Russ Winslow's

1.        IDENTIFICATION                                    (7-54-m)

1.1      MAINDEC 701 Digital - 7- 54-M

1.2      PDP-7 Instruction Test

1.3      3/11/66


To check program operation from

upper 8K bank —

1.   Set   7-9 mem. sw. to 7, extend sw. off, TRAP OFF.

2.   READ  IN  "UPPER CORE INTERRUPT LINK"  AT LOC. 0.

3.   READ  IN  RIM LOADER IN UPPER 8K

4.   PUT  MAINDEC 701  IN READER, START  AT
     37777  WITH EXT. SW. DOWN,

5.   START  PGM  AT  20170 —

        WILL  RUN  UNTIL  HALT, MA=22561 and
        AC = 020000 — THIS IS NORMAL —
     ANY OTHER HALT  IS  ABNORMAL —

2. ABSTRACT

Instruction test is a sequence of fourteen programs which tests the operation of all PDP-7 instructions except the IOT group. Indirect addressing and automatic indexing are also checked. ADD, TAD, ISZ are checked with random numbers as well as noise patterns.

RIM Format.

The program tape is supplied ~~in Hardware Read-in format and should be loaded via the Read-in key. Hardware read-in is employed so that no reliance need be placed in untested instructions.~~

3. REQUIREMENTS

3.1 Storage

The program occupies _____ locations of core storage.

3.2 Subprograms and/or Subroutines

The entire Instruction test consists of 14 individual tests which are described fully in the following sections.

3.3 Equipment

Standard PDP-7

High Speed Reader and/or Teletype 33/35.

# 4. USAGE

## 4.1 Loading

1) Load MAINDEC 701 program tape into reader.

2) ~~Set all ADDRESS REGISTER switches (ADS) to the zero (down) state.~~  *RIM LOADEI 17770*

-3) ~~Depress and release the READ-IN key.~~

4) Program will read into core and computer will halt.

## 4.2 Calling Sequence     (Not Applicable)

## 4.3 Switch Settings

| Switch | Setting | Function |
|--------|---------|----------|
| ACSØ-17 | non-zero | A non-zero AC is essential in the second test (PR2) to the testing of the OAS instruction. |
| ACSØ | 1 | Causes current program to iterate. |
|  | Ø | Causes program to give way to the next in sequence. |
| ACS17 | 1 | Causes series of test programs to be repeated from the beginning automatically. |
|  | Ø | Causes a halt to be executed at ___2623___ signifying the end of the test series. Pressing CONTINUE causes the series to be repeated from the beginning. |

4.4     Start Up and/ or Entry

1.      Set the ACCUMULATOR SWITCH REGISTER (ACS) to some non-zero state.

2.      Set ACS∅ and ACS17 as desired (see para. 4.3)

3.      Set ADDRESS SWITCH REGISTER (ADS) to 17∅. Press START.

4.      Normally, four HLT instructions will be encountered in succession requiring  that

the operator press CONTINUE after examining the contents of the AC and LINK, subsequent

to each HLT.  Pressing CONTINUE after execution of the fourth HLT causes the computer

to begin cycling through the first test in the series.

        The code involved in this series of HLT's is as follows:

| | | |
|---|---|---|
| 170 | HLT | /Test of HLT instruction.  AC should = ∅. ~~LINK should = 1~~    L=0 |
| 171 | CMA!CML | /Change state of AC and LINK.   if a 1 after CML |
| 172 | HLT | /AC should = 777777.   LINK should = 1. |
| 173 | SPA | |
| 174 | CLA!CLL | |
| 175 | HLT | /AC and LINK should = ∅.  If AC=777777, SPA failed. |
| 176 | SPA | |
| 177 | HLT | /SPA failed. |
| 2∅∅ | OAS!HLT | /AC should = C(ACS). |

        . . . . . .

        . . . . . .

## 4.5   Errors In Usage

| Program | C(MA) | Cause and Remedial Action |
|---|---|---|
| PR13 | 0003 | Trap Mode Switch is on. Turn Trap Mode Switch off and press CONTINUE. |
| General | 0005 | A non-programmed interrupt has occurred. Press CONTINUE. |
| PR13 | 0020 | CAL instruction jumped to 20. Restart at loc. 2546 (PR13). |
| General | 0021 | A non-programmed CAL has occurred from the location addressed in register 20 less one. Re-examine your operating procedure and/or reload program. |
| PR13 | 0022-0027 | CAL instruction forced incorrect address to PC. Restart at loc 2546 (PR13). |
| PR13 | 0032 | Execution of CAL failed to preserve the LINK. Press CONTINUE to ignore error. Restart at loc 2546 (PR13) to repeat the CAL. |
| PR13 | 0036 | Execution of CAL failed to save the PC. Press CONTINUE to ignore error. Restart at loc 2546 (PR13) to repeat the CAL. |
| PR13 | 0042 | Execution of CAL failed to clear bits 1-4 of loc. 20. Press CONTINUE to ignore error. Restart at loc 2546 (PR13) to repeat the CAL. |
| PR0 | 0170 | This is the test for HLT. AC and LINK should = 0. If not, pressing START has failed to clear AC and LINK. Press CONTINUE if AC and LINK = 0. |

