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 I AND EXECUTING ENTIRE DIAGNOSTIC).

LOAD ADDRESS = 200
START

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 15C,
   $TESTN=77.

2. LOAD STARTING ADDRESS OF SUBTEST IN LOC. 215
3. LOAD ADDRESS = ADDRESS OF SUBTEST
   START

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 APTII MONITOR.

2.3 PROGRAM OPTIONS

THIS PROGRAM IS INTENDED TO BE A BASIC PROCESSOR TEST.
IT IS INTENDED TO BE THE LOWEST LEVEL DIAGNOSTIC RUN.
IT PROVIDES FOR NO SELECTABLE OPTIONS.

IN ORDER THAT THE TEST BE RUNNABLE ON A PROCESSOR WITHOUT A
TELETYPewriter. IT IS POSSIBLE TO SUPPRESS THE END OF PASS MESSAGE.
IF NO TELETYPewriter IS AVAILABLE, ALTER THE BYTE, $ENVM, WHICH
IS LOCATED IN THE APT MAILBOX. SETTING $ENVM TO 40(B) WILL
SUPPRESS ALL CONSOLE OUTPUT.
The exact location of this byte can be found in the symbol table at the end of the listing.

2.4 EXECUTION TIMES

The diagnostic completes one pass in less than 1 sec. 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. 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.)

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 to diagnostic interface specification. It is the primary medium for identifying errors.

3.2.1 $MSGTP
3.2.2 $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.

3.2.3 $PASS

This location is incremented for every complete pass of the diagnostic. Monitoring this 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.

3.2.4 $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 the subtest in which they were discovered.

4.0 PROGRESS REPORT

At the end of each successful pass the program increments the location $PASS which is in the APT mailbox. This location will
ALWAYS CONTAIN THE NUMBER OF SUCCESSFUL PASSES COMPLETED. $PASS IS RESET WITH EVERY RETRY FROM LOC. 200.

ADDITIONALLY, THE MESSAGE END OF DGKAA IS PRINTED ON THE CONSOLE TELETYPewriter AFTER THE FIRST PASS AND FOLLOWING EVERY 400TH PASS THEREAFTER.

IF NO TELETYPewriter IS AVAILABLE, THE CONSOLE OUTPUT MUST BE SUPPRESSED. (SEE SECTION 2.3)

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 APT 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 RB 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 NOT TELETYPewriter IS AVAILABLE TO REPORT THE PRINTED PROGRESS REPORTS.

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
-----

% .TITLE MAINDEC-11-DGKAA 11/04 CPU TEST
 .ENABLE ABS
 STBOT=500
 .MLIST CMD,MC,MD
 .LIST ME
 SCOPE=NOP
 R7+%7
 R6+%6
 PS=177776
 TPS=177564
 TPB=177566
 406
 .MCALL $APTHDR,$APTLS,$ACTII
 .SBTTL ACTII HOOKS

;***********************************************************************************************
;HOOKS REQUIRED BY ACTII
 $SVPC=. ;SAVE PC
.SBTL APT MAILBOX-ETABLE

EVEN
$MAIL: .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
$ENVNM: .BYTE AENVNM ; ENVIRONMENT MODE BITS
$SWREG: .WORD ASWREG ; USER SWITCH REGISTER
$USRWR: .WORD AUSRWR ; CPU TYPE, OPTIONS
$CPUOP: .WORD ACPUOP ; CPU TYPE, OPTIONS

000330

.SBTL APT PARAMETER BLOCK

;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;******************************************************************************
.46 .SET LOCATION
+46 .24 .SET POWER FAIL TO POINT TO START OF PROGRAM
+52 .200 .FOR APT START UP
+55 .44 .POINT TO APT INDIRECT ADDRESS PNTN.
+$APTHDR .5X .POINT TO APT HEADER BLOCK
+$APTHDR .5X .RESET LOCATION COUNTER

;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.

$APTHD: .WORD 0 ; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$SHIBTS: .WORD 0 ; ADDRESS OF APT MAILBOX (BITS 0-15)
$ADDR: .WORD 0 ; ADDRESS OF MAILBOX (BITS 0-15)
$STTM: .WORD 2 ; RUN TIM OF LONGEST TEST
$PSTM: .WORD 2 ; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNIT: .WORD 0 ; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT

.GO1
**Test 1: Check Branches on Z Bit**

```
**ftar parameter block**

500 000554 005212 000001
501 000556 027122 000001
502 000562 001024 000001
503 000564 000257 000001
504 000566 001401 000001
505 000570 000044 000001

**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**
**BR1: MOV #1,-(R2) ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**
**INC -(R2) ; BRANCH INSTRUCTION AND**
**HALT ; REPLACE THE MOVE INSTRUCTION**
**SHOULD HAVE BRANCHED: Z=0**

**BR2: BNE BR3 ; FOLLOWING W/ 774**
**MOV #2,-(R2) ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**
**INC -(R2) ; CONDITIONAL BRANCH INST. AND**
**SET MSGTY P TO FATAL ERROR**
**HALT ; REPLACE THE MOVE INSTRUCTION**
**SHOULD HAVE BRANCHED: Z=0**

**BR3: BNE BR4 ; WHICH FOLLOWS W/ 770**
**MOV #3,-(R2) ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**
**INC -(R2) ; BRANCH INSTRUCTION AND**
**SET MSGTY P TO FATAL ERROR**
**HALT ; REPLACE THE MOVE INSTRUCTION**
**SHOULD NOT HAVE BRANCHED HERE ON Z=1**

**BR4: BNE BR5 ; FOLLOWING W/ 760**
**MOV #4,-(R2) ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**
**INC -(R2) ; CONDITIONAL BRANCH INST. AND**
**SET MSGTY P TO FATAL ERROR**
**HALT ; REPLACE THE MOVE INSTRUCTION**
**SHOULD HAVE BRANCHED ON Z=1**
**OR SEQUENCE ERROR**
```
THE DATA PATH TESTS ARE USED TO VERIFY THAT VARIOUS
DATA PATTERNS CAN BE SUCCESSFULLY MOVED THROUGH THE DATA PATHS
AND COMPARE MODE 2, 3 INSTRUCTIONS ARE USED TO PASS AND
TEST VARIOUS DATA PATTERNS IN THE DATA PATHS.
THE TEST EXERISES INTERNAL DATA PATHS, THE UNIBUS
DATA TRANSCEIVERS, AND ALU CONTROL FOR ALU AND UNIBUS INPUTS.
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

TEST 3 TEST OF PATTERN 125252 IN DATA PATH

TEST 4 TEST OF PATTERN 052525 IN DATA PATH
TEST 5: TEST PATTERNS 052525 IN DATA PATH

**TST4:**
```
600 000736 005212
601 000740 022712 000004
602 000744 001007
603 000746 012737 052525 000000
604 000754 022737 052525 000000
605 000762 301404
```
- **INC (R2)**: Update test number
- **CMP #4,(R2)**: Sequence error?
- **BNE TST5-ID**: BR to error halt on seq error
- **MOV #052525,0#0**: Move alternating zeroes and ones
- **THRU DATA PATH**: Thru data path
- **SUCCESSFUL?**: Successful?
- **TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**: Conditional branch inst. and replace the move instruction which follows w/771

**TST5:**
```
600 000774 005212
601 000776 022712 000005
602 001002 001007
603 001004 012737 177777 000000
604 001012 022737 177777 000000
605 001020 001404
```
- **INC (R2)**: Update test number
- **CMP #5,(R2)**: Sequence error?
- **BNE TST6-ID**: BR to error halt on seq error
- **MOV #177777,0#0**: Move ones thru data path
- **SUCCESSFUL**: Successful
- **TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**: Conditional branch inst. and replace the move instruction which follows w/771

**TST6:**
```
600 010022 012742 000010
601 001026 005242
602 001030 000000
```
- **MOV #10,(R2)**: Move to mailbox
- **INC (R2)**: Set msgtyp to fatal error
- **HALT**: Data incorrect
- **SUCCESSFUL**: Successful
- **TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**: Conditional branch inst. and replace the move instruction which follows w/771

**TST7:**
```
600 000774 005212
601 000776 022712 000005
602 001002 001007
603 001004 012737 177777 000000
604 001012 022737 177777 000000
605 001020 001404
```
- **INC (R2)**: Update test number
- **CMP #5,(R2)**: Sequence error?
- **BNE TST6-ID**: BR to error halt on seq error
- **MOV #177777,0#0**: Move ones thru data path
- **SUCCESSFUL**: Successful
- **TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**: Conditional branch inst. and replace the move instruction which follows w/771
THE SCRATCH PAD TESTS ARE USED TO VERIFY THAT VARIOUS DATA PATTERNS CAN BE SUCCESSFULLY HELD IN THE SCRATCH PAD CIRCUITRY. MOVE AND COMPARE INSTRUCTIONS ARE USED TO TEST THAT RO CAN HOLD VARIOUS DATA PATTERNS. EACH DATA PATTERN IS MOVED AND TESTED IN A SMALL LOOP CONVENIENT FOR SCOPIING. 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.

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 RO ARE SUCCESSFUL BUT THE TESTS WITH THE OTHER REGISTERS FAIL, SUSPECT THE REGISTER SELECT LINES AND THEN THE SCRATCH PAD ITSELF.

TEST 6 TEST IF RO CAN HOLD ALL ZEROES

TEST 7 TEST IF RO CAN HOLD ONES AND ZEROS
MO1

TEST IF RO CAN HOLD ONES AND ZEROES

WHICH follows W/ 773

MOVE TO MAILBOX * ******* 12 *******
SET MSGTYP TO FP;M.L ERROR
OR SEQUENCE ERROR

;TEST 10 TEST IF RO CAN HOLD ZEROES AND ONES

;******************************************************************************

;ST10:  INC  (R2)  ;UPDATE TEST NUMBER
       CMP  #10-(R2)  ;SEQUENCE ERROR?
         BNE  TST11-10  ;BR TO ERROR HALT ON SEQ ERROR
       MOV  #052525,RO  ;MOVE ALTERNATING ZEROES AND ONES TO RO
       CMP  RO,#052525  ;SUCCESSFUL?
         BEQ  TST11
         TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
                     CONDITIONAL BRANCH INST. AND
                     REPLACE THE MOVE INSTRUCTION
                     WHICH follows W/ 773
       MOV  #13,-(R2)  ;MOVE TO MAILBOX * ******* 13 *******
       INC  -(R2)  ;SET MSGTYP TO FATAL ERROR
       HALT  ;RO NOT 52525
                OR SEQUENCE ERROR

;******************************************************************************

;TEST 11 TEST IF RO CAN HOLD ALL ONES

;******************************************************************************

;ST11:  INC  (R2)  ;UPDATE TEST NUMBER
       CMP  #11-(R2)  ;SEQUENCE ERROR?
         BNE  TST12-10  ;BR TO ERROR HALT ON SEQ ERROR
       MOV  #177777,RO  ;MOVE ALL ONES TO RO
       CMP  RO,#177777  ;SUCCESSFUL?
         BEQ  TST12
         TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
                     CONDITIONAL BRANCH INST. AND
                     REPLACE THE MOVE INSTRUCTION
                     WHICH follows W/ 773
       MOV  #14,-(R2)  ;MOVE TO MAILBOX * ******* 14 *******
       INC  -(R2)  ;SET MSGTYP TO FATAL ERROR
       HALT  ;RO NOT 17777
                OR SEQUENCE ERROR

;******************************************************************************

;TEST 12 TEST IF RI CAN HOLD A ONE IN ALL BITS

;******************************************************************************

;ST12:  INC  (R2)  ;UPDATE TEST NUMBER
       CMP  #12-(R2)  ;SEQUENCE ERROR?
         BNE  TST13-10  ;BR TO ERROR HALT ON SEQ ERROR
       MOV  #1,R1  ;SET BIT 0
       CLC  ;CLEAR C-BIT
       REG1:  ROL  R1  ;ROTATE 1 POSITION
         BCC  REG1
         ALL DONE
         BEQ  TST13
         TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
                     CONDITIONAL BRANCH INST. AND
                     REPLACE THE MOVE INSTRUCTION

;******************************************************************************
NO1

WHICH FOLLOWS W/ 772

TEST 13

**TEST 13**

TEST IF R2 CAN HOLD A ONE IN ALL BITS

**TEST 14**

TEST IF R3 CAN HOLD A ONE IN ALL BITS

**TEST 15**

TEST IF R4 CAN HOLD A ONE IN ALL BITS

; ST13:

INC (R2)
CMP $13(R2)
BNE REG2A-14
MOV $1,R2
CLC
REG2: ROL R2
BCC REG2
BEG REG2A

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

; ST14:

INC (R2)
CMP $14(R2)
BNE TST15-10
MOV $1,R3
CLC
REG3: ROL R3
BCC REG3
BEG TST15

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

; ST15:

INC (R2)
CMP $15(R2)
BNE TST16-10
MOV $1,R4
CLC
REG4: ROL R4
BCC REG4
BEG TST16

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
BRANCH INSTRUCTION AND
REPLACE THE MOVE INSTRUCTION
FOLLOWING W/ 772

; ST16:

INC (R2)
CMP $16(R2)
BNE TST17-10
MOV $1,R5
CLC
REG5: ROL R5
BCC REG5
BEG TST17

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
BRANCH INSTRUCTION AND
REPLACE THE MOVE INSTRUCTION
FOLLOWING W/ 772

...
001356 001-CH

001360 012742 000020
001364 005242 000001
001366 000000

001370 005212
001372 022712 000016
001376 001006
001400 012706 000001
001404 000041
001406 006105
001410 003376
001412 001404

001414 012742 000021
001420 005242 000000
001422 000000

001424 005212
001428 022712 000017
00142c 001006
001430 012706 000001
001434 000041
001436 006106
00143a 003376
00143c 001404

001450 012742 000022
001454 005242
001456 000000

BEQ TST16

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

MOV #20,-(R2)
INC -(R2)
HALT

MOVE TO MAILBOX # ****** 20 ******
SET MSGTYP TO FATAL ERROR
FAILURE WITH R6
OR SEQUENCE ERROR

*******************************************************************************

; TEST 16 TEST IF RS CAN HOLD A ONE IN ALL BITS

*******************************************************************************

TST16: INC (R2)

CMP #16 (R2)
BNE TST17-10
BR TO ERROR HALT ON SEQ ERROR

MOV #1,R5
CLC
CLEAR C-BIT

REG5: ROL R5
ROTATE 1 POSITION

BCC REG5
ALL DONE

BEQ TST17

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

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

MOVE TO MAILBOX # ****** 21 ******
SET MSGTYP TO FATAL ERROR
FAILURE WITH R6
OR SEQUENCE ERROR

*******************************************************************************

; TEST 17 TEST IF R6 CAN HOLD A ONE IN ALL BITS

*******************************************************************************

TST17: INC (R2)

CMP #17 (R2)
BNE TST20-10
BR TO ERROR HALT ON SEQ ERROR

MOV #1,R6

CLC
CLEAR C-BIT

REG6: ROL R6
ROTATE 1 POSITION

BCC REG6
ALL DONE

BEQ TST20

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

MOV #22,-(R2)
INC -(R2)
HALT

MOVE TO MAILBOX # ****** 22 ******
SET MSGTYP TO FATAL ERROR
FAILURE WITH R6
OR SEQUENCE ERROR
**SBTL: 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 AMUX 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 20** TEST IF PSW WILL HOLD ZEROS

```
**ST20:**
INC (R2) ; UPDATE TEST NUMBER
CMP #20,(R2) ; SEQUENCE ERROR?
BNE ST21-10 ; BR TO ERROR HALT ON SEQ ERROR
MOV #STBOT,R6
MOV #0,(R5) ; SET PSW TO ZERO
TST #R5 ; SUCCESSFUL
BEG ST21 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
            ; CONDITIONAL BRANCH INST. AND
            ; REPLACE THE MOVE INSTRUCTION
            ; WHICH FOLLOWS W/ 770
MOV #23,-(R2) ; MOVE TO MAILBOX # ******** 23 ******
INC -(R2) ; SET MSGTYP TO FATAL ERROR
HALT ; PSW NOT 0
          ; OR SEQUENCE ERROR
```

**TEST 21** TEST IF PSW WILL HOLD ONES AND ZEROS

```
**ST21:**
INC (R2) ; UPDATE TEST NUMBER
CMP #21,(R2) ; SEQUENCE ERROR?
BNE ST22-10 ; BR TO ERROR HALT ON SEQ ERROR
MOV #25E,(R5) ; MOVE ALL ONES AND ZEROS TO PSW
CMP #25E,#25E ; SUCCESSFUL?
BEG ST22 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
            ; CONDITIONAL BRANCH INST. AND
            ; REPLACE THE MOVE INSTRUCTION
            ; WHICH FOLLOWS W/ 771
MOV #24,-(R2) ; MOVE TO MAILBOX # ******** 24 ******
INC -(R2) ; SET MSGTYP TO FATAL ERROR
HALT ; PSW NOT 252
          ; OR SEQUENCE ERROR
```

**TEST 22** TEST IF PSW (EXCEPT T-BIT) WILL HOLD ZEROS AND ONES

```
```
TST22: INC (R2)  
UPDATE TEST NUMBER
CMP #25,(R2)  
SEQUENCE ERROR?
BNE TST23-10  
BR TO ERROR HALT ON SEQ ERROR
MOV #105,#PS  
MOVE ALT. ONES AND ZEROS TO PSW
BEQ TST23  
SUCCESSFUL?

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

MOVE TO MAILBOX # ******** 25 *******
SET MSGTYP TO FATAL ERROR
PSW NOT 105
OR SEQUENCE ERROR

TST23: INC (R2)  
UPDATE TEST NUMBER
CMP #23,(R2)  
SEQUENCE ERROR?
BNE TST24-10  
BR TO ERROR HALT ON SEQ ERROR
MOV #357,#PS  
MOVE ONES TO PSW
CMP #357,#357  
SUCCESSFUL
BEQ TST24  

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

MOVE TO MAILBOX # ******** 26 *******
SET MSGTYP TO FATAL ERROR
PSW NOT 357
OR SEQUENCE ERROR
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 BIT 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 24: SHIFT BIT 0 TO BIT 1

;*****************************************************************************
; TST24: INC (R2); UPDATE TEST NUMBER
;        CMP #24,(R2); SEQUENCE ERROR?
;        BNE TST25-10; BR TO ERROR HALT ON SEQ ERROR
;        CLC; CLEAR CARRY BIT
;        MOV #1, A; LOAD A 1
;        ROL A; SHIFT LEFT
;        CMP #2, A; SUCCESSFUL
;        BEQ TST25; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 766
; MOVE TO MAILBOX # 27 27
; SET MSGTYP TO FATAL ERROR
; BIT 1 NOT SET
; OR SEQUENCE ERROR

;*****************************************************************************
; TEST 25: SHIFT CARRY INTO BIT 0

;*****************************************************************************
; TST25: INC (R2); UPDATE TEST NUMBER
;        CMP #25,(R2); SEQUENCE ERROR?
;        BNE TST26-10; BR TO ERROR HALT ON SEQ ERROR
;        MOV #0, A; CLEAR LOCATION
;        SEC; SET CARRY
;        ROL A; ROTATE CARRY BIT TO BIT 0
;        BCC TST26; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 771
; MOVE TO MAILBOX # 30 30
; SET MSGTYP TO FATAL ERROR
; CARRY CLEAR
; OR SEQUENCE ERROR
; BIT 0 SET
CLEAR CARRY ONE-PASS CLEAR
BRANCH AT ONE-PASS ADJUST TO SCOPED CONDITIONAL BRANCH.

NOTE: CLEARING THE CARRY TO LOGICAL ERROR OR SEQUENCE ERROR.

MOVE TO MAIN BOX OR ERROR LOGICAL ERROR.

