```
  LOC        OBJECT CODE       ADDR1     ADDR2     STMT
                                                      2 ************************************************************************
                                                      3 *
                                                      4 *          Concurrent Block Update Consistency (CBUC) test
                                                      5 *
                                                      6 ************************************************************************
                                                      7 *
                                                      8 *  According to the POP, when storing a doubleword into a doubleword
                                                      9 *  using a memory copy operations, the destination storage area as
                                                     10 *  seen by other CPUs should ALWAYS present the complete operation,
                                                     11 *  and not any intermediate value.
                                                     12 *
                                                     13 *  What this means is, if the destination doubleword is 111... and
                                                     14 *  another CPU moves 222... to that area, any CPU that accesses the
                                                     15 *  destination doubleword should ALWAYS see either all 111... or all
                                                     16 *  222... but NEVER any intermediate value such as 1122111122221122.
                                                     17 *
                                                     18 *  Even though the 'MVC' and other instructions behave as if they
                                                     19 *  were moving one byte at a time, the hardware ensures that all
                                                     20 *  "Block Updates" (doubleword updates) are always CONSISTENT (i.e.
                                                     21 *  atomic), such that all bytes of a block are always updated at the
                                                     22 *  same time and never piecemeal.
                                                     23 *
                                                     24 *  This test attempts to detect any discrepancy in this area.
                                                     25 *
                                                     26 ************************************************************************
                                                     27 *
                                                     28 *                          Example test scripts
                                                     29 *
                                                     30 *                               (CBUC.tst)
                                                     31 *
                                                     32 * *Testcase CBUC (Concurrent Block Update Consistency)
                                                     33 * defsym      testdur  30   # (maximum test duration in seconds)
                                                     34 * mainsize    1
                                                     35 * numcpu      2
                                                     36 * sysclear
                                                     37 * archlvl     z/Arch
                                                     38 * loadcore    "$(testpath)/CBUC.core"
                                                     39 * script      "$(testpath)/CBUC.subtst"  &     # ('&' = async thread!)
                                                     40 * runtest     300                              # (subtst will stop it)
                                                     41 * *Done
                                                     42 * numcpu 1
                                                     43 *
                                                     44 *                             (CBUC.subtst)
                                                     45 *
                                                     46 * # CBUC test 'stop' thread...
                                                     47 * # This script is designed to run in a separate thread!
                                                     48 * pause $(testdur)    # Sleep for desired number of seconds
                                                     49 * r 500=FF            # And then force our test to stop
                                                     50 *
                                                     51 *
                                                     52 ************************************************************************
```

```
 LOC        OBJECT CODE       ADDR1     ADDR2     STMT

                                                    54 ********************************************************************************
                                                    55 *
                                                    56 *                          PROGRAMMING NOTE
                                                    57 *
                                                    58 *  The below loop values do NOT determine our test duration.  Rather,
                                                    59 *  it is our asynchronous 'cbuc.subtst' script that controls how long
                                                    60 *  our test runs by sleeping for the desired test duration number of
                                                    61 *  seconds and then sets the 'STOPFLAG' to a non-zero value to force
                                                    62 *  our test to end.  Using a value of zero for our loop value ensures
                                                    63 *  we can always support the maximum possible test duration.
                                                    64 *
                                                    65 ********************************************************************************

                          00000000  00000001      67 WRLOOPS  EQU   0                    Number of writer thread loops
                          00000000  00000001      68 RDLOOPS  EQU   0                    Number of reader thread loops


                                                    70 ********************************************************************************
                                                    71 *
                                                    72 *  CPU 1, in a tight loop, moves to the test area, using, in turn,
                                                    73 *  MVC, MVCL, and MVCLE, two alternate values: X'1111111111111111'
                                                    74 *  and X'2222222222222222'.