| Program | C(MA) | Cause and Remedial Action |
|---|---|---|
| PRØ | 172 | This is a preliminary test of CMA and CML. If AC $\neq$ 777777, CMA failed. If LINK $\neq$ 1, CML failed. In either case, restart at Ø17Ø. If AC = 777777 and LINK = 1, press CONTINUE. |
| PRØ | 175 | If AC and LINK = Ø, press CONTINUE. If AC = 777777, SPA failed; restart at 17Ø. |
| PRØ | 177 | SPA failed. Restart at 17Ø. |
| PRØ | 200 | Test for OAS. If AC = C(SR), press CONTINUE. If not, restart at 17Ø. |
| PRØ | 204 | JMP failed to jump. Restart at 2Ø3. |
| PRØ | 215 | ISZ skipped on negative result. Restart at 21Ø. |
| PRØ | 216-223 | ISZ forced 2,3,4,5,6 or 7 to the PC. Restart at 21Ø. |
| PRØ | 225 | ISZ failed to skip on Ø result. Restart at 21Ø. |
| PRØ | 227-234 | ISZ forced 2,3,4,5,6, or 7 to the PC. Restart at 21Ø. |
| PRØ | 237 | ISZ skipped on non-zero positive result. Restart at 21Ø. |
| PRØ | 24Ø-245 | ISZ forced 2,3,4,5,6 or 7 to the PC. Restart at 21Ø. |
| PR1 | 257 | NOP skipped. Restart at 253. |
| PR1 | 260-265 | NOP forced 2,3,4,5,6, or 7 to the PC. Restart at 253. |
| PR1 | 207 | SKP failed to skip. Restart at 253. |
| PR1 | 271-276 | SKP forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 302 | If LINK= 1, CLL failed. If LINK= Ø, SNL failed. Restart at 253. |
| PR1 | 303 - 31Ø | SNL forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 312 | SZL failed to skip. Restart at 253. |
| PR1 | 314 - 321 | SZL forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 324 | If LINK = 1, SNL failed. If LINK = Ø, CML failed. Restart at 253. |

| Program | C(MA) | Cause and Remedial Action |
|---------|-------|---------------------------|
| PR1 | 326-333 | SNL forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 336 | NOP skipped. Restart at 253. |
| PR1 | 337-344 | NOP forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 347 | If LINK = 1, CLL failed. If LINK = ∅, SZL failed. Restart at 253. |
| PR1 | 351 - 356 | SZL forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 362 | If LINK = 1, SZL failed. If LINK = ∅ , CML failed. Restart at 253. |
| PR1 | 363 - 37∅ | SZL forced 2,3,4,5,6, or 7 to the PC. Restart at 253. |
| PR1 | 373 | If LINK = 1, SNL failed. If LINK = ∅ , STL failed. Restart at 253. |
| PR1 | 375 - 4∅2 | SNL forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 4∅6 | If AC ≠ ∅, SZA failed. Otherwise, CLA failed. Restart at 253. |
| PR1 | 41∅ - 415 | SZA forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 42∅ | NOP skipped. Restart at 253. |
| PR1 | 421 - 426 | NOP forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 432 | If AC = ∅ , CMA failed . Otherwise, SZA failed. Restart at 253. |
| PR1 | 433 - 44∅ | SZA forced 2,3,4,5 6 or 7 to the PC. Restart at 253. |
| PR1 | 444 | If AC = ∅, SZA failed. Otherwise CLC and/or CMA failed. Restart at 253. |
| PR1 | 446 - 453 | SZA forced 1,2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 457 | IF AC = ∅, SZA failed. Otherwise CLA failed. Restart at 253. |
| PR1 | 461 - 466 | SZA forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 471 | If AC = ∅, CMA failed. Otherwise SMA failed. Restart at 253. |
| PR1 | 473 - 5∅∅ | SMA forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 503 | NOP skipped. Restart at 253. |
| PR1 | 5∅4 - 511 | NOP forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |

4.5    Errors in Usage        (continued)

| Program | | C(MA) | Cause and Remedial Action |
|---------|---|-------|---------------------------|
| PR1 | | 515 | If AC = ∅, SMA failed.  Otherwise CMA failed. Restart at 253. |
| PR1 | | 51∅ - 523 | SMA forced 2,3,4,5,6 or 7 to the PC.  Restart at 253. |
| PR1 | | 525 | SPA failed. Restart at 253. |
| PR1 | | 527 - 534 | SPA forced 2,3,4,5,6 or 7 to the PC.  Restart at 253. |
| PR1 | | 54∅ | If AC = ∅, CMA failed.  Otherwise, SPA failed. Restart at 253. |
| PR1 | | 541 - 546 | SPA forced 2,3,4,5,6 or 7 to the PC.  Restart at 253. |
| PR1 | | 551 | If AC = ∅, SZA failed.  Otherwise CLA failed. Restart at 253. |
| PR1 | | 553 - 56∅ | SZA forced 2,3,4,5,6 or 7 to the PC. Restart at 253. |
| PR1 | 564 | 564 | If AC = ∅, OAS failed. Otherwise SNA failed. (be sure that ACS hold non-zero number) Restart at 253 |
| PR1 | | 566-573 | SNA forced 2,3,4,5,6 or 7 to the PC.  Restart at 253. |
| PR2 | | 631 | JMS at 627 failed to jump.  Restart at 621. |
| PR2 | | 64∅ | JMS at 3673 failed to increment operand address (Y) in PC. Restart at 621. |
| PR2 | | 642 - 647 | JMS at 3673 incremented operand address (Y) 2,3,4,5,6 or 7 times.  Restart at 621. |
| PR2 | | 651 | JMS at 3673 failed to preserve LINK.  Restart at 621. |
| PR2 | | 654 | JMP I at 653 failed to jump.  Restart at 652. |