SET MSG TYPE TO FATAL ERROR.
**CONDITION CODE TEST**

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 31**

TEST BRANCHES AROUND Z-BIT

```
ST31: INC (R2) ;UPDATE TEST NUMBER
CMP #31,(R2) ;SEQUENCE ERROR?
BNE TST32-10 ;BR TO ERROR HALT ON SEQ ERROR

BRZ1: MOV #35,-(R2) ;MOVE TO MAILBOX & 35
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;IMPROPER BR W/ Z=1

;CHECK WITH Z-BIT OFF
BRZ2: SCC
CLZ
BNE TST32

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

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

BRZ3: MOV #36,-(R2) ;MOVE TO MAILBOX & 36
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;IMPROPER BR W/ Z=0

;OR SEQUENCE ERROR
```
**THIS TEST CHECKS THE CONDITIONAL BRANCHES INVOLVING THE N-BIT.**

THE N-BIT IS SET WITH ALL OTHER CC BITS ZERO AND BOTH CONDITIONS
BANI 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 32 TEST BRANCHES AROUND N-BIT**

```assembly
ST32: INC (R2) ; UPDATE TEST NUMBER
       CMP #32,(R2) ; SEQUENCE ERROR?
       BNE TST33-10 ; BR TO ERROR HALT ON SEQ ERROR
       ; FIRST WITH N-BIT ON
       CCC
       SED
       BRA BRN1
       ; CHECK OPPOSITE CONDITION
       ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       ; CONDITIONAL BRANCH INST. AND
       ; REPLACE THE MOVE INSTRUCTION
       ; WHICH FOLLOWS W/ 774

BRN1: MOV #37,-(R2) ; MOVE TO MAILBOX 8
       INC -(R2) ; SET MSGYP TO FATAL ERROR
       HALT ; IMPROPER BR W/ N=1
       ; CHECK WITH N-BIT OFF
       ACC
       CLN
       BRA BRN3
       ; CHECK OPPOSITE CONDITION
       ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       ; CONDITIONAL BRANCH INST. AND
       ; REPLACE THE MOVE INSTRUCTION
       ; WHICH FOLLOWS W/ 764

BRN2: MOV #127,-(R2) ; MOVE TO MAILBOX 8
       INC -(R2) ; SET MSGYP TO FATAL ERROR
       HALT ; IMPROPER BR W/ N=0
       ; OR SEQUENCE ERROR

BRN3: MOV #40,-(R2) ; MOVE TO MAILBOX 8
       INC -(R2) ; SET MSGYP TO FATAL ERROR
       HALT ; IMPROPER BR W/ N=0
```

---

**END OF TEST 32**
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
TESTED IN THE TEST ARE VERIFIED HERE.

*******************************************************************************

** TEST 33 **
TEST BRANCHES AROUND V-BIT

** TEST 33 **
TEST BRANCHES AROUND V-BIT

** TEST 33 **
TEST BRANCHES AROUND V-BIT

TST33:  INC  (R2) ; UPDATE TEST NUMBER
        CMP  #33,(R2) ; SEQUENCE ERROR?
        BNE  TST34-I0 ; BR TO ERROR HALT ON SEQ ERROR
        BVS  BRV11 ; FIRST WITH V-BIT ON
        BVC  BRV2 ; CC=0010: JUST V-BIT
        MOV  #41, -(R2) ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
        INC  -(R2) ; CONDITIONAL BRANCH INST. AND
        HALT ; REPLACE THE MOVE INSTRUCTION
        BVS  BRV12 ; WHICH FOLLOWS W/ 774
        BVC  BRV13
        BVS  BRV3 ; CHECK OPPOSITE CONDITION
        BRV1:  MOV  #41,-(R2) ; MOVE TO MAILBOX
        INC  -(R2) ; 41 41 41 41 41 41
        HALT ; 41 41 41 41 41 41
        BVC  TST34 ; IMPROPER BR W/ V=1
        BRV2:  MOV  #42,-(R2) ; CHECK WITH V-BIT OFF
        INC  -(R2) ; CC=1101: ALL BUT V-BIT
        HALT ; CC=1101: ALL BUT V-BIT
        BVC  TST34
        BRV3:  MOV  #42,-(R2) ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
        INC  -(R2) ; CONDITIONAL BRANCH INST. AND
        HALT ; REPLACE THE MOVE INSTRUCTION
        BVC  TST34 ; WHICH FOLLOWS W/ 764
        OR SEQUENCE ERROR
**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 34: Test branches around C-bit**

**Test branches around C-bit**

```
TST34: INC (R2) ; Update test number
        CMP #34, (R2) ; Sequence error?
        BNE TST35-10 ; BR to error halt on seq error
        ; FIRST WITH C-BIT ON
        SEC
        BNE TST35-10 ; BR to error halt on cond error
        BCS BRC1
        ; CHECK OPPOSITE CONDITION
        BCC BRC2
        ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND
        ; REPLACE THE MOVE INSTRUCTION
        ; WHICH FOLLOWS W/ 774

BRC1: MOV #43, -(R2) ; Move to mailbox # 43
        INC -(R2) ; Set msgtyp to fatal error
        BNE TST35-10 ; Improper br w/ C=1
        ; CHECK WITH V-BIT OFF

BRC2: SCC
        CLV
        BVS BRC3
        BMI TST35
        ; CHECK OPPOSITE CONDITION
        ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND
        ; REPLACE THE MOVE INSTRUCTION
        ; WHICH FOLLOWS W/ 784

BRC3: MOV #44, -(R2) ; Move to mailbox # 44
        INC -(R2) ; Set msgtyp to fatal error
        BNE TST35-10 ; Improper br w/ C=0
        ; OR SEQUENCE ERROR
        ```
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.

********************

THE CLR INSTRUCTION IS USED TO INTRODUCE EACH ADDRESSING
MODE WITH THE SINGLE OPERAND INSTRUCTION. FOLLOWING THE SEQUENCE CHECK,
The CLR INSTRUCTION IS EXECUTED AND A BRANCH TEST IS EXECUTED WHICH
CHECKS THAT THE 2-BIT WAS PROPERLY SET. THIS SMALL TEST IS SELF-SUFFICIENT
AND CAN BE SCOPED TO TROUBLE SHOOT ALL OF THE IR DECODE LOGIC AND
MICROCODE FOR SOP INSTRUCTIONS WITH MODE 0. FOLLOWING THIS TEST
SEVERAL OTHER SOP INSTRUCTIONS ARE INTRODUCED WITH MODE 0. THESE
INSTRUCTIONS MANIPULATE DATA AND SERVE TO CHECK THE DATA RESULTS
OF THE SOP INSTRUCTIONS IN THIS TEST. THE DATA IN THIS TEST IS
OPERATED ON BY EACH INSTRUCTION WITHOUT REINITIALIZING.

********************

TEST 35  TEST MODE 0 USING SOP INST.

TST35: INC (R2) ; UPDATE TEST NUMBER
COP #5 (R2) ; SEQUENCE ERROR?
BNE TS136-10 ; BR TO ERROR HALT ON SEQ ERROR
CLR RO ; TRY THE CLEAR INST.
BEQ SOPDA

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; WHICH FOLLOWS W/ 775

MOV #55,-(R2) ; MOVE TO MAILBOX # ******* 45 *******
INC - (R2) ; SET MSGTYD TO FATAL ERROR
HALT ; CLR DID NOT SET 2-BIT
SOPDA: INC RO ; TRY THE INCREMENT INST.
NEG RO ; TRY THE NEGATE INST.
BMI SOPDA

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

REPLACE THE MOVE INSTRUCTION WHICH Follows w/ 775

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

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH Follows w/ 760

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

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 TROUBLEHOUSING EFFORTS SHOULD BE AIMED AT THE ACTUAL IR DECODE AND ALU FUNCTIONING.

TEST 36 TEST REMAINDER OF SOP INSTS IN MODE 0

UPDATE TEST NUMBER

BR TO ERROR HALT ON SEQ ERROR

TRY ADD CARRY INST

TRY SUBTRACT-CARRY INST

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

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH Follows w/ 760
MOV TO MAILBOX ****** S1 ******
SET MSGTYP TO FATAL ERROR
CUMMULATIVE RESULT OF ADC,SBC,NEG AND DEC INSTRS. FAILED
OR SEQUENCE ERROR
**B03**

**TEST 37**  TEST MODE 0 EVEN BYTE USING SOP INST

**-------------------------------------------------------------------------------------------------------------------------**

**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 37: TEST MODE 0 EVEN BYTE USING SOP INST**

**-------------------------------------------------------------------------------------------------------------------------**

**TST37:**

- **INC (R2)**: UPDATE TEST NUMBER
- **CMP 37,(R2)**: SEQUENCE ERROR?
- **BNE TST40-10**: BR TO ERROR HALT ON SEQ ERROR
- **CLRB RO**: TRY CLEARING EVEN BYTE OF REGISTER
- **BEO SOPBOA**: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
  - **INC (R2)**: CONDITIONAL BRANCH INST. AND
  - **MOV 32,-(R2)**: REPLACE THE MOVE INSTRUCTION
  - **HALT**: WHICH FOLLOWS W/ 776
- **SOPBOA: COMB RO**: TRY SETTING EVEN BYTE OF REGISTER
- **BPL SOPBOB**: TRY INCREMENTING EVEN BYTE OF REGISTER
- **INCBO RO**: TRY INCREMENTING EVEN BYTE OF REGISTER
- **BEQ TST40**: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
  - **INC (R2)**: CONDITIONAL BRANCH INST. AND
  - **MOV 33,-(R2)**: REPLACE THE MOVE INSTRUCTION
  - **HALT**: WHICH FOLLOWS W/ 756

**-------------------------------------------------------------------------------------------------------------------------**

**SOPBOB:**

- **MOV 33,-(R2)**: MOVE TO MAILBOX 53 53
- **INC -(R2)**: SET MSGTYP TO FATAL ERROR
- **HALT**: TEST CUMULATIVE RESULT OF ABOVE BYTE INST. OR SEQUENCE ERROR
THIS TEST USES THE CLR INSTRUCTION TO INTRODUCE ANC 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 WD  "TEST MODE 1 USING SOP INST."
******************************************************************************

ST40:
INC (R2) ; UPDATE TEST NUMBER
CMP #240,(R2) ; SEQUENCE ERROR?
BNE TST41-1D ; BR TO ERROR HALT ON SEQ ERROR
CLR (RO) ; INITIALIZE RO
CLR (R0) ; TRY CLEAR INST W/MODE 1
BEQ SOP1A

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

TST41-1D
MOV #54,-(R2) ; MOVE TO MAILBOX # "" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " 

SOP1A:
DEC (R0) ; INITIALIZE CARRY
BPE SOP18
BPE SOP1B

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

SOP18:
MOV #55,-(R2) ; MOVE TO MAILBOX # " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " 

SOP1B:
INC (R2) ; SEGN MSGTYP TO FATAL ERROR
HALT ; TEST CUMMULATIVE RESULT OF ABOVE INST
OR SEQUENCE ERROR
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 41  TEST MODE 1 EVEN BYTE USING SOP INST

<table>
<thead>
<tr>
<th>TST-41:</th>
<th>INC (R2)</th>
<th>UPDATE TEST NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP</td>
<td>$41.(R2)</td>
<td>SEQUENCE ERROR</td>
</tr>
<tr>
<td>BNE</td>
<td>TST-42-10</td>
<td>BR TO ERROR HALT ON SEQ ERROR</td>
</tr>
<tr>
<td>CLR</td>
<td>RD</td>
<td>INITIALIZE RD</td>
</tr>
<tr>
<td>CLR</td>
<td>(R0)</td>
<td>INITIALIZE LOC. D</td>
</tr>
<tr>
<td>CLR</td>
<td>(R0)</td>
<td></td>
</tr>
<tr>
<td>CLRB</td>
<td>(R0)</td>
<td></td>
</tr>
<tr>
<td>BEQ</td>
<td>SOPBIA</td>
<td>TRY TO CLEAR BYTE D</td>
</tr>
</tbody>
</table>

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

| MOV | $56.- (R2) | MOVE TO MAILBOX # 56 ===== |
| INC | -(R2) | |
| HALT | | CLRB DID NOT SET Z-BIT |

SOPBIA: | INC | (R0) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BPL</td>
<td>SOPBIB</td>
<td></td>
</tr>
<tr>
<td>NEGB</td>
<td>(R0)</td>
<td></td>
</tr>
<tr>
<td>BPL</td>
<td>SOPBIB</td>
<td></td>
</tr>
<tr>
<td>INCB</td>
<td>(R0)</td>
<td></td>
</tr>
<tr>
<td>BEQ</td>
<td>TST-42</td>
<td></td>
</tr>
</tbody>
</table>

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 671

| MOV | $57.- (R2) | MOVE TO MAILBOX # 57 ===== |
| INC | -(R2) | |
| HALT | | CHECK CUMULATIVE RESULT OF ABOVE INST |
| | | OR SEQUENCE ERROR |

| MOV | $58.- (R2) | |
| MOV | $59.- (R2) | |
| MOV | $60.- (R2) | |
**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 42**  TEST MODE 1 ODD BYTE USING SOP INST

------------------------------------------------------------------------

**ST42:**

INC (R2)  ; UPDATE TEST NUMBER
CMP #42 (R2)  ; SEQUENCE ERROR?
BNE TST43-10  ; BR TO ERROR HALT ON SEQ ERROR
CLR RD  ; INITIALIZE RD
CLR (RD)  ; INITIALIZE LOC. 0
COM (RD)  ;
INC RD  ; RD=ODD BYTE
CLR RD  ; TRY TO CLEAR BYTE 1
BEQ SOPB1C  ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

**SOPB1C:**

MOV #60, -(R2)  ; MOVE TO MAILBOX 8 ******* 60 *******
INC -(R2)  ; SET MSGTYP TO FATAL ERROR
HALT  ; CLRB DID NOT SET 2-BIT
DEC RD  ; RD=WORD ADDR.
INC (RD)  ; INCREMENT TO TEST WORD
INC RD  ; RD=ODD BYTE
NEGB (RD)  ; TRY TO NEGATE BYTE 1
BPL SOPB1D  ; ; TRY TO INCREMENT BYTE 1

**SOPB1D:**

MOV #61, -(R2)  ; MOVE TO MAILBOX 8 ******* 61 *******
INC -(R2)  ; SET MSGTYP TO FATAL ERROR
HALT  ; TEST CUMMULATIVE RESULT OF ABOVE INST.

; OR SEQUENCE ERROR
THIS TEST VERIFIES MODE 2 SINGLE-OPERAND INSTRUCTIONS. PREVIOUSLY
TESTED INSTRUCTIONS ARE USED TO SET A POINTER IN RD TO LOC 400.
LOC 400 IS INITIALIZED TO -1 BEFORE A CLR MODE 2 IS EXECUTED.
THEN RD 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 43 -  TEST MODE 2 USING SOP INST.

**UPDATING TEST NUMBER
SEQUENCE ERROR?
BR TO ERROR HALT ON SEQ ERROR
SET RD=400
CLEAR 400
INITIALIZE: 400=-1
TRY CLEARING WITH MODE 2
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 771

MOVE TO MAILBOX # ****** 62 ******
SET MSGTYW TO FATAL ERROR
CLR INST DID NOT SET Z-BIT
RESET RD
TRY COMPLEMENTING WITH MODE 2
RESET RD
TRY INCREMENTING WITH MODE 2
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 755

MOVE TO MAILBOX # ****** 63 ******
SET MSGTYW TO FATAL ERROR
CHECK CUMULATIVE RESULT OF ABOVE INST
OR SEQUENCE ERROR
THIS TEST VERIFIES MODE 2 SINGLE OPERAND INSTRUCTIONS WHICH
ADDRESS EVEN BYTES. RD IS SET TO 400 AND USED TO INITIALIZE LOCATION
400 TO -1. CLR#3 INSTRUCTION IS THEN EXECUTED ON BYTE 400 WITH
MODE 2.
RD 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 44 TEST MODE 2 EVEN BYTE USING SOP INST.

TST44: INC (R2) ; UPDATE TEST NUMBER
        CMP #44,(R2) ; SEQUENCE ERROR?
        BNE TST45-10 ; BR TO ERROR HALT ON SEQ ERROR
        CLR RD
        COMB RD
        INC RD
        CLRB (RD)
        INCB (RD) ; CLEAR 400
        : INITIALIZE: 400=-1
        BEQ SOPB2A
        ; TRY TO CLEAR 400 W/MODE 2
        ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
        ; CONDITIONAL BRANCH INST. AND
        ; REPLACE THE MOVE INSTRUCTION
        ; WHICH FOLLOWS W/ 771
        ; MOVE TO MAILBOX # ******** 64
        MOV #64,-(R2) ; SET MSGTYP TO FATAL ERROR
        INC -(R2) ; CLR DID NOT SET Z-BIT
        HALT
        SOPB2A: DEC RD
        : RESULT RD=400
        INC (RD)
        NEGB (RD)+
        BPL SOPB2B
        DEC RD
        INCB (RD)+
        BEQ TST45
        ; TRY INCREMENT OF EVEN BYTE
        ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
        ; CONDITIONAL BRANCH INST. AND
        ; REPLACE THE MOVE INSTRUCTION
        ; WHICH FOLLOWS W/ 756
        SOPB2B: MOV #65,-(R2) ; MOVE TO MAILBOX # ******** 65
        INC -(R2) ; SET MSGTYP TO FATAL ERROR
        HALT
        ; TEST CUMMULATIVE RESULT OF ABOVE INST.
        ; OR SEQUENCE ERROR
THIS TEST FOLLOWS THE SAME PROCEDURE DESCRIBED IN THE PREVIOUS TESTS. HERE, THE BYTE INSTRUCTION IS USED TO ADDRESS AN ODD BYTE.

**TEST 45**  TEST MODE 2 ODD BYTE USING SOP INST.

```
ST45:  INC  (R2) ; UPDATE TEST NUMBER
        JMP  #45,(R2) ; SEQUENCE ERROR?
        BNE  TST46-10 ; BR TO ERROR HALT ON SEG ERROR
        CLR  RO ; SET RO=400
        COMB  RO
        INC  RO
        CLR  (RO)
        CM  (RO) ; INITIALIZE: 400=-1
        INC  RO ; RO=ODD BYTE
        CLRB  (RO)+ ; 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 $770
                      ; MOVE TO MAILBOX # 66 ****** 66 ******
                      ; SET MSGTYP TO FATAL ERROR
                      ; CLRB DID NOT SET Z-BIT
                      ; RD=WORD ADDR.
                      ; INCREMENT WORD
                      ; POINT TO ODD BYTE
                      ; TRY TO NEGATE ODD BYTE
                      ; RESET RD TO ODD BYTE
                      ; TRY TO INCREMENT ODD BYTE
                      ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
                      ; CONDITIONAL BRANCH INST. AND
                      ; REPLACE THE MOVE INSTRUCTION
                      ; WHICH FOLLOWS $753
                      ; MOVE TO MAILBOX # 67 ****** 67 ******
                      ; SET MSGTYP TO FATAL ERROR
                      ; TEST CUMMULATIVE RESULT OF ABOVE INST.
                      ; OR SEQUENCE ERROR
```

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.
RO IS SET TO 400, THE START OF THE ADDRESS TABLE, AND A CLR
INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR LOC. 0. THEN RO
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 46
TEST MODE 3 USING SOP INST.

TST46:
INC (R2) ;UPDATE TEST NUMBER
CMP #46, (R2) ;SEQUENCE ERROR?
BNE TST47-1D ;BR TO ERROR HALT ON SEQ ERROR
CLR RO ;SET RO=400
COMB RO
INC RO
CLR (RO)
CLR (RO)+
BEQ 50P3A
CLEAR LOC 400
TRY TO CLEAR LOC 0 USING MODE 3
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 772

MOV #70,-(R2) ;MOVE TO MAILBOX & 
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CLR DID NOT SET Z-BIT
RESET RO=400

SOP3A:
DEC RO
DEC RO
COM (RO)+
BPL 50P3B
INC (RO)+
BEQ TST47
TRY TO COMPLEMENT LOC 0 OF MODE 3
TRY TO INCREMENT LOC 0 W/MODE 3
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 760

SOP3B:
MOV #71,-(R2) ;MOVE TO MAILBOX & 
INC -(R2) ;SET MSGTYP TO FATAL ERROR
HALT ;CUMMULATIVE RESULT OF ABOVE INST FAILED
; OR SEQUENCE ERROR
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

R0 TO -1 A CLR B 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 47**

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

******************************************************************************

^ST47: INC (R2) ;UPDATE TEST NUMBER

CMP #73,-(R2) ;SEQUENCE ERROR?

BNE TST50-10 ;BR TO ERROR HALT ON SEQ ERROR

CLR R4

COMB R4

INC R4

CLR (R0)

CMR (R0)

CLRB (R4)+ ;TRY TO CLEAR EVEN BYTE

BEG SOPB3A

;INITIALIZE LOC. 0=-1

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

CONDITIONAL BRANCH INST. AND

REPLACE THE MOVE INSTRUCTION

WHICH FOLLOWS W/ 770

MOVE TO MAILBOX 8 ******** 72 *******

MOV #72,-(R2)

INC -(R2)

HALT

SOPB3A:

DEC R4

DEC R4

INC (R4)+

BPL SOPB3B

NEG R (R4)+

BPL SOPB3B

DEC R4

DEC R4

INCB (R4)+ ;TRY TO INCREAT INEVEN BYTE

BEG TST50

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

CONDITIONAL BRANCH INST. AND

REPLACE THE MOVE INSTRUCTION

WHICH FOLLOWS W/ 752

MOVE TO MAILBOX 8 ******** 73 *******

MOV #73,-(R2)

INC -(R2)

HALT

SOPB3B:

CUMMULATIVE RESULT OF ABOVE INST FAILED

OR SEQUENCE ERROR
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. RO serves as the table pointer. RO is initialized to 400. Loc. 0 is set to -1 using the first two table entries. A loop mode 3 is executed on byte 1 using table address at 400. RO is decremented to 400 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 50  Test mode 3 odd byte using SOP inst.

---------------------------

TST50:  INC  (R2)  Update test number
        CMP  #50, (R2)  Sequence error?
        BNE  TST51-10  BR to error halt on seq error
        CLR  RO
        COMB  RO
        INC  RO
        CLR  @RO+  Initialize
        CMP  @RO+  LOC 05-1 RO=404
        COM  @RO+  Try to clear odd byte
        CLR  SOPB3C
        BEQ  SOPB3C
        TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
        CONDITIONAL BRANCH INST. AND
        REPLACE THE MOVE INSTRUCTION
        WHICH Follows W/771

        MOV  $74,-(R2)  Move to mailbox 8
        INC  -(R2)  Set msgtyp to fatal error
        HALT  CLR did not set 2-bit
        DEC  RO  Reset RO
        SOPB3C:  DEC  RO  Point to even byte addr.
        DEC  RO
        DEC  RO
        DEC  RO
        INC  @RO+  Increment word
        MEGB  @RO+  Try to negate odd byte
        BPL  SOPB3D  Try to increment odd byte
        INCB  @RO+  TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
        BPL  SOPB3D
        CONDITIONAL BRANCH INST. AND
        REPLACE THE MOVE INSTRUCTION
        WHICH Follows W/794

        MOV  $75,-(R2)  Move to mailbox 8
        INC  -(R2)  Set msgtyp to fatal error
        HALT  Cumulative result of above insts failed
        OR sequence error
;********************
;TEST S1: TEST MODE 4 USING SOP INSTS
;********************
TST51: INC (R2); UPDATE TEST NUMBER
CMP #51 (R2); SEQUENCE ERROR?
BNE TST52-10; BR TO ERROR HALT ON SEQ ERROR
CLR (R0); SET RD=400
COMB RO
INCR (R0)
CLR -(R0)
BEQ SOP4A
;TRY TO CLEAR USING MODE 4
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 773
; MOVE TO MAILBOX # 76
;CLR DID NOT SET Z-BIT
SOP4A: INC RO
INCR RO
COM -(R0)
BPL SOP4B
INCR (R0)
INC RO
INCR -(R0)
BEQ TST52
;TRY TO COMPLEMENT USING MODE 4
;MOVE POINTER
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 757
; MOVE TO MAILBOX # 77
;CHECK CUMMULATIVE RESULT OF ABOVE INST.
; OR SEQUENCE ERROR
SOP4B: MOV #77 -(R2)
INC -(R2)
HALT

**THIS TEST VERIFIES MODE 5 SINGLE OPERAND INSTRUCTIONS.**

**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.

RD 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 RD IS INCREMENTED BY TWO AND TWO OTHER MODE 3
INSTRUCTIONS OPERATE ON LOC. 0 TO VERIFY THE DATA RESULTS OF
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 52**

**TEST MODE 5 USING SOP INSTS**

```plaintext
ST52: INC (R2) ; UPDATE TEST NUMBER
       CMP #52,(R2) ; SEQUENCE ERROR?
       BNE TST53-10 ; BR TO ERROR HALT ON SEQ ERROR
       CLR RD ; SET RD=376
       CLR (RD)+
       NEGB RD ; TRY TO CLEAR LOC 0 W/MODE 5
       BEQ SOPS
       ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       ; CONDITIONAL BRANCH INST. AND
       ; REPLACE THE MOVE INSTRUCTION
       ; WHICH FOLLOWS W/ 773

SOPS: INC RD ; RESET RD
       INC RD
       COM A-(RD) ; TRY TO COMPLEMENT LOC. 0 W/MODE 5
       BPL SOPS5
       INC A-(RD) ; TRY TO INCREMENT LOC. 0 W/MODE 5
       BEQ TST53 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       ; CONDITIONAL BRANCH INST. AND
       ; REPLACE THE MOVE INSTRUCTION
       ; WHICH FOLLOWS W/ 761

SOPS5: MOV #101,-(R2) ; MOVE TO MAILBOX # ******* 101 ******
       INC -(R2) ; SET MSGTYP TO FATAL ERROR
       HALT ; CLR DID NOT SET Z-BIT

