```
 LOC        OBJECT CODE       ADDR1     ADDR2    STMT

                                                     2 ************************************************************************
                                                     3 *
                                                     4 *Testcase bim-001-add-sub
                                                     5 *  Test case capability includes condition codes and fixed point
                                                     6 *  overflow interruptions.
                                                     7 *
                                                     8 *
                                                     9 *                      ********************
                                                    10 *                      **   IMPORTANT!   **
                                                    11 *                      ********************
                                                    12 *
                                                    13 *        This test uses the Hercules Diagnose X'008' interface
                                                    14 *        to display messages and thus your .tst runtest script
                                                    15 *        MUST contain a "DIAG8CMD ENABLE" statement within it!
                                                    16 *
                                                    17 *
                                                    18 ************************************************************************


                                                    20 ************************************************************************
                                                    21 *
                                                    22 *                         bim-001-add-sub.asm
                                                    23 *
                                                    24 * Copyright 2018 by Stephen R Orso.
                                                    25 *
                                                    26 * Distributed under the Boost Software License, Version 1.0.  See
                                                    27 * accompanying file BOOST_LICENSE_1_0.txt or a copy at:
                                                    28 *
                                                    29 *      http://www.boost.org/LICENSE_1_0.txt)
                                                    30 *
                                                    31 * Adapted from the original bim-001-add by Peter J. Jansen.
                                                    32 *
                                                    33 ************************************************************************



                                                    35 ************************************************************************
                                                    36 *
                                                    37 * Tests the following ADD and SUB instructions, except those marked (*)
                                                    38 * as these are not (yet) implemented.
                                                    39 *   ADD REGISTER RR    AR   32-bit sum, augend, addend
                                                    40 *              (*) RRF-a ARK  32-bit sum, augend, addend, 3 operand
                                                    41 *                  RRE   AGR  64-bit sum, augend, addend
                                                    42 *              (*) RRF-a ARGK 64-bit sum, augend, addend, 3 operand
                                                    43 *              (*) RRE   AGFR 64-bit augend, sum, 32-bit addend
                                                    44 *   ADD        (*) RX-a  A    32-bit sum, augend, addend
                                                    45 *              (*) RXY-a AY   32-bit sum, augend, addend
                                                    46 *              (*) RXY-a AG   64-bit sum, augend, addend
                                                    47 *              (*) RXY-a AGF  64-bit augend, sum, 32-bit addend
                                                    48 *   SUB REGISTER RR    SR   32-bit sum, minuend, subtrahend
                                                    49 *              (*) RRF-a SRK  32-bit sum, minuend, subtrahend, 3 operand
                                                    50 *                  RRE   SGR  64-bit sum, minuend, subtrahend
                                                    51 *              (*) RRF-a SRGK 64-bit sum, minuend, subtrahend, 3 operand
                                                    52 *              (*) RRE   SGFR 64-bit minuend, sum, 32-bit subtrahend
                                                    53 *   SUB        (*) RX-a  S    32-bit sum, minuend, subtrahend
                                                    54 *              (*) RXY-a SY   32-bit sum, minuend, subtrahend
                                                    55 *              (*) RXY-a SG   64-bit sum, minuend, subtrahend
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

                                                      56 *            (*) RXY-a SGF  64-bit minuend, sum, 32-bit subtrahend
                                                      57 *
                                                      58 * Instructions are test against the definition in the z/Architecture
                                                      59 *   Principles of Operation, SA22-7832-11 (September, 2017), p. 7-27
                                                      60 *   and p. 7-387
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

                                                      62 *
                                                      63 * Test data is compiled into this program.  The test script that runs
                                                      64 * this program can provide alternative test data through Hercules 'r'
                                                      65 * commands.
                                                      66 *
                                                      67 * Basic 32-bit test data is:    2147483647(=M), 0, 1, -1,
                                                      68 *             -2147483647(=-M), -2147483648(=-M-1=-(M+1))
                                                      69 *
                                                      70 * Basic 64-bit test data is:    9223372036854775807(=G) , 0, 1, -1,
                                                      71 *   -9223372036854775807(=-G), -9223372036854775808(=-G-1=-(G+1))
                                                      72 *
                                                      73 * Test Case Order
                                                      74 * 1) AR,  RR, 32-bit addition
                                                      75 * 2) AGR, RR, 64-bit addition
                                                      76 * 3) SR,  RR, 32-bit subtraction
                                                      77 * 4) SGR, RR, 64-bit subtraction
                                                      78 *
                                                      79 * Routines have not been coded for the other ADD / SUB instructions.
                                                      80 *   It would not be hard, and no additional test data would be needed.
                                                      81 *
                                                      82 * Each value is added / subtracted to every value twice, once with
                                                      83 *   interruptions suppressed, once with interruptions enabled.
                                                      84 *   72 results are generated for each such test case.
                                                      85 *
                                                      86 * Opportunities for future development:
                                                      87 * - Use SIGP to change the processor mode and verify correct operation
                                                      88 *   in 390 and 370 mode, including verification that ADD varients
                                                      89 *   that are unsupported generate the expected operation exceptions
                                                      90 * - Add the remaining RR* and the RX* instructions.
                                                      91 *
                                                      92 ********************************************************************************
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT
                                                     94 *
                              00000000  00013BEF     95 BIMADSUB START 0
                              00000000  00000001     96 STRTLABL EQU   *
                              00000000  00000001     97 R0       EQU   0               Work register for cc extraction
                                                     98 *                              ..also augend / minuend and result
                                                     99 *                              ..register for two-operand add and
                                                    100 *                              ..subtract
                              00000001  00000001    101 R1       EQU   1               addend / subtrahend register for
                                                    102 *                              ..RR two-operand variants.
                              00000002  00000001    103 R2       EQU   2               Count of test augends / minuends
                                                    104 *                              ..remaining
                              00000003  00000001    105 R3       EQU   3               Pointer to next test augend /
                                                    106 *                              ..minuend
                              00000004  00000001    107 R4       EQU   4               Count of test addends /
                                                    108 *                              ..subtrahends remaining
                              00000005  00000001    109 R5       EQU   5               Pointer to next test addend /
                                                    110 *                              ..subtrahend
                              00000006  00000001    111 R6       EQU   6               Size of each augend / minuend and
                                                    112 *                              ..addend / subtrahend
                              00000007  00000001    113 R7       EQU   7               Pointer to next result value(s)
                              00000008  00000001    114 R8       EQU   8               Pointer to next cc/fixed point ovfl
                              00000009  00000001    115 R9       EQU   9               Top of inner loop address in tests
                              0000000A  00000001    116 R10      EQU   10              Pointer to test address list
                              0000000B  00000001    117 R11      EQU   11              **Reserved for z/CMS test rig
                              0000000C  00000001    118 R12      EQU   12              Top of outer loop address in tests
                              0000000D  00000001    119 R13      EQU   13              Return address to mainline
                              0000000E  00000001    120 R14      EQU   14              **Return address for z/CMS test rig
                              0000000F  00000001    121 R15      EQU   15              **Base register on z/CMS or Hyperion
                                                    122
                                                    123 *
00000000                      00000000              124          USING *,R15
00000000                      00013840              125          USING HELPERS,R12
                                                    126 *
                                                    127 * Above is assumed to works on real iron (R15=0 after sysclear) and in
                                                    128 * John's z/CMS test rig (R15 points to start of load module).
                                                    129 *
                                                    130 * Selective z/Arch low core layout
                                                    131 *
00000000                      00000000  0000008C    132          ORG   STRTLABL+X'8C'      Program check interrution code
0000008C  00000000                                  133 PCINTCD  DS    F
                                                    134 *
                              00000150  00000001    135 PCOLDPSW EQU   STRTLABL+X'150'     z/Arch Program check old PSW
                                                    136 *
00000090                      00000090  000001A0    137          ORG   STRTLABL+X'1A0'     z/Arch Restart PSW
000001A0  00000001 80000000                         138          DC    X'0000000180000000',AD(START)
                                                    139 *
000001B0                      000001B0  000001D0    140          ORG   STRTLABL+X'1D0'     z/Arch Program check NEW PSW
000001D0  00000000 00000000                         141          DC    X'0000000000000000',AD(PROGCHK)
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT
                                                    143 *
                                                    144 * Program check routine.  If Data Exception, continue execution at
                                                    145 * the instruction following the program check.  Otherwise, hard wait.
                                                    146 * No need to collect data.
                                                    147 *
000001E0                      000001E0  00000200    148          ORG   STRTLABL+X'200'
00000200                                            149 PROGCHK  DS    0H                Program check occured...
00000200  9508 F08F                     0000008F    150          CLI   PCINTCD+3,X'08'  Fixed Point Overflow?
00000204  A774 0004                     0000020C    151          JNE   PCNOTDTA          ..no, fail the test
00000208  B2B2 F150                     00000150    152          LPSWE PCOLDPSW          ..yes, resume program execution

0000020C  900F F23C                     0000023C    154 PCNOTDTA STM   R0,R15,SAVEREGS  Save registers
00000210  58C0 F27C                     0000027C    155          L     R12,AHELPERS     Get address of helper subroutines
00000214  4DD0 C000                     00013840    156          BAS   R13,PGMCK         Report this unexpected program check
00000218  980F F23C                     0000023C    157          LM    R0,R15,SAVEREGS  Restore registers

0000021C  12EE                                      159          LTR   R14,R14        Return address provided?
0000021E  077E                                      160          BNZR  R14            Yes, return to z/CMS test rig.
00000220  B2B2 F228                     00000228    161          LPSWE PROGPSW        Not data exception, enter disabled wait
00000228  00020000 00000000                         162 PROGPSW  DC    0D'0',X'0002000000000000',XL6'00',X'DEAD' Abnormal end
00000238  B2B2 F2C0                     000002C0    163 FAIL     LPSWE FAILPSW        Not data exception, enter disabled wait
0000023C  00000000 00000000                         164 SAVEREGS DC    16F'0'           Registers save area
0000027C  00013840                                  165 AHELPERS DC    A(HELPERS)       Address of helper subroutines
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

                                                     167 ************************************************************************
                                                     168 *
                                                     169 *  Main program.  Enable Advanced Floating Point, process test cases.
                                                     170 *
00000280                                             171 START    DS    0H
                                                     172 *
                                                     173 * ADD REGISTER (32-bit operands, two operand)
                                                     174 *
00000280  41A0 F2D0                 000002D0        175          LA    R10,ARTABL    32-bit test table
00000284  4DD0 F330                 00000330        176          BAS   R13,ARTEST    AR, add register 32-bit
                                                     177 *
                                                     178 * ADD REGISTER (64-bit operands, two operand)
                                                     179 *
00000288  41A0 F2E8                 000002E8        180          LA    R10,AGRTABL   64-bit test table
0000028C  4DD0 F3C6                 000003C6        181          BAS   R13,AGRTEST   AGR, add register 64-bit
                                                     182 *
                                                     183 * SUB REGISTER (32-bit operands, two operand)
                                                     184 *
00000290  41A0 F300                 00000300        185          LA    R10,SRTABL    32-bit test table
00000294  4DD0 F46C                 0000046C        186          BAS   R13,SRTEST    SR, subtract register 32-bit
                                                     187 *
                                                     188 * SUB REGISTER (64-bit operands, two operand)
                                                     189 *
00000298  41A0 F318                 00000318        190          LA    R10,SGRTABL   64-bit test table
0000029C  4DD0 F502                 00000502        191          BAS   R13,SGRTEST   SGR, subtract register 64-bit
                                                     192 *
                                                     193 ************************************************************************
                                                     194 *                    Verify test results...
                                                     195 ************************************************************************
                                                     196 *
000002A0  58C0 F27C                 0000027C        197          L     R12,AHELPERS     Get address of helper subroutines
000002A4  4DD0 C0A4                 000138E4        198          BAS   R13,VERISUB      Go verify results
000002A8  12EE                                      199          LTR   R14,R14          Was return address provided?
000002AA  077E                                      200          BNZR  R14              Yes, return to z/CMS test rig.
