?
3973 026374 001404              BEQ
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 760 <====
026376 012742 000617          MOV      #617,-(R2)  ;MOVE TO MAILBOX # ***** 617 *****
026402 005242              INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
026404 000000              HALT     ;THE PS IS NOT EQUAL TO 10
3974 026406 022700 177525      1$:  CMP      #177525,R0   ;IS THE RESULT 177525?
3975 026412 001404              BEQ      2$
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 751 <====
026414 012742 000620          MOV      #620,-(R2)  ;MOVE TO MAILBOX # ***** 620 *****
026420 005242              INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
026422 000000              HALT     ;R0 IS NOT EQUAL TO 177525
3976 026424 022701 052525      2$:  CMP      #52525,R1   ;IS THE RESULT 52525?
3977 026430 001404              BEQ      TST265
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 742 <====
026432 012742 000621          MOV      #621,-(R2)  ;MOVE TO MAILBOX # ***** 621 *****
026436 005242              INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
026440 000000              HALT     ;R0:1 IS NOT EQUAL TO 52525
; OR SEQUENCE ERROR

```

3978
3979
3980
3981

.SBTTL ASHC SHIFTED BY @-(4) DM7 REG 4
;MODE 7 REG 4

```

:TEST 265      ASHC 1252552,125252,SHIF,@(4)-177525,52525 PS 10
:*****
TST265: INC      (R2)          ;UPDATE TEST NUMBER
          CMP      #265,(R2)   ;SEQUENCE ERROR?
          BNE     TST266-10    ;BR TO ERROR HALT ON SEQ ERROR
          MOV     #125252,R0   ;LOAD R0 WITH 125252
          MOV     #125252,R1   ;LOAD R0:1 WITH 125252
          CLC
          MOV     #54,R4       ;SET UP R4
          ASHC   @(R4),R0      ;SHIFT R0,R0.1 BY @(4)
          MOV     @#PS,SPSW    ;SAVE PS
          CMPB   #10,SPSW     ;IS THE PS 10?
          BEQ
          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
          ;          CONDITIONAL BRANCH INST. AND <====
          ;          REPLACE THE MOVE INSTRUCTION <====
          ;          WHICH FOLLOWS W/ 757 <====
          MOV     #622,-(R2)   ;MOVE TO MAILBOX # ***** 622 *****
          INC     -(R2)
          HALT
          CMP     #177525,R0   ;SET MSGTYP TO FATAL ERROR
          BEQ     2$          ;THE PS IS NOT EQUAL TO 10
          ; IS THE RESULT 177525?
          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
          ;          CONDITIONAL BRANCH INST. AND <====
          ;          REPLACE THE MOVE INSTRUCTION <====
          ;          WHICH FOLLOWS W/ 750 <====
          MOV     #623,-(R2)   ;MOVE TO MAILBOX # ***** 623 *****
          INC     -(R2)
          HALT
          CMP     #52525,R1    ;SET MSGTYP TO FATAL ERROR
          BEQ     TST266      ;R0 IS NOT EQUAL TO 177525
          ; IS THE RESULT 52525?
          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
          ;          CONDITIONAL BRANCH INST. AND <
          ;          REPLACE THE MOVE INSTRUCTION <
          ;          WHICH FOLLOWS W/ 741 <
          MOV     #624,-(R2)   ;MOVE TO MAILBOX # ***** 624 *****
          INC     -(R2)
          HALT
          ;SET MSGTYP TO FATAL ERROR
          ;R0:1 IS NOT EQUAL TO 52525
          ; OR SEQUENCE ERROR
026442 005212
026444 022712 000265
026450 001036
3982 026452 012700 125252
3983 026456 012701 125252
3984 026462 000241
3985 026464 012704 036504
3986 026470 073074 000000
3987 026474 013767 177776 007766
3988 026502 122767 000010 007760
3989 026510 001404
026512 012742 000622
026516 005242
026520 000000
3990 026522 022700 177525 1$:
3991 026526 001404
026530 012742 000623
026534 005242
026536 000000
3992 026540 022701 052525 2$:
3993 026544 001404
026546 012742 000624
026552 005242
026554 000000
3994
3995
3996
3997
```

.SBTTL ASHC SHIFTED BY -32. DM2 REG 7
;MODE 2 REG 7

```

:*****
:TEST 266      ASHC 100000 0 SHIFTED BY -32.=-1 -1 PS 11
:*****
TST266: INC      (R2)          ;UPDATE TEST NUMBER
          CMP     #266,(R2)   ;SEQUENCE ERROR?
          BNE     TST267-10    ;BR TO ERROR HALT ON SEQ ERROR
          MOV     #100000,R0   ;LOAD R0 WITH 100000
          MOV     #0,R1        ;LOAD R0:1 WITH 0
          CLC
          ASHC   #-32.,R0     ;SHIFT R0,R0.1 BY -32.
          MOV     @#PS,SPSW    ;SAVE PS
          CMPB   #11,SPSW     ;IS THE PS 11?
          BEQ
          ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
          ;          CONDITIONAL BRANCH INST. AND <
026556 005212
026560 022712 000266
026564 001034
3998 026566 012700 100000
3999 026572 012701 000000
4000 026576 000241
4001 026600 073027 177740
4002 026604 013767 177776 007656
4003 026612 122767 000011 007650
4004 026620 001404
```

```

;
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 761 <====
; MOVE TO MAILBOX # ***** 625 *****
; SET MSGTYP TO FATAL ERROR
; THE PS IS NOT EQUAL TO 11
; IS THE RESULT -1?
026622 012742 000625 MOV #625,-(R2)
026626 005242 INC -(R2)
026630 000000 HALT
4005 026632 022700 177777 1$: CMP #-1,R0
4006 026636 001404 BEQ 2$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 752 <====
; MOVE TO MAILBOX # ***** 626 *****
; SET MSGTYP TO FATAL ERROR
; R0 IS NOT EQUAL TO -1
; IS THE RESULT -1?
026640 012742 000626 MOV #626,-(R2)
026644 005242 INC -(R2)
026646 000000 HALT
4007 026650 022701 177777 2$: CMP #-1,R1
4008 026654 001404 BEQ TST267

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 743 <====
; MOVE TO MAILBOX # ***** 627 *****
; SET MSGTYP TO FATAL ERROR
; R0!1 IS NOT EQUAL TO -1
; OR SEQUENCE ERROR
026656 012742 000627 MOV #627,-(R2)
026662 005242 INC -(R2)
026664 000000 HALT

4009
4010
4011
4012
.SBTTL ASHC SHIFTED BY 15 DM2 REG 7
;MODE 2 REG 7
*****
;TEST 267 ASHC 0 -1 SHIFTED BY 15.=77777 100000 PS=0
*****
TST267: INC (R2) ;UPDATE TEST NUMBER
CMP #267,(R2) ;SEQUENCE ERROR?
BNE TST270-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #0,R0 ;LOAD R0 WITH 0
MOV #-1,R1 ;LOAD R0!1 WITH -1
CLC
ASHC #15.,R0 ;SHIFT R0,R0!1 BY 15.
MOV @#PS,SPSW ;SAVE PS
CMPB #0,SPSW ;IS THE PS 0?
BEQ 1$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 761 <====
; MOVE TO MAILBOX # ***** 630 *****
; SET MSGTYP TO FATAL ERROR
; THE PS IS NOT EQUAL TO 0
; IS THE RESULT 77777?
026732 012742 000630 MOV #630,-(R2)
026736 005242 INC -(R2)
026740 000000 HALT
4020 026742 022700 077777 1$: CMP #77777,R0
4021 026746 001404 BEQ 2$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 752 <====
; MOVE TO MAILBOX # ***** 631 *****
; SET MSGTYP TO FATAL ERROR
; R0 IS NOT EQUAL TO 77777
; IS THE RESULT 100000?
026750 012742 000631 MOV #631,-(R2)
026754 005242 INC -(R2)
026756 000000 HALT
4022 026760 022701 100000 2$: CMP #100000,R1

```

4023 026764 001404 BEQ TST270

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <----=
: REPLACE THE MOVE INSTRUCTION <=====
: WHICH FOLLOWS W/ 743 <---=
: MOVE TO MAILBOX # ***** 632 *****
: SET MSGTYP TO FATAL ERROR
: R0!1 IS NOT EQUAL TO 100000
: OR SEQUENCE ERROR

026766 012742 000632 MOV #632,-(R2)
026772 005242 INC -(R2)
026774 000000 HALT

4024
4025
4026
4027

.SBTTL ASHC SHIFTED BY 16. DM2 REG 7
:MODE 2 REG 7

:TEST 270 ASHC 0 52525 SHIFTED BY 16.=52525 0 PS=0

026776 005212
027000 022712 000270
027004 001034
4028 027006 012700 000000
4029 027012 012701 052525
4030 027016 000241
4031 027020 073027 000020
4032 027024 013767 177776 007436
4033 027032 122767 000000 007430
4034 027040 001404

TST270: INC (R2) ;UPDATE TEST NUMBER
CMP #270,(R2) ;SEQUENCE ERROR?
BNE TST271-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #0,R0 ;LOAD R0 WITH 0
MOV #52525,R1 ;LOAD R0!1 WITH 52525
CLC
ASHC #16.,R0 ;SHIFT R0,R0.1 BY 16.
MOV @#PS,SPSW ;SAVE PS
CMPB #0,SPSW ;IS THE PS 0?
BEQ 1\$

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
: CONDITIONAL BRANCH INST. AND <
: REPLACE THE MOVE INSTRUCTION <
: WHICH FOLLOWS W/ 761 <

027042 012742 000633 MOV #633,-(R2)
027046 005242 INC -(R2)
027050 000000 HALT
4035 027052 022700 052525 1\$: CMP #52525,R0
4036 027056 001404 BEQ 2\$

: MOVE TO MAILBOX # ***** 633 *****
: SET MSGTYP TO FATAL ERROR
: THE PS IS NOT EQUAL TO 0
: IS THE RESULT 52525?

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < ==
: CONDITIONAL BRANCH INST. AND < --=
: REPLACE THE MOVE INSTRUCTION <=====
: WHICH FOLLOWS W/ 752 <----=
: MOVE TO MAILBOX # ***** 634 *****

027060 012742 000634 MOV #634,-(R2)
027064 005242 INC -(R2)
027066 000000 HALT
4037 027070 022701 000000 2\$: CMP #0,R1
4038 027074 001404 BEQ TST271

: SET MSGTYP TO FATAL ERROR
: R0 IS NOT EQUAL TO 52525
: IS THE RESULT 0?

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <=====
: CONDITIONAL BRANCH INST. AND <=====
: REPLACE THE MOVE INSTRUCTION <=====
: WHICH FOLLOWS W/ 743 <=====
: MOVE TO MAILBOX # ***** 635 *****

027076 012742 000635 MOV #635,-(R2)
027102 005242 INC -(R2)
027104 000000 HALT

: SET MSGTYP TO FATAL ERROR
: R0!1 IS NOT EQUAL TO 0
: OR SEQUENCE ERROR

4039
4040

:MODE 2 REG 7

4042

:TEST 271 ASHC -1 0 SHIFTED BY 16. =0 0 PS=7

027106 005212
027110 022712 000271
027114 001034
4043 027116 012700 177777
4044 027122 012701 000000
4045 027126 000241
4046 027130 073027 000020
4047 027134 013767 177776 007326
4048 027142 122767 000007 007320
4049 027150 001404

TST271: INC (R2) ;UPDATE TEST NUMBER
CMP #271,(R2) ;SEQUENCE ERROR?
BNE TST272-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #-1,R0 ;LOAD R0 WITH -1
MOV #0,R1 ;LOAD R0!1 WITH 0
CLC
ASHC #16.,R0 ;SHIFT R0,R0!1 BY 16.
MOV @#PS,SPSW ;SAVE PS
CMPB #7,SPSW ;IS THE PS 7?
BEQ 1\$

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 761 <====

027152 012742 000636
027156 005242
027160 000000
4050 027162 022700 000000
4051 027166 001404

MOV #636,-(R2) ;MOVE TO MAILBOX # ***** 636 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;THE PS IS NOT EQUAL TO 7
1\$: CMP #0,R0 ;IS THE RESULT 0?
BEQ 2\$

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 752 <====

027170 012742 000637
027174 005242
027176 000000
4052 027200 022701 000000
4053 027204 001404

MOV #637,-(R2) ;MOVE TO MAILBOX # ***** 637 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;R0 IS NOT EQUAL TO 0
2\$: CMP #0,R1 ;IS THE RESULT 0?
BEQ TST272

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 743 <====

027206 012742 000640
027212 005242
027214 000000

MOV #640,-(R2) ;MOVE TO MAILBOX # ***** 640 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;R0!1 IS NOT EQUAL TO 0
; OR SEQUENCE ERROR

4054
4055
4056
4057

.SBTTL ASHC SHIFTED BY @ (4) DM7 REG 4
:MODE 7 REG 4

:TEST 272 ASHC 125252,125252,SHIF,@(4)=177525,52525 PS 10

027216 005212
027220 022712 000272
027224 001036
4058 027226 012701 125252
4059 027232 012700 125252
4060 027236 000241
4061 027240 012704 036504
4062 027244 073074 000000
4063 027250 013767 177776 007212
4064 027256 122767 000010 007204
4065 027264 001404

TST272: INC (R2) ;UPDATE TEST NUMBER
CMP #272,(R2) ;SEQUENCE ERROR?
BNE TST273-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #125252,R1 ;LOAD R0!1 WITH 125252
MOV #125252,R0 ;LOAD R0 WITH 125252
CLC
MOV #S4,R4 ;SET UP R4
ASHC @ (R4),R0 ;SHIFT R0,R0!1 BY @ (4)
MOV @#PS,SPSW ;SAVE PS
CMPB #10,SPSW ;IS THE PS 10?
BEQ 1\$

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <----

027266 012742 000641
027272 005242
027274 000000
4066 027276 022701 052525
4067 027302 001404

1S:

MOV #641,-(R2)
INC -(R2)
HALT
CMP #52525,R1
BEQ 2S

:
: CONDITIONAL BRANCH INST. AND <----
: REPLACE THE MOVE INSTRUCTION <----
: WHICH FOLLOWS W/ 757 <----
: MOVE TO MAILBOX # ***** 641 *****
: SET MSGTYP TO FATAL ERROR
: THE PS IS NOT EQUAL TO 10
: IS THE RESULT 52525?

027304 012742 000642
027310 005242
027312 000000
4068 027314 022700 177525
4069 027320 001404

2S:

MOV #642,-(R2)
INC -(R2)
HALT
CMP #177525,R0
BEQ TST273

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 750 <====
: MOVE TO MAILBOX # ***** 642 *****
: SET MSGTYP TO FATAL ERROR
: R0!1 IS NOT EQUAL TO 52525
: IS THE RESULT 177525?

027322 012742 000643
027326 005242
027330 000000

MOV #643,-(R2)
INC -(R2)
HALT

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 741 <====
: MOVE TO MAILBOX # ***** 643 *****
: SET MSGTYP TO FATAL ERROR
: R0 IS NOT EQUAL TO 177525
: OR SEQUENCE ERROR

4070
4071

4073
4074

.SBTTL MUL USING DM2 REG 7

:TEST 273 MUL 100000 * #100000 = 40000 0 PS-1

027332 005212
027334 022712 000273
027340 001035
4075 027342 012706 001000
4076 027346 005037 177776
4077 027352 012700 100000
4078 027356 070027 100000
4079 027362 013767 177776 007100
4080 027370 122767 000001 007072
4081 027376 001404

TST273: INC (R2) ;UPDATE TEST NUMBER
CMP #273,(R2) ;SEQUENCE ERROR?
BNE TST274-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #STBOT,R6
CLR @PS
MOV #100000,R0 ;LOAD MULTIPLICAN WITH 100000
MUL #100000,R0 ;MULTIPLY 100000 BY #100000
MOV @PS,SPSW ;SAVE PS
CMPB #1,SPSW ;IS PS = 1
BEQ 1\$

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
: CONDITIONAL BRANCH INST. AND <
: REPLACE THE MOVE INSTRUCTION <
: WHICH FOLLOWS W/ 760 < -

027400 012742 000644
027404 005242
027406 000000
4082 027410 022700 040000
4083 027414 001404

MOV #644,-(R2) ;MOVE TO MAILBOX # ***** 644 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;PS IS WRONG
1\$: CMP #40000,R0 ;IS HIGH ORDER = 40000
BEQ 2\$

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 751 <====

027416 012742 000645
027422 005242
027424 000000
4084 027426 022701 000000
4085 027432 001404

MOV #645,-(R2) ;MOVE TO MAILBOX # ***** 645 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;HIGH ORDER IS WRONG
2\$: CMP #0,R1 ;IS LOWER = 0
BEQ TST274

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
: CONDITIONAL BRANCH INST. AND <====
: REPLACE THE MOVE INSTRUCTION <====
: WHICH FOLLOWS W/ 742 <====

027434 012742 000646
027440 005242
027442 000000

MOV #646,-(R2) ;MOVE TO MAILBOX # ***** 646 *****
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;LOW ORDER IS WRONG
; OR SEQUENCE ERROR

4086
4087
4088
4089

;TEST MSB OF LOWER PRODUCT CLEAR C-BIT
;MODE 2 /REG7

:TEST 274 MUL 70707 * #70707 = 31221 44261 PS-1

027444 005212
027446 022712 000274
027452 001031
4090 027454 012700 070707
4091 027460 070027 070707
4092 027464 013767 177776 006776
4093 027472 122767 000001 006770
4094 027500 001404

TST274: INC (R2) ;UPDATE TEST NUMBER
CMP #274,(R2) ;SEQUENCE ERROR?
BNE TST275-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #70707,R0 ;LOAD MULTIPLICAN WITH 70707
MUL #70707,R0 ;MULTIPLY 70707 BY #70707
MOV @PS,SPSW ;SAVE PS = 1
CMPB #1,SPSW ;IS PS = 1
BEQ 1\$

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
: CCNDITIONAL BRANCH INST. AND <
: REPLACE THE MOVE INSTRUCTION <


```
027502 012742 000647      MOV      #647,-(R2)      ; WHICH FOLLOWS W/ 764      <===
027506 005242      INC      -(R2)          ; MOVE TO MAILBOX # ***** 647 *****
027510 000000      HALT                                ; SET MSGTYP TO FATAL ERROR
4095 027512 022700 031221 1$:  CMP      #31221,R0      ; PS IS WRONG
4096 027516 001404      BEQ      2$            ; IS HIGH ORDER =31221
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS      <====
                                ; CONDITIONAL BRANCH INST. AND                      <====
                                ; REPLACE THE MOVE INSTRUCTION                      <====
                                ; WHICH FOLLOWS W/ 755                          <====

027520 012742 000650      MOV      #650,-(R2)     ; MOVE TO MAILBOX # ***** 650 *****
027524 005242      INC      -(R2)          ; SET MSGTYP TO FATAL ERROR
027526 000000      HALT                                ; HIGH ORDER IS WRONG
4097 027530 022701 044261 2$:  CMP      #44261,R1     ; IS LOWER ORDER =44261
4098 027534 001404      BEQ      TST275        ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS      <====
                                ; CONDITIONAL BRANCH INST. AND                      <====
                                ; REPLACE THE MOVE INSTRUCTION                      <====
                                ; WHICH FOLLOWS W/ 746                          <====