TST53: INC (R2) ; UPDATE TEST NUMBER
       CMP #52,(R2) ; SEQUENCE ERROR?
       BNE TST53-10 ; BR TO ERROR HALT ON SEQ ERROR
       CLR RD ; SET RD=376
       CLR (RD)+
       NEGB RD ; TRY TO CLEAR LOC 0 W/MODE 5
       BEQ SOPS
       ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       ; CONDITIONAL BRANCH INST. AND
       ; REPLACE THE MOVE INSTRUCTION
       ; WHICH FOLLOWS W/ 773
```

**TEST 52**

**TEST MODE 5 USING SOP INSTS**

```plaintext
ST52: INC (R2) ; UPDATE TEST NUMBER
       CMP #52,(R2) ; SEQUENCE ERROR?
       BNE TST53-10 ; BR TO ERROR HALT ON SEQ ERROR
       CLR RD ; SET RD=376
       CLR (RD)+
       NEGB RD ; TRY TO CLEAR LOC 0 W/MODE 5
       BEQ SOPS
       ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       ; CONDITIONAL BRANCH INST. AND
       ; REPLACE THE MOVE INSTRUCTION
       ; WHICH FOLLOWS W/ 773

SOPS: INC RD ; RESET RD
       INC RD
       COM A-(RD) ; TRY TO COMPLEMENT LOC. 0 W/MODE 5
       BPL SOPS5
       INC A-(RD) ; TRY TO INCREMENT LOC. 0 W/MODE 5
       BEQ TST53 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       ; CONDITIONAL BRANCH INST. AND
       ; REPLACE THE MOVE INSTRUCTION
       ; WHICH FOLLOWS W/ 761

SOPS5: MOV #101,-(R2) ; MOVE TO MAILBOX # ******* 101 ******
       INC -(R2) ; SET MSGTYP TO FATAL ERROR
       HALT ; TEST CUMMULATIVE RESULT OF ABOVE INSTS
       ; OR SEQUENCE ERROR
```
THIS TEST VERIFIES MODE 6 SINGLE OPERAND INSTRUCTIONS. IT USES LOCATION 0 AS ITS TARGET DATA. RD IS SET TO 400 USING PREVIOUSLY TESTED INSTRUCTIONS AND A MODE 6 CLR INSTRUCTION IS EXECUTED ON LOC. 0 USING RD AND A -400 OFFSET. COM AND INC INSTRUCTIONS ARE THEN USED TO VERIFY THE DATA.

TESTS:

**TEST S3: TEST MODE 6 USING SOP INSTS**

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>03664</td>
<td>022712</td>
</tr>
<tr>
<td>03664</td>
<td>022712</td>
</tr>
<tr>
<td>03664</td>
<td>001020</td>
</tr>
<tr>
<td>03672</td>
<td>005000</td>
</tr>
<tr>
<td>03604</td>
<td>105100</td>
</tr>
<tr>
<td>03604</td>
<td>005200</td>
</tr>
<tr>
<td>03700</td>
<td>000603</td>
</tr>
<tr>
<td>03700</td>
<td>001404</td>
</tr>
<tr>
<td>03706</td>
<td>012742</td>
</tr>
<tr>
<td>03712</td>
<td>005242</td>
</tr>
<tr>
<td>03714</td>
<td>000000</td>
</tr>
<tr>
<td>03716</td>
<td>005160</td>
</tr>
<tr>
<td>03722</td>
<td>100030</td>
</tr>
<tr>
<td>03724</td>
<td>005260</td>
</tr>
<tr>
<td>03730</td>
<td>001404</td>
</tr>
</tbody>
</table>

**MOP6A:**

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>177400</td>
</tr>
</tbody>
</table>

**MOP6B:**

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>012742</td>
<td>000103</td>
</tr>
<tr>
<td>005242</td>
<td>000000</td>
</tr>
</tbody>
</table>

;TRY TO CLEAR LOCATION 0 W/MODE 6

;TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

;CONDITIONAL BRANCH INST. AND

;REPLACE THE MOVE INSTRUCTION

;WHICH FOLLOWS W/ 772

;TRY TO COMPLEMENT LOCATION 0 W/MODE 6

;TRY TO INCREMENT LOCATION 0 W/MODE 6

;TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

;CONDITIONAL BRANCH INST. AND

;REPLACE THE MOVE INSTRUCTION

;WHICH FOLLOWS W/ 760

; Move to Mailbox ** 103 **

; Set MSGTYP to Fatal Error

; Replace the Move Instruction

; Which follows W/ 760

; Test Cumulative Result of Above Insts

; OR Sequence Error

;
**THIS TEST VERIFIES MODE 7 SINGLE OPERAND INSTRUCTIONS. IT USES**

**THE POINTER TO LOC. 0 WHICH IS STORED AT LOC. 402.**

**RD 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 54  TEST MODE 7 USING SOP INST.**

**-------------------------------**

**ST54:  INC (R2) ; UPDATE TEST NUMBER**

**CMP $54 (R2) ; SEQUENCE ERROR?**

**BNE TS5S-1C ; BR TO ERROR HALT ON SEQ ERROR**

**CLR RD ; SET RD=400**

**COMB RO ; TRY TO CLEAR LOC. 0 W/MODE 7**

**INC RD ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**

**BEQ SOP7A ; CONDITIONAL BRANCH INST. AND**

**; REPLACE THE MOVE INSTRUCTION**

**; WHICH FOLLOWS W/ 772**

**MOV $104 -(R2) ; MOVE TO MAILBOX 0 104 000000**

**INC -(R2) ; SET MSGTYP TO FATAL ERROR**

**HALT ; CLR DID NOT SET Z-BIT**

**; TRY TO COMPLEMENT LOC. 0 W/MODE 7**

**; TRY TO INCREMENT LOC. 0 W/MODE 7**

**; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**

**BPL SOP7B ; CONDITIONAL BRANCH INST. AND**

**; REPLACE THE MOVE INSTRUCTION**

**; WHICH FOLLOWS W/ 760**

**INC $105 -(R2) ; MOVE TO MAILBOX 0 105 000000**

**HALT ; TEST CUMULATIVE RESULT OF ABOVE INSTS.**

**; OR SEQUENCE ERROR**
THIS TEST VERIFIES PROGRAM COUNTER ADDRESSING WITH SOP INSTRUCTIONS. CLR MODE 77 IS USED TO CLEAR THE LOCATION FOLLOWING THE INSTRUCTION (SOP), THEN SINGLE OPERAND INSTRUCTIONS WITH MODES 37, 67, AND 77, USING INDIRECT POINTER SOPXAD ARE USED TO VERIFY THE DATA RESULTS OF THESE INSTRUCTIONS.

TEST 55 TEST S0P INSTRUCTIONS MODES 2,3,6,7 WITH REGISTER 7

ST55: INC (R2) 
CMP #55,(R2) 
BNE SOPB 
CLR (R7)+ 
SCFX: =1 
BEQ S0PA 

S0P: MOV #106,-(R2) 
INC -(R2) 
SOPA: INC #SOPX 
NEG SOPX 
BPL SOPB 
INC #SOPXAD 
BEQ ST556 

S0PB: MOV #107,-(R2) 
INC -(R2) 
HALT 
SOPXAD: S0P
**THIS TEST VERIFIES SINGLE OPERAND NON-MODIFYING INSTRUCTIONS**

- USING MODE 0, RO 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 56**

**TEST MODE 0 SOP NON-MODIFYING**

---

**ST56:** INC (R2)  ; UPDATE TEST NUMBER
CMP #56, (R2) ; SEQUENCE ERROR?
BNE TST57-10 ; BR TO ERROR HALT ON SEQ ERROR
CLR RO ; INITIALIZE RO=0
SCL ; SET CC=10D1
CLZ
TST RO ; TRY TST W/ MODE O
BVS SMDA ; CHECK THAT CC=O1D0
BMI SMDA
BCE SMDA
BEQ TST57

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 77D

**SMDA:**

MOV #11D, (R2) ; MOVE TO MAILBOX # ****** 110 ******
INC -(R2) ; SET MSGTYP TO FATAL ERROR
HALT ; CONDITION CODES NOT SET PROPERLY
; OR SEQUENCE ERROR
THIS TEST VERIFIES SINGLE OPERAND NON-MODIFYING BYTE INSTRUCTIONS WITH MODE 1.
RO 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 57**
TEST MODE 0 EVEN BYTE W/SOP NON-MODIFYING

TST57:
INC (R2); UPDATE TEST NUMBER
CMP #57, (R2); SEQUENCE ERROR?
BNE TST60-10; BR TO ERROR HALT ON SEQ ERROR
CLR RC; INITIALIZE
COMB RO; RO=377
SCC; SET CC=0111
CLN
TSTB RO; TRY TST EVEN BYTE
BVS SNMBDA; CHECK CC=1000
BLO SNMBDA
BMI TST60

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

SNMBDA:
MOV #111-(R2); MOVE TO MAILBOX " ****** 111 ******"
INC -(R2); SET MSGTYP TO FATAL ERROR
HALT; CONDITION CODES NOT SET PROPERLY
OR SEQUENCE ERROR
F04

***********************************************************************

THIS TEST VERIFIES SINGLE OPERAND INSTRUCTIONS WITH MODE 1.
RD 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 JSING RD AND CONDITIONAL BRANCHES TEST
THE RESULTS.

***********************************************************************

TEST 60 TEST MODE 1 SOP NON-MODIFYING

**********inkle INSTRUCTION**********

TST60: INC (R2) ; UPDATE TEST NUMBER
        CMP #60, "C2" ; SEQUENCE ERROR?
        BNE TST61-10 ; BR TO ERROR HALT ON SEQ ERROR
        CLR (RD) ; POINT TO LOC 0
        CLR (RD) ; CLEAR LOC 0
        SCC ; INITIALIZE
        CLZ ; CC=1001
        TST (RD) ; TRY TST W/ MODE 1
        BVS SNMIA ; CHECK CC=0100
        BCS SNMIA
        BMI SNMIA
        BEQ TST61

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

SNMIA: MOV #112, -(R2) ; MOVE TO MAILBOX # 112
        INC -(R2) ; SET MSGTPY TO FATAL ERROR
        HALT ; CC'S NOT SET PROPERLY
        OR SEQUENCE ERROR
G04

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

***********************************************************************

TEST 61  TEST MODE 1 BYTE INST. NON-MODIFYING

***********************************************************************

TS61: INC  (R2) ; UPDATE TEST NUMBER
      CJP  #61 (R2) ; SEQUENCE ERROR?
      BNE TST62-10 ; BR TO ERROR HALT ON SEQ ERROR
      CLR  RO ; POINT TO LOC 0
      CLR  (RO); CLEAR LOC 0
      COMB (RO) ; COMPLEMENT BYTE 0
      SCC  ; SET CC=0111
      CLZ

TS61: BSTB (RO) ; TRY TST ON EVEN BYTE
      BVS SXMBIA
      BLO SXMBIA
      BMI SXMBIB

SNMBIA: MOV  #113, -(R2) ; MOVE TO MAILBOX # ****** 113 ******
       INC  -(R2) ; SET MSGTYP TO FATAL ERROR
       HALT ; CC'S NOT CORRECT

SNMBIB: CLR  RO ; SET CC=1011
       INC  RO
       SCC
       CLZ
       BSTB (RO) ; TRY TO TST AN ODD BYTE
       BVS SXMBIC
       BCE SXMBIC
       BMI SXMBIC
       BEQ TST62

SNMBIC: MOV  #114, -(R2) ; MOVE TO MAILBOX # ****** 114 ******
       INC  -(R2) ; SET MSGTYP TO FATAL ERROR
       HALT ; CC'S NOT CORRECT
       OR SEQUENCE ERROR
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 INCORRECT PROPERLY.

**TEST 62**  TEST MODE 2 WITH SOP NON-MODIFYING

**TST62:**  INC (R2): UPDATE TEST NUMBER
            CMP #62,(R2): SEQUENCE ERROR?
            BNE TST63-10: BR TO ERROR HALT ON SEQ ERROR
            CLR RO: INITIALIZE RO=0
            CLR (RO): CLEAR LOC 0
            SEC
            CLZ
            TST (RO): TRY TST W/ MODE 2
            BV S MN2A: CHECK CC=0100
            BCS SNM2A
            BMI S NM2B
            BEQ SNM2B

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
         REPLACE THE MOVE INSTRUCTION WHICH follows w/ 767

**SNM2A:**  MOV #115,-(R2): MOVE TO MAILBOX 115
            INC -(R2): SET MSGTYP TO FATAL ERROR
            HALT

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND
         REPLACE THE MOVE INSTRUCTION WHICH follows w/ 760

**SNM2B:**  DEC RO: CC'S NOT CORRECT
            DEC RO
            BEQ TST63

**MOVE TO MAILBOX 116****
**SET MSGTYP TO FATAL ERROR**
**MODE 2 DID NOT INC REQ CORRECTLY**
**OR SEQUENCE ERROR**
THIS TEST VERIFIES MODE 2 SINGLE OPERAND NON-MODIFYING BYTE
INSTRUCTIONS IT USES RO TO POINT TO LOC. 0 WITH LOCATION 0
SET TO 377 THE EVEN AND ODD BYTE 12 TESTED WITH TSTB INSTRUCTIONS
TO VERIFY THE CORRECT CC ARE SET THE REGISTER IS CHECKED FOR
PROPER INCREMNETING.

**TEST 63**
**TEST MODE 2 - BYTE W/SOP NON-MODIFYING**

**TST63:** INC (R2) ; UPDATE TEST NUMBER
            CMP $63,(R2) ; SEQUENCE ERROR
            BNE TST64-10 ; BR TO ERROR HALT ON SEQ ERROR
            CLR RO ; CLEAR RO
            CLR (RO) ; CLEAR LOC 0
            COMB (RO) ; SET LOC 0=377
            SCC ; SET CC=0111
            CLN
            TSTB (RO)+ ; TRY TST OF EVEN BYTE
            BVS SMB2A
            BLES SMB2A
            BMI SMB2B

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

**SMB2A:** MOV $117,-(R2) ; MOVE TO MAILBOX # 117
           INC -(R2) ; SET MSGTYP TO FATAL ERROR
           HLT ; CC'S NOT SET CORRECTLY
           SMB2B: DEC RO ; DECREMENT RO
           BEQ SMB2C ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
           CONDITIONAL BRANCH INST. AND
           REPLACE THE MOVE INSTRUCTION
           WHICH follows W/ 761

**SMB2C:** MOV $120,-(R2) ; MOVE TO MAILBOX # 120
           INC -(R2) ; SET MSGTYP TO FATAL ERROR
           HALT ; MODE 2 DID NOT INC REG CORRECTLY
           SMB2D: INC RO ; POINT TO ODD BYTE
           SCC ; SET CC=1011
           SMB2E: TSTB (RO)+ ; TRY TST OF ODD BYTE
           BVS SMB2D ; CHECK CC'S=0100
           BCS SMB2D
           BMI SMB2D
           BEQ SMB2E ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
           CONDITIONAL BRANCH INST. AND
           REPLACE THE MOVE INSTRUCTION
           WHICH follows W/ 748

**SMB2D:** MOV $121,-(R2) ; MOVE TO MAILBOX # 121
           INC -(R2) ; SET MSGTYP TO FATAL ERROR
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2273</td>
<td>004524</td>
<td>000000</td>
</tr>
<tr>
<td>2274</td>
<td>004526</td>
<td>005300</td>
</tr>
<tr>
<td>2275</td>
<td>004530</td>
<td>006300</td>
</tr>
<tr>
<td>2276</td>
<td>004532</td>
<td>001404</td>
</tr>
<tr>
<td>2277</td>
<td>SNMB2E</td>
<td>DEC RO</td>
</tr>
<tr>
<td>2278</td>
<td>BEG</td>
<td>TST64</td>
</tr>
<tr>
<td>2279</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2280</td>
<td>004524</td>
<td>012742</td>
</tr>
<tr>
<td>2281</td>
<td>004526</td>
<td>005300</td>
</tr>
<tr>
<td>2282</td>
<td>004530</td>
<td>006300</td>
</tr>
<tr>
<td>2283</td>
<td>004542</td>
<td>000000</td>
</tr>
</tbody>
</table>

: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 736. MOVE TO MAILBOX ♂ 122  SET MSGTYP TO FATAL ERROR ♂ 122  ITEM  víde 6  DID NOT INCREMENT PROPERLY OR SEQUENCE ERROR
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
1ST MODE 3 INSTRUCTION.

TEST 64 TEST MODE 3 W/ SOP NON-MODIFYING INSTS

TEST64: INC (R2); UPDATE TEST NUMBER
            CMP #64,(R2) ;SEQUENCE ERROR
            BNE TST65-10 ;BR TO ERROR HALT ON SEQ ERROR
            CLR (R2) ;CLEAR LOC 0
            DEC (R2) ;RD=376
            BGE TST65-10
            CLZ ;SET CC=1011
            TST (R6); 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/ 765

SNM3A:
            MOV #123,-(R2) ;MOVE TO MAILBOX # 123
            INC -(R2); SET MSGTYYP TO FATAL ERROR
            HALT ;CC'S NOT CORRECT
            BGE TST65

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

SNM3B:
            MOV #124,-(R2) ;MOVE TO MAILBOX # 124
            INC -(R2); SET MSGTYYP TO FATAL ERROR
            HALT ;MODE 3 DID NOT INC REG CORRECTLY
            OR SEQUENCE ERROR
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 INCRCMENTING AND
THE CC’S ARE VERIFIED.
THE TABLE AT LOC. 402-404 SHOULD CONTAIN 0 AND 1 BEFORE AND
AFTER THE TEST IS RUN.

TEST 65  TEST MODE 3 - BYTES W/ SOP NON-MODIFYING INST.

ST65: INC (R2); UPDATE TEST NUMBER
      CMP #65, (R2); SEQUENCE ERROR?
      BNE TST65-10; BR TO ERROR HALT ON SEQ ERROR
      CLR RO; RO=0
      CLR (R0); CLEAR LOC 0
      COMB (RO); LOC. 0 =377
      COMB RO
      INC RO
      TST (RO)+; RO=402
      SCC; CC=0111
      CLN
      TSTB 0(R0)+; TRY TST OF EVEN BYTE
      BVS SNMB3A; CHECK CC=1000
      BLO8 SNMB3A
      BMI SNMB3B

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

SNMB3A:
      MOV #125,-(R2); MOVE TO MAILBOX 125
      INC -(R2); SET MSGTP TO FATAL ERROR
      HALT; CC'S NOT CORRECT

SNMB3B:
      SNC
      CLR
      TSTB 0(R0)+; TRY TST OF ODD BYTE
      BVS SNMB3C; CHECK CC=0100
      BLO8 SNMB3C
      BMI SNMB3C
      BEQ SNMB3D

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

SNMB3C:
      MOV #126,-(R2); MOVE TO MAILBOX 126
      INC -(R2); SET MSGTP TO FATAL ERROR
      HALT; CC'S NOT CORRECT

SNMB3D:
      TST (RO)+; RO=410
      TST (RO)
      BMI TST66

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
**MO4**

*REPLACE THE MOVE INSTRUCTION * (====)

WHICH FOLLOWS W/ 767 (====)

**MOVE TO MAILBOX # ***** 127 *******

SET MSGTPY TO FATAL ERROR

**HALT - (R2)**

TSTB DID NOT INCREMENT RD CORRECTLY

**OR SEQUENCE ERROR**

******************************************************************************

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

LOC. 0 IS SET TO -1 AND THE CC'S ARE SET TO THE COMPLEMENT OF THE

EXPECTED RESULTS. RD AND SET TO 2 AND A TST MODE 4 IS EXECUTED.

THE CC'S ARE CHECKED WITH CONDITIONAL BRANCH INSTRUCTIONS AND THE REGISTER

IS CHECKED FOR PROPER DECREMENTING.

******************************************************************************

**TEST 66 TEST MODE 4 W/ SOP NON-MODIFYING INSTS**

******************************************************************************

**TST66: INC (R2)**

UPDATE TEST NUMBER

**BNE TST67-10 BR TO ERROR HALT ON SEQ ERROR**

**CLR (R0) RD=0**

**CLR (R0) LOC 0=0**

**COM (R0)+ LOC 0=1**

**SCC SET CC=1011**

**CLZ**

**TST - (R0) TRY TST W/ MODE 4**

**BVS SNM4A CHECK CC=0100**

**BLO SNM4B**

**BMI SNM4B**

**: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**

**CONDITIONAL BRANCH INST. AND**

**REPLACE THE MOVE INSTRUCTION**

**WHICH FOLLOWS W/ 767**

**SNM4A: MOV #130 -(R2) MOVE TO MAILBOX # ***** 130 *******

**SET MSGTPY TO FATAL ERROR**

**HALT -(R2)**

**CC'S NOT CORRECT**

**SNM4B: TST RO**

**BEQ TST67**

**: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**

**CONDITIONAL BRANCH INST. AND**

**REPLACE THE MOVE INSTRUCTION**

**WHICH FOLLOWS W/ 761**

**SNM4A: MOV #131 -(R2) MOVE TO MAILBOX # ***** 131 *******

**SET MSGTPY TO FATAL ERROR**

**HALT -(R2)**

**TST MODE 4 DID NOT DEC RD CORRECTLY**

**OR SEQUENCE ERROR**
NO4

This test verifies Mode 5 SOP non-modifying instructions. It uses a pointer at Loc. 376 to test Loc. O. RO is set to 000, a TST Mode 5 instruction is executed and the cc's checked. RO is checked to insure proper decrementing.

Test 67: Test Mode 5 w/ SOP non-modifying Insts

**TST67: Update test number**

**MOV #132, -(R2)**

**132**

**INC -(R2)**

**SET MSGTP0 TO FATAL ERROR**

**INC RO**

**SET CC=0011**

**SET CC=0100**

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

**TST70: Test Mode 5 w/ Sequence error**

**MOV #133, -(R2)**

**INC -(R2)**

**SET MSGTP0 TO FATAL ERROR**

**INC RO**

**MODE 5 DID NOT DEC RO CORRECTLY OR SEQUENCE ERROR**
THIS TEST VERIFIES MODE 6 SOP NON-MODIFYING INSTRUCTIONS.
RO IS SET TO 377 AND A MODE 6 TST INSTRUCTION IS EXECUTED USING RO AND AN OFFSET OF -377. THE CC'S ARE CHECKED AS WELL AS RO TO ENSURE IT WAS NOT ALTERED.

TEST 70 TEST MODE 6 W/ SOP NON-MODIFYING INSTS

**TST70**

<table>
<thead>
<tr>
<th>INC</th>
<th>(R2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP</td>
<td>370</td>
</tr>
<tr>
<td>BNE</td>
<td>TST71-10</td>
</tr>
<tr>
<td>CLR</td>
<td>RO</td>
</tr>
<tr>
<td>OR</td>
<td>0</td>
</tr>
<tr>
<td>COM</td>
<td>(RO)</td>
</tr>
<tr>
<td>AND</td>
<td>0=1</td>
</tr>
<tr>
<td>SBC</td>
<td>RO</td>
</tr>
<tr>
<td>OR</td>
<td>377</td>
</tr>
<tr>
<td>CLN</td>
<td></td>
</tr>
<tr>
<td>SBC</td>
<td>TST-377 (RO)</td>
</tr>
<tr>
<td>BVS</td>
<td>SNM6A</td>
</tr>
<tr>
<td>BLO</td>
<td>SNM6A</td>
</tr>
<tr>
<td>BMI</td>
<td>SNM6B</td>
</tr>
<tr>
<td>MOV</td>
<td>#134</td>
</tr>
<tr>
<td>INC</td>
<td>-(R2)</td>
</tr>
<tr>
<td>HALT</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>#135</td>
</tr>
<tr>
<td>INC</td>
<td>-(R2)</td>
</tr>
<tr>
<td>HALT</td>
<td></td>
</tr>
</tbody>
</table>

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

SNM6A: MOVE TO MAILBOX 8 134

SNM6B: MOV #135 | -(R2) |
INC | -(R2) |
HALT |

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

SET MSCVY TO FATAL ERROR CC'S INCORRECT

SET MSCVY TO FATAL ERROR TST MODE 6 INCORRECTLY CHANGED RC OR SEQUENCE ERROR
**C05**

**TEST PCHE & W SOP MCN-MODIFYING INSTS**

this test verifies mode 7 sop non-modifying instructions.

it uses a pointer to loc. 0 stored at loc. 400 to tsi loc. 0.

rd is set to 377 and loc. 0 is tested thru the pointer at 400 using

rd and an offset of 1.

**TEST 71**

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC</td>
<td>update test number</td>
</tr>
<tr>
<td>CMP</td>
<td>sequence error?</td>
</tr>
<tr>
<td>BNE</td>
<td>tst72-10: br to error halt on seq error</td>
</tr>
<tr>
<td>CLR</td>
<td>rd=0</td>
</tr>
<tr>
<td>CLR</td>
<td>loc o=0</td>
</tr>
<tr>
<td>COM</td>
<td>loc o=-1</td>
</tr>
<tr>
<td>COMB</td>
<td>rd=377</td>
</tr>
<tr>
<td>BHI</td>
<td>cc=0111</td>
</tr>
<tr>
<td>CLN</td>
<td></td>
</tr>
<tr>
<td>TST</td>
<td>@1(rd)</td>
</tr>
<tr>
<td>BVS</td>
<td>sm7a</td>
</tr>
<tr>
<td>BLO</td>
<td>sm7a</td>
</tr>
<tr>
<td>BHI</td>
<td>sm7b</td>
</tr>
<tr>
<td>SNM7A:</td>
<td>move to mailbox 8 136</td>
</tr>
<tr>
<td>INC</td>
<td>- (r2)</td>
</tr>
<tr>
<td>HALT</td>
<td>set msgtyp to fatal error</td>
</tr>
<tr>
<td>COMB</td>
<td>rd=0</td>
</tr>
<tr>
<td>SNM7B:</td>
<td>loc 377</td>
</tr>
<tr>
<td>MOV</td>
<td>@137 -(r2)</td>
</tr>
<tr>
<td>INC</td>
<td>-(r2)</td>
</tr>
<tr>
<td>HALT</td>
<td>move to mailbox 8 137</td>
</tr>
<tr>
<td>TST</td>
<td>mode 7 incorrectly changed cc or seq error</td>
</tr>
</tbody>
</table>

---

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

---

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

---
**This test verifies mode 0 double operand instructions. It sets data in RO and RA and uses the ADD instruction to test the DCP microcode.**

**Test 72: Test Mode 0 Double-Operand (DOP) Insts.**

**ST72:**