4.5    Errors in Usage          (continued)

| Program | C(MA) | Cause and Remedial Action |
| --- | --- | --- |
| PR3 | 662 | XOR of 777777 into 777777 failed. AC indicates bits in error. Restart at 657. |
| PR3 | 666 | XOR of 000000 into 000000 failed. AC indicates bits in error. Restart at 663. |
| PR3 | 673 | XOR of 777777 into 000000 failed. AC indicates bits in error. Restart at 667. |
| PR3 | 700 | XOR of 000000 into 777777 failed. AC indicates bits in error. Restart at 674. |
| PR3 | 706 | XOR of 070707 into 000000 into 707070 failed. AC indicates bits in error. Restart at 701. |
| PR3 | 714 | XOR of 707070 into 000000 into 070707 failed. AC indicates bits in error. Restart at 707. |
| PR3 | 722 | XOR of 333333 into 000000 into 444444 failed. AC indicates bits in error. Restart at 715. |
| PR3 | 730 | XOR of 444444 into 000000 into 333333 failed. AC indicates bits in error. Restart at 723. |
| PR3 | 736 | XOR of 525252 into 000000 into 252525 failed. AC indicates bits in error. Restart at 731. |
| PR3 | 744 | XOR of 252525 into 000000 into 525252 failed. AC indicates bits in error. Restart at 737. |
| PR4 | 757 | SAD failed to skip when comparing 000000 and 777777. Restart at 755. |
| PR4 | 761 | SAD failed to replace C(AC) which initially contained 000000. |

4.5 Errors In Usage   (continued)

| Program | C(MA) | Cause and Remedial Action |
|---|---|---|
| PR4 | 765 | SAD skipped when comparing equal numbers (777777). Restart at 762. |
| PR4 | 770 | SAD failed to replace C(AC) which initially contained 777777. Restart at 762. |
| PR4 | 774 | SAD failed to skip when comparing 525252 and 252525. Restart at 771. |
| PR4 | 1000 | SAD failed to replace C(AC) which initially contained 525252. Restart at 771. |
| PR4 | 1005 | SAD skipped when comparing equal numbers (525252). Restart at 1001. |
| PR4 | 1011 | SAD failed to replace C(AC) which initially contained 525252. |
| PR5 | 1025 | SNA failed on AC = 000000. |
| PR5 | 1030 | SNA failed with AC17 set.  Restart at 1022. |
| PR5 | 1033 | SNA failed with AC16 set.  Restart at 1022. |
| PR5 | 1036 | SNA failed with AC15 set.  Restart at 1022. |
| PR5 | 1041 | SNA failed with AC14 set.  Restart at 1022 |
| PR5 | 1044 | SNA failed with AC 13 set.  Restart at 1022 |
| PR5 | 1047 | SNA failed with AC 12 set.  Restart at 1022 |
| PR5 | 1052 | SNA failed with AC 11 set.  Restart at 1022. |
| PR5 | 1055 | SNA failed with AC 10 set.  Restart at 1022 |
| PR5 | 1060 | SNA failed with AC 9 set.  Restart at 1022 |
| PR5 | 1063 | SNA failed with AC8 set.  Restart at 1022 |
| PR5 | 1066 | SNA failed with AC 7 set.  Restart at 1022 |
| PR5 | 1071 | SNA failed with AC6 set.  Restart at 1022 |
| PR5 | 1074 | SNA failed with AC 5 set.  Restart at 1022 |
| PR5 | 1077 | SNA failed with AC 4 set.  Restart at 1022 |
| PR5 | 1102 | SNA failed with AC 3 set.  Restart at 1022 |
| PR5 | 1105 | SNA failed with AC 2 set.  Restart at 1022 |
| PR5 | 1110 | SNA failed with AC 1 set.  Restart at 1022 |
| PR5 | 1113 | SNA failed with AC 0 set.  Restart at 1022 |

| Program | C(MA) | Cause and Remedial Action |
|---------|-------|---------------------------|
| PR6 | 1127 | LAC of 000000 into 000000 failed. Restart at 1124. |
| PR6 | 1134 | LAC of 777777 into 000000 failed. Restart at 1130. |
| PR6 | 1141 | LAC of 777777 into 777777 failed. Restart at 1135. |
| PR6 | 1145 | LAC of 000000 into 777777 failed. Restart at 1142. |
| PR6 | 1153 | LAC of 252525 into 525252 failed. Restart at 1146. |
| PR6 | 1161 | LAC of 525252 into 252525 failed. Restart at 1154. |
| PR6 | 1167 | LAC of 070707 into 707070 failed. Restart at 1162. |
| PR6 | 1175 | LAC of 707070 into 070707 failed. Restart at 1170. |
| PR6 | 1203 | LAC of 444444 into 333333 failed. Restart at 1176. |
| PR6 | 1211 | LAC of 333333 into 444444 failed. Restart at 1204. |
| PR6 | 1216 | DAC of 000000 failed. Restart at 1212. |
| PR6 | 1223 | DAC of 777777 into 000000 failed. Restart at 1217. |
| PR6 | 1230 | DAC of 777777 into 777777 failed. Restart at 1224. |
| PR6 | 1235 | DAC of 000000 into 777777 failed. Restart at 1231. |
| PR6 | 1242 | DAC of 000000 into 000000 failed. Restart at 1236. |
| PR6 | 1250 | DAC of 525252 into 000000 failed. Restart at 1243. |
| PR6 | 1256 | DAC of 252525 into 525252 failed. Restart at 1251. |
| PR6 | 1264 | DAC of 525252 into 252525 failed. Restart at 1257. |
| PR6 | 1272 | DAC of 707070 into 525252 failed. Restart at 1265. |
| PR6 | 1300 | DAC of 070707 into 707070 failed. Restart at 1273. |
| PR6 | 1306 | DAC of 707070 into 070707 failed. Restart at 1301. |
| PR6 | 1314 | DAC of 333333 into 707070 failed. Restart at 1307. |
| PR6 | 1322 | DAC of 444444 into 333333 failed. Restart at 1315. |
| PR6 | 1330 | DAC of 333333 into 444444 failed. Restart at 1323. |