027536 012742 000651      MOV      #651,-(R2)     ; MOVE TO MAILBOX # ***** 651 *****
027542 005242      INC      -(R2)          ; SET MSGTYP TO FATAL ERROR
027544 000000      HALT                                ; LOW ORDER IS WRONG
                                ; OR SEQUENCE ERROR

4099
4100                                ; STORE UPPER PRODUCT AND SET C-BIT NEG #
4101                                ; MODE 2 /REG 7
4102

*****
; TEST 275      MUL 107070 * #107070 = 31222 26100 PS=1
*****
TST275: INC      (R2)          ; UPDATE TEST NUMBER
        CMP      #275,(R2)     ; SEQUENCE ERROR?
        BNE     TST276-10      ; BR TO ERROR HALT ON SEQ ERROR
        MOV     #107070,R0      ; LOAD MULTIPLICAN WITH 107070
        MUL     #107070,R0      ; MULTIPLY 107070 BY #107070
        MOV     @#PS,SPSW      ; SAVE PS
        CMPS   #1,SPSW         ; IS PS = 1
        BEQ     1$

                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS      <=
                                ; CONDITIONAL BRANCH INST. AND                      <
                                ; REPLACE THE MOVE INSTRUCTION                      <-
                                ; WHICH FOLLOWS W/ 764                          <-

027604 012742 000652      MOV      #652,-(R2)     ; MOVE TO MAILBOX # ***** 652 *****
027610 005242      INC      -(R2)          ; SET MSGTYP TO FATAL ERROR
027612 000000      HALT                                ; PS IS WRONG
4108 027614 022700 031222 1$:  CMP      #31222,R0      ; IS HIGH ORDER = 31222
4109 027620 001404      BEQ      2$            ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS      <-
                                ; CONDITIONAL BRANCH INST. AND                      <
                                ; REPLACE THE MOVE INSTRUCTION                      <-
                                ; WHICH FOLLOWS W/ 755                          <-

027622 012742 000653      MOV      #653,-(R2)     ; MOVE TO MAILBOX # ***** 653 *****
027626 005242      INC      -(R2)          ; SET MSGTYP TO FATAL ERROR
027630 000000      HALT                                ;
4110 027632 022701 026100 2$:  CMP      #26100,R1     ; IS LOW ORDER = 26100
4111 027636 001404      BEQ      TST276        ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS      <=-
                                ; CONDITIONAL BRANCH INST. AND                      <-
```

```
027640 012742 000654      MOV    #654,-(R2)      ;REPLACE THE MOVE INSTRUCTION <====  
027644 005242              INC    -(R2)          ;WHICH FOLLOWS W/ 746 <====  
027646 000000              HALT                   ;MOVE TO MAILBOX # ***** 654 *****  
                                ;SET MSGTYP TO FATAL ERROR  
                                ;LOW ORDER IS WRONG  
                                ;OR SEQUENCE ERROR  
4112                                ;PUT MULTIPLER IN R12 AND TEST SIGN FOR ZERO  
4113                                ;MODE 2 /REG 7  
4114
```

```
*****  
:TEST 276      MUL 77777 * #100000 = 140000 100000 PS=11  
*****
```

```
TST276: INC    (R2)          ;UPDATE TEST NUMBER  
        CMP    #276,(R2)    ;SEQUENCE ERROR?  
        BNE    TST277-10    ;BR TO ERROR HALT ON SEQ ERROR  
4115 027660 012700 077777    MOV    #77777,R0        ;LOAD MULTIPLICAN WITH 77777  
4116 027664 070027 100000    MUL    #100000,R0      ;MULTIPLY 77777 BY #100000  
4117 027670 013767 177776 006572    MOV    @#PS,SPSW      ;SAVE PS  
4118 027676 122767 000011 006564    CMPB   #11,SPSW      ;IS PS = 11  
4119 027704 001404          BEQ    1$
```

```
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
CONDITIONAL BRANCH INST. AND <====  
REPLACE THE MOVE INSTRUCTION <====  
WHICH FOLLOWS W/ 764 <====
```

```
027706 012742 000655      MOV    #655,-(R2)      ;MOVE TO MAILBOX # ***** 655 *****  
027712 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR  
027714 000000              HALT                   ;PS IS WRONG  
4120 027716 022700 140000    1$:  CMP    #140000,R0    ;IS HIGH ORDER = 140000  
4121 027722 001404          BEQ    2$
```

```
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
CONDITIONAL BRANCH INST. AND <====  
REPLACE THE MOVE INSTRUCTION <====  
WHICH FOLLOWS W/ 755 <====
```

```
027724 012742 000656      MOV    #656,-(R2)      ;MOVE TO MAILBOX # ***** 656 *****  
027730 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR  
027732 000000              HALT                   ;HIGH ORDER IS WRONG  
4122 027734 022701 100000    2$:  CMP    #100000,R1    ;IS LOW ORDER = 100000  
4123 027740 001404          BEQ    TST277
```

```
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
CONDITIONAL BRANCH INST. AND <====  
REPLACE THE MOVE INSTRUCTION <====  
WHICH FOLLOWS W/ 746 <====
```

```
027742 012742 000657      MOV    #657,-(R2)      ;MOVE TO MAILBOX # ***** 657 *****  
027746 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR  
027750 000000              HALT                   ;LOW ORDER IS WRONG  
                                ;OR SEQUENCE ERROR
```

```
4124                                ;SET Z-BIT R17 = 0  
4125                                ;MODE 2 /REG 7  
4126  
4127
```

```
*****  
:TEST 277      MUL -1 * #0 = 0 0 PS=4  
*****
```

```
TST277: INC    (R2)          ;UPDATE TEST NUMBER  
        CMP    #277,(R2)    ;SEQUENCE ERROR?  
        BNE    TST300-10    ;BR TO ERROR HALT ON SEQ ERROR  
4128 027762 012700 177777    MOV    #-1,R0        ;LOAD MULTIPLICAN WITH -1  
4129 027766 070027 000000    MUL    #0,R0        ;MULTIPLY -1 BY #0  
4130 027772 013767 177776 006470    MOV    @#PS,SPSW      ;SAVE PS
```

```
4131 030000 122767 000004 006462      CMPB  #4,SPSW      ;IS PS = 4
4132 030006 001404                      BEQ   1$
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 764 <====
030010 012742 000660      MOV   #660,-(R2)   ;MOVE TO MAILBOX # ***** 660 *****
030014 005242      INC   -(R2)        ;SET MSGTYP TO FATAL ERROR
030016 000000      HALT                ;PS IS WRONG
4133 030020 022700 000000      1$:  CMP   #0,R0     ;IS HIGH ORDER = 0
4134 030024 001404      BEQ   2$
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====
030026 012742 000661      MOV   #661,-(R2)   ;MOVE TO MAILBOX # ***** 661 *****
030032 005242      INC   -(R2)        ;SET MSGTYP TO FATAL ERROR
030034 000000      HALT                ;HIGH ORDER IS WRONG
4135 030036 022701 000000      2$:  CMP   #0,R1     ;IS LOW ORDER = 0
4136 030042 001404      BEQ   TST300
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 746 <====
030044 012742 000662      MOV   #662,-(R2)   ;MOVE TO MAILBOX # ***** 662 *****
030050 005242      INC   -(R2)        ;SET MSGTYP TO FATAL ERROR
030052 000000      HALT                ;LOW ORDER IS WRONG
; OR SEQUENCE ERROR
```

4137
4138
4139
4140

;TEST UPPER PRODUCT FOR ALL ONES,CLEAR C-BIT
;MODE 2 /REG 7

;TEST 300 MUL -1 * #77777 = 100001 100001 PS=10

```
TST300: 030054 005212      INC   (R2)          ;UPDATE TEST NUMBER
030056 022712 000300      CMP   #300,(R2)    ;SEQUENCE ERROR?
030062 001031      BNE  TST301-10     ;BR TO ERROR HALT ON SEQ ERROR
4141 030064 012701 177777      MOV   #-1,R1       ;LOAD MULTIPLICAN WITH -1
4142 030070 070127 077777      MUL   #77777,R1    ;MULTIPLY -1 BY #77777
4143 030074 013767 177776 006366      MOV   @#PS,SPSW    ;SAVE PS
4144 030102 122767 000010 006360      CMPB  #10,SPSW     ;IS PS = 10
4145 030110 001404      BEQ   1$
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 764 <====
```

```
030112 012742 000663      MOV   #663,-(R2)   ;MOVE TO MAILBOX # ***** 663 *****
030116 005242      INC   -(R2)        ;SET MSGTYP TO FATAL ERROR
030120 000000      HALT                ;PS IS WRONG
4146 030122 022701 100001      1$:  CMP   #100001,R1  ;IS HIGH ORDER = 100001
4147 030126 001404      BEQ   2$
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====
```

```
030130 012742 000664      MOV   #664,-(R2)   ;MOVE TO MAILBOX # ***** 664 *****
030134 005242      INC   -(R2)        ;SET MSGTYP TO FATAL ERROR
```

```

4148 030136 000000          HALT
4149 030140 022701 100001 2$:  CMP      #100001,R1    ;HIGH ORDER IS WRONG
                                BEQ      TST301    ;IS LOW ORDER = 100001
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 746 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 665 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; LOW ORDER IS WRONG
                                ; OR SEQUENCE ERROR
030146 012742 000665          MOV      #665,-(R2)
030152 005242          INC      -(R2)
030154 000000          HALT

```

```

4150          ;MODE 2 /REG 7
4151          :*****
          :TEST 301          MUL 2 * #2 = 0 4 PS=0
          :*****

```

```

          TST301: INC      (R2)          ;UPDATE TEST NUMBER
030156 005212          CMP      #301,(R2)    ;SEQUENCE ERROR?
030160 022712 000301          BNE     TST302-10   ;BR TO ERROR HALT ON SEQ ERROR
4152 030166 012700 000002          MOV      #2,R0        ;LOAD MULTIPLICAN WITH 2
4153 030172 070027 000002          MUL      #2,R0        ;MULTIPLY 2 BY #2
4154 030176 013767 177776 006264          MOV      @#PS,SPSW    ;SAVE PS
4155 030204 122767 000000 006256          CMPB    #0,SPSW      ;IS PS=0
4156 030212 001404          BFG     1$

```

```

                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 764 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 666 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; PS IS WRONG
                                ; IS HIGH ORDER =0
030214 012742 000666          MOV      #666,-(R2)
030220 005242          INC      -(R2)
030222 000000          HALT
4157 030224 022700 000000 1$:  CMP      #0,R0
4158 030230 001404          BEQ     2$

```

```

                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 755 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 667 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; HIGH ORDER IS WRONG
                                ; IS LOW ORDER =4
030232 012742 000667          MOV      #667,-(R2)
030236 005242          INC      -(R2)
030240 000000          HALT
4159 030242 022701 000004 2$:  CMP      #4,R1
4160 030246 001404          BEQ     TST302

```

```

                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
                                ;          CONDITIONAL BRANCH INST. AND <====
                                ;          REPLACE THE MOVE INSTRUCTION <====
                                ;          WHICH FOLLOWS W/ 746 <====
                                ;
                                ; MOVE TO MAILBOX # ***** 670 *****
                                ; SET MSGTYP TO FATAL ERROR
                                ; LOW ORDER IS WRONG
                                ; OR SEQUENCE ERROR
030250 012742 000670          MOV      #670,-(R2)
030254 005242          INC      -(R2)
030256 000000          HALT

```

```

4161          ;MODE 2 /REG 7
4162          :*****
4163          :TEST 302          MUL 1 * #-1 = -1 -1 PS=10
          :*****

```

```

          TST302: INC      (R2)          ;UPDATE TEST NUMBER
030260 005212          CMP      #302,(R2)    ;SEQUENCE ERROR?
030262 022712 000302          BNE     TST303-10   ;BR TO ERROR HALT ON SEQ ERROR
030266 001031

```

```

4164 030270 012700 000001      MOV    #1,R0      ;LOAD MULTIPLICAN WITH 1
4165 030274 070027 177777      MUL    #-1,R0    ;MULTIPLY 1 BY #-1
4166 030300 013767 177776 006162  MOV    @#PS,SPSW ;SAVE PS
4167 030306 122767 000010 006154  CMPB  #10,SPSW   ;IS PS =10
4168 030314 001404                      BEQ    1$

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <---
; REPLACE THE MOVE INSTRUCTION <--=
; WHICH FOLLOWS W/ 764 <-

```

```

030316 012742 000671      MOV    #671,-(R2) ;MOVE TO MAILBOX # ***** 671 *****
030322 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
030324 000000      HALT                      ;PS IS WRONG
4169 030326 022700 177777      1$:  CMP    #-1,R0     ;IS HIGH ORDER =-1
4170 030332 001404                      BEQ    2$

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
; CONDITIONAL BRANCH INST. AND <
; REPLACE THE MOVE INSTRUCTION <
; WHICH FOLLOWS W/ 755 <

```

```

030334 012742 000672      MOV    #672,-(R2) ;MOVE TO MAILBOX # ***** 672 *****
030340 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
030342 000000      HALT                      ;HIGH ORDER IS WRONG
4171 030344 022701 177777      2$:  CMP    #-1,R1    ;IS LOW ORDER =-1
4172 030350 001404                      BEQ    TST303

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- --
; CONDITIONAL BRANCH INST. AND <- --
; REPLACE THE MOVE INSTRUCTION < =-
; WHICH FOLLOWS W/ 746 <-

```

```

030352 012742 000673      MOV    #673,-(R2) ;MOVE TO MAILBOX # ***** 673 *****
030356 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
030360 000000      HALT                      ;LOW ORDER IS WRONG
; OR SEQUENCE ERROR

```

```

4173 ;MODE 2 /REG 7
4174

```

```

;*****
;TEST 303 MUL -1 * #100000 = 0 100000 PS=1
;*****

```

```

030362 005212      TST303: INC    (R2)      ;UPDATE TEST NUMBER
030364 022712 000303      CMP    #303,(R2) ;SEQUENCE ERROR?
030370 001031      BNE    TST304-10 ;BR TO ERROR HALT ON SEQ ERROR
4175 030372 012700 177777      MOV    #-1,R0     ;LOAD MULTIPLICAN WITH -1
4176 030376 070027 100000      MUL    #100000,R0 ;MULTIPLY -1 BY 100000
4177 030402 013767 177776 006060  MOV    @#PS,SPSW ;SAVE PS
4178 030410 122767 000001 006052  CMPB  #1,SPSW   ;IS PS =1
4179 030416 001404                      BEQ    1$

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 764 <====

```

```

030420 012742 000674      MOV    #674,-(R2) ;MOVE TO MAILBOX # ***** 674 *****
030424 005242      INC    -(R2)      ;SET MSGTYP TO FATAL ERROR
030426 000000      HALT                      ;PS IS WRONG
4180 030430 022700 000000      1$:  CMP    #0,R0     ;IS HIGH ORDER = 0
4181 030434 001404                      BEQ    2$

```

```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 755 <====

```

```

030436 012742 000675      MOV    #675,-(R2) ;MOVE TO MAILBOX # ***** 675 *****

```

```

030442 005242      INC      -(R2)      ;SET MSGTYP TO FATAL ERROR
030444 000000      HALT     ;HIGH ORDER IS WRONG
4182 030446 022701 100000 2$:  CMP      #100000,R1  ;IS LOW ORDER =100000
4183 030452 001404      BEQ     TST304
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 746 <====
030454 012742 000676      MOV     #676,-(R2)  ;MOVE TO MAILBOX # ***** 676 *****
030460 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
030462 000000      HALT     ;LOW ORDER IS WRONG
; OR SEQUENCE ERROR

```

4184
4185
4186
4187
4188

.SBTTL DIV USING DM6 REG 7

```

*****
;TEST 304      DIV 0 52525/S1=25252 REM=1 PS=0
*****

```

```

030464 005212      INC     (R2)      ;UPDATE TEST NUMBER
030466 022712 000304      CMP     #304,(R2)  ;SEQUENCE ERROR?
030472 001035      BNE     TST305-10 ;BR TO ERROR HALT ON SEQ ERROR
4189 030474 012700 000000      MOV     #0,R0      ;LOAD HIGH ORDER WITH R0
4190 030500 012701 052525      MOV     #52525,R1  ;LOAD LOW ORDER WITH 52525
4191 030504 012703 036476      MOV     #S1,R3     ;SET UP R3
4192 030510 071067 005762      DIV     S1,R0      ;DIVIDE BY S1
4193 030514 013767 177776 005746      MOV     @PS,SPSW   ;SAVE PS
4194 030522 122767 000000 005740      CMPB   #0,SPSW    ;IS PS=0
4195 030530 001404      BEQ     1$
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 760 <====
030532 012742 000677      MOV     #677,-(R2)  ;MOVE TO MAILBOX # ***** 677 *****
030536 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
4196 030542 022700 025252 1$:  CMP     #25252,R0  ;PS IS WRONG
4197 030546 001404      BEQ     2$      ;IS QUOTIENT =25252
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 751 <====
030550 012742 000700      MOV     #700,-(R2)  ;MOVE TO MAILBOX # ***** 700 *****
030554 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
030556 000000      HALT     ;QUOTIENT IS WRONG
4198 030560 022701 000001 2$:  CMP     #1,R1      ;IS REMAINDER =1
4199 030564 001404      BEQ     TST305
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 742 <====
030566 012742 000701      MOV     #701,-(R2)  ;MOVE TO MAILBOX # ***** 701 *****
030572 005242      INC     -(R2)      ;SET MSGTYP TO FATAL ERROR
030574 000000      HALT     ;REMAINDER IS WRONG
; OR SEQUENCE ERROR

```

4200
4201

.SBTTL DIV USING DM7 REG 7

4202

030576 005212
030600 022712 000305
030604 001035
4203 030606 012700 000000
4204 030612 012701 052525
4205 030616 012704 036500
4206 030622 071077 005652
4207 030626 013767 177776 005634
4208 030634 122767 000000 005626
4209 030642 001404

```
*****  
:TEST 305      DIV 0 52525/@S2 =25252 REM=1 PS=0  
*****  
TST305: INC      (R2)          ;UPDATE TEST NUMBER  
          CMP      #305,(R2)    ;SEQUENCE ERROR?  
          BNE     TST306-10     ;BR TO ERROR HALT ON SEQ ERROR  
          MOV     #0,R0         ;LOAD HIGH ORDER WITH R0  
          MOV     #52525,R1     ;LOAD LOW ORDER WITH 52525  
          MOV     #S2,R4       ;SET UP R4  
          DIV     @S2,R0        ;DIVIDE BY @S2  
          MOV     @WPS,SPSW     ;SAVE PS  
          CMPB   #0,SPSW       ;IS PS=0  
          BEQ    1$
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 760 <====
```

030644 012742 000702
030650 005242
030652 000000
4210 030654 022700 025252
4211 030660 001404

```
MOV      #702,-(R2)          ;MOVE TO MAILBOX # ***** 702 *****  
INC      -(R2)              ;SET MSGTYP TO FATAL ERROR  
HALT  
1$: CMP    #25252,R0        ;PS IS WRONG  
        BEQ    2$          ;IS QUOTIENT =25252
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 751 <====
```

030662 012742 000703
030666 005242
030670 000000
4212 030672 022701 000001
4213 030676 001404

```
MOV      #703,-(R2)          ;MOVE TO MAILBOX # ***** 703 *****  
INC      -(R2)              ;SET MSGTYP TO FATAL ERROR  
HALT  
2$: CMP    #1,R1           ;QUOTIENT IS WRONG  
        BEQ    TST306     ;IS REMINDER =1
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 742 <====
```

030700 012742 000704
030704 005242
030706 000000