                                                    75 *
                                                    76 *  At the same time, CPU 0, also in a tight loop, using MVC, copies
                                                    77 *  the test area to a separate work area and verifies that the value
                                                    78 *  is either X'1111111111111111' or X'2222222222222222'. If any other
                                                    79 *  value is seen, then the test fails.
                                                    80 *
                                                    81 *  For the test to be relevant, it is best to perform this test on a
                                                    82 *  host system with more than one processor core. The more processors
                                                    83 *  (cores) that host system has, the better.
                                                    84 *
                                                    85 *  CPU 0:
                                                    86 *
                                                    87 *     MVC    WORK(8),DEST
                                                    88 *     CLC    WORK(4),WORK+4
                                                    89 *     BNE    FAIL
                                                    90 *     CLC    WORK(4),SRC1
                                                    91 *     BE     OK
                                                    92 *     CLC    WORK(4),SRC2
                                                    93 *     BNE    FAIL
                                                    94 *
                                                    95 *  CPU 1:
                                                    96 *
                                                    97 *     MVC    DEST(8),SRC1
                                                    98 *     MVCL   DEST(8),SRC2
                                                    99 *     MVCLE  DEST(8),SRC1
                                                   100 *     MVC    DEST(8),SRC2
                                                   101 *     MVCL   DEST(8),SRC1
                                                   102 *     MVCLE  DEST(8),SRC2
                                                   103 *
                                                   104 ********************************************************************************
```

```
  LOC         OBJECT CODE       ADDR1     ADDR2     STMT

                                                      106          PRINT OFF
                                                     3487          PRINT ON

                                                     3489 ********************************************************************************
                                                     3490 *        SATK prolog stuff...
                                                     3491 ********************************************************************************

                                                     3493          ARCHLVL  MNOTE=NO
                                                     3495+$AL      OPSYN AL
                                                     3496+$ALR     OPSYN ALR
                                                     3497+$B       OPSYN B
                                                     3498+$BAS     OPSYN BAS
                                                     3499+$BASR    OPSYN BASR
                                                     3500+$BC      OPSYN BC
                                                     3501+$BCTR    OPSYN BCTR
                                                     3502+$BE      OPSYN BE
                                                     3503+$BH      OPSYN BH
                                                     3504+$BL      OPSYN BL
                                                     3505+$BM      OPSYN BM
                                                     3506+$BNE     OPSYN BNE
                                                     3507+$BNH     OPSYN BNH
                                                     3508+$BNL     OPSYN BNL
                                                     3509+$BNM     OPSYN BNM
                                                     3510+$BNO     OPSYN BNO
                                                     3511+$BNP     OPSYN BNP
                                                     3512+$BNZ     OPSYN BNZ
                                                     3513+$BO      OPSYN BO
                                                     3514+$BP      OPSYN BP
                                                     3515+$BXLE    OPSYN BXLE
                                                     3516+$BZ      OPSYN BZ
                                                     3517+$CH      OPSYN CH
                                                     3518+$L       OPSYN L
                                                     3519+$LH      OPSYN LH
                                                     3520+$LM      OPSYN LM
                                                     3521+$LPSW    OPSYN LPSW
                                                     3522+$LR      OPSYN LR
                                                     3523+$LTR     OPSYN LTR
                                                     3524+$NR      OPSYN NR
                                                     3525+$SL      OPSYN SL
                                                     3526+$SLR     OPSYN SLR
                                                     3527+$SR      OPSYN SR
                                                     3528+$ST      OPSYN ST
                                                     3529+$STM     OPSYN STM
                                                     3530+$X       OPSYN X
                                                     3531+$AHI     OPSYN AHI
                                                     3532+$B       OPSYN J
                                                     3533+$BC      OPSYN BRC
                                                     3534+$BE      OPSYN JE
                                                     3535+$BH      OPSYN JH
                                                     3536+$BL      OPSYN JL
                                                     3537+$BM      OPSYN JM
                                                     3538+$BNE     OPSYN JNE
```

```
 LOC         OBJECT CODE       ADDR1     ADDR2     STMT