| Program | C(MA) | Cause and Remedial Action |
|---|---|---|
| PR7 | 1345 | DZM failed to clear the bits indicated in the AC. Restart at 1346. |
| PR7 | 1351 | DZM set the bits indicated in the AC. Restart at 1346. |
| PR7 | 1360 | LAW entered defer cycle. Restart at 1352. |
| PR7 | 1364 | LAW 0 failed. AC = result. Restart at 1352. |
| PR7 | 1373 | LAW entered defer cycle. Restart at 1365. |
| PR7 | 1377 | LAW 0 failed. AC = result. Restart at 1365. |
| PR7 | 1406 | LAW entered defer cycle. Restart at 1400. |
| PR7 | 1412 | LAW 17777 failed. AC = results. Restart at 1400. |
| PR7 | 1421 | LAW entered defer cycle. Restart at 1413. |
| PR7 | 1425 | LAW 17777 failed. AC = result. Restart at 1413. |
| PR7 | 1436 | LAC I of 777777 into 000000 failed. AC = result. Restart at 1426. |
| PR7 | 1446 | DAC I of 777777 into 000000 failed. AC = result. Restart at 1437. |
| PR7 | 1452 | DAC I altered operand register. Restart at 1437. |
| PR8 | 1512 | XCT failed to execute the one cycle CML. Restart at 1507. |
| PR8 | 1514 | XCT failed to execute the two cycle JMP. Restart at 1513. |
| PR8 | 1516 | XCT failed to execute JMP correctly. Restart at 1513. |
| PR8 | 1522 | XCT failed to execute the three cycle JMP I. Restart at 1517. |
| PR8 | 1524 | XCT failed to execute JMP I correctly. Restart at 1517. |
| PR8 | 1530 | XCT failed to execute on XCT. Restart at 1525. |
| PR8 | 1532 | XCT failed to execute XCT correctly. Restart at 1525. |
| PR8 | 1547 | LAC I via Auto Index 10 failed. Restart at 1544. |
| PR8 | 1553 | LAC I via Auto Index 11 failed. Restart at 1550. |
| PR8 | 1557 | LAC I via Auto Index 12 failed. Restart at 1554. |
| PR8 | 1563 | LAC I via Auto Index 13 failed. Restart at 1560. |
| PR8 | 1567 | LAC I via Auto Index 14 failed. Restart at 1564. |
| PR8 | 1573 | LAC I via Auto Index 15 failed. Restart at 1570. |
| PR8 | 1577 | LAC I via Auto Index 16 failed. Restart at 1574. |
| PR8 | 1603 | LAC I via Auto Index 17 failed. Restart at 1600. |

| Program | C(MA) | Cause and Remedial Action |
|---|---|---|
| PR9 | 1654 | AND of 777777 and 777777 failed. AC = result. Restart at 1650. |
| PR9 | 1660 | AND of 777777 and 000000 failed. AC = result. Restart at 1655. |
| PR9 | 1670 | AND of two identical numbers failed. Number concerned found at 4107. AC = result. Press CONTINUE or restart at 1671. |
| PR10 | 2074 | Rotate failed. Rotate instruction concerned in AC. Press CONTINUE for further data. |
| PR10 | 2077 | Rotate failed. AC and LINK display their conditions before the rotate. Press CONTINUE for further data. |
| PR10 | 2104 | Rotate failed. AC and LINK display their conditions after the rotate. Press CONTINUE to resume testing. |
| PR10 | 2135 | RAR followed by RAL left LINK altered. Number rotated in 4107. LINK was initially 0. Press CONTINUE or restart at 2134. |
| PR10 | 2140 | The number in 4107 did not survive a RAR, RAL. Result in AC. Press CONTINUE or restart at 2127. |
| PR10 | 2146 | RAR followed by RAL left LINK altered. Number rotated in 4107. LINK was initially 1. Press CONTINUE or restart at 2134. |
| PR10 | 2151 | The number in 4107 did not survive a RAR, RAL. Result in AC. Press CONTINUE or restart at 2127. |
| PR10 | 2157 | RAR failed to move AC right. Press CONTINUE. |
| PR10 | 2163 | RAL failed to move AC left. Press CONTINUE. |