```
MOV      #704,-(R2)          ;MOVE TO MAILBOX # ***** 704 *****  
INC      -(R2)              ;SET MSGTYP TO FATAL ERROR  
HALT  
; REMINDER IS WRONG  
; OR SEQUENCE ERROR
```

4214
4215
4216
4217
4218

.SBTTL DIV USING DM3 REG 7
;CLEAR V-BIT MODE 3 REG 7

030710 005212
030712 022712 000306
030716 001035
4219 030720 012700 000000
4220 030724 012701 052525
4221 030730 012703 036476
4222 030734 071037 036476
4223 030740 013767 177776 005522
4224 030746 122767 000000 005514
4225 030754 001404

```
*****  
:TEST 306      DIV 0 52525/@S1 =25252 REM=1 PS=0  
*****  
TST306: INC      (R2)          ;UPDATE TEST NUMBER  
          CMP      #306,(R2)    ;SEQUENCE ERROR?  
          BNE     TST307-10     ;BR TO ERROR HALT ON SEQ ERROR  
          MOV     #0,R0         ;LOAD HIGH ORDER WITH R0  
          MOV     #52525,R1     ;LOAD LOW ORDER WITH 52525  
          MOV     #S1,R3       ;SET UP R3  
          DIV     @S1,R0        ;DIVIDE BY @S1  
          MOV     @WPS,SPSW     ;SAVE PS  
          CMPB   #0,SPSW       ;IS PS=0  
          BEQ    1$
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
```

```

:          CONDITIONAL BRANCH INST. AND <---
:          REPLACE THE MOVE INSTRUCTION <---
:          WHICH FOLLOWS W/ 760 <----
030756 012742 000705      MOV      #705,-(R2)      ;MOVE TO MAILBOX # ***** 705 *****
030762 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
030764 000000              HALT                    ;PS IS WRONG
4226 030766 022700 025252 1$:  CMP      #25252,R0      ;IS QUOTIENT =25252
4227 030772 001404              BEQ      2$

:          TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <----
:          WHICH FOLLOWS W/ 751 <====
030774 012742 000706      MOV      #706,-(R2)      ;MOVE TO MAILBOX # ***** 706 *****
031000 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
031002 000000              HALT                    ;QUOTIENT IS WRONG
4228 031004 022701 000001 2$:  CMP      #1,R1          ;IS REMAINDER =1
4229 031010 001404              BEQ      TST307

:          TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
:          CONDITIONAL BRANCH INST. AND <====
:          REPLACE THE MOVE INSTRUCTION <----
:          WHICH FOLLOWS W/ 742 <----
031012 012742 000707      MOV      #707,-(R2)      ;MOVE TO MAILBOX # ***** 707 *****
031016 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
031020 000000              HALT                    ;REMAINDER IS WRONG
:          OR SEQUENCE ERROR

```

4230
4231
4232
4233

.SBTTL DIV USING DMO REG 4
 ;MODE 0 REG 2

 ;TEST 307 DIV 0 52525 /R4=25252 REM=1 PS=0

```

TST307: INC      (R2)          ;UPDATE TEST NUMBER
          CMP      #307,(R2)   ;SEQUENCE ERROR?
          BNE     TST310-10    ;BR TO ERROR HALT ON SEQ ERROR
4234 031032 012700 000000      MOV      #0,R0          ;LOAD HIGH ORDER WITH 0
4235 031036 012701 052525      MOV      #52525,R1      ;LOAD LOW ORDER WITH 52525
4236 031042 012704 000002      MOV      #2,R4          ; SET UP R4
4237 031046 071004              DIV      R4,R0          ;DIVIDE BY R4
4238 031050 013767 177776 005412 MOV      @PS,SPSW       ;GET PS
4239 031056 122767 000000 005404 CMPB     #0,SPSW        ;IS PS=0
4240 031064 001404              BEQ      1$

:          TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
:          CONDITIONAL BRANCH INST. AND <
:          REPLACE THE MOVE INSTRUCTION <
:          WHICH FOLLOWS W/ 761 <

```

```

031066 012742 000710      MOV      #710,-(R2)      ;MOVE TO MAILBOX # ***** 710 *****
031072 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
031074 000000              HALT                    ;PS IS WRONG
4241 031076 022700 025252 1$:  CMP      #25252,R0      ;IS QUOTIENT=25252
4242 031102 001404              BEQ      2$

:          TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---
:          CONDITIONAL BRANCH INST. AND <
:          REPLACE THE MOVE INSTRUCTION <
:          WHICH FOLLOWS W/ 752 <
031104 012742 000711      MOV      #711,-(R2)      ;MOVE TO MAILBOX # ***** 711 *****
031110 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
031112 000000              HALT                    ;QUOTIENT IS WRONG

```



```
4243 031114 022701 000001 2$: CMP #1,R1 ;IS REMAINDER 1
4244 031120 001404 BEQ TST310 ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 743 <====
031122 012742 000712 MOV #712,-(R2) ;MOVE TO MAILBOX # ***** 712 *****
031126 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
031130 000000 HALT ;REMAINDER IS WRONG
; OR SEQUENCE ERROR
```

4245
4246
4247
4248

.SBTTL DIV USING DM4 REG 3
;MODE 4 REG 3

;TEST 310 DIV 0 52525 /-(3)=25252 REM=1 PS=0

```
TST310: 031132 005212 INC (R2) ;UPDATE TEST NUMBER
031134 022712 000310 CMP #310,(R2) ;SEQUENCE ERROR?
031140 001034 BNE TST311-10 ;BR TO ERROR HALT ON SEQ ERROR
4249 031142 012700 000000 MOV #0,R0 ;LOAD HIGH ORDER WITH 0
4250 031146 012701 052525 MOV #52525,R1 ;LOAD LOW ORDER WITH 52525
4251 031152 012703 036500 MOV #S1+2,R3 ;SET UP R3
4252 031156 071043 DIV -(R3),R0 ;DIVIDE BY -(3)
4253 031160 013767 177776 005302 MOV @PS,SPSW ;SAVE PS
4254 031166 122767 000000 005274 CMPB #0,SPSW ;IS PS=0
4255 031174 001404 BEQ 1$ ;
```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- - -
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 761 <====

```
031176 012742 000713 MOV #713,-(R2) ;MOVE TO MAILBOX # ***** 713 *****
031202 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
031204 000000 HALT ;PS IS WRONG
4256 031206 022700 025252 1$: CMP #25252,R0 ;IS QUOTIENT =25252
4257 031212 001404 BEQ 2$ ;
```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 752 <====

```
031214 012742 000714 MOV #714,-(R2) ;MOVE TO MAILBOX # ***** 714 *****
031220 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
031222 000000 HALT ;QUOTIENT IS WRONG
4258 031224 022701 000001 2$: CMP #1,R1 ;IS REMAINDER =1
4259 031230 001404 BEQ TST311 ;
```

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 743 <====

```
031232 012742 000715 MOV #715,-(R2) ;MOVE TO MAILBOX # ***** 715 *****
031236 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
031240 000000 HALT ;REMAINDER IS WRONG
; OR SEQUENCE ERROR
```

4260
4261
4262

.SBTTL DIV USING DM5 REG 4
;MODE 5 REG 4

;TEST 311 DIV 0 52525 /@-(4)-25252 REM 1 PS=0

```
031242 005212          TST311: INC      (R2)          ;UPDATE TEST NUMBER
031244 022712 000311  CMP      #311,(R2)      ;SEQUENCE ERROR?
031250 001034          BNE      TST312-10     ;BR TO ERROR HALT ON SEQ ERROR
4263 031252 012700 000000 MOV     #0,R0          ;LOAD HIGH ORDER WITH 0
4264 031256 012701 052525 MOV     #52525,R1      ;LOAD LOW ORDER WITH 52525
4265 031262 012704 036502 MOV     #S2+2,R4       ;SET UP R4
4266 031266 071054          DIV     @-(R4),R0      ;DIVIDE BY a-(4)
4267 031270 013767 177776 005172 MOV     @#PS,SPSW      ;SAVE PS
4268 031276 122767 000000 005164 CMPS   #0,SPSW        ;IS PS = 0
4269 031304 001404          BEQ     1$           ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 761 <====
031306 012742 000716          MOV     #716,-(R2)     ;MOVE TO MAILBOX # ***** 716 *****
031312 005242          INC     -(R2)        ;SET MSGTYP TO FATAL ERROR
031314 000000          HALT                    ;PS IS WRONG
4270 031316 022700 025252 1$:  CMP     #25252,R0     ;IS QUOTIENT =25252
4271 031322 001404          BEQ     2$           ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 752 <====
031324 012742 000717          MOV     #717,-(R2)     ;MOVE TO MAILBOX # ***** 717 *****
031330 005242          INC     -(R2)        ;SET MSGTYP TO FATAL ERROR
031332 000000          HALT                    ;QUOTIENT IS WRONG
4272 031334 022701 000001 2$:  CMP     #1,R1          ;IS REMAINDER =1
4273 031340 001404          BEQ     TST312       ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-
;          CONDITIONAL BRANCH INST. AND <-
;          REPLACE THE MOVE INSTRUCTION <-
;          WHICH FOLLOWS W/ 743 <-
031342 012742 000720          MOV     #720,-(R2)     ;MOVE TO MAILBOX # ***** 720 *****
031346 005242          INC     -(R2)        ;SET MSGTYP TO FATAL ERROR
031350 000000          HALT                    ;REMAINDER IS WRONG
; OR SEQUENCE ERROR
```

4274
4275
4276

;MODE 5 REG 4

;TEST 312 DIV 0 52525 /a-(R4)=100000 REM=0 PS 2

```
031352 005212          TST312: INC      (R2)          ;UPDATE TEST NUMBER
031354 022712 000312  CMP     #312,(R2)      ;SEQUENCE ERROR?
031360 001034          BNE     TST313-10     ;BR TO ERROR HALT ON SEQ ERROR
4277 031362 012700 052525 MOV     #52525,R0      ;LOAD HIGH ORDER WITH 52525
4278 031366 012704 036502 MOV     #S2+2,R4       ;SET UP R4
4279 031372 012701 000000 MOV     #0,R1          ;LOAD LOW ORDER WITH 0
4280 031376 071054          DIV     @-(R4),R0      ;DIVIDE BY a-(4)
4281 031400 013767 177776 005062 MOV     @#PS,SPSW      ;SAVE PS
4282 031406 122767 000002 005054 CMPS   #2,SPSW        ;IS PS=0
4283 031414 001404          BEQ     1$           ;
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 761 <====
031416 012742 000721          MOV     #721,-(R2)     ;MOVE TO MAILBOX # ***** 721 *****
031422 005242          INC     -(R2)        ;SET MSGTYP TO FATAL ERROR
```

```
4284 031424 000000          HALT          :PS IS WRONG
      031426 022700 052525 1$:  CMP          #52525,R0  :IS QUOTIENT 52525
4285 031432 001404          BEQ          2$
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
      :          CONDITIONAL BRANCH INST. AND <
      :          REPLACE THE MOVE INSTRUCTION <
      :          WHICH FOLLOWS W/ 752 <
      :
      031434 012742 000722          MOV          #722,-(R2)  :MOVE TO MAILBOX # ***** 72? *****
      031440 005242          INC          -(R2)    :SET MSGTYP TO FATAL ERROR
      031442 000000          HALT
4286 031444 022701 000000 2$:  CMP          #0,R1    :QUOTIENT IS WRONG
4287 031450 001404          BEQ          TST313   :IS REMAINDER =0
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
      :          CONDITIONAL BRANCH INST. AND <
      :          REPLACE THE MOVE INSTRUCTION <
      :          WHICH FOLLOWS W/ 743 <
      :
      031452 012742 000723          MOV          #723,-(R2)  :MOVE TO MAILBOX # ***** 723 *****
      031456 005242          INC          -(R2)    :SET MSGTYP TO FATAL ERROR
      031460 000000          HALT          :REMAINDER IS WRONG
      : OR SEQUENCE ERROR

4288
4289
4290
4291
      .SBTTL DIV USING DM2 REG 7
      :MODE 2 REG 7
      :*****
      :TEST 313          DIV -1 -4/#2= -2 REM=0 PS=10
      :*****
TST313: 031462 005212          INC          (R2)      :UPDATE TEST NUMBER
      031464 022712 000313  CMP          #313,(R2)  :SEQUENCE ERROR?
      031470 001033          BNE          TST314-10 :BR TO ERROR HALT ON SEQ ERROR
4292 031472 012700 177777          MOV          #-1,R0    :LOAD HIGH ORDER WITH -1
4293 031476 012701 177774          MOV          #-4,R1    ;LOAD LOW ORDER WITH -4
4294 031502 071027 000002          DIV          #2,R0    :DIVIDE BY #2
4295 031506 013767 177776 004754  MOV          @#PS,SPSW  :SAVE PS
4296 031514 122767 000010 004746  CMPB         #10,SPSW  :IS PS -10
4297 031522 001404          BEQ          1$
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
      :          CONDITIONAL BRANCH INST. AND <====
      :          REPLACE THE MOVE INSTRUCTION <====
      :          WHICH FOLLOWS W/ 762 <====
      :
      031524 012742 000724          MOV          #724,-(R2)  :MOVE TO MAILBOX # ***** 724 *****
      031530 005242          INC          -(R2)    :SET MSGTYP TO FATAL ERROR
      031532 000000          HALT
4298 031534 022700 177776 1$:  CMP          #-2,R0    :PS IS WRONG
4299 031540 001404          BEQ          2$      :IS QUOTIENT = -2
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
      :          CONDITIONAL BRANCH INST. AND <====
      :          REPLACE THE MOVE INSTRUCTION <====
      :          WHICH FOLLOWS W/ 753 <====
      :
      031542 012742 000725          MOV          #725,-(R2)  :MOVE TO MAILBOX # ***** 725 *****
      031546 005242          INC          -(R2)    :SET MSGTYP TO FATAL ERROR
      031550 000000          HALT
4300 031552 022701 000000 2$:  CMP          #0,R1    :QUOTIENT IS WRONG
4301 031556 001404          BEQ          TST314   :IS REMAINDER =0
      :
      : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
      :          CONDITIONAL BRANCH INST. AND <====
      :          REPLACE THE MOVE INSTRUCTION <====
      :          WHICH FOLLOWS W/ 744 <====
```

031560 012742 000726 MOV #726,-(R2) ;MOVE TO MAILBOX # ***** 726 *****
031564 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
031566 000000 HALT ;REMAINDER IS WRONG
; OR SEQUENCE ERROR

4302
4303
4304

:MODE 2 REG 7

:TEST 314 DIV 0 0/ #1 = 0 REM=0 PS=4

031570 005212 TST314: INC (R2) ;UPDATE TEST NUMBER
031572 022712 000314 CMP #314,(R2) ;SEQUENCE ERROR?
031576 001033 BNE TST315-10 ;BR TO ERROR HALT ON SEQ ERROR
4305 031600 012700 000000 MOV #0,R0 ;LOAD HIGH ORDER WITH 0
4306 031604 012701 000000 MOV #0,R1 ;LOAD LOW ORDER WITH 0
4307 031610 071027 000001 DIV #1,R0 ;DIVIDE BY #1
4308 031614 013767 177776 004646 MOV @#PS,SPSW ;SAVE PS
4309 031622 122767 000004 004640 CMPB #4,SPSW ;IS PS =4
4310 031630 001404 BEQ 1\$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
; CONDITIONAL BRANCH INST. AND <====
; REPLACE THE MOVE INSTRUCTION <====
; WHICH FOLLOWS W/ 762 <====

031632 012742 000727 MOV #727,-(R2) ;MOVE TO MAILBOX # ***** 727 *****
031636 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
031640 000000 HALT ;PS IS WRONG
4311 031642 022700 000000 1\$: CMP #0,R0 ;IS QUOTIENT =0
4312 031646 001404 BEQ 2\$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
; CONDITIONAL BRANCH INST. AND <
; REPLACE THE MOVE INSTRUCTION <=
; WHICH FOLLOWS W/ 753 <

031650 012742 000730 MOV #730,-(R2) ;MOVE TO MAILBOX # ***** 730 *****
031654 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
031656 000000 HALT ;QUOTIENT IS WRONG
4313 031660 022701 000000 2\$: CMP #0,R1 ;IS REMAINDER =0
4314 031664 001404 BEQ TST315

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
; CONDITIONAL BRANCH INST. AND <
; REPLACE THE MOVE INSTRUCTION <
; WHICH FOLLOWS W/ 744 <

031666 012742 000731 MOV #731,-(R2) ;MOVE TO MAILBOX # ***** 731 *****
031672 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
031674 000000 HALT ;REMAINDER IS WRONG
; OR SEQUENCE ERROR

4315
4316
4317

:MODE 2 REG 7

:TEST 315 DIV 0 52525 / #2-25252 REM=1 PS=0

031676 005212 TST315: INC (R2) ;UPDATE TEST NUMBER
031700 022712 000315 CMP #315,(R2) ;SEQUENCE ERROR?
031704 001033 BNE TST316-10 ;BR TO ERROR HALT ON SEQ ERROR
4318 031706 012700 000000 MOV #0,R0 ;LOAD HIGH ORDER WITH 0
4319 031712 012701 052525 MOV #52525,R1 ;LOAD LOW ORDER WITH 52525
4320 031716 071027 000002 DIV #2,R0 ;DIVIDE BY #2
4321 031722 013767 177776 004540 MOV @#PS,SPSW ;SAVE PS
4322 031730 122767 000000 004532 CMPB #0,SPSW ;IS PS =0

```
4323 031736 001404 BEQ 1$ ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 762 <====  
031740 012742 000732 MOV #732,-(R2) ;MOVE TO MAILBOX # ***** 732 *****  
031744 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR  
031746 000000 HALT ;PS IS WRONG  
4324 031750 022700 025252 1$: CMP #25252,R0 ;IS QUOTIENT =25252  
4325 031754 001404 BEQ 2$ ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 753 <====  
031756 012742 000733 MOV #733,-(R2) ;MOVE TO MAILBOX # ***** 733 *****  
031762 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR  
031764 000000 HALT ;QUOTIENT IS WRONG  
4326 031766 022701 000001 2$: CMP #1,R1 ;IS REMAINDER =1  
4327 031772 001404 BEQ TST316 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 744 <====  
031774 012742 000734 MOV #734,-(R2) ;MOVE TO MAILBOX # ***** 734 *****  
032000 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR  
032002 000000 HALT ;REMAINDER IS WRONG  
; OR SEQUENCE ERROR  
4328  
4329 ;MODE 2 REG 7  
4330 ;*****  
;TEST 316 DIV 25253 0 / #125252=100000 REM=0 PS=10  
;*****  
TST316: INC (R2) ;UPDATE TEST NUMBER  
CMP #316,(R2) ;SEQUENCE ERROR?  
BNE TST317-10 ;BR TO ERROR HALT ON SEQ ERROR  
4331 032014 012700 025253 MOV #25253,R0 ;LOAD HIGH ORDER WITH 25253  
4332 032020 012701 000000 MOV #0,R1 ;LOAD LOW ORDER WITH 0  
4333 032024 071027 125252 DIV #125252,R0 ;DIVIDE BY #125252  
4334 032030 013767 177776 004432 MOV @RPS,SPSW ;SAVE PS  
4335 032036 122767 000010 004424 CMPB #10,SPSW ;IS PS =10  
4336 032044 001404 BEQ 1$ ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 762 <====  
032046 012742 000735 MOV #735,-(R2) ;MOVE TO MAILBOX # ***** 735 *****  
032052 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR  
032054 000000 HALT ;PS IS WRONG  
4337 032056 022700 100000 1$: CMP #100000,R0 ;IS QUOTIENT =100000  
4338 032062 001404 BEQ 2$ ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 753 <====  
032064 012742 000736 MOV #736,-(R2) ;MOVE TO MAILBOX # ***** 736 *****  
032070 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR  
032072 000000 HALT ;QUOTIENT IS WRONG
```