```
INC (R2)   ; Update test number
CMP #72, (R2)   ; Sequence error?
BNE TST73-10   ; BR to Error halt on Seq Error
CLR RO   ; RO=0
COM RO   ; RO=1
CLR R4   ; R4=0
ADD R0, R4   ; Try Add: R4=1
INC R4   ; R4=0
BEQ TST73
```

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

```
MOV #140, -(R2)   ; Move to mailbox # ****** 140 ******
INC -(R2)   ; Set MSGTYP to Fatal Error
HALT   ; Add inst. failed w/ mode 0
```

**This test verifies the move instruction with mode 0 to mode 0. This test is necessary because this particular instruction utilizes unique microcode.**

**Test 73: Move Mode 0 to Mode 0**

**ST73:**

```
INC (R2)   ; Update test number
CMP #73, (R2)   ; Sequence error?
BNE TST74-10   ; BR to Error halt on Seq Error
CLR RO   ; RO=0
CLR R4   ; R4=0
COM RO   ; RO=1
MOV R0, R4   ; Try move -1 to R4
INC R4   ; R4=0
BEQ TST74
```

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

```
MOV #141, -(R2)   ; Move to mailbox # ****** 141 ******
INC -(R2)   ; Set MSGTYP to Fatal Error
HALT   ; Move failed mode 0 to mode 0
```

OR SEQUENCE ERROR
**THIS TEST QUICKLY VERIFIES THE REMAINING DOP MODIFYING INSTRUCTIONS WITH MODE 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 74** TEST ALL THE DOP INSTRUCTIONS W/ SOURCE MODE 0

**ST74:**
- INC \#142 (R2) **(R2)**
  - UPDATE TEST NUMBER
- CMP \#74 (R2)
- BR TO ERROR HALT ON SEQ ERROR
- CLR R0
- MOV R0,R4
- SEQ DOPDA
  - TRY MOVE MODE 0,0

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

**DOPDA:**
- MOV \#142 -(R2)
  - MOVE TO MAILBOX #
- INC -(R2)
  - SET MSGTYP TO FATAL ERROR
- HALT
  - 7-BIT NOT SET
- R0=1
- R0=177776
- R4=177777
- TRY BIC: R4=1
- R4=0

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

**DOPOB:**
- MOV \#143 -(R2)
  - MOVE TO MAILBOX #
- INC -(R2)
  - SET MSGTYP TO FATAL ERROR
- HALT
  - BIC CLEAR RESULT INCORRECT
- R0=177777
- R4=0
- TRY BIS: R4=177777

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

**DOPOC:**
- MOV \#144 -(R2)
  - MOVE TO MAILBOX #
- INC -(R2)
  - SET MSGTYP TO FATAL ERROR
- HALT
  - RESULT OF BIS INCORRECT
- R0=0
- R0=377
- R4=0
- R4=177777
- R4=177777
- TRY ADD: R4=177777
- R4=0
FOS

TO SCOPE: CLEAR THE RIGHT BYTE OF "145" CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH follows W/ 737

MOVE TO MAILBOX 145
SET MSGTYP TO FATAL ERROR
RESULT OF ADD INCORRECT
177401=R4
R4=17777
RD=0

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH follows W/ 727

MOVE TO MAILBOX 146
SET MSGTYP TO FATAL ERROR
RESULT OF SUB INCORRECT OR SEQUENCE ERROR
TEST 75  TEST DOP NON-MODIFYING INST. W/ SOURCE MODE 0

005504  005212
005506  002712  000075
005512  001046
005514  005000
005516  005004
005518  005204
005520  005204
005522  020400
005524  003004
005526  012742  000147
005528  002449
005530  002449
005532  000000
005534  000000
005536  020004
005538  002404
005540  002404
005542  012742  000150
005544  002449
005546  002449
005548  000000
005550  000000
005552  002000
005554  002400
005556  001404
005558  001404
005560  012742  000151
005562  002449
005564  002449
005566  000000
005568  000000
005570  005000
005572  005000
005574  005000
005576  005000
005578  030004
005580  001404
005582  012742  000152
005584  002449
005586  002449
005588  000000
005590  000000
005592  000000
005594  000000
005596  000000
005598  000000
005590  000000

This test verifies mode 0 DOP non-modifying instructions. RO and R4 are preset to 0 and 1 respectively. Compare instructions are then executed and checked. First R4 is compared to RO then RO to R4.

**UPDATE TEST NUMBER**

**SEQUENCE ERROR?**

**BR TO ERROR HALT ON SEQ ERROR**

**RO=0**

**R4=0**

**R4=1**

**INC R4**

**INC R4**

**CMP R4,R0**

**BGT DNM1**

**TRY COMPARE R4 TO RO**

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

**MOVE TO MAILBOX **

**SET MSGTYP TO FATAL ERROR**

**CC'S NOT CORRECT FOR CMP**

**TRY COMPARE RO TO R4**

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

**MOVE TO MAILBOX **

**SET MSGTYP TO FATAL ERROR**

**CC'S NOT CORRECT FOR CMP**

**TRY COMPARE R4=1 TO RO=1**

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

**MOVE TO MAILBOX **

**SET MSGTYP TO FATAL ERROR**

**CC'S NOT CORRECT (Z=1) FOR CMP**

**TRY BIT RD TO R4**

**TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 745**

**MOVE TO MAILBOX **

**SET MSGTYP TO FATAL ERROR**

**CC'S NOT CORRECT FOR BIT**
H05

INN4: DCB RM
TRY BIT AGAIN

MOV #153,-(R2)
INC -(R2)
HALT

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 736

MOVE TO MAILBOX # 153
SET MSG TYP TO FATAL ERROR
CC'S NOT CORRECT FOR BIT
OR SEQUENCE ERROR

;*******************************************************************************

; THIS TEST VERIFIES MODE 1 DOP INSTRUCTIONS. RO 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 RO AND THE
; RESULTS VERIFIED.

;*******************************************************************************

; TEST 76 TEST MODE 1 W/ DOP INST.

TST76: INC (R2); UPDATE TEST NUMBER
CMP #76,(R2); SEQUENCE ERROR?
BNE TST77-10; BR TO ERROR HALT ON SEQ ERROR
CLR RO
R4=0
CLR (R4)
INC (R4)
LOC 0=1
ADD (R4),RO; TRY ADD SOURCE MODE 1
BEQ TST77

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

MOVE TO MAILBOX # 154
SET MSG TYP TO FATAL ERROR
RESULT OF ADD INCORRECT
OR SEQUENCE ERROR
**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 77 - TEST MODE 1 - EVEN BYTE W/ Dop Insts.**

**------------------------------------------**

**ST77: INC (R2) ;UPDATE TEST NUMBER**
**CMP #7 (R2) ;SEQUENCE ERROR?**
**SNE TST100-10 ;BR TO ERROR HALT ON SEQ ERROR**
**CLR R0 ;R0=0**
**CLR (R0) ;Loc. 0=0**
**COM (R0) ;Loc. 0=17777**
**CLR R4 ;R4=0**
**BISB (R0),R4 ;TRY MODE 1- EVEN BYTE W/ Dop**
**COMB R4 ;R4=0**
**BEQ TST100**

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ ?7? **

**MOV #155,-(R2) ;MOVE TO MAILBOX # ******* 155 *********
**INC -(R2) ;SET MSGTYEP TO FATAL ERROR**
**HALT ;RESULT OF BISB IS INCORRECT OR SEQUENCE ERROR**
THIS TEST VERIFIES MODE 1 DOP NON-MODIFYING INSTRUCTIONS.

WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO -1 AND RD 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 100

TEST MODE 1 - EVEN BYTE W/ DOP NON-MODIFYING INST.

ST100: INC (R2) ; UPDATE TEST NUMBER

CMP #100,(R2) ; SEQUENCE ERROR?

BNE TST101-10 ; BR TO ERROR HALT ON SEQ ERROR

CLR RD ; RD=0

CLR (RD) ; LOC O=0

CLR R4 ; R4=0

CMPB (RD),R4 ; TRY MODE 1 - EVEN BYTE W/ DOP NON-MODIFYING

BEQ TST101

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

MOV #156,-(R2) ; MOVE TO MAILBOX # 156

INC -(R2) ; SET MSGTYP TD FATAL ERROR

HALT ; RESULT OF CMPB INCORRECT

; OR SEQUENCE ERROR
KOS

**MAIN MENU** 12-01 04 CPU TEST
MACY11 127073 06-OCT-76 14:01 PAGE 221

DGKAAA.P11 "10C TEST MODE" - EVEN BYTE W/ DOP NON-MODIFYING INST.

```
K05

;***********************************************************************************************
; THIS TEST VERIFIES MODE 1,0 MOVB INSTRUCTIONS
; WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO 177400. RO IS CLEARED AND
; R4 IS SET TO -1. MOVB ARE USED TO MOVE BYTE 0 TO R4. THIS
; VERIFIES THAT THE PROPER BYTE WAS SELECTED AND THAT THE 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 101 TEST MOV INSTRUCTION MODE 1,0 EVEN BYTE
;**************************************************************************************

TST101: INC (R2) ; UPDATE TEST NUMBER
               CMP #101,(R2) ; SEQUENCE ERROR?
               BNE TST102-10 ; BR TO ERROR HALT ON SEQ ERROR
               CLR R0
               CLR (R0)
               LOC 0=0
               COMB (R0)
               LOC 0=177400
               CLR R4
               COM R4
               R4=0
               MOVB (R0),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/ 767

TST102: MOV #157,-(R2)
               INC -(R2)
               MOV #157,-(R2)
               MOVE TO MAILBOX 1 157
               INC -(R2)
               HALT
               MOVB (R0),R4
               LOC 0=177777
               DO MOVB W/ EVEN BYTE

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

TST103: MOV #160,-(R2)
               INC -(R2)
               MOVB (R0),R4
               LOC 0=177777
               DO MOVB W/ EVEN BYTE

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

```

```
;***********************************************************************************************
; THIS TEST VERIFIES MODE 1,0 MOVB INSTRUCTIONS
; WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO 177400. RO IS CLEARED AND
; R4 IS SET TO -1. MOVB ARE USED TO MOVE BYTE 0 TO R4. THIS
; VERIFIES THAT THE PROPER BYTE WAS SELECTED AND THAT THE 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 101 TEST MOV INSTRUCTION MODE 1,0 EVEN BYTE
;**************************************************************************************

TST101: INC (R2) ; UPDATE TEST NUMBER
               CMP #101,(R2) ; SEQUENCE ERROR?
               BNE TST102-10 ; BR TO ERROR HALT ON SEQ ERROR
               CLR R0
               CLR (R0)
               LOC 0=0
               COMB (R0)
               LOC 0=177400
               CLR R4
               COM R4
               R4=0
               MOVB (R0),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/ 767

TST102: MOV #157,-(R2)
               INC -(R2)
               MOV #157,-(R2)
               MOVE TO MAILBOX 1 157
               INC -(R2)
               HALT
               MOVB (R0),R4
               LOC 0=177777
               DO MOVB W/ EVEN BYTE

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

TST103: MOV #160,-(R2)
               INC -(R2)
               MOVB (R0),R4
               LOC 0=177777
               DO MOVB W/ EVEN BYTE