                                                   3539+$BNH     OPSYN JNH
                                                   3540+$BNL     OPSYN JNL
                                                   3541+$BNM     OPSYN JNM
                                                   3542+$BNO     OPSYN JNO
                                                   3543+$BNP     OPSYN JNP
                                                   3544+$BNZ     OPSYN JNZ
                                                   3545+$BO      OPSYN JO
                                                   3546+$BP      OPSYN JP
                                                   3547+$BXLE    OPSYN JXLE
                                                   3548+$BZ      OPSYN JZ
                                                   3549+$CHI     OPSYN CHI
                                                   3550+$AHI     OPSYN AGHI
                                                   3551+$AL      OPSYN ALG
                                                   3552+$ALR     OPSYN ALGR
                                                   3553+$BCTR    OPSYN BCTGR
                                                   3554+$BXLE    OPSYN JXLEG
                                                   3555+$CH      OPSYN CGH
                                                   3556+$CHI     OPSYN CGHI
                                                   3557+$L       OPSYN LG
                                                   3558+$LH      OPSYN LGH
                                                   3559+$LM      OPSYN LMG
                                                   3560+$LPSW    OPSYN LPSWE
                                                   3561+$LR      OPSYN LGR
                                                   3562+$LTR     OPSYN LTGR
                                                   3563+$NR      OPSYN NGR
                                                   3564+$SL      OPSYN SLG
                                                   3565+$SLR     OPSYN SLGR
                                                   3566+$SR      OPSYN SGR
                                                   3567+$ST      OPSYN STG
                                                   3568+$STM     OPSYN STMG
                                                   3569+$X       OPSYN XG
```

```
 LOC        OBJECT CODE       ADDR1     ADDR2    STMT

                                                  3571 ********************************************************************************
                                                  3572 *        Initiate the CBUC CSECT in the CODE region
                                                  3573 *        with the location counter at 0
                                                  3574 ********************************************************************************

                                                  3576 CBUC     ASALOAD  REGION=CODE
                                00000000  00000807 3577+CBUC     START 0,CODE
00000000  00020000 00000000                       3579+          PSW   0,0,2,0,X'008'      64-bit Restart ISR Trap New PSW
00000010                        00000010  00000058 3580+          ORG   CBUC+X'058'
00000058  00020000 00000000                       3582+          PSW   0,0,2,0,X'018'      64-bit External ISR Trap New PSW
00000068  00020000 00000000                       3583+          PSW   0,0,2,0,X'020'      64-bit Supervisor Call ISR Trap New PSW
00000078  00020000 00000000                       3584+          PSW   0,0,2,0,X'028'      64-bit Program ISR Trap New PSW
00000088  00020000 00000000                       3585+          PSW   0,0,2,0,X'030'      64-bit Machine Check Trap New PSW
00000098  00020000 00000000                       3586+          PSW   0,0,2,0,X'038'      64-bit Input/Output Trap New PSW
000000A8                        000000A8  000001A0 3587+          ORG   CBUC+X'1A0'
000001A0  00020000 00000000                       3589+          PSWZ  0,0,2,0,X'120'     Restart ISR Trap New PSW
000001B0  00020000 00000000                       3590+          PSWZ  0,0,2,0,X'130'     External ISR Trap New PSW
000001C0  00020000 00000000                       3591+          PSWZ  0,0,2,0,X'140'     Supervisor Call ISR Trap New PSW
000001D0  00020000 00000000                       3592+          PSWZ  0,0,2,0,X'150'     Program ISR Trap New PSW
000001E0  00020000 00000000                       3593+          PSWZ  0,0,2,0,X'160'     Machine Check Trap New PSW
000001F0  00020000 00000000                       3594+          PSWZ  0,0,2,0,X'170'     Input/Output Trap New PSW


                                                  3596 ********************************************************************************
                                                  3597 *        Define the z/Arch RESTART PSW
                                                  3598 ********************************************************************************

                                00000200  00000001 3600 PREVORG  EQU   *
00000200                        00000200  000001A0 3601          ORG   CBUC+X'1A0'
                                                  3602 *        PSWZ  <sys>,<key>,<mwp>,<prog>,<addr>[,amode]
000001A0  00000001 80000000                       3603          PSWZ  0,0,0,0,X'200',64
000001B0                        000001B0  00000200 3604          ORG   PREVORG


                                                  3606 ********************************************************************************
                                                  3607 *        Create IPL (restart) PSW
                                                  3608 ********************************************************************************

                                                  3610          ASAIPL   IA=BEGIN
                                00000000  00000807 3611+CBUC     CSECT
00000200                        00000200  00000000 3612+          ORG   CBUC
00000000  00080000 00000200                       3613+          PSWE390 0,0,0,0,BEGIN,24
00000008                        00000008  00000200 3614+          ORG   CBUC+512         Reset CSECT to end of assigned storage area
                                00000000  00000807 3615+CBUC     CSECT
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT

                                                  3617 ********************************************************************************
                                                  3618 *                The actual CBUC program itself...