```
4339 032074 022701 000000      2$:  CMP      #0,R1      ;IS REMAINDER =0
4340 032100 001404              BEQ      TST317
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <- --
;          CONDITIONAL BRANCH INST. AND <= --
;          REPLACE THE MOVE INSTRUCTION <-- --
;          WHICH FOLLOWS W/ 744 <- ==
032102 012742 000737          MOV      #737,-(R2)    ;MOVE TO MAILBOX # ***** 737 *****
032106 005242              INC      -(R2)        ;SET MSGTYP TO FATAL ERROR
032110 000000              HALT                    ;REMAINDER IS WRONG
; OR SEQUENCE ERROR
```

4341
4342

```
;MODE 2 REG 7
;*****
;TEST 317      DIV -1 -1 / #-1=1 REM=0 PS=0
;*****
```

```
TST317: INC      (R2)          ;UPDATE TEST NUMBER
032112 005212              CMP      #317,(R2)    ;SEQUENCE ERROR?
032114 022712 000317          BNE     TST320-10     ;BR TO ERROR HALT ON SEQ ERROR
4343 032122 012700 177777      MOV      #-1,R0       ;LOAD HIGH ORDER WITH -1
4344 032126 012701 177777      MOV      #-1,R1      ;LOAD LOW ORDER WITH -1
4345 032132 071027 177777      DIV     #-1,R0       ;DIVIDE BY #-1
4346 032136 013767 177776 004324  MOV     @RPS,SPSW     ;SAVE PS
4347 032144 122767 000000 004316  CMPB    #0,SPSW     ;IS PS =0
4348 032152 001404              BEQ     1$
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <---=
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION < --=
;          WHICH FOLLOWS W/ 762 <====
```

```
032154 012742 000740          MOV      #740,-(R2)    ;MOVE TO MAILBOX # ***** 740 *****
032160 005242              INC      -(R2)        ;SET MSGTYP TO FATAL ERROR
032162 000000              HALT                    ;PS IS WRONG
4349 032164 022700 000001      1$:  CMP      #1,R0       ;IS QUOTIENT =1
4350 032170 001404              BEQ     2$
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 753 <====
```

```
032172 012742 000741          MOV      #741,-(R2)    ;MOVE TO MAILBOX # ***** 741 *****
032176 005242              INC      -(R2)        ;SET MSGTYP TO FATAL ERROR
032200 000000              HALT                    ;QUOTIENT IS WRONG
4351 032202 022701 000000      2$:  CMP      #0,R1      ;IS REMAINDER =0
4352 032206 001404              BEQ     TST320
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 744 <====
```

```
032210 012742 000742          MOV      #742,-(R2)    ;MOVE TO MAILBOX # ***** 742 *****
032214 005242              INC      -(R2)        ;SET MSGTYP TO FATAL ERROR
032216 000000              HALT                    ;REMAINDER IS WRONG
; OR SEQUENCE ERROR
```

4353
4354
4355

```
;MODE 2 REG 7
;*****
;TEST 320      DIV 0 177777/#0=0 REM=0 PS=3
;*****
```

```
TST320: INC      (R2)          ;UPDATE TEST NUMBER
032220 005212              CMP      #320,(R2)    ;SEQUENCE ERROR?
032222 022712 000320          BNE     TST321-10     ;BR TO ERROR HALT ON SEQ ERROR
032226 001020
```

```
4356 032230 012704 177777      MOV      #177777,R4      ;LOAD HIGH ORDER WITH 177777
4357 032234 012705 000000      MOV      #0,R5      ;LOAD LOW ORDER WITH 0
4358 032240 071427 000000      DIV      #0,R4      ;DIVIDE BY #2
4359 032244 013767 177776 004216  MOV      @#PS,SPSW      ;SAVE PS
4360 032252 042767 000014 004210  BIC      #14,SPSW
4361 032260 122767 000003 004202  CMPB     #3,SPSW      ;IS PS =3
4362 032266 001404                      BEQ      TST321
```

```
032270 012742 000743      MOV      #743,-(R2)      ;MOVE TO MAILBOX # ***** 743 *****
032274 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
032276 000000      HALT                    ;PS IS WRONG
; OR SEQUENCE ERROR
```

4363
4364
4365

```
;MODE 2 REG 7
*****
:TEST 321      DIV 0 100000 0000 /#100000 REM=0 PS=2
*****
```

```
TST321: INC      (R2)      ;UPDATE TEST NUMBER
        CMP      #321,(R2) ;SEQUENCE ERROR?
        BNE     TST322-10 ;BR TO ERROR HALT ON SEQ ERROR
4366 032310 012704 100000      MOV      #100000,R4     ;LOAD HIGH ORDER WITH 100000
4367 032314 012705 000000      MOV      #0,R5      ;LOAD LOW ORDER WITH 0
4368 032320 071427 000002      DIV      #2,R4      ;DIVIDE BY #2
4369 032324 013767 177776 004136  MOV      @#PS,SPSW      ;SAVE PS
4370 032332 042767 000014 004130  BIC      #14,SPSW
4371 032340 122767 000002 004122  CMPB     #2,SPSW      ;IS PS=2
4372 032346 001404                      BEQ      TST322
```

```
032350 012742 000744      MOV      #744,-(R2)      ;MOVE TO MAILBOX # ***** 744 *****
032354 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
032356 000000      HALT                    ;PS IS WRONG
; OR SEQUENCE ERROR
```

4373
4374
4375

```
;MODE 2 REG 7
*****
:TEST 322      DIV 0 77777/#0=0 REM=0 PS=3
*****
```

```
TST322: INC      (R2)      ;UPDATE TEST NUMBER
        CMP      #322,(R2) ;SEQUENCE ERROR?
        BNE     TST323-10 ;BR TO ERROR HALT ON SEQ ERROR
4376 032370 012704 000000      MOV      #0,R4      ;LOAD HIGH ORDER WITH 0
4377 032374 012705 077777      MOV      #77777,R5     ;LOAD LOW ORDER WITH 77777
4378 032400 071427 000000      DIV      #0,R4      ;DIVIDE BY 0
4379 032404 013767 177776 004056  MOV      @#PS,SPSW      ;SAVE PS
4380 032412 042767 000014 004050  BIC      #14,SPSW
4381 032420 122767 000003 004042  CMPB     #3,SPSW      ;IS PS =3
4382 032426 001430                      BEQ      2$
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <
; CONDITIONAL BRANCH INST. AND <
; REPLACE THE MOVE INSTRUCTION <
; WHICH FOLLOWS W/ 757 <
```

```

032430 012742 000745      MOV      #745,-(R2)      ;MOVE TO MAILBOX # ***** 745 *****
032434 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
032436 000000      HALT                    ;PS IS WRONG

4383
4384 032440 012704 000000      MOV      #0,R4          ;LOAD HIGH ORDER WITH 0
4385 032444 012705 077777      MOV      #77777,R5      ;LOAD LOW ORDER WITH 77777
4386 032450 071427 000000      DIV      #0,R4          ;DIVIDE BY 0
4387 032454 013767 177776 004006      MOV      @#PS,SPSW      ;SAVE PS
4388 032462 042767 000014 004000      BIC      #14,SPSW
4389 032470 122767 000003 003772      CMPB    #3,SPSW        ;IS PS =3
4390 032476 001413      BEQ      TST323

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-
;          CONDITIONAL BRANCH INST. AND <-
;          REPLACE THE MOVE INSTRUCTION <-
;          WHICH FOLLOWS W/ 733 <-

032500 012742 000746      MOV      #746,-(R2)      ;MOVE TO MAILBOX # ***** 746 *****
032504 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
032506 000000      HALT                    ;PS IS WRONG
; OR SEQUENCE ERROR

4391
4392 032510 022705 077777      2$:     CMP      #77777,R5      ;IS REMAINDER =1
4393 032514 001404      BEQ      TST323

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <-- -
;          CONDITIONAL BRANCH INST. AND <-- -
;          REPLACE THE MOVE INSTRUCTION <-- -
;          WHICH FOLLOWS W/ 724 <-- -

032516 012742 000747      MOV      #747,-(R2)      ;MOVE TO MAILBOX # ***** 747 *****
032522 005242      INC      -(R2)          ;SET MSGTYP TO FATA ERROR
032524 000000      HALT                    ;REMAINDER IS WRONG
; OR SEQUENCE ERROR

4394
4395
4396
;MODE 2 REG 7
;*****
;TEST 323 DIV 25253 0 / #125252=100000 REM=0 PS=10
;*****
TST323: INC      (R2)          ;UPDATE TEST NUMBER
        CMP      #323,(R2)      ;SEQUENCE ERROR?
        BNE     TST324-10      ;BR TO ERROR HALT ON SEQ ERROR
        MOV     #25253,R0      ;LOAD HIGH ORDER WITH 25253
        MOV     #0,R1          ;LOAD LOW ORDER WITH 0
        DIV     #125252,R0      ;DIVIDE BY #125252
        MOV     @#PS,SPSW      ;SAVE PS
        CMPB   #10,SPSW        ;IS PS =10
        BEQ    1$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 762 <====

032570 012742 000750      MOV      #750,-(R2)      ;MOVE TO MAILBOX # ***** 750 *****
032574 005242      INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
032576 000000      HALT                    ;PS IS WRONG
4403 032600 022700 100000      1$:     CMP      #100000,R0      ;IS QUOTIENT =100000
4404 032604 001404      BEQ      2$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 753 <====

```



```
032606 012742 000751      MOV    #751,-(R2)      ;MOVE TO MAILBOX # ***** 751 *****
032612 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
032614 000000              HALT                    ;QUOTIENT IS WRONG
4405 032616 022701 000000    2$:    CMP    #0,R1      ;IS REMAINDER =0
4406 032622 001404              BEQ    TST324

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <==--
;          CONDITIONAL BRANCH INST. AND <
;          REPLACE THE MOVE INSTRUCTION <==--
;          WHICH FOLLOWS W/ 744 <====
```

```
032624 012742 000752      MOV    #752,-(R2)      ;MOVE TO MAILBOX # ***** 752 *****
032630 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
032632 000000              HALT                    ;REMAINDER IS WRONG
; OR SEQUENCE ERROR
```

4407 ;MODE 2 RFG 7

4408

```
*****
;TEST 324 DIV -1 -1 /#-1=1 REM=0 PS=0
*****
```

```
TST324: INC    (R2)          ;UPDATE TEST NUMBER
032634 005212              CMP    #324,(R2)      ;SEQUENCE ERROR?
032636 022712 000324      BNE    TST325-10      ;BR TO ERROR HALT ON SEQ ERROR
4409 032644 012700 177777    MOV    #-1,R0         ;LOAD HIGH ORDER WITH -1
4410 032650 012701 177777    MOV    #-1,R1         ;LOAD LOW ORDER WITH -1
4411 032654 071027 177777    DIV    #-1,R0         ;DIVIDE BY #-1
4412 032660 013767 177776 003602  MOV    @#PS,SPSW      ;SAVE PS
4413 032666 122767 000000 003574  CMPB   #0,SPSW        ;IS PS =0
4414 032674 001404              BEQ    1$

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 762 <====
```

```
032676 012742 000753      MOV    #753,-(R2)      ;MOVE TO MAILBOX # ***** 753 *****
032702 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
032704 000000              HALT                    ;PS IS WRONG
4415 032706 022700 000001    1$:    CMP    #1,R0         ;IS QUOTIENT =1
4416 032712 001404              BEQ    2$
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====
;          CONDITIONAL BRANCH INST. AND <====
;          REPLACE THE MOVE INSTRUCTION <====
;          WHICH FOLLOWS W/ 753 <====

032714 012742 000754      MOV    #754,-(R2)      ;MOVE TO MAILBOX # ***** 754 *****
032720 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
032722 000000              HALT                    ;QUOTIENT IS WRONG
4417 032724 022701 000000    2$:    CMP    #0,R1      ;IS REMAINDER =0
4418 032730 001404              BEQ    TST325

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS < -
;          CONDITIONAL BRANCH INST. AND < -
;          REPLACE THE MOVE INSTRUCTION <
;          WHICH FOLLOWS W/ 744 <--
```

```
032732 012742 000755      MOV    #755,-(R2)      ;MOVE TO MAILBOX # ***** 755 *****
032736 005242              INC    -(R2)          ;SET MSGTYP TO FATAL ERROR
032740 000000              HALT                    ;REMAINDER IS WRONG
; OR SEQUENCE ERROR
```

4419 ;MODE 2 REG 7

4420

```
*****
;TEST 325 DIV 0 177777/#0=0 REM=0 PS=3
*****
```

```
TST325: INC    (P2)          ;UPDATE TEST NUMBER
```

```
032744 022712 000325      CMP      #325,(R2)      ;SEQUENCE ERROR?  
032750 001020      BNE      TST326-10     ;BR TO ERROR HALT ON SEQ ERROR  
4421 032752 012704 177777      MOV      #177777,R4    ;LOAD HIGH ORDER WITH 177777  
4422 032756 012705 000000      MOV      #0,R5         ;LOAD LOW ORDER WITH 0  
4423 032762 071427 000000      DIV      #0,R4         ;DIVIDE BY #2  
4424 032766 013767 177776 003474      MOV      @#PS,SPSW     ;SAVE PS  
4425 032774 042767 000014 003466      BIC      #14,SPSW      ;  
4426 033002 122767 000003 003460      CMPB    #3,SPSW       ;IS PS =3  
4427 033010 001404      BEQ      TST326
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====  
; CONDITIONAL BRANCH INST. AND <====  
; REPLACE THE MOVE INSTRUCTION <====  
; WHICH FOLLOWS W/ 757 <====
```

```
033012 012742 000756      MOV      #756,-(R2)    ;MOVE TO MAILBOX # ***** 756 *****  
033016 005242      INC      -(R2)         ;SET MSGTYP TO FATAL ERROR  
033020 000000      HALT
```

```
; PS IS WRONG  
; OR SEQUENCE ERROR
```

4428
4429

:MODE 2 REG 7

```
*****  
:TEST 326      DIV 0 100000 000000 /#2=100000 REM=0 PS=2  
*****
```

```
TST326: INC      (R2)      ;UPDATE TEST NUMBER  
033022 005212      CMP      #326,(R2)    ;SEQUENCE ERROR?  
033024 022712 000326      BNE      TST327-10     ;BR TO ERROR HALT ON SEQ ERROR  
4430 033032 012704 100000      MOV      #100000,R4    ;LOAD HIGH ORDER WITH 100000  
4431 033036 012705 000000      MOV      #0,R5         ;LOAD LOW ORDER WITH 0  
4432 033042 071427 000002      DIV      #2,R4         ;DIVIDE BY #2  
4433 033046 013767 177776 003414      MOV      @#PS,SPSW     ;SAVE PS  
4434 033054 042767 000014 003406      BIC      #14,SPSW      ;  
4435 033062 122767 000002 003400      CMPB    #2,SPSW       ;IS PS=2  
4436 033070 001404      BEQ      TST327
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <= -  
; CONDITIONAL BRANCH INST. AND <  
; REPLACE THE MOVE INSTRUCTION <--  
; WHICH FOLLOWS W/ 757 <- -
```

```
033072 012742 000757      MOV      #757,-(R2)    ;MOVE TO MAILBOX # ***** 757 *****  
033076 005242      INC      -(R2)         ;SET MSGTYP TO FATAL ERROR  
033100 000000      HALT
```

```
; PS IS WRONG  
; OR SEQUENCE ERROR
```

4437
4438

:MODE 2 REG 7

```
*****  
:TEST 327      DIV 0 77777 /#0=0 REM=0 PS=3  
*****
```

```
TST327: INC      (R2)      ;UPDATE TEST NUMBER  
033102 005212      CMP      #327,(R2)    ;SEQUENCE ERROR?  
033104 022712 000327      BNE      TST330-10     ;BR TO ERROR HALT ON SEQ ERROR  
4439 033112 012704 000000      MOV      #0,R4         ;LOAD HIGH ORDER WITH 0  
4440 033116 012705 077777      MOV      #77777,R5     ;LOAD LOW ORDER WITH 77777  
4441 033122 071427 000000      DIV      #0,R4         ;DIVIDE BY 0  
4442 033126 013767 177776 003334      MOV      @#PS,SPSW     ;SAVE PS  
4443 033134 042767 000014 003326      BIC      #14,SPSW      ;  
4444 033142 122767 000003 003320      CMPB    #3,SPSW       ;IS PS =3  
4445 033150 001404      BEQ      TST330
```

```
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <= -  
; CONDITIONAL BRANCH INST. AND <--  
; REPLACE THE MOVE INSTRUCTION <  
; WHICH FOLLOWS W/ 757 <
```



```
033322 012742 000765      MOV      #765,-(R2)      ;REPLACE THE MOVE INSTRUCTION <====-
033326 005242              INC      -(R2)          ;WHICH FOLLOWS W/ 723 <===
033330 000000              HALT                    ;MOVE TO MAILBOX # ***** 765 *****
4464 033332 012711 177777 5$: MOV      #-1,(R1)      ;SET MSGTYP TO FATAL ERROR
4465 033336 160061 000000   SUB      R0,0(R1)      ;SUB -1 FROM -1 FAILED TO GIVE ZERO RESULT
4466 033342 001404   BEQ      6$           ;RESET SUBT TO A -1
                                ;MODE-6
```

```
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====-
                                ; CONDITIONAL BRANCH INST. AND <====-
                                ; REPLACE THE MOVE INSTRUCTION <====-
                                ; WHICH FOLLOWS W/ 712 <====-
033344 012742 000766      MOV      #766,-(R2)      ;MOVE TO MAILBOX # ***** 766 *****
033350 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
033352 000000              HALT                    ;SUB -1 FROM -1 FAILED TO GIVE ZERO RESULT
4467 033354 010011 000002 6$: MOV      R0,(R1)      ;RESET SUBT TO A -1
4468 033356 160071 000002   SUB      R0,@2(R1)      ;MODE-7
4469 033362 001404   BEQ      TST331
```

```
                                ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS <====-
                                ; CONDITIONAL BRANCH INST. AND <====-
                                ; REPLACE THE MOVE INSTRUCTION <====-
                                ; WHICH FOLLOWS W/ 702 <====-
033364 012742 000767      MOV      #767,-(R2)      ;MOVE TO MAILBOX # ***** 767 *****
033370 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
033372 000000              HALT                    ;SUB -1 FROM -1 FAILED TO GIVE ZERO RESULT
                                ; OR SEQUENCE ERROR
```

4470 *****
:TEST 331 TEST RTI KERNAL/USER MODE MICRO FLOW