000002AC  B2B2 F2B0                 000002B0        201          LPSWE GOODPSW          Load SUCCESS PSW

000002B0                                             203          DS    0D            Ensure correct alignment for PSW
000002B0  00020000 00000000                          204 GOODPSW  DC    X'0002000000000000',AD(0)  Normal end - disabled wait
000002C0  00020000 00000000                          205 FAILPSW  DC    X'0002000000000000',XL6'00',X'0BAD' Abnormal end
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT
                                                    207 *
                                                    208 * Input values parameter list, six fullwords:
                                                    209 *      1) Count of augends / minuends (and addends / subtrahends)
                                                    210 *      2) Address of augends / minuends
                                                    211 *      3) Address of addends / subtrahends
                                                    212 *      4) Address to place sums / differences
                                                    213 *      5) Address to place condition code and interruption code
                                                    214 *      6) Size of augends / minuends, addends / subtrahends
                                                    215 *         ..and sums / differences
                                                    216 *
000002D0                                            217 ARTABL   DS    0F           Inputs for 32-bit/32-bit tests
000002D0  00000006                                  218          DC    A(VALCT/4)
000002D4  000005B0                                  219          DC    A(A32VALS)   Address of augends
000002D8  000005B0                                  220          DC    A(A32VALS)   Address of addends
000002DC  00001000                                  221          DC    A(ARSUM)     Address to store sums
000002E0  00002000                                  222          DC    A(ARFLG)     Address to store cc, int code
000002E4  00000004                                  223          DC    A(4)         4 byte augends, addends and sums
                                                    224 *
000002E8                                            225 AGRTABL  DS    0F           Inputs for 64-bit/64-bit tests
000002E8  00000006                                  226          DC    A(VALCT64/8)
000002EC  000005C8                                  227          DC    A(A64VALS)   Address of augends
000002F0  000005C8                                  228          DC    A(A64VALS)   Address of addends
000002F4  00001400                                  229          DC    A(AGRSUM)    Address to store sums
000002F8  00002400                                  230          DC    A(AGRFLG)    Address to store cc, int code
000002FC  00000008                                  231          DC    A(8)         8 byte augends, addends and sums
                                                    232 *
00000300                                            233 SRTABL   DS    0F           Inputs for 32-bit/32-bit tests
00000300  00000006                                  234          DC    A(VALCT/4)
00000304  000005B0                                  235          DC    A(A32VALS)   Address of minuends
00000308  000005B0                                  236          DC    A(A32VALS)   Address of subtrahends
0000030C  00001800                                  237          DC    A(SRSUM)     Address to store differences
00000310  00002800                                  238          DC    A(SRFLG)     Address to store cc, int code
00000314  00000004                                  239          DC    A(4)         4 byte minuends, subtrahends and
                                                    240 *                           ..differences
                                                    241 *
00000318                                            242 SGRTABL  DS    0F           Inputs for 64-bit/64-bit tests
00000318  00000006                                  243          DC    A(VALCT64/8)
0000031C  000005C8                                  244          DC    A(A64VALS)   Address of minuends
00000320  000005C8                                  245          DC    A(A64VALS)   Address of addends
00000324  00001C00                                  246          DC    A(SGRSUM)    Address to store differences
00000328  00002C00                                  247          DC    A(SGRFLG)    Address to store cc, int code
0000032C  00000008                                  248          DC    A(8)         8 byte minuends, subtrahends and
                                                    249 *                           ..differences
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

                                                    251 ********************************************************************************
                                                    252 *
                                                    253 * ADD REGISTER (AR, RRE) - 32-bit addend, 32-augend, 32-bit sum.
                                                    254 * Result replaces augend in operand 1.
                                                    255 *
                                                    256 ********************************************************************************


00000330  B222 0000                                 258 ARTEST   IPM    R0             Get cc, program mask
00000334  5000 F5A8                 000005A8        259          ST     R0,ARCCPM      Save for later disable of ints
00000338  5000 F5AC                 000005AC        260          ST     R0,ARCCPMOV    Save for overflow enablement
0000033C  9608 F5AC                 000005AC        261          OI     ARCCPMOV,X'08'  Enable fixed point overflow ints
00000340  9823 A000                 00000000        262          LM     R2,R3,0(R10)   Get count and addresses of augends
00000344  9878 A00C                 0000000C        263          LM     R7,R8,12(R10) Get address of result area and flag area.
00000348  5860 A014                 00000014        264          L      R6,20(,R10)    Get size of augends, addends and results.
0000034C  1222                                      265          LTR    R2,R2          Any test cases?
0000034E  078D                                      266          BZR    R13            ..No, return to caller
00000350  0DC0                                      267          BASR   R12,0          Set top of loop
                                                    268 *
                                                    269 * Top of outer loop.  Process next augend
                                                    270 *
00000352  5840 A000                 00000000        271          L      R4,0(,R10)     Get count of addends
00000356  5850 A008                 00000008        272          L      R5,8(,R10)     Get address of addend table
0000035A  0D90                                      273          BASR   R9,0           Set top of loop
                                                    274 *
0000035C  5800 3000                 00000000        275          L      R0,0(,R3)      Initialize augend
00000360  5810 5000                 00000000        276          L      R1,0(,R5)      Initialize addend
00000364  1A01                                      277          AR     R0,R1          Replace augend with sum
00000366  5000 7000                 00000000        278          ST     R0,0(,R7)      Store sum
0000036A  B222 0000                                 279          IPM    R0             Retrieve condition code
0000036E  8800 001C                 0000001C        280          SRL    R0,28          Move CC to low-order r0
00000372  4200 8000                 00000000        281          STC    R0,0(,R8)      Store condition code
00000376  4176 7000                 00000000        282          LA     R7,0(R6,R7)    Point to next sum slot
0000037A  4180 8004                 00000004        283          LA     R8,4(,R8)      Point to next cc-int code slot
                                                    284 *
                                                    285 * Repeat the instruction with Fixed Point Overflow interruptions
                                                    286 * enabled.
                                                    287 *
0000037E  5800 F5AC                 000005AC        288          L      R0,ARCCPMOV    Get cc/program mask for overflow ints
00000382  0400                                      289          SPM    R0             Enable Fixed Point Overflow inter.
00000384  D703 F08C F08C      0000008C  0000008C    290          XC     PCINTCD,PCINTCD    Zero out PC interruption code
                                                    291 *
0000038A  5800 3000                 00000000        292          L      R0,0(,R3)      Initialize augend
0000038E  5810 5000                 00000000        293          L      R1,0(,R5)      Initialize addend
00000392  1A01                                      294          AR     R0,R1          Replace augend with sum
00000394  5000 7000                 00000000        295          ST     R0,0(,R7)      Store sum
00000398  D202 8001 F08D      00000001  0000008D    296          MVC    1(3,R8),PCINTCD+1   Save interruption code
0000039E  B222 0000                                 297          IPM    R0             Retrieve condition code
000003A2  8800 001C                 0000001C        298          SRL    R0,28          Move CC to low-order r0
000003A6  4200 8000                 00000000        299          STC    R0,0(,R8)      Store condition code
                                                    300 *
000003AA  5800 F5A8                 000005A8        301          L      R0,ARCCPM      Get cc/program mask for no o'flow ints
000003AE  0400                                      302          SPM    R0             Disable Fixed Point Overflow inter.
                                                    303 *
000003B0  4156 5000                 00000000        304          LA     R5,0(R6,R5)    Point to next addend
000003B4  4176 7000                 00000000        305          LA     R7,0(R6,R7)    Point to next sum slot
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

000003B8  4180 8004                      00000004   306          LA    R8,4(,R8)     Point to next cc-int code slot
000003BC  0649                                      307          BCTR  R4,R9         Loop through addends
                                                    308 *
                                                    309 * End of addends.  Process next augend
                                                    310 *
000003BE  4136 3000                      00000000   311          LA    R3,0(R6,R3)   Point to next augend
000003C2  062C                                      312          BCTR  R2,R12        Loop through augends
                                                    313 *
000003C4  07FD                                      314          BR    R13           All converted; return.
                                                    315 *
                                                    316 ********************************************************************
                                                    317 *
                                                    318 * ADD REGISTER (AGR, RRE) - 64-bit addend, 64-augend, 64-bit sum.
                                                    319 * Result replaces augend in operand 1.
                                                    320 *
                                                    321 ********************************************************************


000003C6  B222 0000                                 323 AGRTEST  IPM   R0            Get cc, program mask
000003CA  5000 F5A8                      000005A8   324          ST    R0,ARCCPM     Save for later disable of ints
000003CE  5000 F5AC                      000005AC   325          ST    R0,ARCCPMOV   Save for overflow enablement
000003D2  9608 F5AC                      000005AC   326          OI    ARCCPMOV,X'08'  Enable fixed point overflow ints
000003D6  9823 A000                      00000000   327          LM    R2,R3,0(R10)  Get count and addresses of augends
000003DA  9878 A00C                      0000000C   328          LM    R7,R8,12(R10) Get address of result area and flag area.
000003DE  5860 A014                      00000014   329          L     R6,20(,R10)   Get size of augends, addends and results.
000003E2  1222                                      330          LTR   R2,R2         Any test cases?
000003E4  078D                                      331          BZR   R13           ..No, return to caller
000003E6  0DC0                                      332          BASR  R12,0         Set top of loop
                                                    333 *
                                                    334 * Top of outer loop.  Process next augend
                                                    335 *
000003E8  5840 A000                      00000000   336          L     R4,0(,R10)    Get count of addends
000003EC  5850 A008                      00000008   337          L     R5,8(,R10)    Get address of addend table
000003F0  0D90                                      338          BASR  R9,0          Set top of loop
                                                    339 *
000003F2  E300 3000 0004                 00000000   340          LG    R0,0(,R3)     Initialize augend
000003F8  E310 5000 0004                 00000000   341          LG    R1,0(,R5)     Initialize addend
000003FE  B908 0001                                 342          AGR   R0,R1         Replace augend with sum
00000402  E300 7000 0024                 00000000   343          STG   R0,0(,R7)     Store sum
00000408  B222 0000                                 344          IPM   R0            Retrieve condition code
0000040C  8800 001C                      0000001C   345          SRL   R0,28         Move CC to low-order r0
00000410  4200 8000                      00000000   346          STC   R0,0(,R8)     Store condition code
00000414  4176 7000                      00000000   347          LA    R7,0(R6,R7)   Point to next sum slot
00000418  4180 8004                      00000004   348          LA    R8,4(,R8)     Point to next cc-int code slot
                                                    349 *
                                                    350 * Repeat the instruction with Fixed Point Overflow interruptions
                                                    351 * enabled.
                                                    352 *
0000041C  5800 F5AC                      000005AC   353          L     R0,ARCCPMOV   Get cc/program mask for overflow ints
00000420  0400                                      354          SPM   R0            Enable Fixed Point Overflow inter.
00000422  D703 F08C F08C       0000008C  0000008C   355          XC    PCINTCD,PCINTCD    Zero out PC interruption code
                                                    356 *
00000428  E300 3000 0004                 00000000   357          LG    R0,0(,R3)     Initialize augend
0000042E  E310 5000 0004                 00000000   358          LG    R1,0(,R5)     Initialize addend
00000434  B908 0001                                 359          AGR   R0,R1         Replace augend with sum
00000438  E300 7000 0024                 00000000   360          STG   R0,0(,R7)     Store sum
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

0000043E  D202 8001 F08D      00000001  0000008D   361          MVC   1(3,R8),PCINTCD+1   Save interruption code
00000444  B222 0000                               362          IPM   R0                 Retrieve condition code
00000448  8800 001C                     0000001C  363          SRL   R0,28              Move CC to low-order r0
0000044C  4200 8000                     00000000  364          STC   R0,0(,R8)          Store condition code
                                                  365 *
00000450  5800 F5A8                     000005A8  366          L     R0,ARCCPM          Get cc/program mask for no o'flow ints
00000454  0400                                    367          SPM   R0                 Disable Fixed Point Overflow inter.
                                                  368 *
00000456  4156 5000                     00000000  369          LA    R5,0(R6,R5)        Point to next addend
0000045A  4176 7000                     00000000  370          LA    R7,0(R6,R7)        Point to next sum slot
0000045E  4180 8004                     00000004  371          LA    R8,4(,R8)          Point to next cc-int code slot
00000462  0649                                    372          BCTR  R4,R9              Loop through addends
                                                  373 *
                                                  374 * End of addends.  Process next augend
                                                  375 *
00000464  4136 3000                     00000000  376          LA    R3,0(R6,R3)        Point to next augend
00000468  062C                                    377          BCTR  R2,R12             Loop through augends
                                                  378 *
0000046A  07FD                                    379          BR    R13                All converted; return.
                                                  380 *
                                                  381 ***************************************************************
                                                  382 *
                                                  383 * SUB REGISTER (SR, RRE) - 32-bit subtrahend, 32-minuend, 32-bit sum.
                                                  384 * Result replaces subtrahend in operand 1.
                                                  385 *
                                                  386 ***************************************************************


0000046C  B222 0000                               388 SRTEST   IPM   R0                 Get cc, program mask
00000470  5000 F5A8                     000005A8  389          ST    R0,ARCCPM          Save for later disable of ints
00000474  5000 F5AC                     000005AC  390          ST    R0,ARCCPMOV        Save for overflow enablement
00000478  9608 F5AC                     000005AC  391          OI    ARCCPMOV,X'08'     Enable fixed point overflow ints
0000047C  9823 A000                     00000000  392          LM    R2,R3,0(R10)       Get count and addresses of minuends
00000480  9878 A00C                     0000000C  393          LM    R7,R8,12(R10)      Get address of result area and flag area.
00000484  5860 A014                     00000014  394          L     R6,20(,R10)        Get size of minuends, subtrahends
                                                  395 *                                  ..and results.
00000488  1222                                    396          LTR   R2,R2              Any test cases?
0000048A  078D                                    397          BZR   R13                ..No, return to caller
0000048C  0DC0                                    398          BASR  R12,0              Set top of loop
                                                  399 *
                                                  400 * Top of outer loop.  Process next minuend
                                                  401 *
0000048E  5840 A000                     00000000  402          L     R4,0(,R10)         Get count of subtrahends
00000492  5850 A008                     00000008  403          L     R5,8(,R10)         Get address of subtrahend table
00000496  0D90                                    404          BASR  R9,0               Set top of loop
                                                  405 *
00000498  5800 3000                     00000000  406          L     R0,0(,R3)          Initialize minuend
0000049C  5810 5000                     00000000  407          L     R1,0(,R5)          Initialize subtrahend
000004A0  1B01                                    408          SR    R0,R1              Replace minuend with difference
000004A2  5000 7000                     00000000  409          ST    R0,0(,R7)          Store difference
000004A6  B222 0000                               410          IPM   R0                 Retrieve condition code
000004AA  8800 001C                     0000001C  411          SRL   R0,28              Move CC to low-order r0
000004AE  4200 8000                     00000000  412          STC   R0,0(,R8)          Store condition code
000004B2  4176 7000                     00000000  413          LA    R7,0(R6,R7)        Point to next sum slot
000004B6  4180 8004                     00000004  414          LA    R8,4(,R8)          Point to next cc-int code slot
                                                  415 *
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT
                                                    416 * Repeat the instruction with Fixed Point Overflow interruptions
                                                    417 * enabled.