| Program | C(MA) | Cause and Remedial Action |
|---------|-------|---------------------------|
| PR11 | 2201 | ADD failed on XOR function (707070 + 070707). Restart at 2174 or press CONTINUE. |
| PR11 | 2207 | ADD failed on XOR function (070707 + 707070). Restart at 2202 or press CONTINUE. |
| PR11 | 2215 | ADD failed on XOR function (444444 + 333333). Restart at 2210 or press CONTINUE. |
| PR11 | 2223 | ADD failed on XOR function. (333333 + 444444). Restart at 2216 or press CONTINUE. |
| PR11 | 2231 | ADD failed on XOR function (525252 + 252525). Restart 2224 or press CONTINUE. |
| PR11 | 2236 | ADD failed on XOR function (252525 + 525252). Restart at 2232 or press CONTINUE. |
| PR11 | 2244 | ADD failed a carry function (525252 + 525252). Restart at 2237 or press CONTINUE. |
| PR11 | 2252 | ADD failed a carry function (333333 + 333333). Restart at 2245 or press CONTINUE. |
| PR11 | 2260 | ADD failed a carry function (673567 + 673567). Restart at 2253 or press CONTINUE. |
| PR11 | 2266 | TAD a carry function (525252 + 525252). Restart at 2261 or press CONTINUE. |
| PR11 | 2274 | TAD failed a carry function (333333 + 333333). Restart at 2267 or press CONTINUE. |
| PR11 | 2302 | TAD failed a carry function (673567 + 673567). Restart at 2275 or press CONTINUE. |
| PR11 | 2210 | ADD failed attempting 671671 + 257257. Restart at 2303 or press CONTINUE. |
| PR11 | 2312 | ADD of 671671 + 257257 set an initially zeroed LINK. Restart at 2303 and press CONTINUE. |
| PR11 | 2400 | ADD of random numbers failed. First halt in this address displays AUGEND in AC. Press CONTINUE. Second halt displays ADDEND. Press CONTINUE to try another random set. |
| PR11 | 2403 | TAD of random numbers failed. Proceed as for halt in 2400 above. |

4.5    Errors In Usage    (continued)

| Program | C(MA) | Cause and Remedial Action |
|---------|-------|---------------------------|
| PR12 | 2521 | ISZ failed. If AC = $\emptyset$, ISZ failed to skip. Otherwise, ISZ lost a count (compare C(AC) withC(41$\emptyset$7). Restart at 2511. |
| PR12 | 2524 | ISZ skipped on non-zero result. Restart at 2511. |
| PR12 | 2536 | ISZ failed with random number. Expected result in AC. Press CONTINUE to obtain actual result. |
| PR12 | 254$\emptyset$ | ISZ failed with random number (in 14$\emptyset$3). Result in AC. Press CONTINUE or restart at 2541. |
| PR13 | 2556 | JMS failed to preserve a previously set LINK. Restart at 255$\emptyset$. |
| PR13 | 2561 | JMS failed to clear Y1-4. Restart at 2556. |
| PR13 | 257$\emptyset$ | JMS failed to preserve a previously cleared LINK. Restart at 2562. |
| PR13 | 2573 | JMS failed to clear Y1-4. Restart at 25$\emptyset$2. |
| PR13 | 26$\emptyset$6 | CAL failed to jump. Restart at 2574. |
| General | 2623 | Not an Error Halt! Program halts here when test series complete. AC17=1 prevents this halt. Press CONTINUE to repeat test series. |
| PR2 | 366$\emptyset$ - 3662 | JMS at 3732 altered PC incorrectly. Restart at 3732. |
| PR2 | 3663 | JMS at 3732 failed to increment operand address Y. Restart at 3732. |
| PR2 | 3665 - 3672 | JMS at 3732 incremented operand address (Y)2,3,4,5,6 or 7 times. Restart at 3732. |
| PR2 | 37$\emptyset\emptyset$ | JMS at 4$\emptyset\emptyset$1 altered PC incorrectly. Restart at 4$\emptyset$61. |
| PR2 | 37$\emptyset$1 | JMS at 4$\emptyset$61 failed to increment operand address Y. Restart at 4$\emptyset$61. |
| PR2 | 37$\emptyset$3 - 371$\emptyset$ | JMS at 4$\emptyset$61 incremented operand address (Y)2,3,4,5,6 or 7 times. Restart at 4$\emptyset$61. |
| PR2 | 3715 | JMS at 3713 failed to jump. Restart at 3713. |

*Halts here if run in upper 8 K* [handwritten note]

4.5        Errors in Usage        (continued)

| Program | C(MA) | Cause and Remedial Action |
|---|---|---|
| PR2 | 3717 | JMP i at 3716 failed to jump. Restart at 3716. |
| PR2 | 3720 - 3722 | JMS at 3752 altered PC incorrectly. Restart at 3752. |
| PR2 | 3723 | JMS at 3752 failed to increment operand address Y. Restart at 3752. |
| PR2 | 3725 - 3731 | JMS at 3752 incremented operand address(Y) 2,3,4,5,6 or 7 times. Restart at 3752. |
| PR2 | 3734 | JMS at 3732 failed to jump. Restart at 3732. |
| PR2 | 3736 | JMP I at 3735 failed to jump. Restart at 3735. |
| PR2 | 3737 | JMS at 3713 altered PC incorrectly. Restart at 3713. |
| PR2 | 3742 -3747 | JMS at 3713 incremented operand address(Y) 2,3,4,5,6 or 7 times. Restart at 3713. |
| PR2 | 3754 | JMS at 3752 failed to jump. Restart at 3752. |
| PR2 | 3756 | JMP I at 3755 failed to jump. Restart at 3755. |
| PR2 | 3757 - 4026 | JMP altered PC incorrectly. Restart at 203. |
| PR2 | 4027 | JMS at 627 failed to increment operand address Y. Restart at 627. |
| PR2 | 4031-4036 | JMS at 627 incremented operand address(Y) 2,3,4,5,6 or 7 times. Restart at 627. |
| PR2 | 4043 | JMS at 4041 failed to jump. Restart at 4041. |
| PR2 | 4045 | JMP I at 4044 failed to jump. Restart at 4041. |
| PR2 | 4047 | JMS at 4041 failed to increment operand address Y. Restart at 4041. |
| PR2 | 4051 - 4056 | JMS at 4041 incremented operand address(Y)2,3,4,5,6 or 7 times. Restart at 4041. |
| PR2 | 4063 | JMS at 4061 failed to jump. Restart at 4061. |
| PR2 | 4070 | JMS at 3675 failed to jump. Restart at 3675. |
| PR2 | 4072 | JMP I at 4071 failed to jump. Restart at 4071. |

5.        RESTRICTIONS

None.