```
TST331: INC      (R2)          ;UPDATE TEST NUMBER
033374 005212              CMP      #331,(R2)      ;SEQUENCE ERROR?
033376 022712 000331      BNE      TST332-10     ;BR TO ERROR HALT ON SEQ ERROR
4471 033404 012706 001000   MOV      #STBOT,R6     ;SETUP KERNAL STACK
4472 033410 012767 140000 144360 MOV      #USRM,PS      ;SETUP USER MODE
4473 033416 012706 036370   MOV      #USTBOT,R6    ;SETUP USER STACK POINTER
4474 033422 012746 140000   MOV      #USRM,-(SP)   ;PUSH USER R6 ON STACK
4475 033426 012746 033434   MOV      #REN,-(SP)    ;SETUP RETURN ADDRESS
4476 033432 000002              RTI
4477 033434 022767 140000 144334 REN: CMP      #140000,PS   ;CHECK PSW
4478 033442 001407              BEQ      1$           ;
4479 033444 042767 140000 144324 BIC      #USRM,PS      ;CLEAR USER MODE
4480 033452 012742 000770   MOV      #770,-(R2)    ;MOVE TO MAILBOX # ***** 770 *****
033456 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
033460 000000              HALT                    ;USER MODE DID NOT SET
4481 033462 042767 140000 144306 1$: BIC      #USRM,PS      ;SET KERNAL MODE
4482 033470 012746 000340   MOV      #340,-(SP)    ;SET LEVEL 7
4483 033474 012746 033512   MOV      #TST332,-(SP)
4484 033500 000002              RTI
4485 033502 012742 000771   MOV      #771,-(R2)    ;MOVE TO MAILBOX # ***** 771 *****
033506 005242              INC      -(R2)          ;SET MSGTYP TO FATAL ERROR
033510 000000              HALT                    ;RTI DID NOT HAPPEN
```

```
4486  
4487  
4488 :* THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE  
4489 :* SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION  
4490 :* OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE  
4491 :* THEN BIT 08 OF $SWREG IS SET TO 1 THRU APT SCRIPTING.
```

4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505

```

:*      TEST DESCRIPTION:
:*      THIS TEST VERIFIES THAT THE SWITCH REGISTER SELECT SIGNAL IS
:*      ISSUED FROM THE CPU DATA PATH MODULE TO THE MFM MODULE. WHEN
:*      THE HARDWARE SWITCH REGISTER IS READ, THE ADDRESS OF THE SWITCH
:*      REGISTER IS DECODED ON THE CPU DATA PATH CAUSING THE 'SR SELECT'
:*      SIGNAL TO BE ISSUED TO THE MFM. THE MFM USES THIS SIGNAL TO PLACE
:*      THE REGISTER CONTENTS ONTO THE PAX DATA BUS. CONTENTS OF THE SWITCH
:*      REGISTER OTHER THAN ALL 1'S SHOULD BE READ. (THERE IS A
:*      HI PROBABILITY THAT IF THE 'SR SELECT' SIGNAL IS NOT ISSUED
:*      ALL 1'S WILL BE READ ON PAX DATA.) IT IS ASSUMED THAT DATA OTHER
:*      THAN ALL 1'S IS LOCATED IN HARDWARE SWITCH REGISTER .

```

```

:*****
:TEST 332      SWITCH REGISTER SELECT TEST
:*****

```

```

033512 005212
033514 022712 000332
033520 001022
4506
4507 033522 132767 000200 144571      BITB #200,$ENVM      ;IS APT SIZING?
4508 033530 001405                      BEQ 4$              ;NO;TRY HARDWARE SWITCH REGISTER
4509 033532 032767 000400 144562      BIT #400,$SWREG    ;YES APT IS SIZING;DOES APT SAY TO DO
4510                                     ;THIS TEST
4511 033540 001413                      BEQ 10$            ;NO;SKIP TEST
4512 033542 000404                      BR 1$              ;YES,DO TEST
4513 033544 032737 000400 177570 4$:  BIT #400,@#177570 ;DOES HARDWARE SWITCH REGISTER SAY TO
4514                                     ;TO DO TEST?
4515 033552 001406                      BEQ 10$            ;NO,SKIP TEST
4516
4517 033554 013700 177570 1$:         MOV @#177570,R0     ;READ HARDWARE SWITCH REGISTER CONTENTS
4518                                     ;AND SAVE IN R0
4519 033560 022700 177777              CMP #-1,R0         ;WERE ALL 1'S RECEIVED
4520 033564 001001                      BNE 10$            ;NO,TEST PASSES
4521 033566 000000                      HALT               ;'SR SELECT' ERROR OR SEQUENCE ERROR
4522                                     ;ALL 1'S WERE RECEIVED WHEN HARDWARE SWITCH
4523                                     ;REGISTER WAS READ. THIS INDICATES THAT THE
4524                                     ;THE SWITCH REGISTER CONTENTS WAS NOT PUT ON
4525                                     ;THE PAX DATA LINES AS A RESULT OF THE 'SR SELECT'
4526                                     ;SIGNAL BEING RECEIVED BY THE MFM FROM THE
4527                                     ;CPU DATA PATH MODULE
4528 033570 000240 10$:              NOP                ;END OF TEST
4529 033572 000240                      NOP
4530 033574 000240                      NOP

```

4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543

```

:*****
:*      THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
:*      SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
:*      OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
:*      THEN BIT 08 OF $SWREG IS SET TO 1 THRU APT SCRIPTING.
:*
:*      THIS TEST VERIFIES THE CACHE RESTART SIGNAL ON THE CPU CONTROL
:*      MODULE ISSUED FROM THE CACHE.
:*
:*      THIS TEST ASSUMES THAT ALL MODULES EXCEPT THE CPU DATA PATH/CONTROL
:*      STORE ARE KNOWN GOOD MODULES.

```

4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555

```

:* THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
:* HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
:* VERIFY TESTING OF THE CPU MODULE.
:* TEST DESCRIPTION:
:* VERIFY THAT A CACHE READ HIT WILL RESULT IN DATA BEING READ
:* FROM CACHE DATA STORE, ASSURING THAT THE CACHE HAS ISSUED A
:* A CPU CLOCK RESTART SIGNAL. ASSURE THAT ALL 0'S CAN BE CACHED
:* OUT OF CACHE DATA STORE.

```

TES 333 CACHE RESTART SIGNAL TEST

```

033576 005212
033600 022712 000333
033604 001115
4556 033606 132767 000200 144505
4557 033614 001405
4558 033616 032767 000400 144476
4559
4560 033624 001506
4561 033626 000404
4562 033630 032737 000400 177570 4$:
4563
4564 033636 001501
4565
4566 033640 1$:
4567
4568 033640 012701 040000
4569 033644 012705 060000
4570 033650 012737 033734 000014
4571 033656 012737 000340 000016
4572 033664 005003
4573 033666 005004
4574 033670 012706 060002
4575 033674 005037 060000
4576
4577 033700 012767 000340 144070
4578 033706 112737 000002 177750
4579
4580
4581
4582 033714 012737 000011 177746
4583
4584 033722 000257
4585 033724 005711
4586 033726 005715
4587
4588
4589 033730 000003
4590
4591
4592
4593
4594
4595

TST333: INC (R2) ;UPDATE TEST NUMBER
CMP #333,(R2) ;SEQUENCE ERROR?
BNE TST334-10 ;BR TO ERROR HALT ON SEQ ERROR
BITB #200,$ENVM ;IS APT SIZING?
BEQ 4$ ;NO;TRY HARDWARE SWITCH REGISTER
BIT #400,$SWREG ;YES APT IS SIZING;DOES APT SAY TO DO
;THIS TEST
BEQ 10$ ;NO;SKIP TEST
BR 1$ ;YES,DO TEST
BIT #400,@#177570 ;DOES HARDWARE SWITCH REGISTER SAY TO
;TO DO TEST?
BEQ 10$ ;NO,SKIP TEST

MOV #40000,R1 ;ADDRESS 40000 TO R1
MOV #60000,R5 ;ADDRESS 60000 TO R5
MOV #3$,@#14 ;SETUP BPT TRAP VECTORS
MOV #340,@#16
CLR R3 ;CLEAR ERROR FLAGS
CLR R4
MOV #60002,R6 ;STACK POINTER NOW POINTS TO ADDRESS 60002
CLR @#60000 ;PRECONDITION MAIN MEMORY ADDRESS LOCATION
;60000 WITH ALL 0'S
MOV #340,PS ;PRECONDITION PS TO 340
MOV #2,@#177750 ;'HIT ON DESTINATION ONLY'(HODO) ALLOWS
;CACHE UPDATES AND HITS
;ONLY DURING THE DESTINATION MEMORY ACCESS
;OF AN INSTRUCTION.
;NO BYPASS TO ALLOW WRITES TO CACHE STORES.
;ENABLE LOW CACHE
;CLEAR ALL CONDITION CODES
;
;CACHE READ UPDATE. WRITE ALL 0'S FROM
;MAIN MEMORY LOCATION TO CACHE DATA STORE
;LOCATION 0000.
;BREAKPOINT TRAP. DUE TO A TRAP,THE PSW
;WILL BE WRITTEN TO THE STACK, WHICH NOW
;POINTS TO ADDRESS 60000.THE TRAP INSTRUCTION
;IS A NON-DESTINATION ACCESS INSTR.THEREFORE,
;SINCE HODO IS BEING USED, A CACHE UPDATE
;WILL BE INHIBITED. MAIN MEMORY
;ADDRESS 60000 WILL CONTAIN PSW DATA OF 344,AND

```

```

4596                                     ;THE LOCATION IN CACHE CORRESPONDING TO ADDRESS
4597                                     ;60000 WILL BE LEFT WITH ALL 0'S DATA.
4598
4599 033732 000000                       HALT                               ;BPT TRAP DID NOT OCCUR
4600
4601 033734 042737 000002 177750 3$:    BIC      #2,@#177750           ;TRAP TO HERE;DISABLE HODO
4602 033742 011500                       MOV      (R5),R0              ; WHEN THIS INSTRUCTION READS
4603                                     ;ADDRESS 60000
4604                                     ;A CACHE READ HIT SHOULD RESULT AND A CPU CLOCK
4605                                     ;RESTART SIGNAL SHOULD BE ISSUED.
4606                                     ;THE CPU SHOULD READ DATA FROM CACHE DATA STORE
4607                                     ;RATHER THAN MAIN MEMORY.
4608 033744 000240                       NOP
4609 033746 000240                       NOP
4610 033750 005700                       TST      R0                  ;R0 SHOULD CONTAIN ALL 0'S
4611 033752 001406                       BEQ      7$
4612 033754 022700 000344               CMP      #344,R0            ;DID THE CPU READ MAIN MEMORY?
4613 033760 001002                       BNE      6$                 ;NO,MUST HAVE READ CACHE BUT BAD DATA WAS RECEIVED.
4614 033762 005203                       INC      R3                  ;INDICATE ERROR THAT MAIN MEMORY WAS READ
4615 033764 000401                       BR       7$
4616 033766 005204 6$:                 INC      R4                  ;INDICATE ERROR THAT DATA WAS CACHED BUT
4617                                     ;BAD DATA WAS RECEIVED.
4618 033770 105037 177750 7$:          CLRB    @#177750            ;DISABLE MAINTENANCE MODE
4619 033774 012737 000000 177746       MOV      #0,@#177746        ;TURN ON CACHE
4620 034002 012706 001000               MOV      #STBOT,R6          ;RESET STACK POINTER
4621 034006 012737 036410 000014       MOV      #T014,@#14        ;RESTORE BPT VECTORS
4622 034014 005037 000016               CIR      @#16
4623 034020 005703                       TST      R3                  ;WAS MAIN MEMORY READ?
4624 034022 001402                       BEQ      8$                 ;NO
4625 034024 000000                       HALT                          ;ERROR
4626                                     ;CPU CLOCK RESTART-CACHED DATA TESTS
4627
4628                                     ;ATTEMPTING TO CAUSE A READ HIT IN ORDER TO
4629                                     ;CACHE DATA RESULTED IN MAIN MEMORY BEING READ
4630 034026 000405                       BR       10$
4631 034030 005704 8$:                 TST      R4                  ;NEXT TEST
4632 034032 001403                       BEQ      10$                ;WAS BAD DATA CACHED FROM CACHE DATA STORE?
4633 034034 000000                       HALT                          ;NO, NEXT TEST
4634                                     ;ERROR
4635                                     ;CPU CLOCK RESTART-CACHED DATA TESTS
4636
4637                                     ;CREATING A READ HIT BY READING ADDRESS 60000
4638 034036 000401                       BR       10$                ;CACHED BAD DATA FROM CACHE DATA STORE
4639 034040 000000                       HALT                          ;SEQUENCE ERROR
4640 034042 000402 10$:                BR       .+6
4641 034044 000240                       NOP
4642 034046 000240                       NOP
4643
4644                                     ;*****
4645                                     ;THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
4646                                     ;SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
4647                                     ;OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
4648                                     ;THEN BIT 08 OF $SWREG IS SET TO 1 THRU APT SCRIPTING.
4649
4650                                     ;THIS TEST VERIFIES
4651                                     ;THE CPU CONTROL SIGNAL TO THE CACHE WHICH INDICATES THAT AN ACCESS
4652                                     ;TO THE UNIBUS IS BEING PERFORMED(PA TOP 128K L).

```

```

4653      : *
4654      : * THIS TEST ASSUMES THAT ALL MODULES OTHER THAN CPU CONTROL/DATA PATH
4655      : * ARE KNOWN GOOD MODULES.
4656      : *
4657      : * THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
4658      : * HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
4659      : * VERIFY TESTING OF THE CPU MODULE.
4660      : * TEST DESCRIPTION:
4661      : * VERIFY THAT THE CACHE WRITE CONTROL LOGIC WILL INHIBIT A CACHE
4662      : * READ UPDATE TO CACHE TAG STORE DUE TO AN ACCESS TO I/O PAGE.
4663      : *
4664      : *
*****
:TEST 334      CACHE WRITE CONTROL LOGIC TEST
*****
TST334: INC      (R2)          ;UPDATE TEST NUMBER
          CMP      #334,(R2)   ;SEQUENCE ERROR?
          BNE     TST335-10    ;BR TO ERROR HALT ON SEQ ERROR
4665 034060 132767 000200 144233 BITB    #200,$ENVM        ;IS APT SIZING?
4666 034066 001405          BEQ     4$          ;NO;TRY HARDWARE SWITCH REGISTER
4667 034070 032767 000400 144224 BIT      #400,$SWREG      ;YES APT IS SIZING;DOES APT SAY TO DO
4668          ;THIS TEST
4669 034076 001442          BEQ     10$         ;NO;SKIP TEST
4670 034100 000404          BR      1$          ;YES,DO TEST
4671 034102 032737 000400 177570 4$: BIT    #400,@#177570 ;DOES HARDWARE SWITCH REGISTER SAY TO
4672          ;TO DG TEST?
4673 034110 001435          BEQ     10$         ;NO,SKIP TEST
4674          ;
4675 034112          ;1$:
4676 034112 112737 000002 177750      MOVB #2,@#177750      ;ALLOWS CACHE TAG FIELD BITS TO BE
4677          ;WRITTEN TO CHR<15:07> ONLY DURING
4678          ;THE DESTINATION MEMORY ACCESS
4679          ;OF AN INSTRUCTION
4680 034120 012737 000015 177746      MOV #15,@#177746    ;NO UCB SO AS TO WRITE ENABLE CACHE STORE
4681 034126 005737 057744          TST @#57744
4682 034132 005737 077744          TST @#77744
4683          ;READ UPDATE;LOAD BIT PATTERN
4684          ;00000011 INTO TAG STORE LOCATION
4685 034136 005737 177744          TST @#177744
4686          ;7762
4687          ;ACCESS I/O PAGE BY READING CME REGISTER.
4688          ;THE CACHE COULD DO AN UPDATE TO
4689          ;TAG STORE LOCATION 7762 BUT THE ACCESS
4690 03 142 005737 057744          TST @#57744
4691          ;TO I/O PAGE WILL INHIBIT WRITE CONTROL
4692          ;LOGIC
4693 034146 013700 177752          MOV @#177752,R0    ;WRITE TAG STORE DATA FROM LOCATION
4694 034152 000240          NOP                ;7762 INTO CHR<15:07>.
4695          ;SAVE CHR DATA
4696 034156 105037 177750          CLRB @#177750     ;DISABLE MAINTENANCE MODE
4697 034162 012737 000000 177746      MOV #0,@#177746    ;TURN ON CACHE
4698 034170 042700 000177          BIC #177,R0       ;PREPARE R0 FOR ERROR CHECK
4699 034174 022700 000600          CMP #600,R0       ;BITS 15:07 SHOULD BE BIT PATTERN 00000011
4700 034200 001401          BEQ 10$           ;PASS
4701 034202 000000          HALT              ;CACHE WRITE CONTROL CPU LOGIC SIGNAL
4702          ;OR SEQUENCE ERROR
4703          ;READING TAG DATA BITS<21:13> THRU CACHE HIT REGISTE
4704          ;DID NOT RESULT IN BIT PATTERN 00000011.

```


4705
4706
4707
4708
4709
4710 034204 000240
4711 034206 000240
4712 034210 000240
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727

:THIS INDICATES THAT A READ UPDATE
:MAY HAVE OCCURED TO CACHE TAG STORE
:DUE TO AN ACCESS TO I/O PAGE
:RO BITS 15:07 CONTAINS DATA RECEIVED
:FROM CACHE HIT REGISTER BITS 15:07
10\$: NOP ;END OF TEST
NOP
NOP
:*****
:THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
:SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
:OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
:THEN BIT 08 OF \$SWREG IS SET TO 1 THRU APT SCRIPTING.
:THIS TEST VERIFIES
:THE CACHE BYPASS SIGNAL GENERATED FROM THE CPU TO THE CACHE(CACHE BYPASS L).
:THIS SIGNAL IS GENERATED WHEN THE ASRB INSTRUCTION IS EXECUTED.
:THIS TEST ASSUMES THAT ALL MODUIES EXCEPT CPU DATA PATH/CONTROL STORE
:ARE KNOWN GOOD MODULES.
:THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE

4729
 4730
 4731
 4732
 4733
 4734
 4735

```

: *   HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
: *   VERIFY TESTING OF THE CPU MODULE.
: *   TEST DESCRIPTION:
: *   VERIFY THAT THE ASRB INSTRUCTION WILL CAUSE A CACHE BYPASS
: *   UNDER A READ HIT CONDITION.
  
```

 :TEST 335 ASRB CACHE BYPASS TEST

```

034212 005212
034214 022712 000335
034220 001043
4736 034222 132767 000200 144071
4737 034230 001405
4738 034232 032767 000400 144062
4739
4740 034240 001434
4741 034242 000404
4742 034244 032737 000400 177570 4$:
4743
4744 034252 001427
4745
4746 034254 1$:
4747 034254 012700 060000
4748
4749 034260 112737 000002 177750
4750
4751
4752
4753 034266 012737 000011 177746
4754
  
```

```

TST335: INC (R2) ;UPDATE TEST NUMBER
CMP #335,(R2) ;SEQUENCE ERROR?
BNE TST336-10 ;BR TO ERROR HALT ON SEQ ERROR
BITB #200,$ENVM ;IS APT SIZING?
BEQ 4$ ;NO;TRY HARDWARE SWITCH REGISTER
BIT #400,$SWREG ;YES APT Ic SIZING;DOES APT S TO DO
;THIS TEST
BEQ 10$ ;NO;SKIP TEST
BR 1$ ;YES,DO TEST
BIT #400,@#177570 ;DOES HARDWARE SWITCH REGISTER SAY TO
;TO DO TEST?
BEQ 10$ ;NO;SKIP TEST