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

```
**L05**

**THIS TEST VERIFIES MODE 1 Dop INSTRUCTIONS WHICH REFERENCE ODD BYTES. LOC. 0 IS SET TO 177400. R4 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 102 TEST MODE 1-ODD BYTE W/ Dop Insts.**

**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) ; LOC. 0=0 CLR R4 ; R4=0 INC R4 ; R4=1 COMB (R4) ; LOC. 0=177400 BISB (R4),(R0) ; TRY TO BIS LOW ORDER BITS W/ MODE 1 INC (R0) ; CHECK RESULT BEQ TST103 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND WHICH FOLLOWS W' 770 MOV #161,-(R2) ; MOVE TO MAILBOX 161 ** *** INC -(R2) ; SET MSGTYP TO FATAL ERROR HALT ; RESULT OF BISB INCORRECT ; OR SEQUENCE ERROR
This test verifies mode 2 Dop instructions. LOC. 0 is set to -1.
RO 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 103: Test mode 2 w/ Dop insts.

ST103: INC (R2) ; Update test number
       CMP #103,(R2) ; Sequence error?
       BNE TST104-10
       CLR RO ; RO=0
       CLR (R0) ; LOC. = 0
       COM (R0) ; LOC. = 00000
       MOV (R0)+,R4 ; TRY MOVE MODE 2,0
       INC R4 ; CHECK R4
       BEQ DOP2

       TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       CONDITIONAL BRANCH INST. AND
       REPLACE THE MOVE INSTRUCTION
       WHICH FOLLOWS W/ 772
       MOVE TO MAILBOX # **** 162 *****
       INC -(R2) ; SET MSGTYP TO FATAL ERROR
       HALT
       DOP2: DEC RO
              DEC RO
              BEQ TST104

       TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       CONDITIONAL BRANCH INST. AND
       REPLACE THE MOVE INSTRUCTION
       WHICH FOLLOWS W/ 763
       MOVE TO MAILBOX # **** 163 *****
       INC -(R2) ; SET MSGTYP TO FATAL ERROR
       HALT
       REGISTER NOT INCREMENTED IN MODE 2
       OR SEQUENCE ERROR
**NO5**

THIS TEST VERIFIES MODE 2 DOP BYTE INSTRUCTIONS WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO -. RD 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 104  TEST MODE 2 - EVEN BYTE W/ DOP INST.**

**TST104:** INC (R2)  UPDATE TEST NUMBER

CMP #104 (R2): SEQUENCE ERROR?

BNE TST105-10: BR TO ERROR HALT ON SEQ ERROR

CLR RD: RD=0

MOV RD (RD): LOC. 0=0

COM (RD): LOC. 0=17777

BICB (RD)+(R2): TRY TO CLEAR BYTE 1 FROM BYTE 0 W/ BICB

ISTB #1: CHECK RESULT

BEQ DOPB2A: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH Follows W/ 771

MOV #164 -(R2): MOVE TO MAILBOX 8 **164** **164**

INC -(R2): SET MSGTYP TO FATAL ERROR

HALT: BICB DESTINATION INCORRECT

DOPB2A: COMB @0: CHECK BICB SOURCE

BEQ TST105: TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH Follows W/ 762

MOV #165 -(R2): MOVE TO MAILBOX 8 **165** **165**

INC -(R2): SET MSGTYP TO FATAL ERROR

HALT: BICB SOURCE INCORRECTLY CHANGED OR SEQUENCE ERROR
THIS TEST VERIFIES MODE 2 DOP BYTE INSTRUCTIONS WHICH REFERENCE
ODD BYTES. RC IS SET TO 1, LOC. O IS SET TO 177400, AND R4 IS CLEARED.
A MODE 2 MOVBE USES RD TO MOVE BYTE 1 TO R4. AN INCREMENT
IS USED TO CHECK THAT THE PROPER BYTE WAS MOVED AND SIGN X-TENDED.

TEST 105:  TEST MODE 2 - ODD BYTE W/ DOP INST.

**----------**
**ST105: INC (R2), UPDATE TEST NUMBER**
**CMP #105(R2), SEQUENCE ERROR**
**BNE TST106-10, BR TO ERROR HALT ON SEQ ERROR**
**CLR R4, :LOC. = 0**
**CLR R4, :LOC. = 177777**
**MOVBE (R0)+, R4, TRY DOP MODE 2 W/ ODD BYTE**
**INC R4, CHECK RESULT OF MOVBE**
**BEQ DOPB2B, TO SCOPE THE RIGHT BYTE OF THIS**
**COND. BRANCH INST. AND WHICH FOLLOWS W/ 776**

**DOPB2B: MOV #166, -(R2), MOVE TO MAILBOX ***** 166 *******
**INC -(R2), SET MSGTYP TO FATAL ERROR**
**HALT, RESULT OF MOVBE INCORRECT**
**DOPB2B: TST -(R0), BUMP RD DOWN BY 2**
**BEQ TST106, CHECK RD**
**TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**
**COND. BRANCH INST. AND REPLACE THE MOVE INSTRUCTION**
**WHICH FOLLOWS W/ 761**

**MOV #167, -(R2), MOVE TO MAILBOX ***** 167 *******
**INC -(R2), SET MSGTYP TO FATAL ERROR**
**HALT, MODE 2 BYTE DID NOT INCREMENT REG. CORRECTLY**
**OR SEQUENCE ERROR**
THIS TEST VERIFIES MODE 3 DOUBLE-Operand INSTRUCTIONS. LOC. 0 IS LOADED WITH ALTERNATING ZEROS AND ONES; AND RD IS LOADED WITH ALTERNATING ONES AND ZEROS. A MODE 3 BIS IS Used TO SET RD TO -1 BY USING LOC. 0 AS THE SOURCE TO BIS THE ZEROS IN RD. THE RESULT IS TESTED BY INCREMENTSING RD AND CHECKING FOR ZERO.

TEST 106: TEST MODE 3 W/ DOP INSTS.

TST106: INC (R2; UPDATE TEST NUMBER
CMP #106 (R2; SEQUENCE ERROR?
BNE TST107-10; BR TO ERROR HALT ON SEQ ERROR
MOV #052525.00; MOVE 1'S AND 0'S TO LOC. 0
MOV #125252,RO; SET ALL ONE AND ZERO IN RD
BIS #0,RO; TRY TO SET ALL OTHER BITS W/ MODE 3
INC RD; TEST RESULT
BEQ TST107-10; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
MOV #170,-(R2; CONDITIONAL BRANCH INST. AND
INC -(R2; REPLACE THE MOVE INSTRUCTION
HALT; WHICH FOLLOWS W/ 767

TST107: MOV TO MAILBOX # 170 *****
SET MSGTYP TO FATAL ERROR
BIS W/ MODE 3 INCORRECT RESULT
OR SEQUENCE ERROR

TEST 107: TEST MODE 3 - EVEN BYTE W/ DOP INSTS.

TST107: INC (R2; UPDATE TEST NUMBER
CMP #107 (R2; SEQUENCE ERROR?
BNE TST110-10; BR TO ERROR HALT ON SEQ ERROR
MOV #52652.00; MOVE 1'S AND 0'S PATTERN TO LOC. 0
CLR RD; MOVE RD TO 0
BISB #0,RO; TRY RD=252 W/ MODE 3 - EVEN BYTE
CMP #252,RO; BISB W/ EVEN BYTE SUCCESSFUL?
BEQ; TST110; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
MOV #171,-(R2; CONDITIONAL BRANCH INST. AND
INC -(R2; REPLACE THE MOVE INSTRUCTION
HALT; WHICH FOLLOWS W/ 767
MOV TO MAILBOX # 171 *****
SET MSGTYP TO FATAL ERROR
BISB W/ MODE 3 - EVEN BYTE FAILED
OR SEQUENCE ERROR
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: RO = 0125.

TEST 110   TEST MODE 3 - ODD BYTE W/ DOP INSTS.

TST110: INC (R2); UPDATE TEST NUMBER
CMP #110 (R2); SEQUENCE ERROR?
BNE TST110-10; BR TO ERROR HALT ON SEQ ERROR
MOV #S2652,3#D; MOVE 1'S AND 0'S PATTERN TO LOC 0
CLR RO; RO=0
BISB #1, RO; TRY RO=152 W/ MODE 3 - ODD BYTE
CMP #126, RO; RO=125
BEQ TST111; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION

MOV #172,-(R2); MOVE TO MAILBOX # 172
INC -(R2); SET MSGTYP TO FATAL ERROR
HALT; BISB W/ MODE 3 - ODD BYTE FAILED

OR SEQUENCE ERROR

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
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 111   TEST MODE 4 W/ DOP INSTS.

TST111: INC (R2); UPDATE TEST NUMBER
CMP #111,(R2); SEQUENCE ERROR?
BNE DOP4; BR TO ERROR HALT ON SEQ ERROR
MOV #TBL, RO; INITIALIZE RO
MOV -(R0), @TBL; TBL=12552
ADD -(R0), @TBL; TBL=12552
BICB -(R0), @TBL; TBL=02552
BISB -(R0), @TBL+1; TBL=12552
CMP -(R0), @TBL; CHECK RESULT
BEQ TST112; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION

---
THIS TEST VERIFIES MODE 5 DOUBLE OPERAND INSTRUCTIONS.

THE TEST USES AN ADDRESS TABLE STORED FOLLOWING THE TEST CODE.

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 112  TEST MODE 5 W/ DOP INSTS.

TST112: INC R2, R2  ; UPDATE TEST NUMBER
        CMP $112, R2  ; SEQUENCE ERROR?
        BNE DOPS  ; BR TO ERROR HALT ON SEQ ERROR

        MOV #TBL2+2, R0  ; INITIALIZE RD

        MOV R0, #TBL1  ; TBL1=125252

        ADD R0, #TBL1  ; TBL1=125252

        BICB R0, #TBL1  ; TBL1=125252

        BISB R0, #TBL+1  ; TBL1=125252

        CMP R0, #TBL1  ; CHECK RESULT

        BEQ TST113  ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
                      ; CONDITIONAL BRANCH INST. AND
                      ; REPLACE THE MOVE INSTRUCTION
                      ; WHICH Follows W/ 753

        MOV $174, -(R2)  ; MOVE TO MAILBOX # 174 174

        INC -(R2)  ; SET MSGTYP TO FATAL ERROR

        HALT  ; RESULT OF MODE 5 INSTS. INCORRECT

        OR SEQUENCE ERROR

DOPS:

        MOV #173, -(R2)  ; MOVE TO MAILBOX # 173 173

        INC -(R2)  ; SET MSGTYP TO FATAL ERROR

        HALT  ; RESULT OF MODE 4 INSTS. INCORRECT

        OR SEQUENCE ERROR
THIS TEST VERIFIES MODE 6 DOUBLE OPERAND INSTRUCTIONS.
IT USES THE SAME DATA AS THAT USED IN THE MODE 4 TESTS.
THIS TIME THE DATA IS ACCESSED USING MODE 6. RO IS SET TO POINT TO THE MIDDLE OF THE ADDRESS TABLE. THE TABLE IS ACCESSED FROM BOTTOM TO TOP BY VARYING THE OFFSET IN THE MODE 6 INSTRUCTIONS.
THE DATA RESULTS ARE IDENTICAL TO THOSE EXPECTED IN THE MODE 4 TESTS.

TEST 113: TEST MODE 6 W/ DOP INSTS.

**ST113:** INC (R2) : UPDATE TEST NUMBER
CMP #113 (R2) : SEQUENCE ERROR?
BNE TST114-10 : BR TO ERROR HALT ON SEQ ERROR
MOV #TBL-4, RO : INITIALIZE RO
MOV 2 (R0), #TBL1 : TBL1=125252
ADD 0 (R0), #TBL1 : TBL1=000000
BICB -1 (R0), #TBL1 : TBL1=000000
BISB -1 (R0), #TBL1+1 : TBL1=125252
CMP -4 (R0), #TBL1 : CHECK RESULT
BEQ TST114

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

THIS TEST VERIFIES MODE 7 DOUBLE OPERAND INSTRUCTIONS.
THIS TEST USES THE SAME ADDRESS TABLE AND DATA TABLE USED BY THE MODE 5 TESTS. THIS TIME THE DATA IS ACCESSED USING MODE 7.
RO IS SET TO POINT TO THE MIDDLE OF THE ADDRESS TABLE IN THE MODE 5 TEST. THE TABLE IS ACCESSED FROM BOTTOM TO TOP BY VARYING THE OFFSET IN THE MODE 7 INSTRUCTIONS. THE DATA RESULTS ARE IDENTICAL TO THOSE EXPECTED IN THE MODE 5 TESTS.

TEST 114: TEST MODE 7 W/ DOP INSTS.

**ST114:** INC (R2) : UPDATE TEST NUMBER
CMP #114 (R2) : SEQUENCE ERROR?
BNE TST115-10 : BR TO ERROR HALT ON SEQ ERROR
MOV #TBL2-4, RO : INITIALIZE RO
MOV #4 (R0), #TBL1 : TBL1=125252
ADD #0 (R0), #TBL1 : TBL1=000000
BICB #0 (R0), #TBL1 : TBL1=000000
BISB #0 (R0), #TBL1+1 : TBL1=125252
BISB #0 (R0), #TBL1 : TBL1=125252
CMP 3-4 (R0), #TBL1 : CHECK RESULT
BEQ TST115
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH Follows W/ 756

MOVE TO MAILBOX: ********* 176

RESULT OF MODE 7 INSTS INCORRECT OR SEQUENCE ERROR

THIS TEST VERIFIES THE ROTATE MODE 0 INSTRUCTIONS.
RO 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 115 TEST ROTATE INSTRUCTIONS OF MODE 0

ST115: INC (R2) UPDATE TEST NUMBER

CMP #115, (R2) SEQUENCE ERROR?
BNE TST116-10 BR TO ERROR HALT ON SEQ ERROR
MOV #125252,RO INITIALIZE DATA
SEC
ROL RO TRY ROL W/ MODE 0
BVC ROTOA CC=OO1
BCC ROTORA; CHECK DATA
CMP #O52525,RO
BEQ ROTOB

ROTOA:

MOV #177,-(R2) MOVE TO MAILBOX: ********* 177
INC -(R2)
HALT SET MSGTYP TO FATAL ERROR

ROTOB:

MOV #125252,RO INITIALIZED DATA
SEC SET C-BIT
ROLB RO TRY ROL W/ MODE 0 EVEN BYTE
BVC ROTOCC=001
BCC ROTO; CHECK DATA
CMP #125125,RO
BEQ TST116

ROTOC:

MOV #200,-(R2) MOVE TO MAILBOX: ********* 200
INC -(R2)
HALT SET MSGTYP TO FATAL ERROR
ROLB MODE 0 FAILED OR SEQUENCE ERROR
THIS TEST VERIFIES THE ROTATE MODE 1 INSTRUCTIONS.
THE DATA TO BE ROTATED IS IN LOC D. RD IS USED AS THE
ADDRESSING REGISTER. THE C-BIT IS LOAD AND AN ROL IS EXECUTED.
THE RESULTS ARE CHECKED BY COMPARING THE DATA RESULTS AND TESTING
THE C AND V BITS. THIS PROCEDURE IS THEN REPEATED TWICE MORE
TO TEST THE BYTE ROTATES. FIRST ON BYTE D, THEN ON BYTE E.

TEST 16: TEST ROTATE INSTRUCTIONS W/ MODE 1

TST116: INC (R2) : UPDATE TEST NUMBER
         CMP #116, (R2) : SEQUENCE ERROR?
         BNE TST117-10 : BR TO ERROR HALT ON SEQ ERROR
         CLR RO : POINT TO LOC 0
         MOV #5252, (R0) : INITIALIZE DATA
         CLC : CLEAR C-BIT
         ROL (R0) : TRY ROL W/ MODE 1
         BVC ROTIA : CC=1000
         BCS ROTIA : CHECK RESULT
         BEQ ROT16 : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
         ;CONDITIONAL BRANCH INST. AND
         ;REPLACE THE MOVE INSTRUCTION
         ;WHICH FOLLOWS W/ 765
         ;CHECK RESULT

ROTIA: MOV #201, -(R2) : MOVE TO MAILBOX & ******* 201 *******
       INC -(R2) : SET MSGTYP TO FATAL ERROR
       HALT : ROL MODE 1 FAILED

ROTIB: SEC : INITIALIZE DATA
        MOV #12525, (R0) : TRY ROLB W/ MODE 1 EVEN BYTE
        BVC ROTIC : CC=1011
        BCC ROTIC : TEST RESULT
        BEQ ROT16 : TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
        ;CONDITIONAL BRANCH INST. AND
        ;REPLACE THE MOVE INSTRUCTION
        ;WHICH FOLLOWS W/ 747

ROTIC: MOV #202, -(R2) : MOVE TO MAILBOX & ******* 202 *******
       INC -(R2) : SET MSGTYP TO FATAL ERROR
       HALT : ROLB W/ MODE 1 EVEN BYTE FAILED

ROTID: MOV #12525, (R0) : POINT TO ODD BYTE
       CLR RO : SET C-BIT
       INC RO : TRY ROLB W/ MODE 1 ODD BYTE
       BVC ROTIE : CC=0011
       BCC ROTIE : CHECK DATA
       CMP #052652, #0 :
**ROTATE INSTRUCTIONS W/ MODE 1**

**ROT1:**
MOV  #203,-(R2) ; MOVE TO MAILBOX 203
INC  -(R2) ; SET MSGTYP TO FATAL ERROR
HALT ; ROLB W/ MODE 1 ODD BYTE FAILED

---

**ROTATE INSTRUCTIONS W/ MODE 2**

**TST117:**
INC  (R2) ; UPDATE TEST NUMBER
CMP  #17676,(R0) ; BR TO ERROR HALT ON SEQ ERROR
BNE  TST120-10 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CLR  R0 ; CONDITIONAL BRANCH INST. AND
POINT TO LOC 0 ; WHICH follows W/ 763
MOV  #173737,(R0) ; REPLACE THE MOVE INSTRUCTION
CLC  ; MODE 2
ROL  (R0)+ ; INCORRECT
ROLB  W/ MODE 2
BCC  ROTA ; CHECK C-BIT
BNE  ROTA ; CHECK DATA
DEC  R0 ; BRANCH IF RESULT INCORRECT
DEC  R0 ; TEST RD
BEQ  ROTA ;

---

**ROT2:**
MOV  #204,-(R2) ; MOVE TO MAILBOX 204
INC  -(R2) ; SET MSGTYP TO FATAL ERROR
HALT ; ROLB W/ MODE 2 FAILED

---

**ROT2B:**
CLR  R0 ; POINT TO LOC 0
MOV  #1040,(R0) ; INITIALIZE DATA
CLC  ; CLEAR C-BIT
ROLB  (R0)+ ; TRY ROLB W/ MODE 2 EVEN BYTE
BCS  ROTA ; CHECK C-BIT
CMP  #1100,2#0 ; CHECK DATA
BNE  ROTA ; BRANCH IF DATA INCORRECT
DEC  R0 ; CHECK RD
BEQ  ROTA ;

---

**ROT2C:**
MOV  #205,-(R2) ; MOVE TO MAILBOX 205
INC -(R2) : SET MSGTP TO FATAL ERROR
HALT : ROLB W/ MODE 2 EVEN BYTE FAILED

SET C-BIT
POINT TO LOC D
INITIALIZE DATA
POINT TO ODD BYTE OF DATA

ROTB (R0)+
TRY ROL W/ MODE 2 ODD BYTE
CHECK C-BIT
CHECK DATA
CHECK DATA
CHECK C-BIT

BRANCH IF DATA INCORRECT
CHECK RD

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 721

MOVE TO MAILBOX

MOV #206, -(R2) : MOVE TO MAILBOX

INC -(R2) : SET MSGTP TO FATAL ERROR
ROLB W/ MODE 2 ODD BYTE FAILED
HALT
OR SEQUENCE ERROR
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 120 TEST ROTATE INSTRUCTIONS /W MODE 3

TST120: INC (R2) ; UPDATE TEST NUMBER
          CMP #120, (R2) ; SEQUENCE ERROR?
          BNE TST121-10 ; BR TO ERROR HALT ON SEQ ERROR
          MOV #52525, R0 ; INITIALIZE DATA IN LOC 0
          SEC
          ROL 300 ; TRY ROL W/ MODE 3
          BCS ROT3A ; CHECK C-BIT
          BEQ . ROT3B ; CHECK DATA
          TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 765

          MOV #207, -(R2) ; MOVE TO MAILBOX # 207
          INC -(R2) ; SET MSGTYTP TO FATAL ERROR
          HALT W/ MODE 3 FAILED
          ROT3A:
          MOV #12525, R0 ; INITIALIZE DATA
          CLC
          ROL 300 ; TRY ROL W/ MODE 3 EVEN BYTE
          BCC ROT3C ; CHECK C-BIT
          BEQ ROT3D ; CHECK DATA
          TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 746

          MOV #210, -(R2) ; MOVE TO MAILBOX # 210
          INC -(R2) ; SET MSGTYTP TO FATAL ERROR
          HALT W/ MODE 3 EVEN BYTE FAILED
          ROT3C:
          MOV #12525, R0 ; INITIALIZE DATA IN LOC. 0
          SEC
          ROLB #1 ; TRY ROL W/ MODE 3 ODD BYTE
          BCC ROT3E ; CHECK C-BIT
          BEQ TST121 ; CHECK DATA
          TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 727

          MOV #211, -(R2) ; MOVE TO MAILBOX # 211
          INC -(R2) ; SET MSGTYTP TO FATAL ERROR
          HALT W/ MODE 3 ODD BYTE FAILED
3494

; OR SEQUENCE ERROR
************

**THIS TEST VERIFIES MODE 4 ROTATE INSTRUCTIONS. THE DATA IS STORED IN LOC. 0. RO IS SET TO 2 AND THE CARRY IS SET. AN ROL MODE 4 IS USED TO ROTATE LOCATION 0 USING RO. THE DATA IS CHECKED AND THE C AND V BITS ARE TESTED. THE PROPER DECREMENTING OF RO IS VERIFIED.**

**TEST 121: TEST MODE 4 W/ ROTATE INSTRUCTIONS**

**TST121:** INC (R2), UPDATA TEST NUMBER
CMP #121, (R2), SEQUENCE ERROR?
BNE TST122-10, BR TO ERROR HALT ON SEQ ERROR
MOV #00707, #0, INITIALIZE DATA IN LOC. 0
MOV #2, RO, INITIALIZE RO AS PCINTER
SET C-REG
ROL -(RO), TRY ROL W/ MODE 4
BCS ROT4, CHECK C-BIT
CMP #16167, #0, CHECK DATA
BNE ROT4, BRANCH IF DATA INCORRECT
TST RO, CHECK MODE 4 REGISTER
BEQ TST122, TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND WHICH FOLLOWS W/ 762

**ROT4:**
MOV #212, -(R2), MOVE TO MAILBOX
INC -(R2), SET MSGTYP TO FATAL ERROR
HALT, ROL MODE 4 FAILED; OR SEQUENCE ERROR

************

**TEST 122: TEST MODE 5 W/ ROTATE INSTRUCTIONS**

**TST122:** INC (R2), UPDATA TEST NUMBER
CMP #122, (R2), SEQUENCE ERROR?
BNE ROT5, BR TO ERROR HALT ON SEQ ERROR
MOV #ROTX, #0, MOVE POINTER TO LOC. 0
MOV #2, RD, SET MODE 5 REG. TO LOC. 0
MOV #10707, ROTX, INITIALIZE DATA
CLC, CLEAR C-BIT
ROL 2-(RO), TRY ROL W/ MODE 5
BCC ROT5, CHECK C-BIT

************
NO6

MAINDEC-11-DGKAA 11 04 CPU TEST
DGKAA.P11 T122 TEST MODE 5 W/ ROTATE INSTRUCTIONS

3551 007664 022737 016160 007710
3552 007672 001002
3553 007674 005700
3554 007676 001405
3555
3556
3557
3558
3559 007700 012742 000213
3560 007706 005242
3561 007706 000000
3562
3563 007710 000000

ROTX: 0

ROTS:

MOV #213,-(R2) ;MOVE TO MAILBOX ** 213 **
INC -(R2) ;SET MSGTYC TO FATAL ERROR
HALT ;ROL MODE 6 FAILED

; OR SEQUENCE ERROR

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH follows w/ 757

; 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 123: TEST NO6 W/ ROTATE INSTRUCTIONS
******************************************************************************

TST123: INC (R2) ;UPDATE TEST NUMBER

CMP #123,(R2) ;SEQUENCE ERROR?
BNE TST124-10 ;BR TO ERROR HALT ON SEQ ERROR

MOV #125252,3#ROTX ;INITIALIZE DATA
SEC
ROL ROTX
BCC ROT6 ;CHECK B-ĐT
CMP #52525,3#ROTX ;CHECK DATA
BEQ TST124

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH follows w/ 765

ROTX: MOV #214,-(R2) ;MOVE TO MAILBOX ** 214 **
INC -(R2) ;SET MSGTYC TO FATAL ERROR
HALT ;ROL W/ MODE 6 FAILED

; OR SEQUENCE ERROR

; CONDITIONS
**TEST 124 TEST MODE 7 W/ ROTATE INSTRUCTIONS**

**TST124:** INC (R2) ; UPDATE TEST NUMBER

CMP #124, (R2) ; SEQUENCE ERROR
BNE ROT7 ; BR TO ERROR HALT ON SEQ ERROR
MOV #52525, ROX ; INITIALIZE DATA
MOV #ROX, #ROTX ; INITIALIZE ADDRESS POINTER
CLC
ROL #ROTXAD ; TRY ROL W/ MODE 7
BCE SRO7 ; CHECK C-BIT
CMP #ROTX, #125252 ; CHECK DATA
BEQ TST125 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

**ROT7:**

MOV #215, (R2) ; MOVE TO MAILBOX #... 215
INC -(R2) ; SET MSGTY9 TO FATAL ERROR
HALT ; ROL W/ MODE 7 FAILED

**ROTXAD:** D

---

**TEST 125 TEST MODE 0 W/ SWAB INST.**

**TST125:** INC (R2) ; UPDATE TEST NUMBER

CMP #125, (R2) ; SEQUENCE ERROR
BNE TST126 ; BR TO ERROR HALT ON SEQ ERROR
MOV #177400, RO ; MOVE TEST PATTERN TO RO
SWAB RO ; TRY SWAB MODE 0
BMI SBO ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

**SBO:**

MOV #216, -(R2) ; MOVE TO MAILBOX #... 216
INC -(R2) ; SET MSGTY9 TO FATAL ERROR
HALT ; SWAB DID NOT SET CC'S CORRECT
; CHECK RESULT

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

MOVE TO MAILBOX & 217
SET MSGTYP TO FATAL ERROR
RESULT OF SWAB MODE C FAILED
OR SEQUENCE ERROR
**THIS TEST VERIFIES MODE 1 SWAB INSTRUCTION. THE TEST PATTERN IS MOVED TO LOC 0. RD IS CLEARED AND USED AS THE ADDRESSING REGISTER IN THE MODE 1 SWAB. THE DATA RESULTS ARE CHECKED WITH A COMPARE.**

**TEST 126 TEST MODE 1 W/ SWAB INST**

**TST126: INC (R2) UPDATE TEST NUMBER\n CMP $126, (R2) SEQUENCE ERROR?\n BNE TST127-10 BR TO ERROR HALT ON SEQ ERROR\n MOV #125652, W INCREASE TEST PATTERN TO LOC. 0\n CLR RD RD=0\n SWAB (RD) TRY SWAB MODE 1\n CMP #125253, W CHECK RESULT\n BEQ TST127 TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 767\n INC -(R2) SET MSGTYP TO Fatal ERROR\n HALT RESULT OF SWAB MODE 1 FAILED OR SEQUENCE ERROR

**THIS TEST VERIFIES MODE 2 SWAB INSTRUCTION. THE TEST PATTERN IS MOVED TO LOC 0. RD IS CLEARED AND USED AS THE MODE 2 ADDRESSING REGISTER. THE RESULTS ARE CHECKED WITH A COMPARE. RD IS CHECKED FOR PROPER DECREMENTING.**

**TEST 127 TEST MODE 2 W/ SWAB INST**

**TST127: INC (R2) UPDATE TEST NUMBER\n CMP $127, (R2) SEQUENCE ERROR?\n BNE TST130-10 BR TO ERROR HALT ON SEQ ERROR\n MOV #125152, W INCREASE TEST PATTERN TO LOC. 0\n CLR RD RD=0\n SWAB (RD) TRY SWAB MODE 2\n CMP #125253, W CHECK RESULT\n BEQ SB2 TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 767\n INC -(R2) SET MSGTYP TO Fatal ERROR\n HALT RESULT OF SWAB MODE 0 FAILED OR SEQUENCE ERROR
**E07**

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

MOV  #222, -(R2)  MOVE TO MAILBOX & ********  222 ********
INC  -(R2)  SET MSGTYP TO FATAL ERROR
HALT  REGISTER VALUE INCORRECT

**********************************************************************************************

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.

**********************************************************************************************

<table>
<thead>
<tr>
<th>TEST 130</th>
<th>TEST MODE 3 W/SWAB INST.</th>
</tr>
</thead>
</table>

TST130: INC (R2)  UPDATE TEST NUMBER
CMP #130, (R2)  SEQUENCE ERROR?
BNE TST131-10  BR TO ERROR HALT ON SEQ ERROR
MOV #377, #0  MOVE TEST PATTERN TO LOC. 3
SWAB #0  TRY SWAB W/ MODE 3
CMP #1777400, #0  CHECK RESULT
BEQ TST131

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

MOV  #223, -(R2)  MOVE TO MAILBOX & ********  223 ********
INC  -(R2)  SET MSGTYP TO FATAL ERROR
HALT  RESULT OF SWAB INCORRECT

OR SEQUENCE ERROR
```
THIS TEST VERIFIES MODE 4 SWAB INSTRUCTIONS. THE DATA IS MOVED TO LOC 0. RD IS SET TO 2 AND USED AS THE MODE 4 ADDRESSING REGISTER. THE DATA IS CHECKED WITH A COMPARE AND RD IS CHECKED FOR PROPER DECREMENTING.

**TEST 131**
**TEST MODE 4 W/ SWAB INST**

**TST131: INC (R2); UPDATE TEST NUMBER**
**CMP #131,(R2); SEQUENCE ERROR?**
**BNE TST132-10; BR TO ERROR HALT ON SEQ ERROR**
**MOV #125652, R0; MOVE TEST PATTERN TO LOC. 0**
**MOV #2, R0; SET UP REGISTER POINTER**
**SHRD -(R0); TRY SWAB MODE 4**
**CMP #125253, R0; CHECK RESULT**
**BEQ SB4; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**
**... CONDITIONAL BRANCH INST. AND**
**... REPLACE THE MOVE INSTRUCTION**
**... WHICH follows @ 760**

**SB4: INC -(R2); MOVE TO MAILBOX @ 224**
**HALT; SET MSGTYP TO FATAL ERROR**
**RESULT OF SWAB INCORRECT**
**BEQ TST132; CHECK EFFECT ON REG.**

**TST132: INC (R2); TO SCOPE: CLEAR THE RIGHT BYTE OF THIS**
**MOV #225, -(R2); CONDITIONAL BRANCH INST. AND**
**HALT; REPLACE THE MOVE INSTRUCTION**
**RESULT OF SWAB INCORRECT**
**OR SEQUENCE ERROR**
THIS TEST VERIFIES MODE 5 SWAB INSTRUCTION. THE TEST USES
TWO LOCATIONS FOLLOWING THE TEST CODE. SBsxH HOLDS THE DATA:
SBsxAD IS A POINTER TO THE DATA LOCATION. THE DATA IS MOVED TO
SBsx AND RO IS SET TO TWO PLUS THE ADDRESS OF SBsxAD FOLLOWING
THE MODE 5 SWAB SBsx IS CHECKED FOR THE PROPER DATA. RO IS
CHECKED TO SEE THAT IT WAS DECREMENTED PROPERLY.