                                                    418 *
000004BA  5800 F5AC                     000005AC   419          L     R0,ARCCPMOV   Get cc/program mask for overflow ints
000004BE  0400                                     420          SPM   R0            Enable Fixed Point Overflow inter.
000004C0  D703 F08C F08C      0000008C  0000008C   421          XC    PCINTCD,PCINTCD    Zero out PC interruption code
                                                    422 *
000004C6  5800 3000                     00000000   423          L     R0,0(,R3)     Initialize minuend
000004CA  5810 5000                     00000000   424          L     R1,0(,R5)     Initialize subtrahend
000004CE  1B01                                     425          SR    R0,R1         Replace minuend with difference
000004D0  5000 7000                     00000000   426          ST    R0,0(,R7)     Store difference
000004D4  D202 8001 F08D      00000001  0000008D   427          MVC   1(3,R8),PCINTCD+1   Save interruption code
000004DA  B222 0000                                428          IPM   R0            Retrieve condition code
000004DE  8800 001C                     0000001C   429          SRL   R0,28         Move CC to low-order r0
000004E2  4200 8000                     00000000   430          STC   R0,0(,R8)     Store condition code
                                                    431 *
000004E6  5800 F5A8                     000005A8   432          L     R0,ARCCPM     Get cc/program mask for no o'flow ints
000004EA  0400                                     433          SPM   R0            Disable Fixed Point Overflow inter.
                                                    434 *
000004EC  4156 5000                     00000000   435          LA    R5,0(R6,R5)   Point to next subtrahend
000004F0  4176 7000                     00000000   436          LA    R7,0(R6,R7)   Point to next difference slot
000004F4  4180 8004                     00000004   437          LA    R8,4(,R8)     Point to next cc-int code slot
000004F8  0649                                     438          BCTR  R4,R9         Loop through subtrahends
                                                    439 *
                                                    440 * End of subtrahends.  Process next minuend
                                                    441 *
000004FA  4136 3000                     00000000   442          LA    R3,0(R6,R3)   Point to next minuend
000004FE  062C                                     443          BCTR  R2,R12        Loop through minuends
                                                    444 *
00000500  07FD                                     445          BR    R13           All converted; return.
                                                    446 *
                                                    447 ********************************************************************
                                                    448 *
                                                    449 * SUB REGISTER (SGR, RRE) - 64-bit subtrahend, 64-minuend, 64-bit sum.
                                                    450 * Result replaces subtrahend in operand 1.
                                                    451 *
                                                    452 ********************************************************************


00000502  B222 0000                                454 SGRTEST  IPM   R0            Get cc, program mask
00000506  5000 F5A8                     000005A8   455          ST    R0,ARCCPM     Save for later disable of ints
0000050A  5000 F5AC                     000005AC   456          ST    R0,ARCCPMOV   Save for overflow enablement
0000050E  9608 F5AC                     000005AC   457          OI    ARCCPMOV,X'08'  Enable fixed point overflow ints
00000512  9823 A000                     00000000   458          LM    R2,R3,0(R10)  Get count and addresses of minuends
00000516  9878 A00C                     0000000C   459          LM    R7,R8,12(R10) Get address of result area and flag area.
0000051A  5860 A014                     00000014   460          L     R6,20(,R10)   Get size of minuends, subtrahends and
                                                    461 *                            ..and results.
0000051E  1222                                     462          LTR   R2,R2         Any test cases?
00000520  078D                                     463          BZR   R13           ..No, return to caller
00000522  0DC0                                     464          BASR  R12,0         Set top of loop
                                                    465 *
                                                    466 * Top of outer loop.  Process next minuend
                                                    467 *
00000524  5840 A000                     00000000   468          L     R4,0(,R10)    Get count of subtrahends
00000528  5850 A008                     00000008   469          L     R5,8(,R10)    Get address of subtrahend table
0000052C  0D90                                     470          BASR  R9,0          Set top of loop
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT
                                                   471 *
0000052E  E300 3000 0004                 00000000  472          LG    R0,0(,R3)     Initialize minuend
00000534  E310 5000 0004                 00000000  473          LG    R1,0(,R5)     Initialize subtrahend
0000053A  B909 0001                               474          SGR   R0,R1         Replace minuend with difference
0000053E  E300 7000 0024                 00000000  475          STG   R0,0(,R7)     Store difference
00000544  B222 0000                               476          IPM   R0            Retrieve condition code
00000548  8800 001C                     0000001C  477          SRL   R0,28         Move CC to low-order r0
0000054C  4200 8000                     00000000  478          STC   R0,0(,R8)     Store condition code
00000550  4176 7000                     00000000  479          LA    R7,0(R6,R7)   Point to next difference slot
00000554  4180 8004                     00000004  480          LA    R8,4(,R8)     Point to next cc-int code slot
                                                   481 *
                                                   482 * Repeat the instruction with Fixed Point Overflow interruptions
                                                   483 * enabled.
                                                   484 *
00000558  5800 F5AC                     000005AC  485          L     R0,ARCCPMOV   Get cc/program mask for overflow ints
0000055C  0400                                    486          SPM   R0            Enable Fixed Point Overflow inter.
0000055E  D703 F08C F08C       0000008C  0000008C  487          XC    PCINTCD,PCINTCD    Zero out PC interruption code
                                                   488 *
00000564  E300 3000 0004                 00000000  489          LG    R0,0(,R3)     Initialize minuend
0000056A  E310 5000 0004                 00000000  490          LG    R1,0(,R5)     Initialize subtrahend
00000570  B909 0001                               491          SGR   R0,R1         Replace minuend with difference
00000574  E300 7000 0024                 00000000  492          STG   R0,0(,R7)     Store difference
0000057A  D202 8001 F08D       00000001  0000008D  493          MVC   1(3,R8),PCINTCD+1   Save interruption code
00000580  B222 0000                               494          IPM   R0            Retrieve condition code
00000584  8800 001C                     0000001C  495          SRL   R0,28         Move CC to low-order r0
00000588  4200 8000                     00000000  496          STC   R0,0(,R8)     Store condition code
                                                   497 *
0000058C  5800 F5A8                     000005A8  498          L     R0,ARCCPM     Get cc/program mask for no o'flow ints
00000590  0400                                    499          SPM   R0            Disable Fixed Point Overflow inter.
                                                   500 *
00000592  4156 5000                     00000000  501          LA    R5,0(R6,R5)   Point to next subtrahend
00000596  4176 7000                     00000000  502          LA    R7,0(R6,R7)   Point to next difference slot
0000059A  4180 8004                     00000004  503          LA    R8,4(,R8)     Point to next cc-int code slot
0000059E  0649                                    504          BCTR  R4,R9         Loop through subtrahends
                                                   505 *
                                                   506 * End of subtrahends.  Process next minuend
                                                   507 *
000005A0  4136 3000                     00000000  508          LA    R3,0(R6,R3)   Point to next minuend
000005A4  062C                                    509          BCTR  R2,R12        Loop through minuends
                                                   510 *
000005A6  07FD                                    511          BR    R13           All converted; return.
                                                   512 *
000005A8  00000000                                513 ARCCPM   DC    F'0'          Savearea for cc/program mask
000005AC  00000000                                514 ARCCPMOV DC    F'0'          cc/program mask with interupts enabled
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT

                                                    516 ********************************************************************************
                                                    517 *
                                                    518 * Integer inputs.  The same values are used for 32-bit addends /
                                                    519 * subtrahends and augends / minuends.  Each addend is added to each
                                                    520 * augend, and each subtrahend is subtracted from each minuend.
                                                    521 *
                                                    522 * N.B., the number of 32-bit and 64-bit test values must be the same.
                                                    523 *
                                                    524 ********************************************************************************



                                                    526 *
                                                    527 * 32-bit test inputs.
                                                    528 *
000005B0                                            529 A32VALS  DS    0D                      32-bit operands
000005B0  7FFFFFFF                                  530          DC    F'2147483647'           32-bit max pos. int.     ( M)
000005B4  00000001                                  531          DC    F'1'
000005B8  00000000                                  532          DC    F'0'
000005BC  FFFFFFFF                                  533          DC    F'-1'
000005C0  80000001                                  534          DC    F'-2147483647'          32-bit max neg. int. + 1 (-M)
000005C4  80000000                                  535          DC    F'-2147483648'          32-bit max neg. int.     (-M-1)
                                                    536 *
                          00000018  00000001        537 VALCT    EQU   *-A32VALS               Count of integers in list * 4

                                                    539 *
                                                    540 * 64-bit test inputs.
                                                    541 *
000005C8                                            542 A64VALS  DS    0D                      64-bit operands
000005C8  7FFFFFFF FFFFFFFF                         543          DC    D'9223372036854775807'  64-bit max pos. int.     ( G)
000005D0  00000000 00000001                         544          DC    D'1'
000005D8  00000000 00000000                         545          DC    D'0'
000005E0  FFFFFFFF FFFFFFFF                         546          DC    D'-1'
000005E8  80000000 00000001                         547          DC    D'-9223372036854775807' 64-bit max neg. int. + 1 (-G)
000005F0  80000000 00000000                         548          DC    D'-9223372036854775808' 64-bit max neg. int.     (-G-1)
                            00000030  00000001      549 VALCT64  EQU   *-A64VALS               Count of integers in list * 8
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT
                                                      551 ********************************************************************************
                                                      552 *                    ACTUAL results saved here
                                                      553 ********************************************************************************
                                                      554 *
                                                      555 *                 Locations for ACTUAL results
                                                      556 *
                              00001000  00000001      557 ARSUM    EQU   STRTLABL+X'1000'    AR results
                                                      558 *                                  ..72 results used
                              00002000  00000001      559 ARFLG    EQU   STRTLABL+X'2000'    Condition and interrupt codes
                                                      560 *                                  ..72 results used
                              00001400  00000001      561 AGRSUM   EQU   STRTLABL+X'1400'    AGR results
                                                      562 *                                  ..72 results used
                              00002400  00000001      563 AGRFLG   EQU   STRTLABL+X'2400'    Condition and interrupt codes
                                                      564 *                                  ..72 results used
                              00001800  00000001      565 SRSUM    EQU   STRTLABL+X'1800'    SR results
                                                      566 *                                  ..72 results used
                              00002800  00000001      567 SRFLG    EQU   STRTLABL+X'2800'    Condition and interrupt codes
                                                      568 *                                  ..72 results used
                              00001C00  00000001      569 SGRSUM   EQU   STRTLABL+X'1C00'    SGR results
                                                      570 *                                  ..72 results used
                              00002C00  00000001      571 SGRFLG   EQU   STRTLABL+X'2C00'    Condition and interrupt codes