                                                  3619 ********************************************************************************

00000200                       00000000           3621          USING CBUC,R0                    No base registers needed

00000200  1F00                                    3623 BEGIN    SLR   R0,R0                    Start clean
00000202  4110 0001                      00000001 3624          LA    R1,1                     Request z/Arch mode
00000206  1F22                                    3625          SLR   R2,R2                    Start clean
00000208  1F33                                    3626          SLR   R3,R3                    Start clean
0000020A  AE02 0012                      00000012 3627          SIGP  R0,R2,X'12'              Request z/Arch mode

0000020E  1F11                                    3629          SLR   R1,R1                    Start clean
00000210  4120 0000                      00000000 3630          LA    R2,0                     Get our CPU number
00000214  4140 0224                      00000224 3631          LA    R4,BEGIN2                Our restart entry point
00000218  4040 01AE                      000001AE 3632          STH   R4,X'1AE'                Update restart PSW
0000021C  AE02 0006                      00000006 3633          SIGP  R0,R2,X'06'              Restart our CPU
00000220  47F0 0338                      00000338 3634          B     SIG1FAIL                 WTF?! How did we get here?!

00000224  4120 0001                      00000001 3636 BEGIN2   LA    R2,1                     Second CPU number
00000228  4140 026E                      0000026E 3637          LA    R4,WRITER                Point to its entry point
0000022C  4040 01AE                      000001AE 3638          STH   R4,X'1AE'                Update restart PSW
00000230  AE02 0006                      00000006 3639          SIGP  R0,R2,X'06'              Restart second CPU
00000234  4770 0348                      00000348 3640          BNZ   SIG2FAIL                 WTF?! (SIGP failed!)
                                                  3641 *       B     READER                   Enter our own work loop



00000238  5800 0358                      00000358 3643 READER   L     R0,RDCOUNT               R0 <== loop count
0000023C  9500 0500                      00000500 3644 READLOOP CLI   STOPFLAG,X'00'          Are we being asked to stop?
00000240  4770 0304                      00000304 3645          BNE   STOPTEST                 Yes, then do so.

00000244  D207 0800 0400       00000800  00000400 3647          MVC   WORK,READDEST            Grab copy of test value

0000024A  D507 0800 0501       00000800  00000501 3649          CLC   WORK,PATTERN1           Is it all the first pattern?
00000250  4770 025C                      0000025C 3650          BNE   READ2                    No, check if second pattern
00000254  4600 023C                      0000023C 3651          BCT   R0,READLOOP              Otherwise keep looping...
00000258  47F0 0304                      00000304 3652          B     STOPTEST                 Done!

0000025C  D507 0800 0513       00000800  00000513 3654 READ2    CLC   WORK,PATTERN2           Is it all the second pattern?