TEST 132 TEST MODE 5 W/ SWAB INST.

TST132: INC (R2); UPDATE TEST NUMBER
          CMP #132,(R2); SEQUENCE ERROR?
          BNE SB5; BR TO ERROR HALT ON SEQ ERROR
          MOV #SBsxAD+2, RO; SET UP POINTER TO WORK LOCATION
          MOV #125125, SBsx; MOVE PATTERN TO WORK LOCATION
          SWAB a-(RO); TRY SWAB MODE 5
          CMP #52652, SBsx; CHECK RESULT
          BEQ SB5A; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
                                    (====)
                                    CONDITIONAL BRANCH INST. AND (====)
                                    WHICH FOLLOW W/ 766 (====)
                                    MOVE TO MAILBOX #226 (====)
                                    SET MSGTYP TO FATAL ERROR
                                    RESULT OF SWAB INCORRECT
                                    CHECK RESULT OF REG.
          MOV #226,-(R2); TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
          INC -(R2); (====)
          HALT; REPLACE THE MOVE INSTRUCTION (====)
          SB5A: CMP RO,#SBsxAD; WHICH FOLLOW W/ 757 (====)
          BEQ TST133; MOVE TO MAILBOX #227 (====)
          SBS: MOV #227,-(R2); SET MSGTYP TO FATAL ERROR
          INC -(R2); REGISTER VALUE INCORRECT
          HALT; OR SEQUENCE ERROR
          SBsx: 0; WORK LOCATION
          SBsxAD: SBsx
THIS TEST VERIFIES MODE 6 SWAP INSTRUCTION. THIS TEST USES A WORK LOCATION (SBBX) FOLLOWING THE TEST CODE. TEST DATA IS LOADED INTO THE WORK LOCATION. RD, THE ADDRESSING REGISTER IS LOADED WITH 6 LESS THAN THE ADDRESS OF THE WORK LOCATION. THE MODE 6 SWAP IS EXECUTED WITH A +6 OFFSET. THE DATA IS VERIFIED WITH A COMPARE.

TEST 123 TEST MODE 6 W/ SWAP INST.

TST133: INC (R2) ;UPDATE TEST NUMBER
         CMP #133,(R2) ;SEQUENCE ERROR?
         BNE SBB6 ;BR TO ERROR HALT ON SEQ ERROR
         MOV #25125,SBBX ;MOVE PATTERN TO WORK LOCATION
         MOV #SBBX-6,RO ;MOVE OFFSET POINTER TO RO
         SWAB 6(RO) ;TRY SWAP W/ MODE 6
         CMP #52562.6(RO) ;CHECK RESULT
         BEQ TST134

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

SBB6: MOV #230,-(R2) ;MOVE TO MAILBOX
       INC -(R2) ;SET MSGTYPE TO FATAL ERROR
       HLT ;RESULT OF SWAP INCORRECT
         ; OR SEQUENCE ERROR
       SBBX: 0 ;WORK LOCATION
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 TO THE WORK LOCATION. RD 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.

I07

**ST134: TEST MODE 7 W/ SWAB INST.**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC (R2)</td>
<td>Update test number</td>
</tr>
<tr>
<td>CMP #134,(R2)</td>
<td>Sequence error?</td>
</tr>
<tr>
<td>BNE SB7</td>
<td>BR to error halt on seq error</td>
</tr>
<tr>
<td>MOV #177400 SB7X</td>
<td>Move pattern to work location</td>
</tr>
<tr>
<td>MOV #SB7XAD-72,RO</td>
<td>Move offset pointer to RD</td>
</tr>
<tr>
<td>SWAB @72(RO)</td>
<td>Try swab mode 7</td>
</tr>
<tr>
<td>CMP @72(RO),#377</td>
<td>Check results</td>
</tr>
<tr>
<td>BEQ ST135</td>
<td></td>
</tr>
</tbody>
</table>

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

**SB7:**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV #231,-(R2)</td>
<td>Move to mailbox</td>
</tr>
<tr>
<td>INC -(R2)</td>
<td>Set msgtyp to fatal error</td>
</tr>
<tr>
<td>HALT</td>
<td>Result of swab incorrect or sequence error</td>
</tr>
<tr>
<td>WORK LOCATION</td>
<td></td>
</tr>
<tr>
<td>SB7AD: SB7X</td>
<td>Pointer to work location</td>
</tr>
</tbody>
</table>

- JMP MODE 1
- JMP MODE 2
- JMP MODE 3
- 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 BLOCK HAS 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 JMP20 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 SET UP 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 WHICH ERROR OCCURRED. IF THE SEQUENCE IS CORRECT THEN THE ERROR DETECTED WAS A MODE FAILURE (E.G. FAILURE OF THE REGISTER TO BE INCREMENTED IN MODE 2 JUMP.)

This page contains a test sequence for the JMP instruction in various modes. The test is designed to verify the correctness of JMP operations across different modes. The sequence involves comparing results of previous operations, ensuring that jumps occur according to the programmed sequence, and using an internal sequence test (JMPSEQ) to verify this. Each block of code is labeled to indicate the mode being executed, and a simple procedure is followed, ensuring that any register changes are verified before proceeding to the next block.
MOVE TO MAILBOX # 233

SET MSGTYP TO FATAL ERROR

SHOULD BE HERE FROM JMP MODE 2 ONLY

POINT RO TO INDIRECT JMP ADDR.

UPDATE SEQUENCE CHECKER

TRY JMP MODE 3

ADDRESS INDIRECT JMP

CHECK THAT JMAPs ARE IN SEQUENCE: JMPSEQ=0?

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS (==)

CONDITIONAL BRANCH INST. AND (==)

REPLACE THE MOVE INSTRUCTION (==)

WHICH FOLLOWS W/ 743 (==)

MOVE TO MAILBOX # 234

SET MSGTYP TO FATAL ERROR

SHOULD BE HERE FROM JMP MODE 1 ONLY

UPDATE SEQUENCE CHECKER

SET RO-JUMP TARGET

TRY A JUMP MODE 2 TO "JMP3"

CHECK RESULT OF REGISTER IN MODE 3 JUMP

MOVE TO MAILBOX # 235

SET MSGTYP TO FATAL ERROR

REGISTER VALUE AFTER MODE 3 JUMP INCORRECT

CHECK JUMP SEQUENCE: JMPSEQ=E2?

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS (==)

CONDITIONAL BRANCH INST. AND (==)

REPLACE THE MOVE INSTRUCTION (==)

WHICH FOLLOWS W/ 727 (==)

MOVE TO MAILBOX # 236

SET MSGTYP TO FATAL ERROR

SHOULD BE ONLY FROM MODE 3 JUMP

SET UP POINTER TO JUMP TARGET

UPDATE SEQUENCE CHECKER

TRY JUMP MODE 4 TO "JMP4"

CHECK THAT JMAPs ARE IN SEQUENCE: JMPSEQ=4?

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS (==)

CONDITIONAL BRANCH INST. AND (==)

REPLACE THE MOVE INSTRUCTION (==)

WHICH FOLLOWS W/ 717 (==)

MOVE TO MAILBOX # 237

SET MSGTYP TO FATAL ERROR

SHOULD BE HERE ONLY FROM MODE 5 JUMP

SET UP OFFSET POINTER TO JUMP TARGET

UPDATE JUMP SEQUENCE

TRY MODE 6 & JUMP

CHECK THAT JMAPs ARE IN SEQUENCE: JMPSEQ=3?
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 664

MOVE TO MAILBOX # ****** 240 ******
SET MSGTYP TO FATAL ERROR
SHOULD ONLY BE HERE FROM MODE 4 JUMP
SET UP POINTER TO INDIRECT JUMP ADDR.
UPDATE JUMP SEQUENCE
TRY JUMP MODE 5 TO "JMP6"
INDIRECT ADDRESS POINTER

; CHECK JUMPS IN SEQUENCE: JMPSEQ=5?

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 646

MOVE TO MAILBOX # ****** 241 ******
SET MSGTYP TO FATAL ERROR
SHOULD ONLY BE HERE FROM MODE 6 JUMP
SET UP OFFSET POINTER TO INDIRECT ADDR.
UPDATE JUMP SEQUENCE
TRY MODE 7 JUMP
INDIRECT ADDRESS

; CHECK JUMPS IN SEQUENCE: JMPSEQ
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 or mode 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. RI 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 136: TEST JSR INSTRUCTION W/ ALL MODES
***************************************************************************

TST136: INC #3136,(R2); UPDATE TEST NUMBER
        CMP #136,(R2); SEQUENCE ERROR?
        BNE JSR1; BR TO ERROR HALT ON SEQ ERROR
        JSR0: JMP #03136
        JSR1: MOV #STB0T,R6; SET STACK POINTER
              MOV #JSR2,R0; SET TARGET ADDRESS
        JSR1A: MOV #243,-(R2); MOVE TO MAILBOX # 243
               INC -(R2); SET MSGTYE TO FATAL ERROR
               HALT; JSR MODE 1 FAILED
        JSR3: CMP #8300001,JSRSEQ; CHECK SEQUENCE, JSRSEQ=1?
              BNE JSR3A; BRANCH IF OUT OF SEQUENCE
              CMP R1,#JSR4; PROPER PC SAVED?
              BNE JSR3A; BRANCH IF SP WRONG
              CMP #STB0T-2,R6; STACK POINTER DECREMENTED?
              BNE JSR3A; BRANCH IF REG NOT SAVED
              CMP #JSR3+2,R0; MODE 2 INCREMENT CORRECT?
              BEQ JSR3B; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
                             CONDITIONAL BRANCH INST. AND
              JSR3A: MOV #244,-(R2); MOVE TO MAILBOX # 244
                    INC -(R2); REPLACE THE MOVE INSTRUCTION
                    JSR3B: SET MSGTYE TO FATAL ERROR

***************************************************************************
UPDATE SEQUENCE CHECKER

TRY JSR MODE 6

CHECK SEQUENCE: JSRSEQ=3

BRANCH IF OUT OF SEQUENCE

PROPER PC SAVED?

BRANCH IF PC WRONG

CHECK MODE 4 REGISTER

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

CONDITIONAL BRANCH INST. AND

REPLACE THE MOVE INSTRUCTION

WHICH FOLLOWS W/ 623

MOVE TO MAILBOX 250

SET MSGTPY TO ERROR

JSR MODE 4 MALFUNCTIONED

UPDATE SEQUENCE CHECKER

POINT RD TO TARGET ADDRESS

TRY JSR MODE 5

CHECK SEQUENCE: JSRSEQ=5

BRANCH IF OUT OF SEQUENCE

PROPER PC SAVED?

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

CONDITIONAL BRANCH INST. AND

REPLACE THE MOVE INSTRUCTION

WHICH FOLLOWS W/ 603

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

CONDITIONAL BRANCH INST. AND

REPLACE THE MOVE INSTRUCTION

WHICH FOLLOWS W/ 603

MOVE TO MAILBOX 251

SET MSGTPY TO ERROR

JSR MODE 5 FAILED

UPDATE SEQUENCE CHECKER

TRY JSR MODE 7

MODE 5 TARGET ADDRESS

MODE 7 TARGET ADDRESS

SEQUENCE CHECKER

CHECK SEQUENCE: JSRSEQ=5

BRANCH IF OUT OF SEQUENCE

PROPER PC SAVED?

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS

CONDITIONAL BRANCH INST. AND

REPLACE THE MOVE INSTRUCTION

WHICH FOLLOWS W/ 561

MOVE TO MAILBOX 252

SET MSGTPY TO ERROR

JSR MODE 7 MALFUNCTIONED

OR SEQUENCE ERROR
THIS TEST VERIFIES THE RTS INSTRUCTION. THE STACK POINTER
IS INITIALIZED AND A TEST PATTERN STORED ON STACK. RO IS LOADED
WITH RETURN ADDRESS. AN RTS IS EXECUTED, AND, AT THE TARGET
ADDRESS, A CHECK IS MADE THAT RO WAS PROPERLY RESTORED FROM THE
STACK.

************************************************************

**TEST 137 TEST RTS INSTRUCTION

**************************************************************

TEST 137: INC (R2) ; UPDATE TEST NUMBER

CMP #137, (R2) ; SEQUENCE ERROR?

BNE TS1140-10 ; BR TO ERROR HALT ON SEQ ERROR

MOV #51807, R6 ; INITIALIZE STACK POINTER

MOV #55256, -(R6) ; INITIALIZE TOP OF STACK

MOV #RTS1, R0 ; INITIALIZE RETURN REGISTER

RTS R0 ; TRY RTS THROUGH RD

; TO SCOPE: REPLACE THE MOVE INSTRUCTION (====
; FOLLOWING W/ 770) (====

MOV #253, -(R2) ; MOVE TO MAILBOX # ***** 253 *****

INC (R2) ; SET MSGTP TO FATAL ERROR

HALT ; RTS FAILED

RTS1: CMP #55256, R0 ; CHECK THAT RO RESTORED FROM STACK

BEQ TS1140 ; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION (====
; WHICH Follows W/ 762) (====

MOV #254, -(R2) ; MOVE TO MAILBOX # ***** 254 *****

INC (R2) ; SET MSGTP TO FATAL ERROR

HALT ; RTS MALFUNCTIONED

; OR SEQUENCE ERROR
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 NO. TEST MOV INSTRUCTION

| ST140: INC (R2) | UPDATE TEST NUMBER |
| ST141: MOV #140,(R2) | MOV TO MAILBOX |
| ST143: CMP #140,(R2) | MOV TO MAILBOX |
| ST144: BNE TST141-10 | MOV TO MAILBOX |
| ST145: SCU | MOV TO MAILBOX |
| ST146: CLC | MOV TO MAILBOX |
| ST147: MOV #100000,RO | MOV TO MAILBOX |
| ST148: BLO MOV1 | MOV TO MAILBOX |
| ST149: BVS MOV1 | MOV TO MAILBOX |
| ST150: BMI MOV2 | MOV TO MAILBOX |
| MOV: MOV #255,-(R2) | MOV TO MAILBOX |
| INC -(R2) | MOV TO MAILBOX |
| HALT | MOV TO MAILBOX |
| MOV2: SCC | MOV TO MAILBOX |
| CLZ | MOV TO MAILBOX |
| MOV #0,RO | MOV TO MAILBOX |
| BHI MOV3 | MOV TO MAILBOX |
| BVS MOV3 | MOV TO MAILBOX |
| BPL TST141 | MOV TO MAILBOX |
| MOV3: MOV #256,-(R2) | MOV TO MAILBOX |
| INC -(R2) | MOV TO MAILBOX |
| HALT | MOV TO MAILBOX |
| MOV4: TST141: INC (R2) | MOV TO MAILBOX |
| ;UPDATE TEST NUMBER | MOV TO MAILBOX |
431b 0011274 0002772 000140 0011274
432b 0112750 001024 00112750
433b 0112720 012700 100001
434b 0112760 0027077
435b 0112750 002510 100000
436b 0112720 012700 100000
437b 0112760 101402
438b 0112750 102401
439b 0112720 100401
43ab 0112760 100404
43bc 0112750 002772
43bd 0112720 012742 000257
43be 012000 005242
43bf 012000 000000
43c0 012004 000227
43c1 012006 001024
43c2 012010 032700 077776
43c3 012014 101002
43c4 012016 102401
43c5 012020 100004
43c6 012022 012742 000260
43c7 012026 005242
43c8 012030 000000
43c9 012034 005212
43ca 012038 002712
43cbeb 000142
43cbe 012042 012700 177777
43cfc 012046 0027077
43d0 012050 0027077
43d1 012052 012700 077777
43d2 012056 012700 077777
43d3 012060 102401
43d4 012062 102401
43d5 012064 102401
43d6 012064 012742 000261
43d7 012064 000277
43d8 012064 012742 000261
43d9 012064 000277

**SEQUENCE ERROR**

**BR TO ERROR HALT ON SEG ERROR**

**SCC**

**CC=0110**

**CC=1000**

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

**SET MSGTYP TO FATAL ERROR**

**BIT DID NOT SET CC'S CORRECTLY**

**SET MSGTYP TO FATAL ERROR**

**BIT DID NOT SET CC'S CORRECTLY**

**OR SEQUENCE ERROR**

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

**SET MSGTYP TO FATAL ERROR**

**BIT DID NOT SET CC'S CORRECTLY**

**OR SEQUENCE ERROR**

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

**SET MSGTYP TO FATAL ERROR**

**BIT DID NOT SET CC'S CORRECTLY**

**OR SEQUENCE ERROR**
CLZ
BIC #100000,RO ;CC=0101

BIC3:

MOV #262 -(R2) ;MOVE TO MAILBOX  ***  262  ******
INC -(R2)
HALT

BIC1:

MOV #263 -(R2) ;MOVE TO MAILBOX  ***  263  ******
INC -(R2)
HALT

BIC2:

MOV #17777,RO ;CC=1001

BIC3:

MOV #264 -(R2) ;MOVE TO MAILBOX  ***  264  ******
INC -(R2)
HALT

--------------------------------------------------------------------------

TEST 144: TEST INC INSTRUCTION
--------------------------------------------------------------------------

TS144: INC (R2)

; UPDATE TEST NUMBER
CMP #144, (R2)
; SEQUENCE ERROR?
BNE TST145-10
; BR TO ERROR HALT ON SEQ ERROR
MOV #077777, R0
; RO=077777
; CC=0100
INC R0
; CC=1010 RO=10000
BLS INC1
BPL INC1
BVS INC2

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 7777

INC1: MOV #265 (R2)
; MOVE TO MAILBOX # 265
INC -(R2)
; SET MSGTYP TO FATAL ERROR
HALT
; INC DIOD NOT SET CC'S CORRECTLY
INC
; CC=0101 RO=0

INC2: BIS #777777, R0
; RO=177777
SEC
; CC=1011
CLZ
INC R0
; CC=0101 RC=0
BMX 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/ 753

INC3: MOV #266 (R2)
; MOVE TO MAILBOX # 266
INC -(R2)
; SET MSGTYP TO FATAL ERROR
HALT
; INC DID NOT SET CC'S CORRECTLY
INC
; CC=1110

INC4: SCC
CLC
INC R0
; CC=0000 RO=1
BLS INC5
BMI INC5
BPL TST145
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 746

MOV #267, -(R2)
INC -(R2)
HALT

MOVE TO MAILBOX & ********** 267 ********
SET MSGTYP TO FATAL ERROR
INC DID NOT SET CC'S CORRECTLY
OR SEQUENCE ERROR

POSTSC: TEST 145
TEST DEC INSTRUCTION

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

INC -(R2)
CMP #145-(R2)
BNE TST146-10
BR TO ERROR HALT ON SEQ ERROR
RD=2
SCC
DEC RO
BMI DEC1
BEQ DEC1
BVS DEC1
BCS DEC2

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

MOV #270, -(R2)
INC -(R2)
HALT
DEC1

MOVE TO MAILBOX & ********** 270 ********
SET MSGTYP TO FATAL ERROR
DEC DID NOT SET CC'S CORRECTLY
DEC2

SECC
CLF
DEC RO
BMI DEC3
BVC DEC4

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

MOV #271, -(R2)
INC -(R2)
HALT
DEC3

MOVE TO MAILBOX & ********** 271 ********
SET MSGTYP TO FATAL ERROR
DEC DID NOT SET CC'S CORRECTLY
DEC4

SCC
+CLN: CLC
DEC RO
BMI DEC6

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 744
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01244</td>
<td>MOV  #273,-(R2)  MOVE TO MAILBOX # 273</td>
</tr>
<tr>
<td>01247</td>
<td>INC  -.R2)  SET MSGTP TO FATAL ERROR</td>
</tr>
<tr>
<td>01247</td>
<td>HALT  DEC DID NOT SET CC'S CORRECTLY</td>
</tr>
<tr>
<td></td>
<td>RG=100000  OR SEQUENCE ERROR</td>
</tr>
<tr>
<td>01248</td>
<td>GCC=1011  DEC DID NOT SET CC'S CORRECTLY</td>
</tr>
<tr>
<td>01255</td>
<td>CC=0011  OR SEQUENCE ERROR</td>
</tr>
<tr>
<td>01262</td>
<td>GSC  TO SCOPE: CLEAR THE RIGHT BYTE OF THIS</td>
</tr>
<tr>
<td></td>
<td>100403    CONDITIONAL BRANCH INST. AND</td>
</tr>
<tr>
<td>01262</td>
<td>BSI  REPLACE THE MOVE INSTRUCTION</td>
</tr>
<tr>
<td></td>
<td>102001    WHICH FOLLOWS WITH 727</td>
</tr>
<tr>
<td>01262</td>
<td>BCV  $E=газ$</td>
</tr>
<tr>
<td></td>
<td>103404    $E=газ$</td>
</tr>
<tr>
<td>01265</td>
<td>DEC7:</td>
</tr>
<tr>
<td>01265</td>
<td>MOV  #273,-(R2)  MOVE TO MAILBOX # 273</td>
</tr>
<tr>
<td>01268</td>
<td>INC  -.R2)  SET MSGTP TO FATAL ERROR</td>
</tr>
<tr>
<td>01268</td>
<td>HALT  DEC DID NOT SET CC'S CORRECTLY</td>
</tr>
<tr>
<td>01270</td>
<td>RG=100000  OR SEQUENCE ERROR</td>
</tr>
<tr>
<td></td>
<td>GCC=1011  DEC DID NOT SET CC'S CORRECTLY</td>
</tr>
<tr>
<td>01274</td>
<td>CC=0011  OR SEQUENCE ERROR</td>
</tr>
<tr>
<td>01274</td>
<td>GSC  TO SCOPE: CLEAR THE RIGHT BYTE OF THIS</td>
</tr>
<tr>
<td></td>
<td>000273    CONDITIONAL BRANCH INST. AND</td>
</tr>
<tr>
<td>01277</td>
<td>BSI  REPLACE THE MOVE INSTRUCTION</td>
</tr>
<tr>
<td></td>
<td>000273    WHICH FOLLOWS WITH 727</td>
</tr>
<tr>
<td>01277</td>
<td>BCV  $E=газ$</td>
</tr>
<tr>
<td></td>
<td>000000    $E=газ$</td>
</tr>
</tbody>
</table>

------------------------------------------------------------------------------------------------------

**TEST 146**  TEST CLR INSTRUCTION

------------------------------------------------------------------------------------------------------

**TST146**: INC (R2): UPDATE TEST NUMBER

CMP #146, (R2): SEQUENCE ERROR?