                                                      572 *                                  ..72 results used
                              00003000  00000001      573 ENDRES   EQU   STRTLABL+X'3000'    next location for results
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT

                                                      575 ************************************************************************
                                                      576 *                      EXPECTED results
                                                      577 ************************************************************************
                                                      578 *
000005F8                        000005F8  00010000    579          ORG   STRTLABL+X'10000'   (FAR past end of actual results)
                                                      580 *
                                00010000  00000001    581 ARSUM_GOOD EQU *
00010000  D47EF7C6 4B4BC6C6                           582  DC CL64'M=7F..FF+   M=7F..FF,   M=7F..FF+   1=00..01'
00010040  FFFFFFFE FFFFFFFE                           583  DC XL16'FFFFFFFEFFFFFFFE8000000080000000'
00010050  D47EF7C6 4B4BC6C6                           584  DC CL64'M=7F..FF+   0=00..00,   M=7F..FF+  -1=FF..FF'
00010090  7FFFFFFF 7FFFFFFF                           585  DC XL16'7FFFFFFF7FFFFFFF7FFFFFFE7FFFFFFE'
000100A0  D47EF7C6 4B4BC6C6                           586  DC CL64'M=7F..FF+-M  =80..01,   M=7F..FF+-M-1=80..00'
000100E0  00000000 00000000                           587  DC XL16'0000000000000000FFFFFFFFFFFFFFFF'
000100F0  F17EF0F0 4B4BF0F1                           588  DC CL64'1=00..01+   M=7F..FF,   1=00..01+   1=00..01'
00010130  80000000 80000000                           589  DC XL16'8000000080000000000000200000002'
00010140  F17EF0F0 4B4BF0F1                           590  DC CL64'1=00..01+   0=00..00,   1=00..01+  -1=FF..FF'
00010180  00000001 00000001                           591  DC XL16'0000000100000001000000000000000'
00010190  F17EF0F0 4B4BF0F1                           592  DC CL64'1=00..01+-M  =80..01,   1=00..01+-M-1=80..00'
000101D0  80000002 80000002                           593  DC XL16'8000000280000002800000180000001'
000101E0  F07EF0F0 4B4BF0F0                           594  DC CL64'0=00..00+   M=7F..FF,   0=00..00+   1=00..01'
00010220  7FFFFFFF 7FFFFFFF                           595  DC XL16'7FFFFFFF7FFFFFFF0000000100000001'
00010230  F07EF0F0 4B4BF0F0                           596  DC CL64'0=00..00+   0=00..00,   0=00..00+  -1=FF..FF'
00010270  00000000 00000000                           597  DC XL16'0000000000000000FFFFFFFFFFFFFFFF'
00010280  F07EF0F0 4B4BF0F0                           598  DC CL64'0=00..00+-M  =80..01,   0=00..00+-M-1=80..00'
000102C0  80000001 80000001                           599  DC XL16'8000000180000001800000080000000'
000102D0  60F17EC6 C64B4BC6                           600  DC CL64'-1=FF..FF+   M=7F..FF,  -1=FF..FF+   1=00..01'
00010310  7FFFFFFE 7FFFFFFE                           601  DC XL16'7FFFFFFE7FFFFFFE0000000000000000'
00010320  60F17EC6 C64B4BC6                           602  DC CL64'-1=FF..FF+   0=00..00,  -1=FF..FF+  -1=FF..FF'
00010360  FFFFFFFF FFFFFFFF                           603  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFE'
00010370  60F17EC6 C64B4BC6                           604  DC CL64'-1=FF..FF+-M  =80..01,  -1=FF..FF+-M-1=80..00'
000103B0  80000000 80000000                           605  DC XL16'8000000800000007FFFFFF7FFFFFFF'
000103C0  60D44040 7EF8F04B                           606  DC CL64'-M  =80..01+   M=7F..FF,-M  =80..01+   1=00..01'
00010400  00000000 00000000                           607  DC XL16'0000000000000000800000280000002'
00010410  60D44040 7EF8F04B                           608  DC CL64'-M  =80..01+   0=00..00,-M  =80..01+  -1=FF..FF'
00010450  80000001 80000001                           609  DC XL16'8000000180000001800000080000000'
00010460  60D44040 7EF8F04B                           610  DC CL64'-M  =80..01+-M  =80..01,-M  =80..01+-M-1=80..00'
000104A0  00000002 00000002                           611  DC XL16'0000000200000002000000100000001'
000104B0  60D460F1 7EF8F04B                           612  DC CL64'-M-1=80..00+   M=7F..FF,-M-1=80..00+  +1=00..01'
000104F0  FFFFFFFF FFFFFFFF                           613  DC XL16'FFFFFFFFFFFFFFFF8000000180000001'
00010500  60D460F1 7EF8F04B                           614  DC CL64'-M-1=80..00+   0=00..00,-M-1=80..00+  -1=FF..FF'
00010540  80000000 80000000                           615  DC XL16'8000000800000007FFFFFF7FFFFFFF'
00010550  60D460F1 7EF8F04B                           616  DC CL64'-M-1=80..00+-M  =80..01,-M-1=80..00+-M-1=80..00'
00010590  00000001 00000001                           617  DC XL16'0000000100000001000000000000000'
                                00000012  00000001    618 ARSUM_NUM EQU (*-ARSUM_GOOD)/80
                                                      619 *
                                                      620 *
                                000105A0  00000001    621 ARFLG_GOOD EQU *
000105A0  83836186 97964040                           622  DC CL64'cc/fpo    M=EF..FF+   M=7F..FF,   M=EF..FF+   1=00..01'
000105E0  03000000 03020008                           623  DC XL16'0300000003020008030000003020008'
000105F0  83836186 97964040                           624  DC CL64'cc/fpo    M=EF..FF+   0=00..00,   M=EF..FF+  -1=FF..FF'
00010630  02000000 02000000                           625  DC XL16'0200000002000000020000002000000'
00010640  83836186 97964040                           626  DC CL64'cc/fpo    M=EF..FF+-M  =80..01,   M=EF..FF+-M-1=80..00'
00010680  00000000 00000000                           627  DC XL16'0000000000000000010000001000000'
00010690  83836186 97964040                           628  DC CL64'cc/fpo    1=00..01+   M=7F..FF,   1=00..01+   1=00..01'
000106D0  03000000 03020008                           629  DC XL16'0300000003020008020000002000000'
000106E0  83836186 97964040                           630  DC CL64'cc/fpo    1=00..01+   0=00..00,   1=00..01+  -1=FF..FF'
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT

00010720  02000000 02000000                          631  DC XL16'02000000020000000000000000000000'
00010730  83836186 97964040                          632  DC CL64'cc/fpo    1=00..01+-M  =80..01,   1=00..01+-M-1=80..00'
00010770  01000000 01000000                          633  DC XL16'01000000010000000100000001000000'
00010780  83836186 97964040                          634  DC CL64'cc/fpo    0=00..00+   M=7F..FF,   0=00..00+   1=00..01'
000107C0  02000000 02000000                          635  DC XL16'02000000020000000200000002000000'
000107D0  83836186 97964040                          636  DC CL64'cc/fpo    0=00..00+   0=00..00,   0=00..00+  -1=FF..FF'
00010810  00000000 00000000                          637  DC XL16'00000000000000000100000001000000'
00010820  83836186 97964040                          638  DC CL64'cc/fpo    0=00..00+-M  =80..01,   0=00..00+-M-1=80..00'
00010860  01000000 01000000                          639  DC XL16'01000000010000000100000001000000'
00010870  83836186 97964040                          640  DC CL64'cc/fpo   -1=FF..FF+   M=7F..FF,  -1=FF..FF+   1=00..01'
000108B0  02000000 02000000                          641  DC XL16'02000000020000000000000000000000'
000108C0  83836186 97964040                          642  DC CL64'cc/fpo   -1=FF..FF+   0=00..00,  -1=FF..FF+  -1=FF..FF'
00010900  01000000 01000000                          643  DC XL16'01000000010000000100000001000000'
00010910  83836186 97964040                          644  DC CL64'cc/fpo   -1=FF..FF+-M  =80..01,  -1=FF..FF+-M-1=80..00'
00010950  01000000 01000000                          645  DC XL16'01000000010000000300000003020008'
00010960  83836186 97964060                          646  DC CL64'cc/fpo -M  =80..01+   M=7F..FF,-M  =80..01+   1=00..01'
000109A0  00000000 00000000                          647  DC XL16'00000000000000000100000001000000'
000109B0  83836186 97964060                          648  DC CL64'cc/fpo -M  =80..01+   0=00..00,-M  =80..01+  -1=FF..FF'
000109F0  01000000 01000000                          649  DC XL16'01000000010000000100000001000000'
00010A00  83836186 97964060                          650  DC CL64'cc/fpo -M  =80..01+-M  =80..01,-M  =80..01+-M-1=80..00'
00010A40  03000000 03020008                          651  DC XL16'03000000030200080300000003020008'
00010A50  83836186 97964060                          652  DC CL64'cc/fpo -M-1=80..00+   M=7F..FF,-M-1=80..00+  +1=00..01'
00010A90  01000000 01000000                          653  DC XL16'01000000010000000100000001000000'
00010AA0  83836186 97964060                          654  DC CL64'cc/fpo -M-1=80..00+   0=00..00,-M-1=80..00+  -1=FF..FF'
00010AE0  01000000 01000000                          655  DC XL16'01000000010000000300000003020008'
00010AF0  83836186 97964060                          656  DC CL64'cc/fpo -M-1=80..00+-M  =80..01,-M-1=80..00+-M-1=80..00'
00010B30  03000000 03020008                          657  DC XL16'03000000030200080300000003020008'
                            00000012  00000001    658 ARFLG_NUM EQU (*-ARFLG_GOOD)/80
                                                  659 *
                                                  660 *
                            00010B40  00000001    661 AGRSUM_GOOD EQU *
00010B40  C77EF7C6 4B4BC6C6                          662  DC CL64'G=7F..FF+   G=7F..FF'
00010B80  FFFFFFFF FFFFFFFE                          663  DC XL16'FFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFE'
00010B90  C77EF7C6 4B4BC6C6                          664  DC CL64'G=7F..FF+   1=00..01'
00010BD0  80000000 00000000                          665  DC XL16'80000000000000008000000000000000'
00010BE0  C77EF7C6 4B4BC6C6                          666  DC CL64'G=7F..FF+   0=00..00'
00010C20  7FFFFFFF FFFFFFFF                          667  DC XL16'7FFFFFFFFFFFFFFF7FFFFFFFFFFFFFFF'
00010C30  C77EF7C6 4B4BC6C6                          668  DC CL64'G=7F..FF+  -1=FF..FF'
00010C70  7FFFFFFF FFFFFFFE                          669  DC XL16'7FFFFFFFFFFFFFFE7FFFFFFFFFFFFFFE'
00010C80  C77EF7C6 4B4BC6C6                          670  DC CL64'G=7F..FF+-G  =80..01'
00010CC0  00000000 00000000                          671  DC XL16'00000000000000000000000000000000'
00010CD0  C77EF7C6 4B4BC6C6                          672  DC CL64'G=7F..FF+-G-1=80..00'
00010D10  FFFFFFFF FFFFFFFF                          673  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF'
00010D20  F17EF0F0 4B4BF0F1                          674  DC CL64'1=00..01+   G=7F..FF'
00010D60  80000000 00000000                          675  DC XL16'80000000000000008000000000000000'
00010D70  F17EF0F0 4B4BF0F1                          676  DC CL64'1=00..01+   1=00..01'
00010DB0  00000000 00000002                          677  DC XL16'00000000000000002000000000000002'
00010DC0  F17EF0F0 4B4BF0F1                          678  DC CL64'1=00..01+   0=00..00'
00010E00  00000000 00000001                          679  DC XL16'00000000000000001000000000000001'
00010E10  F17EF0F0 4B4BF0F1                          680  DC CL64'1=00..01+  -1=FF..FF'
00010E50  00000000 00000000                          681  DC XL16'00000000000000000000000000000000'
00010E60  F17EF0F0 4B4BF0F1                          682  DC CL64'1=00..01+-G  =80..01'
00010EA0  80000000 00000002                          683  DC XL16'80000000000000002800000000000002'
00010EB0  F17EF0F0 4B4BF0F1                          684  DC CL64'1=00..01+-G-1=80..00'
00010EF0  80000000 00000001                          685  DC XL16'80000000000000001800000000000001'
00010F00  F07EF0F0 4B4BF0F0                          686  DC CL64'0=00..00+   G=7F..FF'
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT

00010F40  7FFFFFFF FFFFFFFF                          687  DC XL16'7FFFFFFFFFFFFFFF7FFFFFFFFFFFFFFF'
00010F50  F07EF0F0 4B4BF0F0                          688  DC CL64'0=00..00+   1=00..01'
00010F90  00000000 00000001                          689  DC XL16'00000000000000010000000000000001'
00010FA0  F07EF0F0 4B4BF0F0                          690  DC CL64'0=00..00+   0=00..00'
00010FE0  00000000 00000000                          691  DC XL16'00000000000000000000000000000000'
00010FF0  F07EF0F0 4B4BF0F0                          692  DC CL64'0=00..00+  -1=FF..FF'
00011030  FFFFFFFF FFFFFFFF                          693  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF'
00011040  F07EF0F0 4B4BF0F0                          694  DC CL64'0=00..00+-G  =80..01'
00011080  80000000 00000001                          695  DC XL16'80000000000000018000000000000001'
00011090  F07EF0F0 4B4BF0F0                          696  DC CL64'0=00..00+-M-1=80..00'
000110D0  80000000 00000000                          697  DC XL16'80000000000000008000000000000000'
000110E0  60F17EC6 C64B4BC6                          698  DC CL64'-1=FF..FF+   G=7F..FF'
00011120  7FFFFFFF FFFFFFFE                          699  DC XL16'7FFFFFFFFFFFFFFE7FFFFFFFFFFFFFFE'
00011130  60F17EC6 C64B4BC6                          700  DC CL64'-1=FF..FF+   1=00..01'
00011170  00000000 00000000                          701  DC XL16'00000000000000000000000000000000'
00011180  60F17EC6 C64B4BC6                          702  DC CL64'-1=FF..FF+   0=00..00'
000111C0  FFFFFFFF FFFFFFFF                          703  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF'
000111D0  60F17EC6 C64B4BC6                          704  DC CL64'-1=FF..FF+  -1=FF..FF'
00011210  FFFFFFFF FFFFFFFE                          705  DC XL16'FFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFE'
00011220  60F17EC6 C64B4BC6                          706  DC CL64'-1=FF..FF+-G  =80..01,'
00011260  80000000 00000000                          707  DC XL16'80000000000000008000000000000000'
00011270  60F17EC6 C64B4BC6                          708  DC CL64'-1=FF..FF+-G-1=80..00'
000112B0  7FFFFFFF FFFFFFFF                          709  DC XL16'7FFFFFFFFFFFFFFF7FFFFFFFFFFFFFFF'
000112C0  60C74040 7EF8F04B                          710  DC CL64'-G  =80..01+   G=7F..FF'
00011300  00000000 00000000                          711  DC XL16'00000000000000000000000000000000'
00011310  60C74040 7EF8F04B                          712  DC CL64'-G  =80..01+   1=00..01'
00011350  80000000 00000002                          713  DC XL16'80000000000000028000000000000002'
00011360  60C74040 7EF8F04B                          714  DC CL64'-G  =80..01+   0=00..00'
000113A0  80000000 00000001                          715  DC XL16'80000000000000018000000000000001'
000113B0  60C74040 7EF8F04B                          716  DC CL64'-G  =80..01+  -1=FF..FF'
000113F0  80000000 00000000                          717  DC XL16'80000000000000008000000000000000'