00000262  4770 0320                      00000320 3655          BNE   FAILTEST                 No?! Then *FAIL* immediately!
00000266  4600 023C                      0000023C 3656          BCT   R0,READLOOP              Otherwise keep looping...
0000026A  47F0 0304                      00000304 3657          B     STOPTEST                 Done!
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2    STMT

0000026E  5800 035C                     0000035C  3659 WRITER   L     R0,WRCOUNT              R0 <== loop count
00000272  9500 0500                     00000500  3660 WRITLOOP CLI   STOPFLAG,X'00'          Are we being asked to stop?
00000276  4770 0304                     00000304  3661          BNE   STOPTEST                Yes, then do so.
                                                  3662
0000027A  9180 0600                     00000600  3663          TM    OPTFLAG,OPTMVC
0000027E  4780 0288                     00000288  3664          BZ              NOMVC1
00000282  D20F 03FD 0501      000003FD  00000501  3665          MVC   WRITDEST,PATTERN1       Move 1st pattern to target
                              00000288  00000001  3666 NOMVC1   EQU   *

00000288  9140 0600                     00000600  3668          TM    OPTFLAG,OPTMVCL
0000028C  4780 02A0                     000002A0  3669          BZ              NOMVCL1
00000290  4160 03FD                     000003FD  3670          LA    R6,WRITDEST             R6 --> destination
00000294  4170 0010                     00000010  3671          LA    R7,L'WRITDEST           R7 <== destination length
00000298  4180 0513                     00000513  3672          LA    R8,PATTERN2             R8 --> source
0000029C  1897                                    3673          LR    R9,R7                   R9 <== source length
0000029E  0E68                                    3674          MVCL  R6,R8                   move source to destination
                              000002A0  00000001  3675 NOMVCL1  EQU   *

000002A0  9120 0600                     00000600  3677          TM    OPTFLAG,OPTMVCLE
000002A4  4780 02BA                     000002BA  3678          BZ              NOMVCLE1
000002A8  4160 03FD                     000003FD  3679          LA    R6,WRITDEST             R6 --> destination
000002AC  4170 0010                     00000010  3680          LA    R7,L'WRITDEST           R7 <== destination length
000002B0  4180 0501                     00000501  3681          LA    R8,PATTERN1             R8 --> source
000002B4  1897                                    3682          LR    R9,R7                   R9 <== source length
000002B6  A868 0000                     00000000  3683          MVCLE R6,R8,0                 move source to destination
                              000002BA  00000001  3684 NOMVCLE1 EQU   *

000002BA  9180 0600                     00000600  3686          TM    OPTFLAG,OPTMVC
000002BE  4780 02C8                     000002C8  3687          BZ              NOMVC2
000002C2  D20F 03FD 0513      000003FD  00000513  3688          MVC   WRITDEST,PATTERN2       Move 1st pattern to target
                              000002C8  00000001  3689 NOMVC2   EQU   *

000002C8  9140 0600                     00000600  3691          TM    OPTFLAG,OPTMVCL
000002CC  4780 02E0                     000002E0  3692          BZ              NOMVCL2
000002D0  4160 03FD                     000003FD  3693          LA    R6,WRITDEST             R6 --> destination
000002D4  4170 0010                     00000010  3694          LA    R7,L'WRITDEST           R7 <== destination length
000002D8  4180 0501                     00000501  3695          LA    R8,PATTERN1             R8 --> source
000002DC  1897                                    3696          LR    R9,R7                   R9 <== source length
000002DE  0E68                                    3697          MVCL  R6,R8                   move source to destination
                              000002E0  00000001  3698 NOMVCL2  EQU   *

000002E0  9120 0600                     00000600  3700          TM    OPTFLAG,OPTMVCLE
000002E4  4780 02FA                     000002FA  3701          BZ              NOMVCLE2
000002E8  4160 03FD                     000003FD  3702          LA    R6,WRITDEST             R6 --> destination
000002EC  4170 0010                     00000010  3703          LA    R7,L'WRITDEST           R7 <== destination length
000002F0  4180 0513                     00000513  3704          LA    R8,PATTERN2             R8 --> source
000002F4  1897                                    3705          LR    R9,R7                   R9 <== source length
000002F6  A868 0000                     00000000  3706          MVCLE R6,R8,0                 move source to destination
                              000002FA  00000001  3707 NOMVCLE2 EQU   *

000002FA  4600 0272                     00000272  3709          BCT   R0,WRITLOOP             Otherwise keep looping...