BNE TST147-10: BR TO ERROR HALT ON SEQ ERROR

SCC: CC=1011

CLZ

BMI CLR1: CC=0100

BVS CLR1

BCS CLR1

BEQ TST147

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

CLR1.

MOV #274, -(R2): MOVE TO MAILBOX # 274

INC -(R2): SET MSGTYP TO FATAL ERROR

HALT: CLR DID NOT SET CC'S CORRECTLY OR SEQUENCE ERROR

------------------------------------------------------------------------------------------------------

**TEST 147**  TEST TST INSTRUCTION

------------------------------------------------------------------------------------------------------

**TST147**: INC (R2): UPDATE TEST NUMBER

CMP #147, (R2): SEQUENCE ERROR?

BNE TST150-10: BR TO ERROR HALT ON SEQ ERROR

SCC: CC=1011

CLZ

TST RO: CC=0100

BMI TEST1

BVS TEST1

BCS TEST1

BEQ TEST2

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

------------------------------------------------------------------------------------------------------

**TEST1**: MOV #275, -(R2): MOVE TO MAILBOX # 275

INC -(R2): SET MSGTYP TO FATAL ERROR

HALT: TEST DID NOT SET CC'S CORRECTLY

------------------------------------------------------------------------------------------------------

**TEST2**: DEC RO: MAKE RO NEGATIVE

SCC: CC=0111
; TEST3: MOV #276, -(R2) ; MOVE TO MAILBOX # 276
INC -(R2) ; SET MSGTYP TO FATAL ERROR
HALT ; TEST DID NOT SET CF'S CORRECTLY

; OR SEQUENCE ERROR

;*****************************
; TEST 150 TEST SWAB INSTRUCTION
;*****************************

; TST150: INC (R2) ; UPDATE TEST NUMBER
; CMP #150, (R2) ; SEQUENCE ERROR?
; BNE TST150, (R2) ; BR TO ERROR HALT ON SEQ ERROR
; MOV #170000, R0 ; R0 = 170000
; INC ; CC = 0111
; CLN
; SWAB R0 ; CC = 1000
; R0 = 360

; SWB1: MOV #277, -(R2) ; MOVE TO MAILBOX # 277
; INC -(R2) ; SET MSGTYP TO FATAL ERROR
; HALT ; SWAB DID NOT SET CC'S CORRECTLY

; SWB2: SCC ; CC = 1011
; CLZ
; SWAB R0
; CC = 0100
; R0 = 170000

; SWB3: MOV #300, -(R2) ; MOVE TO MAILBOX # 300
; INC -(R2) ; SET MSGTYP TO FATAL ERROR
; HALT

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 776

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

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 755
**THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE ADD AND ANC 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 151: TEST ADD INSTRUCTION**

**XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
MOVE TO MAILBOX **** 303 ********
SET MSGTYP TO FATAL ERROR
ADD 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/ 732

MOVE TO MAILBOX **** 304 ********
SET MSGTYP TO FATAL ERROR
ADD 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/ 716

MOVE TO MAILBOX **** 305 ********
SET MSGTYP TO FATAL ERROR
ADD DID NOT SET CC'S CORRECTLY
OR SEQUENCE ERROR

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

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

MOVE TO MAILBOX **** 306 ********
SET MSGTYP TO FATAL ERROR
ADD DID NOT SET CC'S CORRECTLY
ADC: R0
BHI ADC3
BVS ADC3
BPL ADC4

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

MOV #307,-(R2)
MOV TO MAILBOX # 307
SET MSGTYP TO FATAL ERROR
ADC DID NOT SET CC'S CORRECTLY

CC=0101
RO=0

ADC: R0
INC -(R2)
SET MSGTYP TO FATAL ERROR
ADC DID NOT SET CC'S CORRECTLY

CC=0101
RO=0

ADC: R0
HALT

CC=0101
RO=0

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

MOV #310,-(R2)
MOV TO MAILBOX # 310
SET MSGTYP TO FATAL ERROR
ADC DID NOT SET CC'S CORRECTLY

CC=0101
RO=0

HALT
OR SEQUENCE ERROR

**TEST 153: TEST NEG INSTRUCTION**

|013206| 005212| 000153 |
|013210| 027712| 000001 |
|013214| 001032| 000000 |
|013216| 012700| 000001 |
|013218| 000277| 000000 |
|013226| 005400| 000000 |
|013230| 103003| 000000 |
|013232| 012402| 000000 |
|013234| 001401| 000000 |
|013236| 100404| 000000 |

**ST153: INC (R2) UPDATE TEST NUMBER**

CMP #153,(R2) SEQUENCE ERROR?
BNE TST154-10 BR TO ERROR HALT ON SEQ ERROR
MOV #1,RO ;CC=0110
SCC
+CLN+CLC
NEG RO ;CC=1001 RD=77777
BCC NEG1
BVS NEG1
BEO NEG1
BMI NEG2

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

**NEG1: MOV #311,-(R2) MOVE TO MAILBOX 8 311**

INC -(R2) SET MSGTYP TO FATAL ERROR
HALT NEG DID NOT SET CC'S CORRECTLY

|013240| 012742| 00311 |
|013242| 005242| 000000 |
|013244| 000000| 000000 |
|013246| 042700| 077777 |
|013248| 000257| 000254 |
|013250| 005400| 000000 |
|013252| 003000| 000000 |
|013254| 01401| 000000 |
|013256| 100404| 000000 |

**NEG2: BIC #77777,R0**

CCC
SL
NEG RO ;CC=0100
BVC NEG3
BCC NEG3
BEO NEG3
BMI NEG4

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

**NEG3: MOV #312,-(R2) MOVE TO MAILBOX 8 312**

INC -(R2) SET MSGTYP TO FATAL ERROR
HALT NEG DID NOT SET CC'S CORRECTLY

|013272| 012742| 00312 |
|013274| 005242| 000000 |
|013276| 000000| 000000 |
|013278| 00277| 000000 |
|013280| 00242| 000000 |
|013282| 005400| 000000 |
|013284| 005400| 000000 |
|013286| 00277| 000000 |
|013288| 00242| 000000 |
|013290| 00242| 000000 |
|013292| 00242| 000000 |
|013294| 00242| 000000 |
|013296| 00242| 000000 |
|013298| 00242| 000000 |
|013300| 001021| 000000 |

**NEG4: CLC RO**

SCC
CLZ
NEG RO ;CC=1011
BVS NEG5
BCC NEG5
BEO NEG5
BMI NEG5

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

**NEG4: MOV #312,-(R2) MOVE TO MAILBOX 8 312**

INC -(R2) SET MSGTYP TO FATAL ERROR
HALT NEG DID NOT SET CC'S CORRECTLY

|013312| 102403| 000000 |
|013314| 103402| 000000 |
|013316| 001021| 000000 |
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 736

; MOV #313,-(R2)
; MOVE TO MAILBOX & ******** 313 *******

; INC - R2!
; SET MSGTTYP TO FATAL ERROR
; NEG DID NOT SET CC'S CORRECTLY
; OR SEQUENCE ERROR

; EGS:

; TEST 154 TEST CMP INSTRUCTION

; TEST 154: INC (R2) ; UPDATE TEST NUMBER
; CMP #154-(R2) ; SEQUENCE ERROR?
; BNE TST155-10 ; OR TO ERROR HALT ON SEQ ERROR
; MOV #5,RO
; CCC=1010
; CCC=0101

; CMP #5,RO
; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 767

; CMP1:
; MOV #314,-(R2)
; MOVE TO MAILBOX & ******** 314 *******
; INC -(R2)
; SET MSGTTYP TO FATAL ERROR
; HALT
; CMP DID NOT SET CC'S CORRECTLY

; CMP2:
; MOV #100000,RO
; CCC=1101
; CCC=0010

; CMP3:
; MOV #315,-(R2)
; MOVE TO MAILBOX & ******** 315 *******
; INC -(R2)
; SET MSGTTYP TO FATAL ERROR
; HALT
; CMP DID NOT SET CC'S CORRECTLY

; CMP4:
; BIS #40000,RO
; RD=140000
; CCC=0100
; CCC=1011

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CMPS:  MOV  #316-(R2)
       INC  -(R2)
HALT
       CMP  #40000.RO  #1.RO
CC=1111
       CMP  #0000
CC=0000
       TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       CONDITIONAL BRANCH INST. AND
       REPLACE THE MOVE INSTRUCTION
       WHICH FOLLOWS W/ 720

CMPS:  MOV  #317-(R2)
       INC  -(R2)
HALT
       CMP  #40000.RO  #1.RO
CC=1111
       CMP  #0000
CC=0000
       TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       CONDITIONAL BRANCH INST. AND
       REPLACE THE MOVE INSTRUCTION
       WHICH FOLLOWS W/ 720

; TEST 155  TEST COM INSTRUCTION
;********************************************************************
;ST155: INC  (R2)  ;UPDATE TEST NUMBER
CMP  #155-(R2)  ;SEQUENCE ERROR?
BNE  TST156-10  ;BR TO ERROR HALT ON SEG ERROR
MOV  #1.RO
CC=1010
+SEC!SEZ
COM  RO
BHI  COM1
BVS  COM1
BPL  TST156
CC=0101

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

COM1:  MOV  #320-(R2)
       INC  -(R2)
HALT
       CMP  #40000.RO  #1.RO
CC=1111
       CMP  #0000
CC=0000
       TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
       CONDITIONAL BRANCH INST. AND
       REPLACE THE MOVE INSTRUCTION
       WHICH FOLLOWS W/ 770
       OR SEQUENCE ERROR
These next two tests verify the functioning of the SUB and SBC instructions. Both of these instructions handle the C and V bits differently. 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 156: Test SUB Instruction**

ST156: INC (R2) ; Update test number
      CMP #156, (R2) ; Sequence error?
      BNE TST157-10 ; BR to error halt on seq error
      MOV #125252, RC
      ; CC=1010
      +SEM!SEC
      SUB #125252, R0 ; CC=0101 R0=0

      SUB1:
      MOV #312, -(R2) ; Move to mailbox # 312
      INC -(R2) ; Set msgtyp to fatal error
      HALT
      ; SUB2:
      BIS #100000, R0
      ; CC=1101
      SVC
      BLO SUB3
      BVC SUB3
      BPL SUB4

      SUB3:
      MOV #322, -(R2) ; Move to mailbox # 322
      INC -(R2) ; Set msgtyp to fatal error
      HALT
      ; SUB4:
      COM R0
      ; RO=177777
      SCC
      ; CC=111111
      SVC
      BLO SUB5
      BVC SUB5
      BPL SUB6

      ; TO SCOPE: Clear the right byte of this
      ; conditional branch inst. and
      ; replace the move instruction
      ; which follows w/ 767

      SUB1:
      MOV #312, -(R2) ; Move to mailbox # 312
      INC -(R2) ; Set msgtyp to fatal error
      HALT
      ; SUB2:
      BIS #100000, R0
      ; CC=1101
      SVC
      BLO SUB3
      BVC SUB3
      BPL SUB4

      SUB3:
      MOV #322, -(R2) ; Move to mailbox # 322
      INC -(R2) ; Set msgtyp to fatal error
      HALT
      ; SUB4:
      COM R0
      ; RO=177777
      SCC
      ; CC=111111
      SVC
      BLO SUB5
      BVC SUB5
      BPL SUB6

      ; TO SCOPE: Clear the right byte of this
      ; conditional branch inst. and
REPLACE THE MOVE INSTRUCTION

WHICH FOLLOWS W/ 737

SUB5:
MOV #323,-(R2)
INC -(R2)
HALT

SUB6:
SUB #140000,RO
BVC SUB7
BCE SUB7
BMI TST157

SUB7:
MOV #324,-(R2)
INC -(R2)
SET MSIGTMP TO FATAL ERROR
HALT

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 733

TST157: INC (R2)
CMP #157,(R2)
BNE TST160-10
BR TO ERROR HALT ON SEQ ERROR
MOV #1,R0
SSC CC=1011
CLZ SBC RO
BBS SBC1
BVC SBC1
BMI SBC1
BEQ SBC2

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

SBC1:
MOV #325,-(R2)
INC -(R2)
SET MSIGTMP TO FATAL ERROR
HALT

SBC2:
SSC CC=1010
+CLZ!CLC
SBC RO
BBS SBC3
BVC SBC3
BMI SBC3
BEQ SBC4

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

SBC3:
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5062</td>
<td>MOVC #326,-(R2)</td>
<td>MOVE TO MAILBOX # 326</td>
</tr>
<tr>
<td>5063</td>
<td>INC -(R2)</td>
<td>SET MSGTYP TO FATAL ERROR</td>
</tr>
<tr>
<td>5064</td>
<td>HALT</td>
<td>SBC DID NOT SET CC'S CORRECTLY</td>
</tr>
<tr>
<td>5065</td>
<td>SBC4:</td>
<td>CC=0111</td>
</tr>
<tr>
<td>5066</td>
<td>SBC5:</td>
<td>CC=1001, R0=177777</td>
</tr>
<tr>
<td>5067</td>
<td>MOV #327,-(R2);</td>
<td>MOVE TO MAILBOX # 327</td>
</tr>
<tr>
<td>5068</td>
<td>INC -(R2)</td>
<td>SET MSGTYP TO FATAL ERROR</td>
</tr>
<tr>
<td>5069</td>
<td>HALT</td>
<td>SBC DID NOT SET CC'S CORRECTLY</td>
</tr>
<tr>
<td>5070</td>
<td>INC #77777, R0</td>
<td>RO=100000</td>
</tr>
<tr>
<td>5071</td>
<td>INC #1010</td>
<td>CC=1101</td>
</tr>
<tr>
<td>5072</td>
<td>MOV #330,-(R2)</td>
<td>MOVE TO MAILBOX # 330</td>
</tr>
<tr>
<td>5073</td>
<td>INC -(R2)</td>
<td>SET MSGTYP TO FATAL ERROR</td>
</tr>
<tr>
<td>5074</td>
<td>HALT</td>
<td>SBC DID NOT SET CC'S CORRECTLY</td>
</tr>
<tr>
<td>5075</td>
<td>OR SEQUENCE ERROR</td>
<td></td>
</tr>
</tbody>
</table>
 THESE NEXT FOUR TESTS VERIFY THE FUNCTIONING OF THE ROL, ROR, ASL AND LSR INSTRUCTIONS. SPECIFIC 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 COMPARATIVE DATA RESULT. THE DATA PATTERNS HAVE BEEN SELECTED TO PRODUCE ALL COMBINATIONS OF THE C AND V BITS.

******************************************************************************

TEST 160 TEST ROL INSTRUCTION
******************************************************************************

ST160: INC (R2) ;UPDATE TEST NUMBER
CMP #160,(R2) ;SEQUENCE ERROR?
BNE ST161-10 ;BR TO ERROR HALT ON SEQ ERROR
MOV #144000,RO ;RD=144000
CCC ;CC=0110
+SEZ!SEV
ROL RO ;CC=100 1 RD=100000
BCC ROL1
BVS ROL1
BEQ ROL2
BMI ROL2

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

ROL1: MOV #331-(R2) ;MOVE TO MAILBOX 
INC -(R2) ;SET MSBTYP TO FATAL ERROR
HALT

ROL2: SCC ;CC=1100
+CLV!CLC
ROL RO ;CC=0011 RD=020000
BCC ROL3
BVC ROL3
BEQ ROL3
BPL ROL4

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

ROL3: MOV #332-(R2) ;MOVE TO MAILBOX 
INC -(R2) ;SET MSBTYP TO FATAL ERROR
HALT
ROL DID NOT SET CC'S CORRECTLY

ROL4: SCC ;CC=0111
CLN
ROL RO ;CC=0000 RD=040001
BLOE ROL5
BVC ROL5
BPL ROL6

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

ROR3:

MOV #336,-(R2)
INC -(R2)
HALT

ROR4:

SCC
INC 
ROR RO
BLS ROR5
BVS ROR5
BEQ ROR5
BPL ROR5

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

ROR5:

MOV #337,-(R2)
INC -(R2)
HALT

ROR6:

CCC
*SEC!SEZ
ROR RO
BLS ROR6
BVC ROR6
BMI TST162

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

ROR7:

MOV #340,-(R2)
INC -(R2)
HALT

; TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
; CONDITIONAL BRANCH INST. AND
; REPLACE THE MOVE INSTRUCTION
; WHICH FOLLOWS W/ 727

******* 340 *******

; MOVE TO MAILBOX 
; SET MSGTYP TO FATAL ERROR
; ROR DID NOT PRODUCE CORRECT RESULTS
; OR SEQUENCE ERROR

; TEST 162

TST162: INC (R2)
CMP #162,(R2)
BNE TST163-10
BRT TO ERROR HALT ON SEQ ERROR
MOV #14000,RO
RO=14000
CCC
CC=0110

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

; MOVE TO MAILBOX 
; SET MSGTYP TO FATAL ERROR
; ROR DID NOT PRODUCE CORRECT RESULTS
; OR SEQUENCE ERROR

; TEST ASL INSTRUCTION

; UPDATE TEST NUMBER

; SEQUENCE ERROR

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

; MOVE TO MAILBOX 
; SET MSGTYP TO FATAL ERROR
; ROR DID NOT PRODUCE CORRECT RESULTS
; OR SEQUENCE ERROR
K09

; INDEC-11-0G0AA 11 ON CPU TEST
; MACYII 27-30 06-OCT-76 14:01 PAGE 273

014440 005242  INC   -(R2) ; SET MSGTYP TO FATAL ERROR
014442 000000  HALT

; ASL2: SCC
; +CLV!CLC
; ASL  R0
; BCC  ASL3
; BEQ  ASL3
; BPL  ASL4

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

014462 012742 000342  MOV   #342,-(R2) ; MOVE TO MAILBOX # ******** 342 ******
014462 006242  INC   -(R2) ; SET MSGTYP TO FATAL ERROR
014466 000000  HALT

; ASL4: SCC
; +CLC
; ASL  R0
; BLO  ASL5
; BVS  ASL5
; BPL  ASL6

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

014506 012742 000343  MOV   #343,-(R2) ; MOVE TO MAILBOX # ******** 343 ******
014506 006242  INC   -(R2) ; SET MSGTYP TO FATAL ERROR
014510 000000  HALT

; ASL5: +SEZ!SEC
; ASL  R0
; BCS  ASL7
; BEQ  ASL7
; BVC  ASL7
; BPL  ASL7

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

014652 006242  INC   -(R2) ; SET MSGTYP TO FATAL ERROR
014656 000000  HALT

; ASL7: MOV   #344,-(R2) ; MOVE TO MAILBOX # ******** 344 ******
014540 001404  CMP   #100000,R0
014544 012742 000344  INC   -(R2) ; SET MSGTYP TO FATAL ERROR
014548 000000  HALT

; BEQ  TST163

; ASL MALFUNCTIONED
; OR SEQUENCE ERROR

; **************************************************************************
; TEST 163: TEST ASR INSTRUCTION
; **************************************************************************

; TST163: INC   (R2) ; UPDATE TEST NUMBER
014562 000163  CMP   #163,(R2) ; SEQUENCE ERROR?
014564 002712 000163
014566 000160  BNE   TST164-10 ; BR TO ERROR HALT ON SEQ ERROR
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 767

MOV #345, -(R2)
SET MSGTYP TO FATAL ERROR

MOV TO MAILBOX # 345

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

MOV #346, -(R2)
SET MSGTYP TO FATAL ERROR

MOV TO MAILBOX # 346

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 740

MOV #347, -(R2)
SET MSGTYP TO FATAL ERROR

MOV TO MAILBOX # 347

TO SCOPE: CLEAR THE RIGHT BYTE OF THIS CONDITIONAL BRANCH INST. AND REPLACE THE MOVE INSTRUCTION WHICH FOLLOWS W/ 737

MOV #100023, RO
RO=100023
CC=0110

RO=140011

RC=020004

RO=010002

RO=014001
TO SCOPE: CLEAR THE RIGHT BYTE OF THIS
CONDITIONAL BRANCH INST. AND
REPLACE THE MOVE INSTRUCTION
WHICH FOLLOWS W/ 720

MOVE TO MAILBOX # ****** 350 ******
SET MSGTYP TO FATAL ERROR
ASR DID NOT FUNCTION CORRECTLY
OR SEQUENCE ERROR
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 O.