## 6.    DESCRIPTION

## 6.1    DISCUSSION

### General

MAINDEC 701 is a set of fourteen programs (numbered PR0 through PR13) designed to test the operation of PDP-7 instructions. The following instructions are tested:

law, all the operate group (except oas which is only partially tested.), and the memory reference instructions.

add and tad are the only arithmetic instructions tested.) Indirect addressing is checked as is the operation of the auto-index registers. Random numbers are employed in testing the rotate group, isz and the arithmetic instructions add and tad.

Maindec 701 is designed so that it is not necessary to assume that any instruction is working. In order to facilitate usage, an exception is made to this rule. The final five or six instructions of each program comprise a sequence which is responsible for reiterating the current test a set number of times and then sensing AC switch zero to determine if the operator desires that the next program be entered. Some of these instructions are not fully tested until a later program is entered. These sequences are not, however, inherent to the testing ~~purse~~ and may be replaced by a single jmp addressed to the beginning of the following test or the current test. Maindec 701 is loaded by the hardware read-in facility, ~~to obviate dependence on an instruction oriented loading system.~~

## 6. DESCRIPTION (continued)

### 6.2 Examples and/or Applications

### 6.2.1 Program Zero (PRØ)

Program zero provides a complete test for hit and jmp and preliminary tests for isz, spa, cml, cma, and cla. When first entered at loc. 17Ø, a sequence of nine instructions is encountered, involving from normal halts. This sequence is designed to provide a test for hlt, a check-out of the ability to clear the AC and Link by pressing START, and preliminary tests of the instructions cma, cml, spa, cla, ail and oas. The last mentioned preliminary tests made here and in this manner are intended to give a sufficient degree of confidence in the operation of these instructions so that they may be safely employed as elements of "housekeeping" functions such as loop counters and switch sensors.

The jmp is tested by requiring jumps, Ø2Ø3 to 3777, 3775, 3766, 4Ø1Ø, 4Ø17 and Ø21Ø. These addresses were chosen so that every PC bit is forced to transition from Ø to 1, 1 to Ø, Ø to Ø, and 1 to 1. Each jump is protected by error hits from any incorrect alteration of PC bits 15 - 17.

Upon completion of the jump sequence, the number 777776 is successively indexed to ØØØØØ1 to check isz's ability to react to negative, zero, and positive results.

The sequences described above, exclusive of the initial nine instructions are arbitrarily iterated $100_{10}$ times. After the 100th iteration the AC Switch Register is checked. If ACSØ = 1, 1ØØ more iterations are performed. If ACSØ = Ø, Program 1 is entered.

## 6.    DESCRIPTION    (continued)

### 6.2.2    Program One    (PR1)

Program One checks operate instructions cla, clc, cll, nop, skp, smo, inl, stl, sza, and szl and partially tests cma, cml, oas, sna, and spa. Since only hlt and jmp have been completely tested prior to this time, the operate instructions are tested for the most part in pairs. This requires that failures be determined manually by examination of the AC and Link states at error halt time. All skip instructions are "protected" to a depth of seven against excessive increments of the PC.

The last test performed is of oas, and requires that the AC switches be in a non-zero state.

Program One is arbitrarily iterated $100_{10}$ times. Following the 100th iteration, the AC Switch Register is checked. If ACS0 = 1, Program One is iterated 100 additional times. If ACS0 = 0, Program Two is entered.

6.      DESCRIPTION    (continued)

6.2.3    Program Two   (PR2)

Program Two tests Jmp I   and partially tests Jms.   The Link preservation
aspects of JMS are tested by PR13.   JMS is executed from 5627 to 4027, 4047,
3765, 3740, 3722, 3663, and 0640  thus requiring all PC bits to transition from
0 to 1, 1 to 0, 0 to 0 and 1 to 1.   JMS is also tested for ability to jump to the
operand address plus one.   JMP I  is tested as a critical precondition of JMS.
JMS and JMP I  executions are accompanied by protection to a depth of seven
against excessive incrementions of the PC.

Program Two is automatically iterated $100_{10}$ times.   Following the
100th iteration, the AC Switch Register is checked.   If AC S0 = 1, Program Two
is iterated 100 additional times.   If AC S0 = 0, Program Three is entered.

6.    DESCRIPTION    (continued).

6.4    Program Three    (PR3)

Program Three tests the XOR instruction. The exclusive OR is formed sixteen times to include all pertinent combinations. The following combinations are employed; AC = 777777 with 777777, AC = 000000 with 000000, AC = 000000 with 777777, AC = 777777 with 000000, AC = 777777 with 000000, AC = 000000 with 070707, AC = 070707 with 707070, AC = 000000 with 707070, AC = 707070 with 070707, AC = 000000 with 333333, AC = 333333 with 444444, AC = 000000 with 444444, AC = 444444 with 333333, AC = 000000 with 525252, AC = 525252 with 252525, AC = 000000 with 252525, and AC = 252525 with 525252.