00011400  60C74040 7EF8F04B                          718  DC CL64'-G  =80..01+-G  =80..01'
00011440  00000000 00000002                          719  DC XL16'00000000000000020000000000000002'
00011450  60C74040 7EF8F04B                          720  DC CL64'-G  =80..01+-G-1=80..00'
00011490  00000000 00000001                          721  DC XL16'00000000000000010000000000000001'
000114A0  60C760F1 7EF8F04B                          722  DC CL64'-G-1=80..00+   G=7F..FF'
000114E0  FFFFFFFF FFFFFFFF                          723  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF'
000114F0  60C760F1 7EF8F04B                          724  DC CL64'-G-1=80..00+  +1=00..01'
00011530  80000000 00000001                          725  DC XL16'80000000000000018000000000000001'
00011540  60C760F1 7EF8F04B                          726  DC CL64'-G-1=80..00+   0=00..00'
00011580  80000000 00000000                          727  DC XL16'80000000000000008000000000000000'
00011590  60C760F1 7EF8F04B                          728  DC CL64'-G-1=80..00+  -1=FF..FF'
000115D0  7FFFFFFF FFFFFFFF                          729  DC XL16'7FFFFFFFFFFFFFFF7FFFFFFFFFFFFFFF'
000115E0  60C760F1 7EF8F04B                          730  DC CL64'-G-1=80..00+-M  =80..01'
00011620  00000000 00000001                          731  DC XL16'00000000000000010000000000000001'
00011630  60C760F1 7EF8F04B                          732  DC CL64'-G-1=80..00+-G-1=80..00'
00011670  00000000 00000000                          733  DC XL16'00000000000000000000000000000000'
                            00000024  00000001      734 AGRSUM_NUM EQU (*-AGRSUM_GOOD)/80
                                                     735 *
                                                     736 *
                            00011680  00000001      737 AGRFLG_GOOD EQU *
00011680  83836186 97964040                          738  DC CL64'cc/fpo    G=EF..FF+   G=7F..FF,   G=EF..FF+   1=00..01'
000116C0  03000000 03040008                          739  DC XL16'03000000304000803000000003040008'
000116D0  83836186 97964040                          740  DC CL64'cc/fpo    G=EF..FF+   0=00..00,   G=EF..FF+  -1=FF..FF'
00011710  02000000 02000000                          741  DC XL16'02000000200000002000000002000000'
00011720  83836186 97964040                          742  DC CL64'cc/fpo    M=EF..FF+-G  =80..01,   G=EF..FF+-G-1=80..00'
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT

00011760  00000000 00000000                          743  DC XL16'00000000000000000100000001000000'
00011770  83836186 97964040                          744  DC CL64'cc/fpo    1=00..01+   G=7F..FF,   1=00..01+   1=00..01'
000117B0  03000000 03040008                          745  DC XL16'03000000030400080200000002000000'
000117C0  83836186 97964040                          746  DC CL64'cc/fpo    1=00..01+   0=00..00,   1=00..01+  -1=FF..FF'
00011800  02000000 02000000                          747  DC XL16'02000000020000000000000000000000'
00011810  83836186 97964040                          748  DC CL64'cc/fpo    1=00..01+-G  =80..01,   1=00..01+-G-1=80..00'
00011850  01000000 01000000                          749  DC XL16'01000000010000000100000001000000'
00011860  83836186 97964040                          750  DC CL64'cc/fpo    0=00..00+   M=7F..FF,   0=00..00+   1=00..01'
000118A0  02000000 02000000                          751  DC XL16'02000000020000000200000002000000'
000118B0  83836186 97964040                          752  DC CL64'cc/fpo    0=00..00+   0=00..00,   0=00..00+  -1=FF..FF'
000118F0  00000000 00000000                          753  DC XL16'00000000000000000100000001000000'
00011900  83836186 97964040                          754  DC CL64'cc/fpo    0=00..00+-G  =80..01,   0=00..00+-G-1=80..00'
00011940  01000000 01000000                          755  DC XL16'01000000010000000100000001000000'
00011950  83836186 97964040                          756  DC CL64'cc/fpo   -1=FF..FF+   M=7F..FF,  -1=FF..FF+   1=00..01'
00011990  02000000 02000000                          757  DC XL16'02000000020000000000000000000000'
000119A0  83836186 97964040                          758  DC CL64'cc/fpo   -1=FF..FF+   0=00..00,  -1=FF..FF+  -1=FF..FF'
000119E0  01000000 01000000                          759  DC XL16'01000000010000000100000001000000'
000119F0  83836186 97964040                          760  DC CL64'cc/fpo   -1=FF..FF+-G  =80..01,  -1=FF..FF+-G-1=80..00'
00011A30  01000000 01000000                          761  DC XL16'01000000010000000300000003040008'
00011A40  83836186 97964060                          762  DC CL64'cc/fpo -G  =80..01+   M=7F..FF,-G  =80..01+   1=00..01'
00011A80  00000000 00000000                          763  DC XL16'00000000000000000100000001000000'
00011A90  83836186 97964060                          764  DC CL64'cc/fpo -G  =80..01+   0=00..00,-G  =80..01+  -1=FF..FF'
00011AD0  01000000 01000000                          765  DC XL16'01000000010000000100000001000000'
00011AE0  83836186 97964060                          766  DC CL64'cc/fpo -G  =80..01+-G  =80..01,-G  =80..01+-G-1=80..00'
00011B20  03000000 03040008                          767  DC XL16'03000000030400080300000003040008'
00011B30  83836186 97964060                          768  DC CL64'cc/fpo -G-1=80..00+   M=7F..FF,-G-1=80..00+  +1=00..01'
00011B70  01000000 01000000                          769  DC XL16'01000000010000000100000001000000'
00011B80  83836186 97964060                          770  DC CL64'cc/fpo -G-1=80..00+   0=00..00,-G-1=80..00+  -1=FF..FF'
00011BC0  01000000 01000000                          771  DC XL16'01000000010000000300000003040008'
00011BD0  83836186 97964060                          772  DC CL64'cc/fpo -G-1=80..00+-G  =80..01,-G-1=80..00+-G-1=80..00'
00011C10  03000000 03040008                          773  DC XL16'03000000030400080300000003040008'
                            00000012  00000001      774 AGRFLG_NUM EQU (*-AGRFLG_GOOD)/80
                                                    775 *
                                                    776 *
                            00011C20  00000001      777 SRSUM_GOOD EQU *
00011C20  D47EF7C6 4B4BC6C6                          778  DC CL64'M=7F..FF-   M=7F..FF,   M=7F..FF-   1=00..01'
00011C60  00000000 00000000                          779  DC XL16'00000000000000007FFFFFFE7FFFFFFE'
00011C70  D47EF7C6 4B4BC6C6                          780  DC CL64'M=7F..FF-   0=00..00,   M=7F..FF-  -1=FF..FF'
00011CB0  7FFFFFFF 7FFFFFFF                          781  DC XL16'7FFFFFFF7FFFFFFF8000000080000000'
00011CC0  D47EF7C6 4B4BC6C6                          782  DC CL64'M=7F..FF--M  =80..01,   M=7F..FF--M-1=80..00'
00011D00  FFFFFFFE FFFFFFFE                          783  DC XL16'FFFFFFFEFFFFFFFEFFFFFFFFFFFFFFFF'
00011D10  F17EF0F0 4B4BF0F1                          784  DC CL64'1=00..01-   M=7F..FF,   1=00..01-   1=00..01'
00011D50  80000002 80000002                          785  DC XL16'8000000280000002000000000000000'
00011D60  F17EF0F0 4B4BF0F1                          786  DC CL64'1=00..01-   0=00..00,   1=00..01-  -1=FF..FF'
00011DA0  00000001 00000001                          787  DC XL16'00000001000000010000000200000002'
00011DB0  F17EF0F0 4B4BF0F1                          788  DC CL64'1=00..01--M  =80..01,   1=00..01--M-1=80..00'
00011DF0  80000000 80000000                          789  DC XL16'80000000800000008000000180000001'
00011E00  F07EF0F0 4B4BF0F0                          790  DC CL64'0=00..00-   M=7F..FF,   0=00..00-   1=00..01'
00011E40  80000001 80000001                          791  DC XL16'8000000180000001FFFFFFFFFFFFFFFF'
00011E50  F07EF0F0 4B4BF0F0                          792  DC CL64'0=00..00-   0=00..00,   0=00..00-  -1=FF..FF'
00011E90  00000000 00000000                          793  DC XL16'00000000000000000000000100000001'
00011EA0  F07EF0F0 4B4BF0F0                          794  DC CL64'0=00..00--M  =80..01,   0=00..00--M-1=80..00'
00011EE0  7FFFFFFF 7FFFFFFF                          795  DC XL16'7FFFFFFF7FFFFFFF8000000080000000'
00011EF0  60F17EC6 C64B4BC6                          796  DC CL64'-1=FF..FF-   M=7F..FF,  -1=FF..FF-   1=00..01'
00011F30  80000000 80000000                          797  DC XL16'8000000080000000FFFFFFFEFFFFFFFE'
00011F40  60F17EC6 C64B4BC6                          798  DC CL64'-1=FF..FF-   0=00..00,  -1=FF..FF-  -1=FF..FF'
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT

00011F80  FFFFFFFF FFFFFFFF                           799  DC XL16'FFFFFFFFFFFFFFFF0000000000000000'
00011F90  60F17EC6 C64B4BC6                           800  DC CL64'-1=FF..FF--M  =80..01,  -1=FF..FF--M-1=80..00'
00011FD0  7FFFFFFE 7FFFFFFE                           801  DC XL16'7FFFFFFE7FFFFFFE7FFFFFF7FFFFFFF'
00011FE0  60D44040 7EF8F04B                           802  DC CL64'-M  =80..01-   M=7F..FF,-M  =80..01-   1=00..01'
00012020  00000002 00000002                           803  DC XL16'0000000200000002800000080000000'
00012030  60D44040 7EF8F04B                           804  DC CL64'-M  =80..01-   0=00..00,-M  =80..01-  -1=FF..FF'
00012070  80000001 80000001                           805  DC XL16'8000000180000001800000280000002'
00012080  60D44040 7EF8F04B                           806  DC CL64'-M  =80..01--M  =80..01,-M  =80..01--M-1=80..00'
000120C0  00000000 00000000                           807  DC XL16'00000000000000000000000100000001'
000120D0  60D460F1 7EF8F04B                           808  DC CL64'-M-1=80..00-   M=7F..FF,-M-1=80..00-  +1=00..01'
00012110  00000001 00000001                           809  DC XL16'0000000100000017FFFFFFF7FFFFFFF'
00012120  60D460F1 7EF8F04B                           810  DC CL64'-M-1=80..00-   0=00..00,-M-1=80..00-  -1=FF..FF'
00012160  80000000 80000000                           811  DC XL16'8000000080000000800000180000001'
00012170  60D460F1 7EF8F04B                           812  DC CL64'-M-1=80..00--M  =80..01,-M-1=80..00--M-1=80..00'
000121B0  FFFFFFFF FFFFFFFF                           813  DC XL16'FFFFFFFFFFFFFFFF0000000000000000'
                            00000012  00000001    814 SRSUM_NUM EQU (*-SRSUM_GOOD)/80
                                                  815 *
                                                  816 *
                            000121C0  00000001    817 SRFLG_GOOD EQU *
000121C0  83836186 97964040                           818  DC CL64'cc/fpo    M=EF..FF-   M=7F..FF,   M=EF..FF-   1=00..01'
00012200  00000000 00000000                           819  DC XL16'00000000000000000200000002000000'
00012210  83836186 97964040                           820  DC CL64'cc/fpo    M=EF..FF-   0=00..00,   M=EF..FF-  -1=FF..FF'
00012250  02000000 02000000                           821  DC XL16'02000000020000000300000003020008'
00012260  83836186 97964040                           822  DC CL64'cc/fpo    M=EF..FF--M  =80..01,   M=EF..FF--M-1=80..00'
000122A0  03000000 03020008                           823  DC XL16'03000000302000803000000003020008'
000122B0  83836186 97964040                           824  DC CL64'cc/fpo    1=00..01-   M=7F..FF,   1=00..01-   1=00..01'
000122F0  01000000 01000000                           825  DC XL16'01000000010000000000000000000000'
00012300  83836186 97964040                           826  DC CL64'cc/fpo    1=00..01-   0=00..00,   1=00..01-  -1=FF..FF'
00012340  02000000 02000000                           827  DC XL16'02000000020000000200000002000000'
00012350  83836186 97964040                           828  DC CL64'cc/fpo    1=00..01--M  =80..01,   1=00..01--M-1=80..00'
00012390  03000000 03020008                           829  DC XL16'03000000302000803000000003020008'
000123A0  83836186 97964040                           830  DC CL64'cc/fpo    0=00..00-   M=7F..FF,   0=00..00-   1=00..01'
000123E0  01000000 01000000                           831  DC XL16'01000000010000000100000001000000'
000123F0  83836186 97964040                           832  DC CL64'cc/fpo    0=00..00-   0=00..00,   0=00..00-  -1=FF..FF'
00012430  00000000 00000000                           833  DC XL16'00000000000000000200000002000000'
00012440  83836186 97964040                           834  DC CL64'cc/fpo    0=00..00--M  =80..01,   0=00..00--M-1=80..00'
00012480  02000000 02000000                           835  DC XL16'02000000020000000300000003020008'
00012490  83836186 97964040                           836  DC CL64'cc/fpo   -1=FF..FF-   M=7F..FF,  -1=FF..FF-   1=00..01'
000124D0  01000000 01000000                           837  DC XL16'01000000010000000100000001000000'
000124E0  83836186 97964040                           838  DC CL64'cc/fpo   -1=FF..FF-   0=00..00,  -1=FF..FF-  -1=FF..FF'
00012520  01000000 01000000                           839  DC XL16'01000000010000000000000000000000'
00012530  83836186 97964040                           840  DC CL64'cc/fpo   -1=FF..FF--M  =80..01,  -1=FF..FF--M-1=80..00'
00012570  02000000 02000000                           841  DC XL16'02000000020000000200000002000000'
00012580  83836186 97964060                           842  DC CL64'cc/fpo -M  =80..01-   M=7F..FF,-M  =80..01-   1=00..01'
000125C0  03000000 03020008                           843  DC XL16'03000000302000801000000001000000'
000125D0  83836186 97964060                           844  DC CL64'cc/fpo -M  =80..01-   0=00..00,-M  =80..01-  -1=FF..FF'
00012610  01000000 01000000                           845  DC XL16'01000000010000000100000001000000'
00012620  83836186 97964060                           846  DC CL64'cc/fpo -M  =80..01--M  =80..01,-M  =80..01--M-1=80..00'
00012660  00000000 00000000                           847  DC XL16'00000000000000000200000002000000'
00012670  83836186 97964060                           848  DC CL64'cc/fpo -M-1=80..00-   M=7F..FF,-M-1=80..00-  +1=00..01'
000126B0  03000000 03020008                           849  DC XL16'03000000302000803000000003020008'
000126C0  83836186 97964060                           850  DC CL64'cc/fpo -M-1=80..00-   0=00..00,-M-1=80..00-  -1=FF..FF'
00012700  01000000 01000000                           851  DC XL16'01000000010000000100000001000000'
00012710  83836186 97964060                           852  DC CL64'cc/fpo -M-1=80..00--M  =80..01,-M-1=80..00--M-1=80..00'
00012750  01000000 01000000                           853  DC XL16'01000000010000000000000000000000'
                            00000012  00000001    854 SRFLG_NUM EQU (*-SRFLG_GOOD)/80
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

                                                     855 *
                                                     856 *
                              00012760  00000001   857 SGRSUM_GOOD EQU *
00012760  C77EF7C6 4B4BC6C6                          858  DC CL64'G=7F..FF-   G=7F..FF'
000127A0  00000000 00000000                          859  DC XL16'00000000000000000000000000000000'
000127B0  C77EF7C6 4B4BC6C6                          860  DC CL64'G=7F..FF-   1=00..01'