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 164: TEST THE BRANCH ROM**

```
TST164: INC (R2)  ; UPDATE TEST NUMBER
        CMP $164,(HL)  ; CHECK ERROR CONDITION
        JRZ $0000, R1  ; BR TO ERROR HANDLE ON SEQ ERROR

SETUP: MOV #BRTAB,R0  ; INITIALIZE BRANCH TABLE POINTER
        MOV #YNTAB,R4  ; INITIALIZE YES/NO BRANCH MAP POINTER

SETBR: MOV (R0)+,R4  ; GET NEXT BRANCH INST.
        MOV (R4)+,R4  ; GET NEXT BRANCH MAP
        MOV $1,CC      ; INITIALIZE CONDITION CODE VALUE
        MOV $16,R3     ; INITIALIZE CONDITION CODE COUNT

SETCC: INC CC         ; SET FOR NEXT CC VALUE
        BIT $100000,R1  ; SEE IF SHOULD BR W/ THESE CC'S
        MOV $177777,R5  ; SIMULATE A JNE
        JMP #177773,R5  ; (JUMP NOT EQUAL)
                  ; TO SETBR

JMP SET2BR

MOV #CONT,NBR    ; SET TO CONTINUE IF NO BRANCH
MOV #ER,YBR     ; SET TO REPORT ERROR IF BRANCH
JMP AROUND      ; GO AROUND OPPOSITE CONDITION

SETBR: MOV #ER,NBR  ; SET TO REPORT ERROR IF NO BRANCH
MOV #CONT,YBR   ; SET TO CONTINUE IF BRANCH

AROUND: ROL R1     ; UPDATE BIT MAP
           mov (PC)+, (PC)+  ; SET CONDITION CODE
           cc: 0           ; NEW CC VALUE GOES HERE
           brh: 0          ; BRANCH INST. GOES HERE
           nhr: 0          ; WHERE TO GO IF NO BRANCH OCCURS
```
$45 015102 203137 YBR: JMP q(PC)+
$46 015103 000000 WHERE TO GO IF BRANCH OCCURS
$47 015104 012702 003334 ERI: MOV #TESTN,R2
$48 015105 012742 003351 ERV: MOV #351,-(R2)
$49 015106 003342 INC -(R2)
$50 015107 000000 HLT
$51 015108 005303 BRCT: 0
$52 015109 013705 177778 CCX: DEC R3
$53 01510A 02705 177773 MOV #177776,R5
$54 01510B 000165 015142 BIC #177773,R5
$55 01510C 015142 177773 JMP +4(R5)
$56 01510D 005367 17775C JMP SETCC
$57 01510E 013705 177776 DEC BRCT
$58 01510F 02705 177773 MOV #177776,R5
$59 015110 000165 015166 BIC #177773,R5
$5A 015111 177773 JMP +4(R5)
$5B 015112 00166 JMP SETBR
; TEST 165  END OF PASS SEQUENCE

;***********************************************************************

;TST165:  INC  (R2)  ; UPDATE TEST NUMBER
        CMP  #165,(R2)  ; SEQUENCE ERROR?
        BNE  EOPI     ; BR TO ERROR HALT ON SEQ ERR CR
        INC  #8888H  ;$PASS
        INCB PASSPT  ; SHOULD PRINT THIS PASS?
        BITB  #40,SEMVM  ; WILL APT ALLOW PRINTING?
        BNE  ACT      ; NO
        CMP  #42,#42AD  ; UNDER ACT AUTO ACCEPT?
        BEQ  ACT      ; IF SO SKIP PRINTOUT
        MOV  #MSG,RO  ; GET MSG ADDR.
        MOVB (RO)+,#TPB  ; PRINT CHARACTER
        BNE  WAIT     ; NEXT IF NOT DONE.
        MOV  #42,RO  ; CHECK ACT
        BEQ  GOAGAIN  ; KEEP GOING
        JSR  PC,(RO)  ; ACT HOOKS
        GOAGAIN:  JMP  RESTART  ; DO NEXT PASS
        EOI:  MOV  #352-(R2)  ; MOVE TO MAILBOX & 352
        INC  -(R2)  ; SET MSGYP TO FATAL ERROR
        HALT  ; SEQUENCE ERROR
        PASSPT:  -1
        MSG:  .ASCIZ  .END OF DGKAA.(15)(12)
<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>G10</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>J5RCA</td>
<td>011560</td>
</tr>
<tr>
<td>J5RCK1</td>
<td>011576</td>
</tr>
<tr>
<td>J5RCLK1</td>
<td>011582</td>
</tr>
<tr>
<td>J5RCLK1</td>
<td>011588</td>
</tr>
<tr>
<td>J5RCLK2</td>
<td>011594</td>
</tr>
<tr>
<td>J5RCLK3</td>
<td>011600</td>
</tr>
<tr>
<td>J5RCLK4</td>
<td>011606</td>
</tr>
<tr>
<td>J5RCLK5</td>
<td>011612</td>
</tr>
<tr>
<td>J5RCLK6</td>
<td>011618</td>
</tr>
<tr>
<td>J5RCLK7</td>
<td>011624</td>
</tr>
<tr>
<td>J5RCLK8</td>
<td>011630</td>
</tr>
<tr>
<td>J5RCLK9</td>
<td>011636</td>
</tr>
<tr>
<td>J5RCLK10</td>
<td>011642</td>
</tr>
<tr>
<td>J5RCLK11</td>
<td>011648</td>
</tr>
<tr>
<td>J5RCLK12</td>
<td>011654</td>
</tr>
<tr>
<td>PS</td>
<td>013200</td>
</tr>
<tr>
<td>PC</td>
<td>013216</td>
</tr>
<tr>
<td>REMS</td>
<td>013232</td>
</tr>
<tr>
<td>RST</td>
<td>013248</td>
</tr>
<tr>
<td>RET</td>
<td>013264</td>
</tr>
<tr>
<td>RTR</td>
<td>013280</td>
</tr>
<tr>
<td>ROL1</td>
<td>014200</td>
</tr>
<tr>
<td>ROL2</td>
<td>014216</td>
</tr>
<tr>
<td>ROL3</td>
<td>014232</td>
</tr>
<tr>
<td>ROL4</td>
<td>014248</td>
</tr>
<tr>
<td>ROL5</td>
<td>014264</td>
</tr>
<tr>
<td>ROL6</td>
<td>014280</td>
</tr>
<tr>
<td>ROL7</td>
<td>014296</td>
</tr>
<tr>
<td>ROL8</td>
<td>014312</td>
</tr>
<tr>
<td>ROL9</td>
<td>014328</td>
</tr>
<tr>
<td>ROL10</td>
<td>014344</td>
</tr>
<tr>
<td>ROL11</td>
<td>014360</td>
</tr>
<tr>
<td>ROL12</td>
<td>014376</td>
</tr>
<tr>
<td>ROL13</td>
<td>014392</td>
</tr>
<tr>
<td>ROL14</td>
<td>014408</td>
</tr>
<tr>
<td>ROL15</td>
<td>014414</td>
</tr>
<tr>
<td>ROL16</td>
<td>014430</td>
</tr>
<tr>
<td>ROL17</td>
<td>014446</td>
</tr>
<tr>
<td>ROL18</td>
<td>014462</td>
</tr>
<tr>
<td>ROL19</td>
<td>014478</td>
</tr>
<tr>
<td>ROL20</td>
<td>014494</td>
</tr>
<tr>
<td>ROL21</td>
<td>014510</td>
</tr>
<tr>
<td>ROL22</td>
<td>014526</td>
</tr>
<tr>
<td>ROL23</td>
<td>014542</td>
</tr>
<tr>
<td>ROL24</td>
<td>014558</td>
</tr>
<tr>
<td>ROL25</td>
<td>014574</td>
</tr>
<tr>
<td>ROL26</td>
<td>014590</td>
</tr>
<tr>
<td>ROL27</td>
<td>014606</td>
</tr>
<tr>
<td>ROL28</td>
<td>014622</td>
</tr>
<tr>
<td>ROL29</td>
<td>014638</td>
</tr>
<tr>
<td>ROL30</td>
<td>014654</td>
</tr>
</tbody>
</table>

Note: The values are in hexadecimal format.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>00436</td>
<td>2142</td>
<td>2157</td>
<td>2164</td>
</tr>
<tr>
<td>B</td>
<td>00420</td>
<td>2238</td>
<td>2239</td>
<td>2245</td>
</tr>
<tr>
<td>C</td>
<td>00416</td>
<td>2250</td>
<td>2258</td>
<td>2264</td>
</tr>
<tr>
<td>D</td>
<td>00416</td>
<td>2262</td>
<td>2263</td>
<td>2270</td>
</tr>
<tr>
<td>E</td>
<td>00416</td>
<td>2298</td>
<td>2299</td>
<td>2301</td>
</tr>
<tr>
<td>F</td>
<td>00416</td>
<td>2306</td>
<td>2307</td>
<td>2308</td>
</tr>
<tr>
<td>G</td>
<td>00416</td>
<td>2309</td>
<td>2318</td>
<td>2318</td>
</tr>
<tr>
<td>H</td>
<td>00416</td>
<td>2411</td>
<td>2412</td>
<td>2418</td>
</tr>
<tr>
<td>I</td>
<td>00500</td>
<td>2454</td>
<td>2455</td>
<td>2461</td>
</tr>
<tr>
<td>J</td>
<td>00500</td>
<td>2466</td>
<td>2468</td>
<td>2504</td>
</tr>
<tr>
<td>K</td>
<td>00500</td>
<td>2499</td>
<td>2500</td>
<td>2546</td>
</tr>
<tr>
<td>L</td>
<td>00500</td>
<td>2539</td>
<td>2540</td>
<td>2540</td>
</tr>
<tr>
<td>M</td>
<td>00500</td>
<td>2661</td>
<td>2662</td>
<td>2662</td>
</tr>
<tr>
<td>N</td>
<td>00500</td>
<td>2670</td>
<td>2671</td>
<td>2671</td>
</tr>
<tr>
<td>O</td>
<td>00500</td>
<td>2697</td>
<td>2698</td>
<td>2698</td>
</tr>
<tr>
<td>P</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>Q</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>R</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>S</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>T</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>U</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>V</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>W</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>X</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>Y</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>Z</td>
<td>00500</td>
<td>2704</td>
<td>2705</td>
<td>2705</td>
</tr>
<tr>
<td>SOP6</td>
<td>003652</td>
<td>1995</td>
<td>1903*</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>SOP7</td>
<td>003716</td>
<td>1929</td>
<td>1935*</td>
<td></td>
</tr>
<tr>
<td>SOP8</td>
<td>003735</td>
<td>1936</td>
<td>1943*</td>
<td></td>
</tr>
<tr>
<td>SOP9</td>
<td>003776</td>
<td>1968</td>
<td>1976*</td>
<td></td>
</tr>
<tr>
<td>SOP10</td>
<td>004012</td>
<td>1977</td>
<td>1984*</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>START</th>
<th>000500</th>
<th>479</th>
<th>489*</th>
<th>5545</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY</td>
<td>000500</td>
<td>400</td>
<td>481</td>
<td>4976</td>
</tr>
<tr>
<td>SUB1</td>
<td>013604</td>
<td>4971</td>
<td>4972</td>
<td>4978*</td>
</tr>
<tr>
<td>SUB2</td>
<td>013614</td>
<td>4973</td>
<td>4982</td>
<td>4993*</td>
</tr>
<tr>
<td>SUB3</td>
<td>013636</td>
<td>4986</td>
<td>4997</td>
<td>5005*</td>
</tr>
<tr>
<td>SUB4</td>
<td>013646</td>
<td>4988</td>
<td>4997</td>
<td>5008*</td>
</tr>
<tr>
<td>SUB5</td>
<td>013684</td>
<td>5001</td>
<td>5002</td>
<td>5008*</td>
</tr>
<tr>
<td>SUB6</td>
<td>013717</td>
<td>5015</td>
<td>5016</td>
<td>5017*</td>
</tr>
<tr>
<td>SUB7</td>
<td>013717</td>
<td>5015</td>
<td>5016</td>
<td>5023*</td>
</tr>
<tr>
<td>TBL1</td>
<td>006536</td>
<td>4639</td>
<td>4640</td>
<td>4641*</td>
</tr>
<tr>
<td>TBL2</td>
<td>006622</td>
<td>3105</td>
<td>3106*</td>
<td>4641*</td>
</tr>
<tr>
<td>TBL3</td>
<td>006659</td>
<td>3130</td>
<td>3131</td>
<td>3137*</td>
</tr>
<tr>
<td>TBL4</td>
<td>006658</td>
<td>3247</td>
<td>3248</td>
<td>3244*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEST1</th>
<th>012560</th>
<th>4586</th>
<th>4587</th>
<th>4598*</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST2</td>
<td>012560</td>
<td>4589</td>
<td>4598*</td>
<td>4599*</td>
</tr>
<tr>
<td>TEST3</td>
<td>012606</td>
<td>4602</td>
<td>4603</td>
<td>4609*</td>
</tr>
<tr>
<td>TBL5</td>
<td>017566</td>
<td>408*</td>
<td>5478</td>
<td>5543*</td>
</tr>
<tr>
<td>TBL6</td>
<td>017564</td>
<td>407*</td>
<td>5476</td>
<td>5541*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TST1</th>
<th>000554</th>
<th>500*</th>
<th>682</th>
<th>598*</th>
</tr>
</thead>
<tbody>
<tr>
<td>TST10</td>
<td>001114</td>
<td>682</td>
<td>685</td>
<td>698*</td>
</tr>
<tr>
<td>TST11</td>
<td>001154</td>
<td>692</td>
<td>695</td>
<td>698*</td>
</tr>
<tr>
<td>TST12</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST13</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST14</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST15</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST16</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST17</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST18</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST19</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST20</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST21</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST22</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST23</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST24</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST25</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST26</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST27</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST28</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST29</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
<tr>
<td>TST30</td>
<td>001204</td>
<td>693</td>
<td>696</td>
<td>698*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>736</th>
<th>741</th>
<th>754</th>
</tr>
</thead>
<tbody>
<tr>
<td>3701</td>
<td>3715</td>
<td>3736</td>
</tr>
<tr>
<td>Symbol</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>TST134</td>
<td>010270</td>
<td></td>
</tr>
<tr>
<td>TST135</td>
<td>010290</td>
<td></td>
</tr>
<tr>
<td>TST136</td>
<td>010310</td>
<td></td>
</tr>
<tr>
<td>TST137</td>
<td>010330</td>
<td></td>
</tr>
<tr>
<td>TST140</td>
<td>010350</td>
<td></td>
</tr>
<tr>
<td>TST141</td>
<td>010370</td>
<td></td>
</tr>
<tr>
<td>TST142</td>
<td>010390</td>
<td></td>
</tr>
<tr>
<td>TST143</td>
<td>010410</td>
<td></td>
</tr>
<tr>
<td>TST144</td>
<td>010430</td>
<td></td>
</tr>
<tr>
<td>TST145</td>
<td>010450</td>
<td></td>
</tr>
<tr>
<td>TST146</td>
<td>010470</td>
<td></td>
</tr>
<tr>
<td>TST147</td>
<td>010490</td>
<td></td>
</tr>
<tr>
<td>TST150</td>
<td>010510</td>
<td></td>
</tr>
<tr>
<td>TST151</td>
<td>010530</td>
<td></td>
</tr>
<tr>
<td>TST152</td>
<td>010550</td>
<td></td>
</tr>
<tr>
<td>TST153</td>
<td>010570</td>
<td></td>
</tr>
<tr>
<td>TST154</td>
<td>010590</td>
<td></td>
</tr>
<tr>
<td>TST155</td>
<td>010610</td>
<td></td>
</tr>
<tr>
<td>TST156</td>
<td>010630</td>
<td></td>
</tr>
<tr>
<td>TST160</td>
<td>010650</td>
<td></td>
</tr>
<tr>
<td>TST161</td>
<td>010670</td>
<td></td>
</tr>
<tr>
<td>TST162</td>
<td>010690</td>
<td></td>
</tr>
<tr>
<td>TST163</td>
<td>010710</td>
<td></td>
</tr>
<tr>
<td>TST17</td>
<td>001424</td>
<td></td>
</tr>
<tr>
<td>TST18</td>
<td>001444</td>
<td></td>
</tr>
<tr>
<td>TST19</td>
<td>001464</td>
<td></td>
</tr>
<tr>
<td>TST20</td>
<td>001484</td>
<td></td>
</tr>
<tr>
<td>TST21</td>
<td>001504</td>
<td></td>
</tr>
<tr>
<td>TST22</td>
<td>001524</td>
<td></td>
</tr>
<tr>
<td>TST23</td>
<td>001544</td>
<td></td>
</tr>
<tr>
<td>TST24</td>
<td>001564</td>
<td></td>
</tr>
<tr>
<td>TST25</td>
<td>001584</td>
<td></td>
</tr>
<tr>
<td>TST26</td>
<td>001604</td>
<td></td>
</tr>
<tr>
<td>TST27</td>
<td>001624</td>
<td></td>
</tr>
<tr>
<td>TST28</td>
<td>001644</td>
<td></td>
</tr>
<tr>
<td>TST29</td>
<td>001664</td>
<td></td>
</tr>
<tr>
<td>TST30</td>
<td>001684</td>
<td></td>
</tr>
<tr>
<td>TST31</td>
<td>001704</td>
<td></td>
</tr>
<tr>
<td>TST32</td>
<td>001724</td>
<td></td>
</tr>
<tr>
<td>TST33</td>
<td>001744</td>
<td></td>
</tr>
<tr>
<td>TST34</td>
<td>001764</td>
<td></td>
</tr>
<tr>
<td>TST35</td>
<td>001784</td>
<td></td>
</tr>
<tr>
<td>TST36</td>
<td>001804</td>
<td></td>
</tr>
<tr>
<td>TST37</td>
<td>001824</td>
<td></td>
</tr>
<tr>
<td>TST38</td>
<td>001844</td>
<td></td>
</tr>
<tr>
<td>TST39</td>
<td>001864</td>
<td></td>
</tr>
<tr>
<td>TST40</td>
<td>001884</td>
<td></td>
</tr>
<tr>
<td>TST41</td>
<td>002004</td>
<td></td>
</tr>
<tr>
<td>TST42</td>
<td>002014</td>
<td></td>
</tr>
<tr>
<td>TST43</td>
<td>002024</td>
<td></td>
</tr>
<tr>
<td>Time (h)</td>
<td>45h</td>
<td>65h</td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Cross Reference Table -- Permanent Symbols

<table>
<thead>
<tr>
<th>SYMBOLS</th>
<th>DESCRIPTION</th>
<th>SYMBOLS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>L11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Note: The table is too detailed to transcribe fully here, but it provides a structured overview of various symbols that are likely used in a technical or scientific context.
<table>
<thead>
<tr>
<th>XActual</th>
<th>YActual</th>
<th>XTop</th>
<th>YTop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1295</td>
<td>1303</td>
<td>1313</td>
<td>1314</td>
</tr>
<tr>
<td>1408</td>
<td>1437</td>
<td>1453</td>
<td>1462</td>
</tr>
<tr>
<td>1703</td>
<td>1709</td>
<td>1742</td>
<td>1751</td>
</tr>
<tr>
<td>1935</td>
<td>1938</td>
<td>1969</td>
<td>1975</td>
</tr>
<tr>
<td>2256</td>
<td>2260</td>
<td>2289</td>
<td>2298</td>
</tr>
<tr>
<td>2580</td>
<td>2582</td>
<td>2613</td>
<td>2626</td>
</tr>
<tr>
<td>2952</td>
<td>2953</td>
<td>2984</td>
<td>2996</td>
</tr>
<tr>
<td>3257</td>
<td>3269</td>
<td>3305</td>
<td>3317</td>
</tr>
<tr>
<td>3518</td>
<td>3554</td>
<td>3586</td>
<td>3605</td>
</tr>
<tr>
<td>3807</td>
<td>3816</td>
<td>3844</td>
<td>3872</td>
</tr>
<tr>
<td>4159</td>
<td>4166</td>
<td>4175</td>
<td>4182</td>
</tr>
<tr>
<td>4556</td>
<td>4563</td>
<td>4588</td>
<td>4599</td>
</tr>
<tr>
<td>4714</td>
<td>4723</td>
<td>4729</td>
<td>4752</td>
</tr>
<tr>
<td>4966</td>
<td>4965</td>
<td>4991</td>
<td>5013</td>
</tr>
<tr>
<td>5214</td>
<td>5220</td>
<td>5229</td>
<td>5234</td>
</tr>
<tr>
<td>5359</td>
<td>5368</td>
<td>5397</td>
<td>5438</td>
</tr>
</tbody>
</table>

**IFF**

<table>
<thead>
<tr>
<th>417</th>
<th>419</th>
<th>511</th>
<th>523</th>
</tr>
</thead>
<tbody>
<tr>
<td>531</td>
<td>544</td>
<td>568</td>
<td>577</td>
</tr>
<tr>
<td>590</td>
<td>598</td>
<td>606</td>
<td>616</td>
</tr>
<tr>
<td>610</td>
<td>643</td>
<td>633</td>
<td>648</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>607</th>
<th>676</th>
<th>698</th>
<th>703</th>
</tr>
</thead>
<tbody>
<tr>
<td>876</td>
<td>885</td>
<td>921</td>
<td>930</td>
</tr>
<tr>
<td>974</td>
<td>974</td>
<td>974</td>
<td>974</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1191</th>
<th>1195</th>
<th>1206</th>
<th>1238</th>
</tr>
</thead>
<tbody>
<tr>
<td>1303</td>
<td>1305</td>
<td>1307</td>
<td>1309</td>
</tr>
<tr>
<td>1311</td>
<td>1313</td>
<td>1315</td>
<td>1317</td>
</tr>
<tr>
<td>1319</td>
<td>1321</td>
<td>1323</td>
<td>1325</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1524</th>
<th>1525</th>
<th>1526</th>
<th>1527</th>
</tr>
</thead>
<tbody>
<tr>
<td>1530</td>
<td>1532</td>
<td>1534</td>
<td>1536</td>
</tr>
<tr>
<td>1538</td>
<td>1540</td>
<td>1542</td>
<td>1544</td>
</tr>
<tr>
<td>1546</td>
<td>1548</td>
<td>1550</td>
<td>1552</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1796</th>
<th>1800</th>
<th>1804</th>
<th>1806</th>
</tr>
</thead>
<tbody>
<tr>
<td>1808</td>
<td>1810</td>
<td>1812</td>
<td>1814</td>
</tr>
<tr>
<td>1815</td>
<td>1818</td>
<td>1820</td>
<td>1822</td>
</tr>
<tr>
<td>1824</td>
<td>1826</td>
<td>1828</td>
<td>1830</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2106</th>
<th>2108</th>
<th>2109</th>
<th>2111</th>
</tr>
</thead>
<tbody>
<tr>
<td>2113</td>
<td>2115</td>
<td>2117</td>
<td>2119</td>
</tr>
<tr>
<td>2121</td>
<td>2123</td>
<td>2125</td>
<td>2127</td>
</tr>
<tr>
<td>2129</td>
<td>2131</td>
<td>2133</td>
<td>2135</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2690</th>
<th>2694</th>
<th>2697</th>
<th>2702</th>
</tr>
</thead>
<tbody>
<tr>
<td>2704</td>
<td>2707</td>
<td>2709</td>
<td>2712</td>
</tr>
<tr>
<td>2714</td>
<td>2717</td>
<td>2720</td>
<td>2723</td>
</tr>
<tr>
<td>2724</td>
<td>2727</td>
<td>2730</td>
<td>2733</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2877</th>
<th>2879</th>
<th>2882</th>
<th>2888</th>
</tr>
</thead>
<tbody>
<tr>
<td>2890</td>
<td>2893</td>
<td>2897</td>
<td>2902</td>
</tr>
<tr>
<td>2904</td>
<td>2908</td>
<td>2912</td>
<td>2916</td>
</tr>
<tr>
<td>2918</td>
<td>2922</td>
<td>2926</td>
<td>2930</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2805</th>
<th>2834</th>
<th>2869</th>
<th>2877</th>
</tr>
</thead>
<tbody>
<tr>
<td>2879</td>
<td>2909</td>
<td>2937</td>
<td>2945</td>
</tr>
<tr>
<td>2953</td>
<td>2986</td>
<td>3015</td>
<td>3023</td>
</tr>
<tr>
<td>3053</td>
<td>3080</td>
<td>3107</td>
<td>3141</td>
</tr>
<tr>
<td>3179</td>
<td>3217</td>
<td>3259</td>
<td>3297</td>
</tr>
<tr>
<td>3297</td>
<td>3323</td>
<td>3357</td>
<td>3384</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5328</th>
<th>5337</th>
<th>5344</th>
<th>5353</th>
</tr>
</thead>
<tbody>
<tr>
<td>5368</td>
<td>5397</td>
<td>5438</td>
<td>5478</td>
</tr>
<tr>
<td>577</td>
<td>577</td>
<td>577</td>
<td>577</td>
</tr>
</tbody>
</table>
ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

 run-time: 61 BY 12 SECONDS
run-time ratio: 527/158=3.3
core used: 33K  : 65 PAGES.