Program three is arbitrarily iterated 4096 times. Following the 4096th iteration, the AC Switch Register is checked. If ACS0 = 1, Program Three is iterated an additional 4096 times. If ACS0 = 0, Program Four is entered.

# 6. DESCRIPTION (continued)

## 6.5 Program Four (PR 4)

Program Four tests the SAD Instruction. SAD is checked to insure that it (1) skips on unequal comparison, (2) does not skip on equal comparison, and (3) replaces the contents of the AC after comparison. Comparisons are made between AC = 000000 and 777777, AC = 777777 and 777777, AC = 525252 and 252525, and AC = 525252 and 525252.

Program Four is arbitrarily iterated $100_{10}$ times. After the 100th iteration, the AC Switch Register is checked.

If ACS0 = 1, Program Four is iterated an additional 100 times.
If ACS0 = 0, Program Five is entered.

## 6. DESCRIPTION (continued)

### 6.6 Program Five (PR 5)

Program Five tests the operation of SNA (and indirectly, of SZA) with each individual AC bit set. The program also provides a test of the decoder network which generates the AC $\neq +\emptyset$ level.

With zeros in all bits SNA is checked to assure that no skip occurs. With zeros in all bits except AC17, SNA is executed to skip over an error halt. The same procedure is used with only AC16 = 1, then AC15 = 1 etc. and finally with only AC$\emptyset$ = 1.

Program Five is arbitrarily iterated $100_{10}$ times. After the 100th iteration, the AC Switch Register is checked.

If ACS$\emptyset$ = 1, Program Five is iterated an additional 100 times.
If ACS$\emptyset$ = $\emptyset$, Program Six is entered.

6.    DESCRIPTION    (continued)


6.7    Program Six    (PR 6)

Program Six tests Lac and Dac.  Lac is tested by loading the AC with a known number using the lac Instruction and then checking for a correct transfer.  The combinations employed are 000000 loaded into 000000, 777777 loaded into 000000, 777777 loaded into 777777, 000000 loaded into 777777, 252525 loaded into 525252, 525252 loaded into 252525, 070707 loaded into 707070, 707070 loaded into 070707, 444444 loaded into 333333, and 333333 loaded into 444444.

Dac is tested by using this instruction to deposit the contents of the AC into a memory location.  The location is then checked to ensure a correct transfer.  The combinations employed are; deposit of 000000 into unknown, deposit of 777777 into 000000, deposit of 777777 into 777777, deposit of 000000 into 777777, deposit of 000000 into 000000, deposit of 525252 into 000000, deposit of 252525 into 525252, deposit of 525252 into 252525, deposit of 707070 into 525252, deposit of 070707 into 707070, deposit of 707070 into 070707, deposit of 333333 into 707070, deposit of 444444 into 333333, and deposit of 333333 into 444444.

Program Six is arbitrarily iterated $100_{10}$ times.
After the 100th iteration, the AC10 Switch Register is checked.
If ACS0 = 1, Program Six is iterated an additional 100 times.
If ACS0 = 0, Program Seven is entered.

6.    DESCRIPTION    (continued)


6.8    Program Seven

Program Seven tests the operation of  Dzm,  Low,  Lac i,  and Dac i.

Dzm    is tested to ensure that it clears a location containing all ones

and a location containing all zeros.

Low  is checked by loading all zeros into all ones, all zeros into all

zeros, all ones into all ones, and all ones into all zeros.  The possibility of

low causing an entry into the defer cycle is also checked.

Lac i  and  Dac i  are checked by loading all ones into all zeros

indirectly via a reference register.  The reference register is checked for

alteration after performing  dac i.

Program Seven is arbitrarily iterated $100_{10}$ times.

After the 100th iteration, the AC Switch Register is checked.

If  ACS0  = 1,  Program Seven is iterated 100 additional times.
If  ACS0  = 0,  Program Eight is entered.

6.    DESCRIPTION    (continued)


6.9    Program Eight    (PR 0)

Program Eight tests the operation of Xct and auto-index registers 10 - 17.

Xct  is used to execute the one cycle instruction cml, the two cycle

instruction jmp, the three cycle instruction jmp I, and another xct instruction.

The operation of the auto-index registers is checked by using each in

turn to retrieve three known numbers via a lac I instruction.

Program Eight is arbitrarily iterated $100_{10}$ times.

After the 100th iteration, the AC Switch Register is checked.

If  ACS0 = 1, Program Eight is iterated 100 additional times.

If  ACS0 = 0, Program Nine is entered.

## DESCRIPTION   (continued)

6.1$\emptyset$     Program Nine   (PR 9)

Program Nine tests the and instruction.

All ones are anded with all ones,  all ones are anded with all zeros,  and each

AC bit is anded with itself.

Program Nine is arbitrarily iterated $100_{10}$ times.

After the 100th iteration, the AC Switch Register is checked.

If  ACS$\emptyset$ = 1,  Program Nine is iterated 100 additional times.

If  ACS$\emptyset$ = $\emptyset$,  Program Ten is entered.

6.    DESCRIPTION   (continued)


6.11   Program Ten   (PR1Ø)

Program ten tests rotate instructions rar, ral, rtr, and rtl.
All four instructions are used in succession to rotate a one through a field of
zeros compused of all AC bits and the Link 4096 times, checking the state
of the AC and link after each individual rotation.  The same procedure is
repeated, this time rotating a zero through a field of ones.

Ral and Rar are then employed to shift all possible 18 bit numbers
one bit left and right.  The program checks that an apparent movement of
the number in the AC has occurred and that the state of the Link is as it
should be after each rotation.