000127F0  7FFFFFFF FFFFFFFE                          861  DC XL16'7FFFFFFFFFFFFFFE7FFFFFFFFFFFFFFE'
00012800  C77EF7C6 4B4BC6C6                          862  DC CL64'G=7F..FF-   0=00..00'
00012840  7FFFFFFF FFFFFFFF                          863  DC XL16'7FFFFFFFFFFFFFFF7FFFFFFFFFFFFFFF'
00012850  C77EF7C6 4B4BC6C6                          864  DC CL64'G=7F..FF-  -1=FF..FF'
00012890  80000000 00000000                          865  DC XL16'80000000000000008000000000000000'
000128A0  C77EF7C6 4B4BC6C6                          866  DC CL64'G=7F..FF--G  =80..01'
000128E0  FFFFFFFF FFFFFFFE                          867  DC XL16'FFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFE'
000128F0  C77EF7C6 4B4BC6C6                          868  DC CL64'G=7F..FF--G-1=80..00'
00012930  FFFFFFFF FFFFFFFF                          869  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF'
00012940  F17EF0F0 4B4BF0F1                          870  DC CL64'1=00..01-   G=7F..FF'
00012980  80000000 00000002                          871  DC XL16'80000000000000028000000000000002'
00012990  F17EF0F0 4B4BF0F1                          872  DC CL64'1=00..01-   1=00..01'
000129D0  00000000 00000000                          873  DC XL16'00000000000000000000000000000000'
000129E0  F17EF0F0 4B4BF0F1                          874  DC CL64'1=00..01-   0=00..00'
00012A20  00000000 00000001                          875  DC XL16'00000000000000010000000000000001'
00012A30  F17EF0F0 4B4BF0F1                          876  DC CL64'1=00..01-  -1=FF..FF'
00012A70  00000000 00000002                          877  DC XL16'00000000000000020000000000000002'
00012A80  F17EF0F0 4B4BF0F1                          878  DC CL64'1=00..01--G  =80..01'
00012AC0  80000000 00000000                          879  DC XL16'80000000000000008000000000000000'
00012AD0  F17EF0F0 4B4BF0F1                          880  DC CL64'1=00..01--G-1=80..00'
00012B10  80000000 00000001                          881  DC XL16'80000000000000018000000000000001'
00012B20  F07EF0F0 4B4BF0F0                          882  DC CL64'0=00..00-   G=7F..FF'
00012B60  80000000 00000001                          883  DC XL16'80000000000000018000000000000001'
00012B70  F07EF0F0 4B4BF0F0                          884  DC CL64'0=00..00-   1=00..01'
00012BB0  FFFFFFFF FFFFFFFF                          885  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF'
00012BC0  F07EF0F0 4B4BF0F0                          886  DC CL64'0=00..00-   0=00..00'
00012C00  00000000 00000000                          887  DC XL16'00000000000000000000000000000000'
00012C10  F07EF0F0 4B4BF0F0                          888  DC CL64'0=00..00-  -1=FF..FF'
00012C50  00000000 00000001                          889  DC XL16'00000000000000010000000000000001'
00012C60  F07EF0F0 4B4BF0F0                          890  DC CL64'0=00..00--G  =80..01'
00012CA0  7FFFFFFF FFFFFFFF                          891  DC XL16'7FFFFFFFFFFFFFFF7FFFFFFFFFFFFFFF'
00012CB0  F07EF0F0 4B4BF0F0                          892  DC CL64'0=00..00--G-1=80..00'
00012CF0  80000000 00000000                          893  DC XL16'80000000000000008000000000000000'
00012D00  60F17EC6 C64B4BC6                          894  DC CL64'-1=FF..FF-   G=7F..FF'
00012D40  80000000 00000000                          895  DC XL16'80000000000000008000000000000000'
00012D50  60F17EC6 C64B4BC6                          896  DC CL64'-1=FF..FF-   1=00..01'
00012D90  FFFFFFFF FFFFFFFE                          897  DC XL16'FFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFE'
00012DA0  60F17EC6 C64B4BC6                          898  DC CL64'-1=FF..FF-   0=00..00'
00012DE0  FFFFFFFF FFFFFFFF                          899  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF'
00012DF0  60F17EC6 C64B4BC6                          900  DC CL64'-1=FF..FF-  -1=FF..FF'
00012E30  00000000 00000000                          901  DC XL16'00000000000000000000000000000000'
00012E40  60F17EC6 C64B4BC6                          902  DC CL64'-1=FF..FF--G  =80..01,'
00012E80  7FFFFFFF FFFFFFFE                          903  DC XL16'7FFFFFFFFFFFFFFE7FFFFFFFFFFFFFFE'
00012E90  60F17EC6 C64B4BC6                          904  DC CL64'-1=FF..FF--G-1=80..00'
00012ED0  7FFFFFFF FFFFFFFF                          905  DC XL16'7FFFFFFFFFFFFFFF7FFFFFFFFFFFFFFF'
00012EE0  60C74040 7EF8F04B                          906  DC CL64'-G  =80..01-   G=7F..FF'
00012F20  00000000 00000002                          907  DC XL16'00000000000000020000000000000002'
00012F30  60C74040 7EF8F04B                          908  DC CL64'-G  =80..01-   1=00..01'
00012F70  80000000 00000000                          909  DC XL16'80000000000000008000000000000000'
00012F80  60C74040 7EF8F04B                          910  DC CL64'-G  =80..01-   0=00..00'
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

00012FC0  80000000 00000001                          911  DC XL16'80000000000000018000000000000001'
00012FD0  60C74040 7EF8F04B                          912  DC CL64'-G  =80..01-  -1=FF..FF'
00013010  80000000 00000002                          913  DC XL16'80000000000000028000000000000002'
00013020  60C74040 7EF8F04B                          914  DC CL64'-G  =80..01--G  =80..01'
00013060  00000000 00000000                          915  DC XL16'00000000000000000000000000000000'
00013070  60C74040 7EF8F04B                          916  DC CL64'-G  =80..01--G-1=80..00'
000130B0  00000000 00000001                          917  DC XL16'00000000000000010000000000000001'
000130C0  60C760F1 7EF8F04B                          918  DC CL64'-G-1=80..00-   G=7F..FF'
00013100  00000000 00000001                          919  DC XL16'00000000000000010000000000000001'
00013110  60C760F1 7EF8F04B                          920  DC CL64'-G-1=80..00-  +1=00..01'
00013150  7FFFFFFF FFFFFFFF                          921  DC XL16'7FFFFFFFFFFFFFFF7FFFFFFFFFFFFFFF'
00013160  60C760F1 7EF8F04B                          922  DC CL64'-G-1=80..00-   0=00..00'
000131A0  80000000 00000000                          923  DC XL16'80000000000000008000000000000000'
000131B0  60C760F1 7EF8F04B                          924  DC CL64'-G-1=80..00-  -1=FF..FF'
000131F0  80000000 00000001                          925  DC XL16'80000000000000018000000000000001'
00013200  60C760F1 7EF8F04B                          926  DC CL64'-G-1=80..00--G  =80..01'
00013240  FFFFFFFF FFFFFFFF                          927  DC XL16'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF'
00013250  60C760F1 7EF8F04B                          928  DC CL64'-G-1=80..00--G-1=80..00'
00013290  00000000 00000000                          929  DC XL16'00000000000000000000000000000000'
                            00000024  00000001     930 SGRSUM_NUM EQU (*-SGRSUM_GOOD)/80
                                                    931 *
                                                    932 *
                            000132A0  00000001     933 SGRFLG_GOOD EQU *
000132A0  83836186 97964040                          934  DC CL64'cc/fpo    G=EF..FF-   G=7F..FF,   G=EF..FF-   1=00..01'
000132E0  00000000 00000000                          935  DC XL16'00000000000000000200000002000000'
000132F0  83836186 97964040                          936  DC CL64'cc/fpo    G=EF..FF-   0=00..00,   G=EF..FF-  -1=FF..FF'
00013330  02000000 02000000                          937  DC XL16'02000000020000000300000003040008'
00013340  83836186 97964040                          938  DC CL64'cc/fpo    G=EF..FF--G  =80..01,   G=EF..FF--G-1=80..00'
00013380  03000000 03040008                          939  DC XL16'03000000030400080300000003040008'
00013390  83836186 97964040                          940  DC CL64'cc/fpo    1=00..01-   G=7F..FF,   1=00..01-   1=00..01'
000133D0  01000000 01000000                          941  DC XL16'01000000010000000000000000000000'
000133E0  83836186 97964040                          942  DC CL64'cc/fpo    1=00..01-   0=00..00,   1=00..01-  -1=FF..FF'
00013420  02000000 02000000                          943  DC XL16'02000000020000000200000002000000'
00013430  83836186 97964040                          944  DC CL64'cc/fpo    1=00..01--G  =80..01,   1=00..01--G-1=80..00'
00013470  03000000 03040008                          945  DC XL16'03000000030400080300000003040008'
00013480  83836186 97964040                          946  DC CL64'cc/fpo    0=00..00-   M=7F..FF,   0=00..00-   1=00..01'
000134C0  01000000 01000000                          947  DC XL16'01000000010000000100000001000000'
000134D0  83836186 97964040                          948  DC CL64'cc/fpo    0=00..00-   0=00..00,   0=00..00-  -1=FF..FF'
00013510  00000000 00000000                          949  DC XL16'00000000000000000200000002000000'
00013520  83836186 97964040                          950  DC CL64'cc/fpo    0=00..00--G  =80..01,   0=00..00--G-1=80..00'
00013560  02000000 02000000                          951  DC XL16'02000000020000000300000003040008'
00013570  83836186 97964040                          952  DC CL64'cc/fpo   -1=FF..FF-   G=7F..FF,  -1=FF..FF-   1=00..01'
000135B0  01000000 01000000                          953  DC XL16'01000000010000000100000001000000'
000135C0  83836186 97964040                          954  DC CL64'cc/fpo   -1=FF..FF-   0=00..00,  -1=FF..FF-  -1=FF..FF'
00013600  01000000 01000000                          955  DC XL16'01000000010000000000000000000000'
00013610  83836186 97964040                          956  DC CL64'cc/fpo   -1=FF..FF--G  =80..01,  -1=FF..FF--G-1=80..00'
00013650  02000000 02000000                          957  DC XL16'02000000020000000200000002000000'
00013660  83836186 97964060                          958  DC CL64'cc/fpo -G  =80..01-   G=7F..FF,-G  =80..01-   1=00..01'
000136A0  03000000 03040008                          959  DC XL16'03000000030400080100000001000000'
000136B0  83836186 97964060                          960  DC CL64'cc/fpo -G  =80..01-   0=00..00,-G  =80..01-  -1=FF..FF'
000136F0  01000000 01000000                          961  DC XL16'01000000010000000100000001000000'
00013700  83836186 97964060                          962  DC CL64'cc/fpo -G  =80..01--G  =80..01,-G  =80..01--G-1=80..00'
00013740  00000000 00000000                          963  DC XL16'00000000000000000200000002000000'
00013750  83836186 97964060                          964  DC CL64'cc/fpo -G-1=80..00-   G=7F..FF,-G-1=80..00-  +1=00..01'
00013790  03000000 03040008                          965  DC XL16'03000000030400080300000003040008'
000137A0  83836186 97964060                          966  DC CL64'cc/fpo -G-1=80..00-   0=00..00,-G-1=80..00-  -1=FF..FF'
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT

000137E0  01000000 01000000                          967  DC XL16'01000000010000000100000001000000'
000137F0  83836186 97964060                          968  DC CL64'cc/fpo -G-1=80..00--G  =80..01,-G-1=80..00--G-1=80..00'
00013830  01000000 01000000                          969  DC XL16'01000000010000000000000000000000'
                             00000012  00000001    970 SGRFLG_NUM EQU (*-SGRFLG_GOOD)/80
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT

00013840                                            972 HELPERS  DS    0H       (R12 base of helper subroutines)

                                                    974 ************************************************************
                                                    975 *                 REPORT UNEXPECTED PROGRAM CHECK
                                                    976 ************************************************************

00013840                                            978 PGMCK    DS    0H
00013840  F384 C072 F08C      000138B2  0000008C    979          UNPK  PROGCODE(L'PROGCODE+1),PCINTCD(L'PCINTCD+1)
00013846  926B C07A                     000138BA    980          MVI   PGMCOMMA,C','
0001384A  DC07 C072 C18C      000138B2  000139CC    981          TR    PROGCODE,HEXTRTAB

00013850  F384 C080 F150      000138C0  00000150    983          UNPK  PGMPSW+(0*9)(9),PCOLDPSW+(0*4)(5)
00013856  9240 C088                     000138C8    984          MVI   PGMPSW+(0*9)+8,C' '
0001385A  DC07 C080 C18C      000138C0  000139CC    985          TR    PGMPSW+(0*9)(8),HEXTRTAB

00013860  F384 C089 F154      000138C9  00000154    987          UNPK  PGMPSW+(1*9)(9),PCOLDPSW+(1*4)(5)
00013866  9240 C091                     000138D1    988          MVI   PGMPSW+(1*9)+8,C' '
0001386A  DC07 C089 C18C      000138C9  000139CC    989          TR    PGMPSW+(1*9)(8),HEXTRTAB

00013870  F384 C092 F158      000138D2  00000158    991          UNPK  PGMPSW+(2*9)(9),PCOLDPSW+(2*4)(5)
00013876  9240 C09A                     000138DA    992          MVI   PGMPSW+(2*9)+8,C' '
0001387A  DC07 C092 C18C      000138D2  000139CC    993          TR    PGMPSW+(2*9)(8),HEXTRTAB

00013880  F384 C09B F15C      000138DB  0000015C    995          UNPK  PGMPSW+(3*9)(9),PCOLDPSW+(3*4)(5)
00013886  9240 C0A3                     000138E3    996          MVI   PGMPSW+(3*9)+8,C' '
0001388A  DC07 C09B C18C      000138DB  000139CC    997          TR    PGMPSW+(3*9)(8),HEXTRTAB

00013890  4100 0046                     00000046    999          LA    R0,L'PROGMSG     R0 <== length of message
00013894  4110 C05E                     0001389E   1000          LA    R1,PROGMSG       R1 --> the message text itself
00013898  4520 C28E                     00013ACE   1001          BAL   R2,MSG           Go display this message
                                                   1002
0001389C  07FD                                     1003          BR    R13              Return to caller




0001389E                                           1005 PROGMSG  DS   0CL70
0001389E  D7D9D6C7 D9C1D440                        1006          DC    CL20'PROGRAM CHECK! CODE '
000138B2  88888888 88888888                        1007 PROGCODE DC    CL8'hhhhhhhh'
000138BA  6B                                       1008 PGMCOMMA DC    CL1','
000138BB  40D7E2E6 40                              1009          DC    CL5' PSW '
000138C0  88888888 88888888                        1010 PGMPSW   DC    CL36'hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh '
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT

                                                    1012 ************************************************************************
                                                    1013 *                      VERIFICATION ROUTINE
                                                    1014 ************************************************************************

000138E4                                            1016 VERISUB  DS    0H
                                                    1017 *
                                                    1018 **       Loop through the VERIFY TABLE...