MOV #60000,R0 ;SETUP TEST LOCATION ADDRESS
;IN R0
MOV B #2,@#177750 ;HODO ALLOWS READ HITS TO BE CACHED,CACHE UPDATES,AN
;CLOCKING OF OUTPUT OF CACHE HIT NAND
;GATE INTO CMR ONLY DURING THE DESTINATION
;ACCESS OF AN INSTRUCTION.
MOV #11,@#177746 ;NO UCB SO AS TO WRITE CACHE STORES
;ENABLE LOW CACHE FOR A READ HIT
  
```

4756
4757 034274 005710
4758
4759
4760 034276 106210

TST (R0)

ASRB (R0)

;READING LOCATION SPECIFIED BY R0
;WILL ASSURE A READ HIT WHEN THE
;LOCATION IS READ AGAIN
;ASRB INSTRUCTION WILL CAUSE A BYPASS

4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777

034300 013701 177750
034304 000240
034306 000240
034310 105037 177750
034314 012737 000000 177746
034322 032701 C^0400
034326 001001
034330 000000

MOV @#177750,R1
NOP
NOP
CLRB @#177750
MOV #0,@#177746
BIT #400,R1
BNE 10\$
HALT

:TO OCCUR INHIBITING A READ HIT
:TO LOCATION SPECIFIED BY RO.
:THIS SITUATION WILL RESULT IN CMR
:BIT 8 BEING A 1.
:SAVE CMR CONTENTS
:
:DISABLE MAINT MODE
:TURN ON CACHE
:WAS CMR BIT 8 A 1
:PASS
:CACHE BYPASS DID NOT OCCUR OR
:SEQUENCE ERROR
:READING OUTPUT OF CACHE HIT NAND GATE
:THRU CMR<8> DID NOT RESULT IN A 1

4779 034332 000240
4780 034334 000240
4781 034336 000240
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803

10\$: NOP ;END OF TEST
NOP
NOP

```

:*****
: THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
: SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
: OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
: THEN BIT 08 OF $SWREG IS SET TO 1 THRU APT SCRIPTING.
:
: THIS TEST VERIFIES THE MICROCODE IN THE CPU CONTROL STORE ASSOCIATED
: WITH THE CIS INSTRUCTION SET. IT IS ASSUMED THAT ALL MODULES EXCEPT
: THE CPU DATA PATH AND CONTROL MODULE ARE ALL 'KNOWN GOOD MODULES'.
:
: THIS TEST ALLOWS MFG. TO ELIMINATE
: HAVING TO RUN THE CIS DIAGNOSTIC DURING QUICK
: VERIFY TESTING OF THE CPU MODULE.
:
: TEST DESCRIPTION:
: THE CPU MPC MICROADDRESSES ASSOCIATED WITH THE CPU ARE
: 740 THRU 776. IT HAS BEEN DETERMINED THAT BY EXECUTING THIS TEST
: ALL BUT MPC ADDRESS 772 AND 773 ARE COVERED.
:
:*****

```

:TEST 336 CIS CPU MPC ADDRESS COVERAGE TEST
:*****

034340 005212
034342 022712 000336
034346 001101
4804 034350 132767 000200 143745
4805 034356 001405
4806 034360 032767 000400 143734
4807
4808 034366 001472
4809 034370 000404
4810 034372 032737 000400 177570 4\$:
4811
4812 034400 001465
4813
4814 034402 012737 034440 000010 1\$:
4815 034410 012737 000340 000012
4816 034416 076001
4817
4818
4819
4820
4821 034420 000240
4822 034422 012737 036400 000010
4823 034430 005037 000012
4824 034434 000000
4825
4826
4827 034436 000406
4828 034440 022626 2\$:
4829 034442 012737 036400 000010
4830 034450 005037 000012

```

TST336: INC (R2) ;UPDATE TEST NUMBER
CMP #336,(R2) ;SEQUENCE ERROR?
BNE TST337-10 ;BR TO ERROR HALT ON SEQ ERROR
BITB #200,$ENVM ;IS APT SIZING?
BEQ 4$ ;NO;TRY HARDWARE SWITCH REGISTER
BIT #400,$SWREG ;YES APT IS SIZING;DOES APT SAY TO DO
;THIS TEST
BEQ ENDMPC ;NO;SKIP TEST
BR 1$ ;YES,DO TEST
BIT #400,@#177570 ;DOES HARDWARE SWITCH REGISTER SAY TO
;TO DO TEST?
BEQ ENDMPC ;NO,SKIP TEST

1$: MOV #2$,@#10 ;SETUP FOR POSSIBLE TRAP
MOV #340,@#12
76001 ;THIS CIS INSTRUCTION SHOULD CAUSE A TRAP
;TO OCCUR TO VECTOR 10 INDICATING THAT
;SWITCH S1 OF CIS M7092 DATA PATH MODULE
;IS OPEN. THE SWITCH MUST BE OPEN TO ALLOW
;THIS TEST TO RUN.

NOP
MOV #T010,@#10 ;RESTORE VECTORS
LLR @#12
HALT ;CHECK TO SEE THAT SWITCH S1 OF M7092 CIS DATA
;PATH MODULE IS OPEN.

BR 5$
CMP (SP)+,(SP)+ ;READJUST STACK
MOV #T010,@#10 ;RESTORE VECTORS
CLR @#12

```

```

4831
4832 034454 076150      5$:  ADDNI                ;:NOW PERFORM CIS INSTRUCTION WHICH WILL
4833                                     ;:COVER MOST OF THE CPU MPC MICROADDRESSES
4834 034456 034466      .WORD  A
4835 034460 034472      .WORD  B
4836 034462 034476      .WORD  D
4837
4838 034464 000414      BR    CHECK                ;:NOW CHECK RFSULTS
4839
4840                                     ADDNI=76150
4841 034466 000006      A:   .WORD  6
4842 034470 034502      .WORD  SA
4843 034472 000006      B:   .WORD  6
4844 034474 034502      .WORD  SA
4845 034476 000006      D:   .WORD  6
4846 034500 034510      .WORD  DA
4847 034502 001002      SA:  .WORD  1002
4848 034504 001002      .WORD  1002
4849 034506 031002      .WORD  31002
4850 034510 000000      DA:  .WORD  0
4851 034512 000000      .WORD  0
4852 034514 000000      .WORD  0
4853
4854 034516                                     CHECK:
4855 034516 022737 032064 034510      CMP    #32064,@#DA        ;:CHECK RESJLTS
4856 034524 001401                                     BEQ    6$                 ;:PASS
4857 034526 000000                                     HALT                                     ;:ERROR
4858 034530 022737 032064 034512 6$:  CMP    #32064,@#DA+2    ;:CHECK RESULT
4859 034536 001401                                     BEQ    7$                 ;:PASS
4860 034540 000000                                     HALT                                     ;:ERROR
4861 034542 022737 032064 034514 7$:  CMP    #32064,@#DA+4    ;:CHECK RESULT
4862 034550 001401                                     BEQ    ENDMPC            ;:PASS
4863 034552 000000                                     HALT                                     ;:CIS INSTUCTION OR SEQUENCE ERROR
4864
4865 034554 000240      ENDMPC: NOP
4866 034556 000240      NOP
4867 034560 000240      NOP

```

WARNING!

THE FOLLOWING CIS ABORT TEST MUST BE LOCATED BELOW
 MEMORY LOCATION 40000 DUE TO THE MEMORY MANAGEMENT
 SETUP REQUIRED IN THE TEST.

ALSO THE TEST DESTROYS THE CONTENTS OF MEMORY IN THE
 FOLLOWING RANGES:

57670-60070
 76000-76001

THIS WARNING IS PRIMARILY AIMED AT FUTURE MODIFIERS OF THIS CPU DIAGNOSTIC.

4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887

4888
 4889
 4890
 4891
 4892
 4893
 4894
 4895
 4896
 4897
 4898
 4899
 4900
 4901
 4902
 4903
 4904
 4905
 4906
 4907
 4908
 4909
 4910
 4911
 4912
 4913
 4914
 4915
 4916
 4917
 4918
 4919
 4920
 4921
 4922
 4923
 4924
 4925
 4926
 4927
 4928
 4929
 4930
 4931
 4932
 4933
 4934
 4935
 4936
 4937
 4938
 4939
 4940
 4941
 4942
 4943
 4944

000377
 076130

000337
 000200 143521
 000400 143512
 000400 177570
 001100
 001072
 005037 172516
 010667 001022
 000000 172340

```

*****
CIS STACK 'PROBE AHEAD' MEMORY MANAGEMENT ABORT TESTS

THE NEXT THREE SUBTESTS ARE AIMED AT TESTING THE KT
PAGE FAULT ROM (SCHEMATIC PAGE K1-8) AND ASSOCIATED LOGIC.
NOTE: THESE 3 SUBTESTS ARE OPTIONAL AND ARE SELECTED BY SETTING MFM
HARDWARE SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF
STANDALONE OPERATION OF THE DIAGNOSTIC. IN THE CASE OF
MANUFACTURING APT RUNTIME MODE THEN BIT 08 OF $SWREG IS SET TO 1
THRU APT SCRIPTING.

EACH OF THESE 3 SUBTESTS SETUP THE STACK POINTER SUCH THAT
WHEN THE CIS PROCESSOR CHECKS TO SEE IF THERE IS ENOUGH
SPACE ON THE STACK (200 BYTES) THE ANSWER FOUND SHOULD BE
NO BECAUSE A PORTION OF THE STACK IS IN PROTECTED MEMORY.

EACH OF THESE 3 SUBTESTS VERIFY THAT THE CIS INSTRUCTION ABORTS
UNDER SEVERAL CONDITIONS OF MEMORY MANAGEMENT PAGE PROTECTION.
ALL OF THE PAGE PROTECTION DATA IS LISTED IN TABLE 'PDRTAB'.

THIS TEST ALLOWS MFG. TO ELIMINATE
HAVING TO RUN THE CIS DIAGNOSTIC DURING QUICK
VERIFY TESTING OF THE CPU MODULE.
  
```

FILL=377
 MOVCI=76130

```

.SBTTL CIS STACK PROBE AHEAD MEM MGMT ABORT TEST
*****
;TEST 337 CIS STACK PROBE AHEAD MEM MGMT ABORT TEST
*****
  
```

```

TST337: INC (R2)           ;UPDATE TEST NUMBER
        CMP #337,(R2)     ;SEQUENCE ERROR?
        BNE 6$           ;BR TO ERROR HALT ON SEQ ERROR

        BITB #200,$ENVM   ;IS APT SIZING?
        BEQ 4$           ;NO; TRY HARDWARE SWITCH REGISTER
        BIT #400,$SWREG  ;YES APT IS SIZING; DOES APT SAY TO DO THIS TEST
        BEQ 5$           ;NO ; SKIP TEST
        BR 1$           ;YES, DO TEST
4$:     BIT #400,@#177570 ;DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?
        BEQ 5$           ;NO, SKIP TEST
        BR 1$           ;YES, DO TEST
5$:     JMP ENDABO
6$:     JMP ENDPASS-10

;SETUP PAR'S FOR DIRECT MAPPING
1$:     CLR @MMR3         ;CLEAR OUT D-SPACE ENABLES
        MOV SP,STK1     ;SAVE THE STACK POINTER
        MOV #0,@#KIPAR0 ;SETUP KERNEL I-SPACE PAR'S
  
```

4945	034654	012737	000200	172342	MOV #200,@#KIPAR1	
4946	034662	012737	000400	172344	MOV #400,@#KIPAR2	
4947	034670	012737	000600	172346	MOV #600,@#KIPAR3	
4948	034676	012737	001000	172350	MOV #1000,@#KIPAR4	
4949	034704	012737	001200	172352	MOV #1200,@#KIPAR5	
4950	034712	012737	001400	172354	MOV #1400,@#KIPAR6	
4951	034720	012737	177600	172356	MOV #177600,@#KIPAR7	
4952						
4953	034726	012737	000000	172240	MOV #0,@#SIPAR0	;SETUP SUPERVISOR I-SPACE PAR'S
4954	034734	012737	000200	172242	MOV #200,@#SIPAR1	
4955	034742	012737	000400	172244	MOV #400,@#SIPAR2	
4956	034750	012737	000600	172246	MOV #600,@#SIPAR3	
4957	034756	012737	001000	172250	MOV #1000,@#SIPAR4	
4958	034764	012737	001200	172252	MOV #1200,@#SIPAR5	
4959	034772	012737	001400	172254	MOV #1400,@#SIPAR6	
4960	035000	012737	177600	172256	MOV #177600,@#SIPAR7	
4961						
4962	035006	012737	000000	177640	MOV #0,@#UIPAR0	;SETUP USER I-SPACE PAR'S
4963	035014	012737	000200	177642	MOV #200,@#UIPAR1	
4964	035022	012737	000400	177644	MOV #400,@#UIPAR2	
4965	035030	012737	000600	177646	MOV #600,@#UIPAR3	
4966	035036	012737	001000	177650	MOV #1000,@#UIPAR4	
4967	035044	012737	001200	177652	MOV #1200,@#UIPAR5	
4968	035052	012737	001400	177654	MOV #1400,@#UIPAR6	
4969	035060	012737	177600	177656	MOV #177600,@#UIPAR7	
4970						
4971						
4972						;SETUP PDR'S FOR R/W ACCESS
4973						:
4974						
4975	035066				SETPDR:	
4976	035066	012700	172300		MOV #KIPDR0,R0	
4977	035072	012720	177406		1\$: MOV #177406,(R0)+	


```

4979 035076 020027 172316          CMP R0,#KIPDR7
4980 035102 101773                BLOS 1$
4981
4982 035104 012700 172200          MOV #SIPDR0,R0
4983 035110 012720 177406          2$: MOV #177406,(R0)+
4984 035114 020027 172216          CMP R0,#SIPDR7
4985 035120 101773                BLOS 2$
4986
4987 035122 012700 177600          MOV #UIPDR0,R0
4988 035126 012720 177406          3$: MOV #177406,(R0)+
4989 035132 020027 177616          CMP R0,#UIPDR7
4990 035136 101773                BLOS 3$
4991
4992                                ;KERNEL MODE CIS STACK PROBE AHEAD MEM MGMT ABORT SUBTEST
4993                                ;
4994
4995 035140                                KMTSTS:
4996 035140 012737 035546 000250      MOV #MMHDLR,@MMVEC                ;SETUP MEM MGMT INTERRUPT VECTOR
4997 035146 012737 000340 000252      MOV #PR7,@MMVEC+2
4998 035154 012701 035224                MOV #1$,R1                        ;SETUP INTR RETURN ADDRESS
4999 035160 012700 035672                MOV #PDRTAB,R0
5000 035164 012706 060070                MOV #60070,SP
5001
5002 035170 011037 172304          2$: MOV (R0),@#KIPDR2                ;PROTECT PART OF STACK
5003 035174 012737 000001 177572      MOV #1,@MMR0                      ;TURN ON MEMORY MGMT
5004
5005 035202 004767 000362                JSR PC,SAVR                        ;SAVE REGISTERS
5006
5007 035206 076130                MOVCI                              ;EXECUTE THE CIS INSTRUCTION
5008 035210 035560                SRC.PTR
5009 035212 035564                DST.PTR
5010 035214 000377                FILL
5011 035216 000240                NOP
5012 035220 000240                NOP
5013 035222 000000                HALT                                ;CIS INSTRUCTION SHOULD HAVE ABORTED BUT DIDN'T
5014
5015 035224 004767 000372          1$: JSR PC,RESR                      ;RESTORE REGISTERS
5016 035230 062700 000002          ADD #2,R0                          ;UPDATE PROTECTION SCHEME TO NEXT TABLE CASE
5017 035234 005710                TST (R0)                           ;ANY CASES LEFT TO TRY?
5018 035236 001354                BNE 2$                              ;BRANCH IF YES
5019
5020 035240 005037 177572          CLR @MMR0                          ;NO - PREPARE TO EXIT TEST
5021 035244 012737 177406 172304      MOV #177406,@#KIPDR2                ;RESTORE R/W ACCESS TO STACK AREA
5022 035252 000400                BR SMTSTS                          ;GO TO NEXT TEST
5023
5024
5025                                ;SUPERVISOR MODE CIS STACK PROBEAHEAD MEMORY MGMT ABORT SUBTEST
5026                                ;
5027
5028 035254                                SMTSTS:
5029 035254 012737 035546 000250      MOV #MMHDLR,@MMVEC                ;SETUP MEM MGMT INTERRUPT VECTOR
5030 035262 012737 040340 000252      MOV #040340,@MMVEC+2
5031 035270 012701 035354                MOV #1$,R1                        ;SETUP INTR RETURN ADDRESS
5032 035274 012700 035672                MOV #PDRTAB,R0
5033 035300 012737 040340 177776      MOV #040340,@#PSW                  ;SWITCH TO SUPERVISOR MODE
5034 035306 012706 060070                MOV #60070,SP
5035
    
```

```

5036 035312 011037 172204      2$:  MOV (R0),@#SIPDR2      ;PROTECT PART OF STACK
5037 035316 012737 000001 177572  MOV #1,@#MMR0          ;TURN ON MEMORY MGMT
5038
5039 035324 004767 000240      JSR PC,SAVR           ;SAVE REGISTERS
5040
5041 035330 076130      MOVCI                ;EXECUTE THE CIS INSTRUCTION
5042 035332 035560      SRC.PTR
5043 035334 035564      DST.PTR
5044 035336 000377      FILL
5045
5046 035340 000240      NOP
5047 035342 000240      NOP
5048 035344 012737 000340 177776  MOV #340,@#PSW       ;SWITCH BACK TO KERNEL MODE BEFORE HALT
5049 035352 000000      HALT                 ;CIS INSTRUCTION SHOULD HAVE ABORTED BUT DIDN'T
5050
5051 035354 004767 000242      1$:  JSR PC,RESR          ;RESTORE REGISTERS
5052 035360 062700 000002      ADD #2,R0            ;UPDATE PROTECTION SCHEME TO NEXT TABLE CASE
5053 035364 005710      TST (R0)             ;ANY CASES LEFT TO TRY?
5054 035366 001351      BNE 2$              ;BRANCH IF YES
5055
5056 035370 005037 177572      CLR @#MMR0           ;NO - PREPARE TO EXIT TEST
5057 035374 012737 177406 172204  MOV #177406,@#SIPDR2 ;RESTORE R/W ACCESS TO STACK AREA
5058 035402 000400      BR UMTSTS            ;GO TO NEXT TEST
5059
5060
5061      ;USER MODE CIS STACK PROBEAHEAD MEM MGMT ABORT SUBTEST
5062      ;
5063
5064 035404      UMTSTS:
5065 035404 012737 035546 000250  MOV #MMHDLR,@#MMVEC  ;SETUP MEM MGMT INTERRUPT VECTOR
5066 035412 012737 140340 000252  MOV #140340,@#MMVEC+2
5067 035420 012701 035504      MOV #1$,R1           ;SETUP INTR RETURN ADDRESS
5068 035424 012700 035672      MOV #PDRTAB,R0
5069 035430 012737 140340 177776  MOV #140340,@#PSW   ;SWITCH TO USER MODE
5070 035436 012706 060070      MOV #60070,SP
5071
5072 035442 011037 177604      2$:  MOV (R0),@#UIPDR2   ;PROTECT PART OF STACK
5073 035446 012737 000001 177572  MOV #1,@#MMR0       ;TURN ON MEMORY MGMT
5074
5075 035454 004767 000110      JSR PC,SAVR           ;SAVE REGISTERS
5076
5077 035460 076130      MOVCI                ;EXECUTE THE CIS INSTRUCTION
5078 035462 035560      SRC.PTR
5079 035464 035564      DST.PTR
5080 035466 000377      FILL
5081
5082 035470 000240      NOP
5083 035472 000240      NOP
5084 035474 012737 000340 177776  MOV #340,@#PSW       ;SWITCH BACK TO KERNEL MODE BEFORE HALT
5085 035502 000000      HALT                 ;CIS INSTRUCTION SHOULD HAVE ABORTED BUT DIDN'T
5086
5087 035504 004767 000112      1$:  JSR PC,RESR          ;RESTORE REGISTERS
5088 035510 062700 000002      ADD #2,R0            ;UPDATE PROTECTION SCHEME TO NEXT TABLE CASE
5089 035514 005710      TST (R0)             ;ANY CASES LEFT TO TRY?
5090 035516 001351      BNE 2$              ;BRANCH IF YES
5091
5092 035520 005037 177572      CLR @#MMR0           ;NO - PREPARE TO EXIT TEST
  
```