The state of the AC Switch Register is then checked.

If ACSØ = 1,  Program Ten is re-entered.

If ACSØ = Ø,  Program Eleven is entered.

6.    DESCRIPTION    (continued)


6.12    Program Eleven    (PR 11)

Program Eleven tests the operation of <u>add</u> and <u>tad</u>. Using various combinations of $070707$, $707070$, 444444, 333333, 777777, 525252, and 252525, XOR functions and carry functions of <u>add</u> are checked as well as carry functions of <u>tad</u>.

Add <u>and</u> tad are further checked by using random numbers. Two random numbers are created and added/tadded together. This result is then compared with a simulated result to assure a correct mathematical result. The Link is also checked.

Program Eleven is arbitrarily iterated 4096 times
After the 4096th iteration, the AC Switch Register is checked.

If  $ACS0 = 1$,  Program Eleven is iterated an additional 4096 times.
If  $ACS0 = 0$,  Program Twelve is entered.

6. DESCRIPTION (continued)


6.13 Program Twelve (PR 12)

Program Twelve completes the testing of ISZ, and concerns itself with counting accuracy: A memory location is indexed from the zero state to 777777 while a parallel count is maintained in the AC by adding one to the previous contents of the AC. The memory location contents are checked against the number in the AC after each isz to detect a miscount. When the memory location is indexed to zero, the program confirms that isz responded correctly with a skip. This sequence repeats three times before beginning a series of random number tests.

A random number is created and incremented with isz. The program checks the result. This sequence is repeated $262,143_{10}$ times. The state of the AC Switch Register is then checked.

If ACS$\emptyset$ = 1, Program Twelve is entirely repeated.
If ACS$\emptyset$ = $\emptyset$, Program Thirteen is entered.

6.    DESCRIPTION    (continued)

6.14    Program Thirteen    (PR 13)

Program Thirteen checks the Link saving properties of Jms and the instructions cal and cal i.

Two Jms instructions are executed to a memory location containing alternately 36000 and 400000, with the Link equal to one and zero respectively. In each case, the memory location is checked to determine if the link was properly saved in bit 0 and if bits 1-4 were properly cleared.

The cal instruction is tested by an execution of cal 0 with the link set, loc. 20 containing het, and loc 21 containing a jump to 30. Beginning at loc. 30 a sequence of instruction tests loc 20 for presence of the Link state in bit zero, the address of the location following the cal in bits 5 - 17, and zeros in bits 1 - 4. A return is made to the cal location plus one with a cleared link. The state of the link is immediately checked to rule out the possibility that cal never jumped originally. Loc 20 is indexed and a cal i 0 is executed to transfer control to the original cal, loc, plus two.

Program Thirteen arbitrarily iterates $100_{10}$ times.
After the 100th Iteration, the AC Switch Register is checked.

If  ACS0 = 1,  Program Thirteen is iterated an additional 100 times.

If  ACS0 = 0,  the AC Switch Register is checked again.

If  ACS17 = 1,  control is transferred to PR0 at loc. 201.

If  ACS17 = 0,  the program halts in loc 2623 signifying the end of the test
             series. Pressing CONTINUE at this point will also transfer
             control to PR0 at loc 201.

6.2

TABLE 6-1   INSTRUCTIONS TESTED BY MAINDEC 701

(Alphabetically Arranged)

| Instruction | Tested by Program | Instruction | Tested by Program |
|---|---|---|---|
| add | 11 | oas | $\emptyset$, 1 |
| and | 9 | ral | 1$\emptyset$ |
| cal | 13 | rar | 1$\emptyset$ |
| clas | 1 | rtl | 1$\emptyset$ |
| clc | 1 | rtr | 1$\emptyset$ |
| cll | 1 | sad | 4 |
| cma | $\emptyset$, 1 | skp | 1 |
| cml | $\emptyset$, 1 | sma | 1 |
| dac | 6, 7 | sna | 1, 5 |
| dzm | 7 | snl | 1 |
| hlt | $\emptyset$ | spa | $\emptyset$, 1 |
| isz | $\emptyset$, 12 | stl | 1 |
| jmp | $\emptyset$ | sza | 1 |
| jms | 2, 13 | szl | 1 |
| lac | 6, 7 | tad | 11 |
| law | 7 | xct | 8 |
| nop | 1 | xor | 3 |

## TABLE 6-2  PROGRAMS IN MAINDEC 701

| Program Number | Tests | Program Number | Tests |
|---|---|---|---|
| Ø | cla * | | |
| Ø | cma * | 1 | szl |
| Ø | cml * | 2 | jms * |
| Ø | hlt | 3 | xor |
| Ø | isz * | 4 | sad |
| Ø | jmp | 5 | sna * |
| Ø | oas ** | 6 | dac * |
| Ø | spa * | 6 | lac * |
| 1 | cla | 7 | dac * |
| 1 | clc | 7 | dzm |
| 1 | cll | 7 | lac * |
| 1 | cma * | 7 | law |
| 1 | cml * | 8 | xct |
| 1 | nop | 9 | and |
| 1 | oas ** | 1Ø | ral |
| 1 | skp | 1Ø | rar |
| 1 | sma | 1Ø | rtl |
| 1 | sna * | 1Ø | rtr |
| 1 | snl | 11 | add |
| 1 | spa * | 11 | tad |
| 1 | stl | 12 | isz * |
| 1 | sza | 13 | cal |
| | | 13 | jms * |

* completely tested, but only partially by this program

** only partially tested