000002FE  47F0 0304                     00000304  3710          B     STOPTEST                Done.
```

```
 LOC        OBJECT CODE       ADDR1     ADDR2    STMT

                                                 3712 ********************************************************************************
                                                 3713 *                                PSWs
                                                 3714 ********************************************************************************


00000302  00                                     3716 FAILFLAG DC       X'00'                 X'FF' == test has failed

00000304  9500 0302                   00000302  3718 STOPTEST CLI      FAILFLAG,X'00'        Should test end normally?
00000308  4770 0320                   00000320  3719          BNE      FAILTEST              No! Test has failed!

0000030C  92FF 0500                   00000500  3721          MVI      STOPFLAG,X'FF'        Tell the other CPU to stop
                                                 3722          DWAITEND LOAD=YES              Normal completion
00000310  8200 0318                   00000318  3724+         LPSW  DWAT0009
00000318  000A0000 00000000                     3725+DWAT0009 PSWE390 0,0,2,0,X'000000'


00000320  92FF 0302                   00000302  3727 FAILTEST MVI      FAILFLAG,X'FF'        Indicate test has failed!
00000324  92FF 0500                   00000500  3728          MVI      STOPFLAG,X'FF'        Tell the other CPU to stop
                                                 3729          DWAIT    LOAD=YES,CODE=BAD     Abnormal termination
00000328  8200 0330                   00000330  3730+         LPSW  DWAT0010
00000330  000A0000 00010BAD                     3731+DWAT0010 PSWE390 0,0,2,0,X'010BAD'


00000338  92FF 0500                   00000500  3733 SIG1FAIL MVI      STOPFLAG,X'FF'        Tell the other CPU to stop
                                                 3734          DWAIT    LOAD=YES,CODE=111     First SIGP failed
0000033C  8200 0340                   00000340  3735+         LPSW  DWAT0011
00000340  000A0000 00010111                     3736+DWAT0011 PSWE390 0,0,2,0,X'010111'


00000348  92FF 0500                   00000500  3738 SIG2FAIL MVI      STOPFLAG,X'FF'        Tell the other CPU to stop
                                                 3739          DWAIT    LOAD=YES,CODE=222     Second SIGP failed
0000034C  8200 0350                   00000350  3740+         LPSW  DWAT0012
00000350  000A0000 00010222                     3741+DWAT0012 PSWE390 0,0,2,0,X'010222'
```

```
  LOC        OBJECT CODE       ADDR1     ADDR2     STMT
                                                    3743 ********************************************************************************
                                                    3744 *                          Working Storage
                                                    3745 ********************************************************************************

00000358  00000000                                  3747 RDCOUNT  DC    A(RDLOOPS)               Number of reader thread loops
0000035C  00000000                                  3748 WRCOUNT  DC    A(WRLOOPS)               Number of writer thread loops

00000360                                            3750          LTORG ,                        Literals pool




00000360                      00000360  000003F8    3752          ORG   CBUC+X'400'-8

000003F8  00000000 00                               3754          DC    XL5'0000000000'          Unaligned writer destination
000003FD                                            3755 WRITDEST DS    0CL16                    Writer thread destination
000003FD  C1C1C1                                    3756          DC    CL3'AAA'
00000400  C2C2C2C2 C2C2C2C2                         3757 READDEST DC    CL8'BBBBBBBB'            MUST be doubleword ALIGNED!