```

5093 035524 012737 177406 177604      MOV #177406,@#UIPDR2      ;RESTORE R/W ACCESS TO STACK AREA
5094 035532 012737 000340 177776      MOV #340,@#PSW          ;SWITCH BACK TO KERNEL MODE
5095 035540 016706 000124              MOV STK1,SP             ;RESTORE THE STACK POINTER
5096 035544 000472                      BR ENDABO               ;GO TO NEXT TEST
5097
5098
5099
5100      ;MEMORY MANAGEMENT TRAP HANDLER
5101      ;
5102 035546      MMHDLR:
5103 035546 005037 177572      CLR @MMMRC              ;TURN OFF MEM MGMT
5104 035552 005726              TST (SP)+               ;FIX UP STACK
5105 035554 005726              TST (SP)+
5106 035556 000111              JMP (R1)                 ;RETURN VIA R1
5107
5108
5109
5110      ;CIS INSTRUCTION SOURCE AND DESTINATION DESCRIPTORS
5111      ;
5112 035560 000001      SRE.PTR:                .WORD 1
5113 035562 076000              .WORD 76000
5114 035564 000001      DST.PTR:                .WORD 1
5115 035566 076001              .WORD 76001
5116
5117      ;SUBROUTINES
5118      ;
5119 035570 010067 000060      SAVR:  MOV R0,SVR0        ;SAVE, REGISTERS
5120 035574 010167 000056      MOV R1,SVR1
5121 035600 010267 000054      MOV R2,SVR2
5122 035604 010367 000052      MOV R3,SVR3
5123 035610 010467 000050      MOV R4,SVR4
5124 035614 010567 000046      MOV R5,SVR5
5125 035620 000207              RTS PC
5126
5127 035622 016700 000026      RESR:  MOV SVR0,R0        ;RESTORE REGISTERS
5128 035626 016701 000024      MOV SVR1,R1
5129 035632 016702 000022      MOV SVR2,R2
5130 035636 016703 000020      MOV SVR3,R3
5131 035642 016704 000016      MOV SVR4,R4
5132 035646 016705 000014      MOV SVR5,R5
5133 035652 000207              RTS PC
5134
5135 035654 000000      SVR0:  .WORD 0
5136 035656 000000      SVR1:  .WORD 0
5137 035660 000000      SVR2:  .WORD 0
5138 035662 000000      SVR3:  .WORD 0
5139 035664 000000      SVR4:  .WORD 0
5140 035666 000000      SVR5:  .WORD 0
5141
5142 035670 000000      STK1:  .WORD 0
5143
5144
5145      ;PROTECTION TABLE (WORD FORMAT = PDR FORMAT)
5146 035672      PDRTAB:
5147 035672 177000              177000      ;ACF=00      ED=0  PLF=176
5148 035674 177410              177410      ;ACF=00      ED=1  PLF=177
5149 035676 177400              177400      ;ACF=00      ED=0  PLF=177

```

5150	035700	100010	100010	:ACF=00	ED=1	PLF=0
5151						
5152	035702	177002	177002	:ACF=01	ED=0	PLF=176
5153	035704	177412	177412	:ACF=01	ED=1	PLF=177
5154	035706	177402	177402	:ACF=01	ED=0	PLF=177
5155	035710	100012	100012	:ACF=01	ED=1	PLF=0
5156						
5157	035712	177004	177004	:ACF=10	ED=0	PLF=176
5158	035714	177414	177414	:ACF=10	ED=1	PLF=177
5159	035716	177404	177404	:ACF=10	ED=0	PLF=177
5160	035720	100014	100014	:ACF=10	ED=i	PLF=0
5161						
5162	035722	177006	177006	:ACF=11	ED=0	PLF=176
5163	035724	177416	177416	:ACF=11	ED=1	PLF=177
5164	035726	000000	0			
5165						
5166						
5167	035730	000000	HALT	:TEST SEQUENCE ERROR		
5168	035732	000240	ENDABO: NOP			
5169	035734	000240	NOP			
5170	035736	000240	NOP			
5171						

```

5173 :*****
5174 :                               END OF PASS SEQUENCE
5175 :*****
5176
5177 035740 005212 ENDPASS: INC (R2) ;UPDATE TEST NUMBER
5178 035742 022712 000340 CMP #340,(R2) ;SEQUENCE ERROR?
5179 035746 001042 BNE EOP1 ;BR TO ERROR HALT ON SEQ ERROR
5180 035750 005237 000306 INC @#SPASS
5181 035754 105267 000106 INCB PASSPT ;SHOULD PRINT THIS PASS?
5182 035760 001033 BNE GOAGIN ;NO
5183 035762 132767 000040 142331 BITB #40,$ENVM ;WILL APT ALLOW PRINTING?
5184 035770 001017 BNE ACT ;NO
5185 035772 023727 000042 036040 CMP @#42,#SENDAD ;UNDER ACT AUTO ACCEPT?
5186 036000 001413 BEQ ACT ;IF SO SKIP PRINTOUT
5187 036002 012700 036070 MOV #MSG,R0 ;GET MSG ADDR.
5188 036006 105737 177564 WAIT: TSTB @#TPS ;TTY READY
5189 036012 100375 BPL WAIT ;NO WAIT
5190 036014 121027 000377 CMPB (R0),#377 ;IS NEXT CHAR. THE TERMINATOR
5191 036020 001403 BEQ ACT ;YES THEN BR
5192 036022 112037 177566 MOVB (R0)+,@#TPB ;PRINT CHARACTER
5193 036026 000767 BR WAIT ;NEXT IF NOT DONE.
5194 036030 013700 000042 ACT: MOV @#42,R0 ;CHECK ACT
5195 036034 001405 BEQ GOAGIN ;KEEP GOING
5196 036036 000005 RESET
5197 036040 004710 $SENDAD: JSR PC,(R0) ;ACT HOOKS
5198 036042 000240 NOP
5199 036044 000240 NOP
5200 036046 000240 NOP
5201 036050 000167 143046 GOAGIN: JMP RESTRT ;DO NEXT PASS
5202 036054 EOP1:
    036054 012742 000772 MOV #772,-(R2) ;MOVE TO MAILBOX # ***** 772 *****
    036060 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
    036062 000000 HALT ;SEQUENCE ERROR

5203
5204
5205 036064 001100 START
5206
5207 036066 177777 PASSPT: -1
5208 036070 015 012 000 MSG: .ASCII <15><12><0><0><0><0><0><0>.END OF.
    036073 000 000 000
    036076 000 000 105
    036101 116 104 040
    036104 117 106
5209 036106 040 040 103 .ASCII . CKKAAAG 11/44 CPU/EIS.<0><0><0><377>
    036111 113 113 101
    036114 101 101 060
    036117 040 061 061
    036122 057 064 064
    036125 040 103 120
    036130 125 057 105
    036133 111 123 000
    036136 000 000 377

5210 .EVEN
5211 036142 000402 BR .+6
5212 036144 001002 BNE .+6
5213 036146 001402 BEQ .+6
5214 036150 002002 BGE .+6
    
```

```

5215 036152 002402          BLT      .+6
5216 036154 003002          BGT      .+6
5217 036156 003402          BLE      .+6
5218 036160 100002          BPL      .+6
5219 036162 100402          BMI      .+6
5220 036164 101002          BHI      .+6
5221 036166 101402          BLOS     .+6
5222 036170 102002          BVC      .+6
5223 036172 102402          BVS      .+6
5224 036174 103002          BCC      .+6          ;SAME AS BHIS
5225 036176 103402          BCS      .+6          ;SAME AS BLO
5226 036200 000000          $TMP0:   .WORD 0
5227 036202 000000          $TMP1:   .WORD 0
5228 036204 000000          $TMP2:   .WORD 0
5229 036206 000000          $TMP3:   .WORD 0
5230 036210 000000          $EPIRQ:  .WORD 0 ;ERROR PIRQ.
5231 036212 000000          $LPADR:  .WORD 0
5232 036214 000000          $LPERR:  .WORD 0 ;ERROR LOOP
5233          000002
5234 036216 177777          .RADIX   2
                    YNTAB: 1111111111111111          ;BR
5235 036220 170360          1111000011110000          ;BNE:  Z=0
5236 036222 007417          0000111100001111          ;BEQ:  Z=1
5237 036224 146063          1100110000110011          ;BGE:  N XOR V =0
5238 036226 031714          0011001111001100          ;BLT:  N XOR V =1
5239 036230 140060          1100000000110000          ;BGT:  Z+(N XOR V) =0
5240 036232 037717          0011111111001111          ;BLE:  Z+(N XOR V) -1
5241
5242 036234 177400          1111111100000000          ;BPL:  N=0
5243 036236 000377          0000000011111111          ;BMI:  N=1
5244 036240 120240          1010000010100000          ;BHI:  C+Z=0
5245 036242 057537          0101111101011111          ;BLOS: C+Z-1
5246 036244 146314          1100110011001100          ;BVC:  V=0
5247 036246 031463          0011001100110011          ;BVS:  V=1
5248 036250 125252          1010101010101010          ;BCC:  C=0
5249 036252 052525          0101010101010101          ;BCS:  C=1
5250          000010          .RADIX   8
5251
5252 036254 012737 036264 000024 PWRDN:  MOV      #PWRUP,@#24          ;SET UP FOR A POWER UP
5253 036262 000000          HALT
5254
5255 036264 012737 036254 000024 PWRUP:  MOV      #PWRDN,@#24          ;SET UP FOR A POWER FAIL
5256 036272 012706 001000          MOV      #STBOT,R6          ;SET UP STACK POINTER
5257 036276 132767 000040 142015 BITB    #40,$ENVM          ;SHOULD PRINT?
5258 036304 001010          BNE      PWR2              ;IF NOT: BR
5259 036306 012700 036332          MOV      #PFMES,R0          ;GET POWER FAIL MESSG.
5260 036312 105737 177564          WATE:   TSTB   @#TPS          ;TTY READY?
5261 036316 100375          BPL      WATE              ;IF NOT: BR
5262 036320 112037 177566          MOVB    (R0)+,@#TPB          ;PRINT NEXT CHAR.
5263 036324 001372          BNE      WATE              ;IF NOT DONE: BR
5264 036326 000137 001100          PWR2:   JMP      @#START          ;START PROGRAM AGAIN
5265
5266
5267 036332          012          015          120          PFMES:  .ASCIZ  <12><15>.POWER FAILURE.<12><15>
          036335          117          127          105
          036340          122          040          106
          036343          101          111          114
          036346          125          122          105
  
```