                                                    1019 *

000138E4  4110 C340                 00013B80       1021          LA    R1,VERIFTAB      R1 --> Verify table
000138E8  4120 0008                 00000008       1022          LA    R2,VERIFLEN      R2 <== Number of entries
000138EC  0D30                                     1023          BASR  R3,0             Set top of loop

000138EE  9846 1000                 00000000       1025          LM    R4,R6,0(R1)      Load verify table values
000138F2  4D70 C0C6                 00013906       1026          BAS   R7,VERIFY        Verify results
000138F6  4110 100C                 0000000C       1027          LA    R1,12(,R1)       Next verify table entry
000138FA  0623                                     1028          BCTR  R2,R3            Loop through verify table

000138FC  9500 C28C                 00013ACC       1030          CLI   FAILFLAG,X'00'   Did all tests verify okay?
00013900  078D                                     1031          BER   R13              Yes, return to caller
00013902  47F0 F238                 00000238       1032          B     FAIL             No, load FAILURE disabled wait PSW



                                                    1034 *
                                                    1035 **       Loop through the ACTUAL / EXPECTED results...
                                                    1036 *

00013906  0D80                                     1038 VERIFY   BASR  R8,0             Set top of loop

00013908  D50F 4000 5040      00000000  00000040   1040          CLC   0(16,R4),64(R5)  Actual results == Expected results?
0001390E  4770 C0DE                     0001391E   1041          BNE   VERIFAIL         No, show failure
00013912  4140 4010                     00000010   1042 VERINEXT LA    R4,16(,R4)       Next actual result
00013916  4150 5050                     00000050   1043          LA    R5,80(,R5)       Next expected result
0001391A  0668                                     1044          BCTR  R6,R8            Loop through results

0001391C  07F7                                     1046          BR    R7               Return to caller
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT
                                                   1048 ********************************************************************
                                                   1049 *                     Report the failure...
                                                   1050 ********************************************************************

0001391E  9005 C264                      00013AA4  1052 VERIFAIL STM   R0,R5,SAVER0R5   Save registers
00013922  92FF C28C                      00013ACC  1053          MVI   FAILFLAG,X'FF'   Remember verification failure
                                                   1054 *
                                                   1055 **       First, show them the description...
                                                   1056 *
00013926  D23F C1E4 5000      00013A24   00000000  1057          MVC   FAILDESC,0(R5)   Save results/test description
0001392C  4100 0054                      00000054  1058          LA    R0,L'FAILMSG1    R0 <== length of message
00013930  4110 C1D0                      00013A10  1059          LA    R1,FAILMSG1      R1 --> the message text itself
00013934  4520 C28E                      00013ACE  1060          BAL   R2,MSG           Go display this message
                                                   1061 *
                                                   1062 **       Save address of actual and expected results
                                                   1063 *
00013938  5040 C260                      00013AA0  1064          ST    R4,AACTUAL       Save A(actual results)
0001393C  4150 5040                      00000040  1065          LA    R5,64(,R5)       R5 ==> expected results
00013940  5050 C25C                      00013A9C  1066          ST    R5,AEXPECT       Save A(expected results)
                                                   1067 *
                                                   1068 **       Format and show them the EXPECTED ("Want") results...
                                                   1069 *
00013944  D205 C224 C3A4      00013A64   00013BE4  1070          MVC   WANTGOT,=CL6'Want: '
0001394A  F384 C22A C25C      00013A6A   00013A9C  1071          UNPK  FAILADR(L'FAILADR+1),AEXPECT(L'AEXPECT+1)
00013950  9240 C232                      00013A72  1072          MVI   BLANKEQ,C' '
00013954  DC07 C22A C18C      00013A6A   000139CC  1073          TR    FAILADR,HEXTRTAB

0001395A  F384 C235 5000      00013A75   00000000  1075          UNPK  FAILVALS+(0*9)(9),(0*4)(5,R5)
00013960  9240 C23D                      00013A7D  1076          MVI   FAILVALS+(0*9)+8,C' '
00013964  DC07 C235 C18C      00013A75   000139CC  1077          TR    FAILVALS+(0*9)(8),HEXTRTAB

0001396A  F384 C23E 5004      00013A7E   00000004  1079          UNPK  FAILVALS+(1*9)(9),(1*4)(5,R5)
00013970  9240 C246                      00013A86  1080          MVI   FAILVALS+(1*9)+8,C' '
00013974  DC07 C23E C18C      00013A7E   000139CC  1081          TR    FAILVALS+(1*9)(8),HEXTRTAB

0001397A  F384 C247 5008      00013A87   00000008  1083          UNPK  FAILVALS+(2*9)(9),(2*4)(5,R5)
00013980  9240 C24F                      00013A8F  1084          MVI   FAILVALS+(2*9)+8,C' '
00013984  DC07 C247 C18C      00013A87   000139CC  1085          TR    FAILVALS+(2*9)(8),HEXTRTAB

0001398A  F384 C250 500C      00013A90   0000000C  1087          UNPK  FAILVALS+(3*9)(9),(3*4)(5,R5)
00013990  9240 C258                      00013A98  1088          MVI   FAILVALS+(3*9)+8,C' '
00013994  DC07 C250 C18C      00013A90   000139CC  1089          TR    FAILVALS+(3*9)(8),HEXTRTAB

0001399A  4100 0035                      00000035  1091          LA    R0,L'FAILMSG2    R0 <== length of message
0001399E  4110 C224                      00013A64  1092          LA    R1,FAILMSG2      R1 --> the message text itself
000139A2  4520 C28E                      00013ACE  1093          BAL   R2,MSG           Go display this message
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2    STMT
                                                   1095 *
                                                   1096 **       Format and show them the ACTUAL ("Got") results...
                                                   1097 *
000139A6  D205 C224 C3AA      00013A64  00013BEA   1098          MVC   WANTGOT,=CL6'Got:  '
000139AC  F384 C22A C260      00013A6A  00013AA0   1099          UNPK  FAILADR(L'FAILADR+1),AACTUAL(L'AACTUAL+1)
000139B2  9240 C232                     00013A72   1100          MVI   BLANKEQ,C' '
000139B6  DC07 C22A C18C      00013A6A  000139CC   1101          TR    FAILADR,HEXTRTAB

000139BC  F384 C235 4000      00013A75  00000000   1103          UNPK  FAILVALS+(0*9)(9),(0*4)(5,R4)
000139C2  9240 C23D                     00013A7D   1104          MVI   FAILVALS+(0*9)+8,C' '
000139C6  DC07 C235 C18C      00013A75  000139CC   1105          TR    FAILVALS+(0*9)(8),HEXTRTAB

000139CC  F384 C23E 4004      00013A7E  00000004   1107          UNPK  FAILVALS+(1*9)(9),(1*4)(5,R4)
000139D2  9240 C246                     00013A86   1108          MVI   FAILVALS+(1*9)+8,C' '
000139D6  DC07 C23E C18C      00013A7E  000139CC   1109          TR    FAILVALS+(1*9)(8),HEXTRTAB

000139DC  F384 C247 4008      00013A87  00000008   1111          UNPK  FAILVALS+(2*9)(9),(2*4)(5,R4)
000139E2  9240 C24F                     00013A8F   1112          MVI   FAILVALS+(2*9)+8,C' '
000139E6  DC07 C247 C18C      00013A87  000139CC   1113          TR    FAILVALS+(2*9)(8),HEXTRTAB

000139EC  F384 C250 400C      00013A90  0000000C   1115          UNPK  FAILVALS+(3*9)(9),(3*4)(5,R4)
000139F2  9240 C258                     00013A98   1116          MVI   FAILVALS+(3*9)+8,C' '
000139F6  DC07 C250 C18C      00013A90  000139CC   1117          TR    FAILVALS+(3*9)(8),HEXTRTAB

000139FC  4100 0035                     00000035   1119          LA    R0,L'FAILMSG2    R0 <== length of message
00013A00  4110 C224                     00013A64   1120          LA    R1,FAILMSG2      R1 --> the message text itself
00013A04  4520 C28E                     00013ACE   1121          BAL   R2,MSG           Go display this message

00013A08  9805 C264                     00013AA4   1123          LM    R0,R5,SAVER0R5   Restore registers
00013A0C  47F0 C0D2                     00013912   1124          B     VERINEXT         Continue with verification...



00013A10                                           1126 FAILMSG1 DS    0CL84
00013A10  C3D6D4D7 C1D9C9E2                        1127          DC    CL20'COMPARISON FAILURE! '
00013A24  4D8485A2 83998997                        1128 FAILDESC DC    CL64'(description)'


00013A64                                           1130 FAILMSG2 DS    0CL53
00013A64  40404040 4040                            1131 WANTGOT  DC    CL6' '           'Want: ' -or- 'Got:  '
00013A6A  C1C1C1C1 C1C1C1C1                        1132 FAILADR  DC    CL8'AAAAAAAA'
00013A72  407E40                                   1133 BLANKEQ  DC    CL3' = '
00013A75  88888888 88888888                        1134 FAILVALS DC    CL36'hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh '

00013A9C  00000000                                 1136 AEXPECT  DC    F'0'             ==> Expected ("Want") results
00013AA0  00000000                                 1137 AACTUAL  DC    F'0'             ==> Actual ("Got") results
00013AA4  00000000 00000000                        1138 SAVER0R5 DC    6F'0'            Registers R0 - R5 save area
00013ABC  F0F1F2F3 F4F5F6F7                        1139 CHARHEX  DC    CL16'0123456789ABCDEF'
                            000139CC  00000010     1140 HEXTRTAB EQU   CHARHEX-X'F0'    Hexadecimal translation table
00013ACC  00                                       1141 FAILFLAG DC    X'00'            FF = Fail, 00 = Success
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT

                                          1143 ********************************************************************************
                                          1144 *        Issue HERCULES MESSAGE pointed to by R1, length in R0
                                          1145 ********************************************************************************

00013ACE  4900 C3A0                  00013BE0  1147 MSG      CH    R0,=H'0'                Do we even HAVE a message?