00000408  C1C1C1C1 C1                               3758          DC    CL5'AAAAA'




0000040D                      0000040D  00000500    3760          ORG   CBUC+X'500'            Fixed address of 'stop' flag

00000500  00                                        3762 STOPFLAG DC    X'00'                    Set to non-zero to stop test
00000501  C1C1C1C1 C1C1C1C1                         3763 PATTERN1 DC    CL16'AAAAAAAAAAAAAAAA'   Should be unaligned
00000511  0000                                      3764          DC    XL2'0000'
00000513  C2C2C2C2 C2C2C2C2                         3765 PATTERN2 DC    CL16'BBBBBBBBBBBBBBBB'   Should also be unaligned




00000523                      00000523  00000600    3767          ORG   CBUC+X'600'              Fixed address of 'option' flag

                              00000080  00000001    3769 OPTMVC   EQU   X'80'                    Use 'MVC'   in write loop
                              00000040  00000001    3770 OPTMVCL  EQU   X'40'                    Use 'MVCL'  in write loop
                              00000020  00000001    3771 OPTMVCLE EQU   X'20'                    Use 'MVCLE' in write loop

00000600  E0                                        3773 OPTFLAG  DC    AL1(OPTMVC+OPTMVCL+OPTMVCLE)   Test options flag




00000601                      00000601  00000800    3775          ORG   CBUC+X'800'

00000800  40404040 40404040                         3777 WORK     DC    CL8' '                   MUST be doubleword ALIGNED!
```

```
  LOC         OBJECT CODE        ADDR1     ADDR2     STMT

                                 00000000  00000001  3780 R0        EQU   0
                                 00000001  00000001  3781 R1        EQU   1
                                 00000002  00000001  3782 R2        EQU   2
                                 00000003  00000001  3783 R3        EQU   3
                                 00000004  00000001  3784 R4        EQU   4
                                 00000005  00000001  3785 R5        EQU   5
                                 00000006  00000001  3786 R6        EQU   6
                                 00000007  00000001  3787 R7        EQU   7
                                 00000008  00000001  3788 R8        EQU   8
                                 00000009  00000001  3789 R9        EQU   9
                                 0000000A  00000001  3790 R10       EQU   10
                                 0000000B  00000001  3791 R11       EQU   11
                                 0000000C  00000001  3792 R12       EQU   12
                                 0000000D  00000001  3793 R13       EQU   13
                                 0000000E  00000001  3794 R14       EQU   14
                                 0000000F  00000001  3795 R15       EQU   15


                                                     3797           END
```

SYMBOL	TYPE	VALUE	LENGTH	DEFN	REFERENCES								
BEGIN	I	000200	2	3623	3613								
BEGIN2	I	000224	4	3636	3631								
CBUC	J	000000	2056	3577	3580	3587	3601	3612	3614	3752	3760	3767	3775	3621
CODE	2	000000	2056	3577									
DWAT0009	3	000318	8	3725	3724								
DWAT0010	3	000330	8	3731	3730								
DWAT0011	3	000340	8	3736	3735								
DWAT0012	3	000350	8	3741	3740								
FAILFLAG	X	000302	1	3716	3718	3727							