```

036351 012 015 000
5268 .EVEN
5269 036354 .BLKW 6
5270 036370 USTBOT:
5271 :*****
5272 : THE FOLLOWING ARE SPECIAL CPU TRAP
5273 : HANDLERS TO TRAP AND REPORT SPECIAL TRAPS.
5274 :
5275 :*****
5276
5277 036370 T04:
036370 012742 000773 MOV #773,-(R2) ;MOVE TO MAILBOX # ***** 773 *****
036374 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
036376 000000 HALT ;TRAPPED THRU LOC. 4
5278 036400 T010:
036400 012742 000774 MOV #774,-(R2) ;MOVE TO MAILBOX # ***** 774 *****
036404 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
036406 000000 HALT ;TRAPPED THRU LOC. 10
5279 036410 T014:
036410 012742 000775 MOV #775,-(R2) ;MOVE TO MAILBOX # ***** 775 *****
036414 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
036416 000000 HALT ;TRAPPED THRU LOC. 14
5280 036420 T030:
036420 012742 000776 MOV #776,-(R2) ;MOVE TO MAILBOX # ***** 776 *****
036424 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
036426 000000 HALT ;TRAPPED THRU LOC. 30
5281 036430 T034:
036430 012742 000777 MOV #777,-(R2) ;MOVE TO MAILBOX # ***** 777 *****
036434 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
036436 000000 HALT ;TRAPPED THRU LOC. 34
5282 036440 T0114:
036440 012742 001000 MOV #1000,-(R2) ;MOVE TO MAILBOX # ***** 1000 *****
036444 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
036446 000000 HALT ;TRAPPED THRU LOC. 114
5283 036450 T0244:
036450 012742 001001 MOV #1001,-(R2) ;MOVE TO MAILBOX # ***** 1001 *****
036454 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
036456 000000 HALT ;TRAPPED THRU LOC. 244
5284 036460 T0250:
036460 012742 001002 MOV #1002,-(R2) ;MOVE TO MAILBOX # ***** 1002 *****
036464 005242 INC -(R2) ;SET MSGTYP TO FATAL ERROR
036466 000000 HALT ;TRAPPED THRU LOC. 250
5285 036470 SPSW: 0
5286 036472 SUBT .WORD 0
5287 036474 SUBT .WORD SUBT
5288 036476 000002 S1:
5289 036500 036476 S2: S1
5290 036502 177771 S3: -7
5291 036504 036502 S4: S3
5292
5293 036506 000000 END: L
5294 000001
  
```

A	034466	APASS =	00000C	BIT7 =	000200	DEC5	017766	DOP2	010550
ABASE =	000000	APRIOR=	000070	BIT8 =	000400	DEC6	017776	DOP4	014052
ACDW1 =	000000	AROUND	023766	BIT9 =	001000	DEC7	020020	DOP5	014136
ACDW2 =	000000	ASL1	021770	BPTVEC=	000014	DISPRE	000174	DST.PT	035564
ACPUOP=	000000	ASL2	022000	BRCT	024024	DNMB0A	011120	DSWR =	177570
ACT	036030	ASL3	022016	BRC1	003440	DNMB0B	011130	EIS	025214
ADC1	020454	ASL4	022026	BRC2	003450	DNMB2A	011356	EMTVEC=	000030
ADC2	020464	ASL5	022042	BRC3	003460	DNMB2B	011366	END	036506
ADC3	020504	ASL6	022052	BRH	023776	DNMB2C	011402	ENDABO	035732
ADC4	020514	ASL7	022076	BRN1	003320	DNMB2D	011416	ENDMPC	034554
ADC5	020532	ASR1	022140	BRN2	003330	DNMB2E	011426	ENDPAS	035740
ADDNI =	076150	ASR2	022150	BRN3	003340	DNMB2F	011444	EOP1	036054
ADDW0 =	000000	ASR3	022172	BRTAB	036142	DNMB3A	011526	ER	024010
ADDW1 =	000000	ASR4	022202	BRV1	003370	DNMB3B	011536	ERROR =	104000
ADDW10=	000000	ASR5	022216	BRV2	003400	DNMB3C	011554	ERRVEC=	000004
ADDW11=	000000	ASR6	022226	BRV3	003410	DNMB3D	011572	FILL =	000377
ADDW12=	000000	ASR7	022256	BRZ1	003250	DNMB3E	011602	GOAGIN	036050
ADDW13=	000000	ASWREG=	000000	BRZ2	003260	DNMB4A	011772	HIADRS=	177742
ADDW14=	000000	ATESTN=	000000	BRZ3	003270	DNMB4B	012002	HITMIS=	177752
ADDW15=	000000	AUNIT =	000000	BR1	001172	DNMB4C	012020	HT =	000011
ADDW2 =	000000	AUSWR =	000000	BR2	001202	DNMB4D	012030	IJMP	016434
ADDW3 =	000000	AVECT1=	000000	BR3	001214	DNMB4E	012040	IJMP4	016206
ADDW4 =	000000	AVECT2=	000000	BR4	001222	DNMB4F	012054	IJMP5	016376
ADDW5 =	000000	B	034472	BR5	001232	DNM03A	010212	INC1	017600
ADDW6 =	000000	BIC1	017420	BTCON	024314	DNM03B	010222	INC2	017610
ADDW7 =	000000	BIC2	017430	BTERR	024300	DNM03C	010232	INC3	017632
ADDW8 =	000000	BIC3	017446	CACHVE=	000114	DNM1	010064	INC4	017642
ADDW9 =	000000	BIS1	017510	CC	023772	DNM1A	011176	INC5	017656
ADD1	020270	BIS2	017520	CCERR	024534	DNM1B	011206	IOTVEC=	000020
ADD2	020300	BIS3	017540	CC1	024422	DNM2	010100	JMPCK	016436
ADD3	020314	BITCHK	024156	CC2	024436	DNM2A	011254	JMPERR	024350
ADD4	020324	BITCLR	024104	CHECK	034516	DNM2B	011264	JMPSEQ	016456
ADD5	020342	BITCON	024240	CLRCD	024420	DNM2C	011272	JMPT	024340
ADD6	020352	BITSET	024122	CLR1	020056	DNM2D	011302	JMP2	016210
ADD7	020364	BIT0 =	000001	CMP1	020720	DNM3	010116	JMP2A	016226
ADD8	020374	BIT00 =	000001	CMP2	020730	DNM4	010140	JMP3	016136
ADD9	020414	BIT01 =	000002	CMP3	020752	DNM4A	011664	JMP3A	016154
ADEVCT=	000000	BIT02 =	000004	CMP4	020762	DNM4B	011674	JMP3B	016174
ADEVN =	000000	BIT03 =	000010	CMP5	021006	DNM4C	011710	JMP4	016240
AENV =	000000	BIT04 =	000020	CMP6	021016	DNM5A	012134	JMP4A	016256
AENVN =	000000	BIT05 =	000040	CMP7	021036	DNM5B	012144	JMP4B	016276
AFATAL=	000000	BIT06 =	000100	COM1	021076	DNM5C	012162	JMP5	016344
AMADR1=	000000	BIT07 =	000200	CONT	024026	DNM6A	012242	JMP5A	016364
AMADR2=	000000	BIT08 =	000400	CONTRL=	177746	DNM6B	012252	JMP6	016310
AMADR3=	000000	BIT09 =	001000	CON1	024452	DNM6C	012270	JMP6A	016330
AMADR4=	000000	BIT1 =	000002	CON2	024544	DNM7A	012352	JMP7	016400
AMAMS1=	000000	BIT10 =	002000	CPUErr=	177766	DNM7B	012362	JMP7A	016420
AMAMS2=	000000	BIT11 =	004000	CR =	000015	DNM7C	012400	JSRCK	017110
AMAMS3=	000000	BIT12 =	010000	CRLF =	000200	DOPB2A	010624	JSRCKA	017104
AMAMS4=	000000	BIT13 =	020000	D	034476	DOPB2B	010702	JSRCK1	017126
AMSGAD=	000000	BIT14 =	040000	DA	034510	DOPA	007626	JSRSEQ	017106
AMSGLG=	000000	BIT15 =	100000	DAERR	024230	DOP0B	007652	JSR0	016472
AMSGTY=	000000	BIT2 =	000004	DDISP =	177570	DOP0C	007672	JSR1	016476
AMTYP1=	000000	BIT3 =	000010	DEC1	017716	DOP0D	007722	JSR1A	016520
AMTYP2=	000000	BIT4 =	000020	DEC2	017726	DOP03A	010000	JSR2	016610
AMTYP3=	000000	BIT5 =	000040	DEC3	017742	DOP03B	010010	JJR2A	016640
AMTYP4=	000000	BIT6 =	000100	DEC4	017752	DOP1	010436	JSR2B	016650

JSR3	016530	MAPH00=	170202	MAPL2 =	170210	MDM7A	013676	NEG50	005766
JSR3A	016570	MAPH01=	170206	MAPL20=	170300	MDM7B	013706	NEG51	005776
JSR3B	016600	MAPH02=	170212	MAPL21=	170304	MDM7C	013724	NEG52	006012
JSR4	016672	MAPH03=	170216	MAPL22=	170310	MDM7D	013744	NEG60	006070
JSR4A	016710	MAPH04=	170222	MAPL23=	170314	MDM7E	013770	NEG61	006100
JSR4B	016720	MAPH05=	170226	MAPL24=	170320	MEMERR=	177744	NEG70	006150
JSR5	016776	MAPH06=	170232	MAPL25=	170324	MFPD0	024642	NEG71	006160
JSR5A	017022	MAPH07=	170236	MAPL26=	170330	MFPDOA	024670	PASSPT	036066
JSR5B	017032	MAPH1 =	170206	MAPL27=	170334	MFP10	023500	PDRTAB	035672
JSR6	016732	MAPH10=	170242	MAPL3 =	170214	MFP10A	023526	PFMES	036332
JSR6A	016756	MAPH11=	170246	MAPL30=	170340	MFPT =	000007	PIRQ =	177772
JSR6AD	017102	MAPH12=	170252	MAPL31=	170344	MMHDLR	035546	PIRQVE=	000240
JSR6B	016766	MAPH13=	170256	MAPL32=	170350	MMR0 =	177572	PRO =	000000
JSR7	017044	MAPH14=	170262	MAPL33=	170354	MMR1 =	177574	PR1 =	000040
JSR7A	017062	MAPH15=	170266	MAPL34=	170360	MMR2 =	177576	PR2 =	000100
JSR7B	017072	MAPH16=	170272	MAPL35=	170364	MMR3 =	172516	PR3 =	000140
KDPAR0=	172360	MAPH17=	170276	MAPL36=	170370	MMVEC =	000250	PR4 =	000200
KDPAR1=	172362	MAPH2 =	170212	MAPL37=	170374	MOVCI =	076130	PR5 =	000240
KDPAR2=	172364	MAPH20=	170302	MAPL4 =	170220	MOV1	017240	PR6 =	000300
KDPAR3=	172366	MAPH21=	170306	MAPL5 =	170224	MOV2	017250	PR7 =	000340
KDPAR4=	172370	MAPH22=	170312	MAPL6 =	170230	MOV3	017266	PS =	177776
KDPAR5=	172372	MAPH23=	170316	MAPL7 =	170234	MRK1	022646	PSW =	177776
KDPAR6=	172374	MAPH24=	170320	MBDM2A	012614	MRK2	022670	PUSRW =	030000
KDPAR7=	172376	MAPH25=	170326	MBDM2B	012624	MRK3	022700	PWRDN	036254
KDPDR0=	172320	MAPH26=	170332	MBDM2C	012642	MRK4	022722	PWRUP	036264
KDPDR1=	172322	MAPH27=	170336	MBDM2D	012654	MRK5	022734	PWRVEC=	000024
KDPDR2=	172324	MAPH3 =	170216	MBDM2E	012664	MRK6	022750	PWR2	036326
KDPDR3=	172326	MAPH30=	170342	MBDM2F	012702	MSG	036070	REG1	002130
KDPDR4=	172330	MAPH31=	170346	MBDM4A	013236	MTPD0	024750	REG1A	002174
KDPDR5=	172332	MAPH32=	170352	MBDM4B	013254	MTP10	023612	REG1E	002142
KDPDR6=	172334	MAPH33=	170356	MBDM4C	013266	NBR	024002	REG2	002244
KDPDR7=	172336	MAPH34=	170362	MBDM4D	013276	NEG00	004410	REG2A	002272
KERSTK=	001100	MAPH35=	170366	MBDM4E	013312	NEG01	004420	REG2B	002320
KIPAR0=	172340	MAPH36=	170372	MDM1A	012450	NEG02	004434	REG2C	002352
KIPAR1=	172342	MAPH37=	170376	MDM1B	012460	NEG03	004450	REG3	002400
KIPAR2=	172344	MAPH4 =	170222	MDM2A	012522	NEG04	004460	REG3A	002444
KIPAR3=	172346	MAPH5 =	170226	MDM2B	012532	NEG1	020574	REG3E	002412
KIPAR4=	172350	MAPH6 =	170232	MDM2C	012540	NEG10	004524	REG4	002514
KIPAR5=	172352	MAPH7 =	170236	MDM2D	012550	NEG11	004534	REG4A	002560
KIPAR6=	172354	MAPL0 =	170200	MDM3A	012756	NEG12	004552	REG4E	002526
KIPAR7=	172356	MAPL00=	170200	MDM3B	012766	NEG13	004566	REG5	002630
KIPDR0=	172300	MAPL01=	170204	MDM3C	013004	NEG14	004576	REG5A	002674
KIPDR1=	172302	MAPL02=	170210	MDM3D	013024	NEG2	020604	REG5E	002642
KIPDR2=	172304	MAPL03=	170214	MDM3E	013052	NEG20	004644	REG6	002744
KIPDR3=	172306	MAPL04=	170220	MDM4A	013132	NEG21	004654	REG6A	003010
KIPDR4=	172310	MAPL05=	170224	MDM4B	013142	NEG22	004702	REG6E	002756
KIPDR5=	172312	MAPL06=	170230	MDM4C	013156	NEG3	020626	REN	033434
KIPDR6=	172314	MAPL07=	170234	MDM5A	013362	NEG30	005224	RESR	035622
KIPDR7=	172316	MAPL1 =	170204	MDM5B	013372	NEG31	005234	REST	023030
KMTSTS	035140	MAPL10=	170240	MDM5C	013410	NEG32	005250	RESTR	001122
KSP =	X000006	MAPL11=	170244	MDM5D	013426	NEG33	005274	RESVEC=	000010
LF =	000012	MAPL12=	170250	MDM5E	013454	NEG34	005310	ROL1	021460
LKS =	177546	MAPL13=	170254	MDM6A	013524	NEG4	020636	ROL2	021470
LKVEC =	000100	MAPL14=	170260	MDM6B	013534	NEG40	005666	ROL3	021506
LOADRS=	177740	MAPL15=	170264	MDM6C	013552	NEG41	005676	ROL4	021516
MAINT =	177750	MAPL16=	170270	MDM6D	013572	NEG42	005712	ROL5	021532
MAPHO =	170202	MAPL17=	170274	MDM6L	013622	NEG5	020656	ROL6	021542

ROL7	021564	SB2	015544	SIZELO=	177760	SOP1B	003736	SW08	= 0004C0
ROR1	021626	SB4	015670	SMTSTS	035254	SOP2B	004170	SW09	= 0C1000
ROR2	021636	SB5	015756	SNMB0A	006354	SOP3A	004754	SW1	= 000002
ROR3	021654	SB5A	015750	SNMB1A	006460	SOP3B	004770	SW10	= 002000
ROR4	021664	SB5X	015766	SNMB1B	006470	SOP4A	005366	SW11	= 004000
ROR5	021702	SB5XAD	015770	SNMB1C	006512	SOP4B	005406	SW12	= 010000
ROR6	021712	SB6	016030	SNMB2A	006634	SOP5A	005450	SW13	= 020000
ROR7	021726	SB6X	016040	SNMB2B	006644	SOP5B	005464	SW14	= 040000
ROTX	015244	SB7	016100	SNMB2C	006660	SOP6A	005530	SW15	= 100000
ROTXAD	015370	SB7X	016110	SNMB2D	006700	SOP6B	005544	SW2	= 000004
ROTOA	014362	SB7XAD	016112	SNMB2E	006710	SOP7A	005612	SW3	= 000010
ROTOB	014372	SCOPE =	000004	SNMB3A	007052	SOP7B	005626	SW4	= 000020
ROTOC	014414	SC3	024514	SNMB3B	007062	SPSW	036470	SW5	= 000040
ROT1A	014462	SC4	024530	SNMB3C	007100	SRC.PT	035560	SW6	= 000100
ROT1B	014472	SDPAR0=	172260	SNMB3D	007110	SR0	= 177572	SW7	= 000200
ROT1C	014516	SDPAR1=	172262	SNM0A	006314	SR1	= 177574	SW8	= 000400
ROT1D	014526	SDPAR2=	172264	SNM1A	006416	SR2	= 177576	SW9	= 001000
ROT1E	014556	SDPAR3=	172266	SNM2A	006554	SR3	= 172516	SXT0	022324
ROT2A	014630	SDPAR4=	172270	SNM2B	006564	SSP	= %000006	SXT1	022334
ROT2B	014640	SDPAR5=	172272	SNM3A	006764	SSP1A	023100	SXT2	022364
ROT2C	014670	SDPAR6=	172274	SNM3B	006774	SSP2	023114	SYSTID=	177764
ROT2D	014700	SDPAR7=	172276	SNM4A	007160	STACK	= 001100	S1	036476
ROT2E	014734	SDPDR0=	172220	SNM4B	007170	START	001100	S2	036500
ROT3A	015002	SDPDR1=	172222	SNM5A	007242	STBOT	= 001000	S3	036502
ROT3B	015012	SDPDR2=	172224	SNM5B	007252	STKMT=	177774	S4	036504
ROT3C	015040	SDPDR3=	172226	SNM6A	007326	STK1	035670	TBITVE=	000014
ROT3D	015050	SDPDR4=	172230	SNM6B	007336	SUBT	036472	TBL1	014072
ROT3E	015076	SDPDR5=	172232	SNM7A	007410	SUBT1	036474	TBL2	014156
ROT4	015152	SDPDR6=	172234	SNM7B	007420	SUB0	007554	TEST1	020114
ROT5	015234	SDPDR7=	172236	SOB1	022526	SUB0A	007564	TEST2	020124
ROT6	015304	SETBR	023662	SOB2	022534	SUB1	021140	TEST3	020142
ROT7	015360	SETCC	023702	SOB3	022544	SUB2	021150	TKVEC	= 000060
RTS1	017174	SETCD	024512	SOB4	022564	SUB3	021172	TO10	036400
R1ERR	002212	SETPDR	035066	SOPA	006232	SUB4	021202	TO114	036440
R10	=%000000	SETUP	023644	SOPB	006252	SUB5	021220	TO14	036410
R11	=%000001	SET2BR	023752	SOPB0A	003656	SUB6	021230	TO244	036450
R12	=%000002	SHL	001600	SOPB0B	003666	SUB7	021250	TO250	036460
R13	=%000003	SHLE	001614	SOPB1A	004000	SUPSTK=	000700	TO30	036420
R14	=%000004	SHR	001714	SOPB1B	004016	SVR0	035654	TO34	036430
R15	=%000005	SHRE	001730	SOPB1C	004062	SVR1	035656	TO4	036370
R2ERR	002342	SIPAR0=	172240	SOPB1D	004102	SVR2	035660	TPB	= 177566
R3ERR	002462	SIPAJ 1=	172242	SOPB2A	004236	SVR3	035662	TPS	= 177564
R4ERR	002576	SIPAJ 2=	172244	SOPB2B	004256	SVR4	035664	TPVEC	= 000064
R5ERR	002712	SIPAR3=	172246	SOPB2C	004326	SVR5	035666	TRAPVE=	000034
R6	=%000006	SIPAR4=	172250	SOPB2D	004352	SWB1	020202	TRTVEC=	000014
R6ERR	003026	SIPAR5=	172252	SOPB3A	005040	SWB2	020212	TST1	001154
R7	=%000007	SIPAR6=	172254	SOPB3B	005064	SWB3	020230	TST10	001554
SA	03'502	SIPAR7=	172256	SOPB3C	005132	SWREG	000176	TST100	006522
SAVR	035570	SIPDR0=	172200	SOPB3D	005154	SW0	= 000001	TST101	006602
SBC1	021312	SIPDR1=	172202	SOPX	006216	SW00	= 000001	TST102	006726
SBC2	021322	SIPDR2=	172204	SOPXAD	006262	SW01	= 000002	TST103	007012
SBC3	021340	SIPDR3=	172206	SOPZA	004150	SW02	= 000004	TST104	007126
SBC4	021350	SIPDR4=	172210	SOP0A	003514	SW03	= 000010	TST105	007204
SBC5	021366	SIPDR5=	172212	SOP0B	003534	SW04	= 000020	TST106	007270
SBC6	021376	SIPDR6=	172214	SOP0C	003576	SW05	= 000040	TST107	007352
SBC7	021416	SIPDR7=	172216	SOP0D	003622	SW06	= 000100	TST11	007624
SBO	015422	SIZEHI	177762	SOP1A	003724	SW07	= 000200	TST110	007434

TST111	007470	TST174	015562	TST256	025544	TST34	003076	UIPAR1=	177642
TST112	007524	TST175	015624	TST257	025634	TST35	003134	UIPAR2=	177644
TST113	007600	TST176	015704	TST26	002536	TST36	003172	UIPAR3=	177646
TST114	007742	TST177	015772	TST260	025724	TST37	003230	UIPAR4=	177650
TST115	010032	TST2	001244	TST261	026014	TST4	001336	UIPAR5=	177652
TST116	010156	TST20	002152	TST262	026104	TST40	003300	UIPAR6=	177654
TST117	010242	TST200	016042	TST263	026216	TST41	003350	UIPAR7=	177656
TST12	001670	TST201	016114	TST264	026330	TST42	003420	UIPDR0=	177600
TST120	010300	TST202	016460	TST265	026442	TST43	003470	UIPDR1=	177602
TST121	010336	TST203	017136	TST266	026556	TST44	003550	UIPDR2=	177604
TST122	010374	TST204	017212	TST267	026666	TST45	003632	UIPDR3=	177606
TST123	010454	TST205	017276	TST27	002606	TST46	003676	JIPDR4=	177610
TST124	010514	TST206	017366	TST270	026776	TST47	003746	UIPDR5=	177612
TST125	010566	TST207	017456	TST271	027106	TST5	001374	UIPDR6=	177614
TST126	010642	TST21	002222	TST272	027216	TST50	004026	UIPDR7=	177616
TST127	010720	TST210	017550	TST273	027332	TST51	004112	UMTSTS	035404
TST13	001740	TST211	017666	TST274	027444	TST52	004200	USESTK=	000600
TST130	010762	TST212	020030	TST275	027546	TST53	004266	USP	=%000006
TST131	011024	TST213	020066	TST276	027650	TST54	004362	USP1	023054
TST132	011066	TST214	020152	TST277	027752	TST55	004474	USP2	02314C
TST133	011144	TST215	020240	TST3	001300	TST56	004614	USP2A	023224
TST134	011222	TST216	020424	TST30	002652	TST57	004720	USP3	023250
TST135	011320	TST217	020542	TST300	030054	TST6	001432	USP4	023274
TST136	011462	TST22	002276	TST301	030156	TST60	005000	USP5	023350
TST137	011622	TST220	020666	TST302	030260	TST61	005074	USP6	023374
TST14	001770	TST221	021046	TST303	030362	TST62	005164	USRM	= 140000
TST140	011730	TST222	021106	TST304	030464	TST63	005334	USTBOT	036370
TST141	012072	TST223	021260	TST305	030576	TST64	005416	WAIT	036006
TST142	012202	TST224	021426	TST306	030710	TST65	005474	WATE	036312
TST143	012310	TST225	021574	TST307	031022	TST66	005554	XBIT1	017330
TST144	012420	TST226	021736	TST31	002722	TST67	005636	XBIT2	017340
TST145	012474	TST227	022106	TST310	031132	TST7	001476	XBIT3	017356
TST146	012566	TST23	002356	TST311	031242	TST70	005726	XOR1	022440
TST147	012722	TST230	022266	TST312	031352	TST71	006030	XOR2	022450
TST15	002022	TST231	022374	TST313	031462	TST72	006114	XOR3	022476
TST150	013100	TST232	022506	TST314	031570	TST73	006204	YBR	024006
TST151	013174	TST233	022574	TST315	031676	TST74	006264	YNTAB	036216
TST152	013330	TST234	022760	TST316	032004	TST75	006324	\$APTHD	000330
TST153	013472	TST235	023030	TST317	032112	TST76	006364	\$CPUOP	000326
TST154	013642	TST236	023140	TST32	002766	TST77	006426	\$DEVCT	000310
TST155	014010	TST237	023420	TST320	032220	UDPAR0=	177660	\$ENDAD	036040
TST156	014074	TST24	002422	TST321	032300	UDPAR1=	177662	\$ENV	000320
TST157	014160	TST240	023526	TST322	032360	UDPAR2=	177664	\$ENVM	000321
TST16	002054	TST241	023634	TST323	032526	UDPAR3=	177666	\$EPIRQ	036210
TST160	014244	TST242	024074	TST324	032634	UDPAR4=	177670	\$ERN	= 001003
TST161	014330	TST243	024244	TST325	032742	UDPAR5=	177672	\$ERROR=	000302
TST162	014424	TST244	024320	TST326	033022	UDPAR6=	177674	\$ETABL	000320
TST163	014566	TST245	024360	TST327	033102	UDPAR7=	177676	\$ETEND	000330
TST164	014744	TST246	024564	TST33	003036	UDPDR0=	177620	\$FATAL	000302
TST165	015106	TST247	024670	TST330	033162	UDPDR1=	177622	\$HIBTS	000330
TST166	015162	TST25	002472	TST331	033374	UDPDR2=	177624	\$LPADR	036212
TST167	015246	TST250	024772	TST332	033512	UDPDR3=	177626	\$LPERR	036214
TST17	002106	TST251	025024	TST333	033576	UDPDR4=	177630	\$MAIL	000300
TST170	015314	TST252	025214	TST334	034050	UDPDR5=	177632	\$MBADR	000332
TST171	015372	TST253	025310	TST335	034212	UDPDR6=	177634	\$MSGAD	000314
TST172	015440	TST254	025374	TST336	034340	UDPDR7=	177636	\$MSGLG	000316
TST173	015502	TST255	025460	TST337	034562	UIPAR0=	177640	\$MSGTY	000300

SYMBOL TABLE

\$PASS	000306	\$SWREG	000322	\$TMP2	036204	\$TSTNM=	000304	\$X	=	034350
\$PASTM	000336	\$TESTN	000304	\$TMP3	036206	\$UNIT	000312	\$XX	=	177703
\$SVPC -	000200	\$TMP0	036200	\$TN =	000337	\$UNITM	000340	\$XXX	=	000702
\$SWR -	000000	\$TMP1	036202	\$TSTM	000334	\$USWR	000324	.\$X	=	001100

. ABS. 036510 000
000000 001

ERRORS DETECTED: 3

VIRTUAL MEMORY USED: 20354 WORDS (80 PAGES)

DYNAMIC MEMORY: 20434 WORDS (78 PAGES)

ELAPSED TIME: 00:16:02

CKKAAA,CKKAAA/-SP/CRF/NL:TOC=CKKAAA0.MLB/ML,(CKKAAA.P11