00013AD2  07D2                                 1148           BNHR  R2                      No, ignore

00013AD4  9002 C2C4                  00013B04  1150           STM   R0,R2,MSGSAVE           Save registers

00013AD8  4900 C3A2                  00013BE2  1152           CH    R0,=AL2(L'MSGMSG)       Message length within limits?
00013ADC  47D0 C2A4                  00013AE4  1153           BNH   MSGOK                   Yes, continue
00013AE0  4100 005F                  0000005F  1154           LA    R0,L'MSGMSG             No, set to maximum

00013AE4  1820                                 1156 MSGOK     LR    R2,R0                   Copy length to work register
00013AE6  0620                                 1157           BCTR  R2,0                    Minus-1 for execute
00013AE8  4420 C2D0                  00013B10  1158           EX    R2,MSGMVC               Copy message to O/P buffer

00013AEC  4120 200A                  0000000A  1160           LA    R2,1+L'MSGCMD(,R2)      Calculate true command length
00013AF0  4110 C2D6                  00013B16  1161           LA    R1,MSGCMD               Point to true command

00013AF4  83120008                             1163           DC    X'83',X'12',X'0008'     Issue Hercules Diagnose X'008'
00013AF8  4780 C2BE                  00013AFE  1164           BZ    MSGRET                  Return if successful
00013AFC  0000                                 1165           DC    H'0'                    CRASH for debugging purposes

00013AFE  9802 C2C4                  00013B04  1167 MSGRET   LM    R0,R2,MSGSAVE           Restore registers
00013B02  07F2                                 1168           BR    R2                      Return to caller

00013B04  00000000 00000000                    1170 MSGSAVE   DC    3F'0'                   Registers save area
00013B10  D200 C2DF 1000      00013B1F 00000000  1171 MSGMVC   MVC   MSGMSG(0),0(R1)         Executed instruction

00013B16  D4E2C7D5 D6C8405C                    1173 MSGCMD    DC    C'MSGNOH * '            *** HERCULES MESSAGE COMMAND ***
00013B1F  40404040 40404040                    1174 MSGMSG    DC    CL95' '                 The message text to be displayed
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2     STMT

                                                    1176 ********************************************************************************
                                                    1177 *                              VERIFY TABLE
                                                    1178 ********************************************************************************
                                                    1179 *
                                                    1180 *        A(actual results), A(expected results), A(#of results)
                                                    1181 *
                                                    1182 ********************************************************************************

00013B80                                            1184 VERIFTAB DC    0F'0'
00013B80  00001000                                  1185          DC    A(ARSUM)
00013B84  00010000                                  1186          DC    A(ARSUM_GOOD)
00013B88  00000012                                  1187          DC    A(ARSUM_NUM)
                                                    1188 *
00013B8C  00002000                                  1189          DC    A(ARFLG)
00013B90  000105A0                                  1190          DC    A(ARFLG_GOOD)
00013B94  00000012                                  1191          DC    A(ARFLG_NUM)
                                                    1192 *
00013B98  00001400                                  1193          DC    A(AGRSUM)
00013B9C  00010B40                                  1194          DC    A(AGRSUM_GOOD)
00013BA0  00000024                                  1195          DC    A(AGRSUM_NUM)
                                                    1196 *
00013BA4  00002400                                  1197          DC    A(AGRFLG)
00013BA8  00011680                                  1198          DC    A(AGRFLG_GOOD)
00013BAC  00000012                                  1199          DC    A(AGRFLG_NUM)
                                                    1200 *
00013BB0  00001800                                  1201          DC    A(SRSUM)
00013BB4  00011C20                                  1202          DC    A(SRSUM_GOOD)
00013BB8  00000012                                  1203          DC    A(SRSUM_NUM)
                                                    1204 *
00013BBC  00002800                                  1205          DC    A(SRFLG)
00013BC0  000121C0                                  1206          DC    A(SRFLG_GOOD)
00013BC4  00000012                                  1207          DC    A(SRFLG_NUM)
                                                    1208 *
00013BC8  00001C00                                  1209          DC    A(SGRSUM)
00013BCC  00012760                                  1210          DC    A(SGRSUM_GOOD)
00013BD0  00000024                                  1211          DC    A(SGRSUM_NUM)
                                                    1212 *
00013BD4  00002C00                                  1213          DC    A(SGRFLG)
00013BD8  000132A0                                  1214          DC    A(SGRFLG_GOOD)
00013BDC  00000012                                  1215          DC    A(SGRFLG_NUM)
                                                    1216 *
                          00000008  00000001        1217 VERIFLEN EQU   (*-VERIFTAB)/12    #of entries in verify table
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT

00013BE0                                            1219          END
00013BE0  0000                                      1220                =H'0'
00013BE2  005F                                      1221                =AL2(L'MSGMSG)
00013BE4  E68195A3 7A40                             1222                =CL6'Want: '
00013BEA  C796A37A 4040                             1223                =CL6'Got:  '
```

SYMBOL	TYPE	VALUE	LENGTH	DEFN	REFERENCES
A32VALS	D	0005B0	8	529	537 219 220 235 236
A64VALS	D	0005C8	8	542	549 227 228 244 245
AACTUAL	F	013AA0	4	1137	1064 1099
AEXPECT	F	013A9C	4	1136	1066 1071
AGRFLG	U	002400	1	563	230 1197
AGRFLG_GOOD	U	011680	1	737	774 1198
AGRFLG_NUM	U	000012	1	774	1199
AGRSUM	U	001400	1	561	229 1193
AGRSUM_GOOD	U	010B40	1	661	734 1194
AGRSUM_NUM	U	000024	1	734	1195
AGRTABL	F	0002E8	4	225	180
AGRTEST	I	0003C6	4	323	181
AHELPERS	A	00027C	4	165	155 197
ARCCPM	F	0005A8	4	513	259 301 324 366 389 432 455 498
ARCCPMOV	F	0005AC	4	514	260 261 288 325 326 353 390 391 419 456 457 485
ARFLG	U	002000	1	559	222 1189
ARFLG_GOOD	U	0105A0	1	621	658 1190
ARFLG_NUM	U	000012	1	658	1191
ARSUM	U	001000	1	557	221 1185
ARSUM_GOOD	U	010000	1	581	618 1186
ARSUM_NUM	U	000012	1	618	1187
ARTABL	F	0002D0	4	217	175
ARTEST	I	000330	4	258	176
BIMADSUB	J	000000	80880	95	
BLANKEQ	C	013A72	3	1133	1072 1100
CHARHEX	C	013ABC	16	1139	1140
ENDRES	U	003000	1	573	
FAIL	I	000238	4	163	1032
FAILADR	C	013A6A	8	1132	1071 1073 1099 1101
FAILDESC	C	013A24	64	1128	1057
FAILFLAG	X	013ACC	1	1141	1030 1053
FAILMSG1	C	013A10	84	1126	1058 1059
FAILMSG2	C	013A64	53	1130	1091 1092 1119 1120
FAILPSW	X	0002C0	8	205	163
FAILVALS	C	013A75	36	1134	1075 1076 1077 1079 1080 1081 1083 1084 1085 1087 1088 1089 1103 1104 1105 1107 1108 1109 1111 1112 1113 1115 1116 1117
GOODPSW	X	0002B0	8	204	201
HELPERS	H	013840	2	972	125 165
HEXTRTAB	U	0139CC	16	1140	981 985 989 993 997 1073 1077 1081 1085 1089 1101 1105 1109 1113 1117
IMAGE	1	000000	80880	0	
MSG	I	013ACE	4	1147	1001 1060 1093 1121
MSGCMD	C	013B16	9	1173	1160 1161
MSGMSG	C	013B1F	95	1174	1154 1171 1152
MSGMVC	I	013B10	6	1171	1158
MSGOK	I	013AE4	2	1156	1153
MSGRET	I	013AFE	4	1167	1164
MSGSAVE	F	013B04	4	1170	1150 1167
PCINTCD	F	00008C	4	133	150 290 296 355 361 421 427 487 493 979
PCNOTDTA	I	00020C	4	154	151
PCOLDPSW	U	000150	1	135	152 983 987 991 995
PGMCK	H	013840	2	978	156
PGMCOMMA	C	0138BA	1	1008	980
PGMPSW	C	0138C0	36	1010	983 984 985 987 988 989 991 992 993 995 996 997
PROGCHK	H	000200	2	149	141
PROGCODE	C	0138B2	8	1007	979 981

SYMBOL	TYPE	VALUE	LENGTH	DEFN	REFERENCES
PROGMSG	C	01389E	70	1005	999 1000
PROGPSW	D	000228	8	162	161
R0	U	000000	1	97	154 157 258 259 260 275 277 278 279 280 281 288 289 292 294 295 297 298 299 301 302 323 324 325 340 342 343 344 345 346 353 354 357 359 360 362 363 364 366 367 388 389 390 406 408 409 410 411 412 419 420 423 425 426 428 429 430 432 433 454 455 456 472 474 475 476 477 478 485 486 489 491 492 494 495 496 498 499 999 1052 1058 1091 1119 1123 1147 1150 1152 1154 1156 1167
R1	U	000001	1	101	276 277 293 294 341 342 358 359 407 408 424 425 473 474 490 491 1000 1021 1025 1027 1059 1092 1120 1161 1171
R10	U	00000A	1	116	175 180 185 190 262 263 264 271 272 327 328 329 336 337 392 393 394 402 403 458 459 460 468 469
R11	U	00000B	1	117	
R12	U	00000C	1	118	125 155 197 267 312 332 377 398 443 464 509
R13	U	00000D	1	119	156 176 181 186 191 198 266 314 331 379 397 445 463 511 1003 1031
R14	U	00000E	1	120	159 160 199 200
R15	U	00000F	1	121	124 154 157
R2	U	000002	1	103	262 265 312 327 330 377 392 396 443 458 462 509 1001 1022 1028 1060 1093 1121 1148 1150 1156 1157 1158 1160 1167 1168
R3	U	000003	1	105	262 275 292 311 327 340 357 376 392 406 423 442 458 472 489 508 1023 1028
R4	U	000004	1	107	271 307 336 372 402 438 468 504 1025 1040 1042 1064 1103 1107 1111 1115
R5	U	000005	1	109	272 276 293 304 337 341 358 369 403 407 424 435 469 473 490 501 1040 1043 1052 1057 1065 1066 1075 1079 1083 1087 1123
R6	U	000006	1	111	264 282 304 305 311 329 347 369 370 376 394 413 435 436 442 460 479 501 502 508 1025 1044
R7	U	000007	1	113	263 278 282 295 305 328 343 347 360 370 393 409 413 426 436 459 475 479 492 502 1026 1046
R8	U	000008	1	114	263 281 283 296 299 306 328 346 348 361 364 371 393 412 414 427 430 437 459 478 480 493 496 503 1038 1044
R9	U	000009	1	115	273 307 338 372 404 438 470 504
SAVER0R5	F	013AA4	4	1138	1052 1123
SAVEREGS	F	00023C	4	164	154 157
SGRFLG	U	002C00	1	571	247 1213
SGRFLG_GOOD	U	0132A0	1	933	970 1214
SGRFLG_NUM	U	000012	1	970	1215
SGRSUM	U	001C00	1	569	246 1209
SGRSUM_GOOD	U	012760	1	857	930 1210
SGRSUM_NUM	U	000024	1	930	1211
SGRTABL	F	000318	4	242	190
SGRTEST	I	000502	4	454	191
SRFLG	U	002800	1	567	238 1205
SRFLG_GOOD	U	0121C0	1	817	854 1206
SRFLG_NUM	U	000012	1	854	1207
SRSUM	U	001800	1	565	237 1201
SRSUM_GOOD	U	011C20	1	777	814 1202
SRSUM_NUM	U	000012	1	814	1203
SRTABL	F	000300	4	233	185
SRTEST	I	00046C	4	388	186
START	H	000280	2	171	138
STRTLABL	U	000000	1	96	132 135 137 140 148 557 559 561 563 565 567 569 571 573 579
VALCT	U	000018	1	537	218 234

```
     SYMBOL         TYPE  VALUE   LENGTH  DEFN  REFERENCES

VALCT64              U    000030       1   549   226   243
VERIFAIL             I    01391E       4  1052  1041
VERIFLEN             U    000008       1  1217  1022
VERIFTAB             F    013B80       4  1184  1217  1021
VERIFY               I    013906       2  1038  1026
VERINEXT             I    013912       4  1042  1124
VERISUB              H    0138E4       2  1016   198
WANTGOT              C    013A64       6  1131  1070  1098
=AL2(L'MSGMSG)       R    013BE2       2  1221  1152
=CL6'Got:   '        C    013BEA       6  1223  1098
=CL6'Want: '         C    013BE4       6  1222  1070
=H'0'                H    013BE0       2  1220  1147
```

MACRO DEFN REFERENCES

No defined macros

```
   DESC      SYMBOL     SIZE       POS           ADDR

Entry: 0

Image       IMAGE      80880  00000-13BEF  00000-13BEF
  Region               80880  00000-13BEF  00000-13BEF
    CSECT   BIMADSUB   80880  00000-13BEF  00000-13BEF
```

```
   STMT                                                FILE NAME
1     c:\Users\Fish\Documents\Visual Studio 2008\Projects\MyProjects\ASMA-0\bim-001-add-sub\bim-001-add-sub.asm

** NO ERRORS FOUND **
```