FAILTEST	I	000320	4	3727	3655	3719							
IMAGE	1	000000	2056	0									
NOMVC1	U	000288	1	3666	3664								
NOMVC2	U	0002C8	1	3689	3687								
NOMVCL1	U	0002A0	1	3675	3669								
NOMVCL2	U	0002E0	1	3698	3692								
NOMVCLE1	U	0002BA	1	3684	3678								
NOMVCLE2	U	0002FA	1	3707	3701								
OPTFLAG	R	000600	1	3773	3663	3668	3677	3686	3691	3700			
OPTMVC	U	000080	1	3769	3663	3686	3773						
OPTMVCL	U	000040	1	3770	3668	3691	3773						
OPTMVCLE	U	000020	1	3771	3677	3700	3773						
PATTERN1	C	000501	16	3763	3649	3665	3681	3695					
PATTERN2	C	000513	16	3765	3654	3672	3688	3704					
PREVORG	U	000200	1	3600	3604								
R0	U	000000	1	3780	3621	3623	3627	3633	3639	3643	3651	3656	3659	3709
R1	U	000001	1	3781	3624	3629							
R10	U	00000A	1	3790									
R11	U	00000B	1	3791									
R12	U	00000C	1	3792									
R13	U	00000D	1	3793									
R14	U	00000E	1	3794									
R15	U	00000F	1	3795									
R2	U	000002	1	3782	3625	3627	3630	3633	3636	3639			
R3	U	000003	1	3783	3626								
R4	U	000004	1	3784	3631	3632	3637	3638					
R5	U	000005	1	3785									
R6	U	000006	1	3786	3670	3674	3679	3683	3693	3697	3702	3706	
R7	U	000007	1	3787	3671	3673	3680	3682	3694	3696	3703	3705	
R8	U	000008	1	3788	3672	3674	3681	3683	3695	3697	3704	3706	
R9	U	000009	1	3789	3673	3682	3696	3705					
RDCOUNT	A	000358	4	3747	3643								
RDLOOPS	U	000000	1	68	3747								
READ2	I	00025C	6	3654	3650								
READDEST	C	000400	8	3757	3647								
READER	I	000238	4	3643									
READLOOP	I	00023C	4	3644	3651	3656							
SIG1FAIL	I	000338	4	3733	3634								
SIG2FAIL	I	000348	4	3738	3640								
STOPFLAG	X	000500	1	3762	3644	3660	3721	3728	3733	3738			
STOPTEST	I	000304	4	3718	3645	3652	3657	3661	3710				
WORK	C	000800	8	3777	3647	3649	3654						
WRCOUNT	A	00035C	4	3748	3659								

```
     SYMBOL         TYPE   VALUE    LENGTH  DEFN  REFERENCES

WRITDEST             C    0003FD       16  3755  3665  3670  3671  3679  3680  3688  3693  3694  3702  3703
WRITER               I    00026E        4  3659  3637
WRITLOOP             I    000272        4  3660  3709
WRLOOPS              U    000000        1    67  3748
```

```
 MACRO     DEFN  REFERENCES

ANTR        172
APROB       304
ARCHIND     464   3494
ARCHLVL     605   3493
ASAIPL      731   3610
ASALOAD     811   3576
ASAREA      866
ASAZAREA   1051
CPUWAIT    1134
DSECTS     1460
DWAIT      1663   3723   3729   3734   3739
DWAITEND   1720   3722
ENADEV     1728
ESA390     1828
IOCB       1839
IOCBDS     2015
IOFMT      2049
IOINIT     2387
IOTRFR     2428
ORB        2476
POINTER    2665
PSWFMT     2693
RAWAIT     2827
RAWIO      2923
SIGCPU     3081
SMMGR      3139
SMMGRB     3239
TRAP128    3288   3588
TRAP64     3265   3578   3581
TRAPS      3301
ZARCH      3375
ZEROH      3387
ZEROL      3415
ZEROLH     3443
ZEROLL     3466
```

```
   DESC    SYMBOL   SIZE    POS      ADDR

Entry: 0

Image      IMAGE    2056  000-807  000-807
  Region   CODE     2056  000-807  000-807
    CSECT  CBUC     2056  000-807  000-807
```

```
   STMT                                          FILE NAME

1     c:\Users\Fish\Documents\Visual Studio 2008\Projects\MyProjects\ASMA-0\CBUC\CBUC.asm
2     C:\Users\Fish\Documents\Visual Studio 2008\Projects\Hercules\_Git\_Harold\SATK-0\srcasm\satk.mac


** NO ERRORS FOUND **
```