```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                               2 ***********************************************************************
                                               3 *                                                                     *
                                               4 *             Testcase for CDSG, STPQ and LPQ Instructions            *
                                               5 *                                                                     *
                                               6 ***********************************************************************
                                               7 *                                                                     *
                                               8 *  The CDSG, STPQ and LPQ instructions all have 'Interlocked-Update   *
                                               9 *  References' as described in the Principles of Operation (POP),     *
                                              10 *  also known as being 'atomic'.  Hercules implements these using the *
                                              11 *  "cmpxchg16" function, which can be hardware assisted for certain   *
                                              12 *  host architectures.  On X86-64 / AMD64 processors, the             *
                                              13 *  "cmpxchg16b" instruction can be used when available.               *
                                              14 *                                                                     *
                                              15 *  This means that whilst one CPU performs a CDSG instruction, its    *
                                              16 *  memory references are interlocked against those by other CPU's.    *
                                              17 *  This test attempts to verify this, with an approach, similar to    *
                                              18 *  the one in the CBUC test.                                          *
                                              19 *                                                                     *
                                              20 *  The STPQ and LPQ instructions also use "cmpxchg16" and are tested  *
                                              21 *  in here as well.                                                   *
                                              22 *                                                                     *
                                              23 ***********************************************************************

                                              25 ***********************************************************************
                                              26 *                                                                     *
                                              27 *                       Example test scripts                          *
                                              28 *                                                                     *
                                              29 *                            (CDSG.tst)                               *
                                              30 *                                                                     *
                                              31 * *Testcase for CDSG, STPQ and LPQ Instructions                       *
                                              32 * mainsize    1                                                       *
                                              33 * numcpu      2                                                       *
                                              34 * sysclear                                                            *
                                              35 * archlvl     z/Arch                                                  *
                                              36 * loadcore    "$(testpath)/CDSG.core"                                 *
                                              37 * runtest     1                                                       *
                                              38 * v 900.38                                                            *
                                              39 * v 940.70                                                            *
                                              40 * *Compare                                                            *
                                              41 * v 940.10                                                            *
                                              42 * *Want "Success! CDSQ STPQ LPQ" E2A48383 85A2A240 5A40C3C4 E2C76B40  *
                                              43 * #v 960.100                                                          *
                                              44 * *Done                                                               *
                                              45 *                                                                     *
                                              46 ***********************************************************************
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                              48 ***********************************************************************
                                              49 *                                                                     *
                                              50 *                        PROGRAMMING NOTE                             *
                                              51 *                                                                     *
                                              52 *  During initialisation we test the functionality of the Store Pair  *
                                              53 *  to Quadword (STPQ) and load Pair from Quadword (LPQ) instructions. *
                                              54 *  We merely do this as these two intructions also rely on the        *
                                              55 *  Hercules macro cmpchxg16 (as of the most recent updates), as does  *
                                              56 *  the CDSG implementation.  This cmpxchg16 macro may be assisted,    *
                                              57 *  depending on the Hercules host architecture.                       *
                                              58 *                                                                     *
                                              59 *  After initialisation, a second CPU is started, at which point in   *
                                              60 *  time both CPU's perform a loop lasting LOOPMAX iterations.  The    *
                                              61 *  first of the loop centers around the CDSG instruction.  The second *
                                              62 *  loop performs 2 CSG instructions, the first one of which overlaps  *
                                              63 *  with the DEST2 destination of CDSG.  This overlapping causes CC=1  *
                                              64 *  when it occurs.  As a result, the non-overlapping destinations     *
                                              65 *  get incremented exactly LOOPMAX times, the overlapping part twice  *
                                              66 *  that.  This is what is being checked for success of the test.  By  *
                                              67 *  inspecting the CDSGCNTR and CSG_CNTR, one can see how many         *
                                              68 *  attempts were needed.  These are allmost always higher than        *
                                              69 *  LOOPMAX, the exact number varies on every testrun.                 *
                                              70 *                                                                     *
                                              71 *  The DESTination area DEST1+DEST2+DEST3 is 3 * 8 bytes long,        *
                                              72 *  and is initialised as follows :                                    *
                                              73 *                                                                     *
                                              74 *     '1A1B2A2B3A3B00004A4B5A5B6A6B00007A7B8A8B9A9B0000'X             *
                                              75 *                                                                     *
                                              76 *  The first loop works against DEST1+DEST2 (the leftmost 16 bytes),  *
                                              77 *  the second loop against DEST2+DEST3 (the rightmost 16 bytes).      *
                                              78 *  Both loops implement an atomic "increment" with the value :        *
                                              79 *                                                                     *
                                              80 *     '00000000000000010000000000000001'X                             *
                                              81 *                                                                     *
                                              82 *  Thus also for the second loop :                                    *
                                              83 *                                                                     *
                                              84 *                     '00000000000000010000000000000001'X             *
                                              85 *                                                                     *
                                              86 *  The process ends when both loops have incremented LOOPMAX times,   *
                                              87 *  and the doubleword in the middle will have been incremented        *
                                              88 *  exactly twice that amount -- that is, if the CDSG operation is     *
                                              89 *  really atomic.  And that is what will decide a successfull CDSG    *
                                              90 *  instruction or not.                                                *
                                              91 *                                                                     *
                                              92 ***********************************************************************
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                              94          PRINT OFF
                                            3475          PRINT ON

                                            3477 ***********************************************************************
                                            3478 *        SATK prolog stuff...                                          *
                                            3479 ***********************************************************************

                                            3481          ARCHLVL  MNOTE=NO
                                            3483+$AL      OPSYN AL
                                            3484+$ALR     OPSYN ALR
                                            3485+$B       OPSYN B
                                            3486+$BAS     OPSYN BAS
                                            3487+$BASR    OPSYN BASR
                                            3488+$BC      OPSYN BC
                                            3489+$BCTR    OPSYN BCTR
                                            3490+$BE      OPSYN BE
                                            3491+$BH      OPSYN BH
                                            3492+$BL      OPSYN BL
                                            3493+$BM      OPSYN BM
                                            3494+$BNE     OPSYN BNE
                                            3495+$BNH     OPSYN BNH
                                            3496+$BNL     OPSYN BNL
                                            3497+$BNM     OPSYN BNM
                                            3498+$BNO     OPSYN BNO
                                            3499+$BNP     OPSYN BNP
                                            3500+$BNZ     OPSYN BNZ
                                            3501+$BO      OPSYN BO
                                            3502+$BP      OPSYN BP
                                            3503+$BXLE    OPSYN BXLE
                                            3504+$BZ      OPSYN BZ
                                            3505+$CH      OPSYN CH
                                            3506+$L       OPSYN L
                                            3507+$LH      OPSYN LH
                                            3508+$LM      OPSYN LM
                                            3509+$LPSW    OPSYN LPSW
                                            3510+$LR      OPSYN LR
                                            3511+$LTR     OPSYN LTR
                                            3512+$NR      OPSYN NR
                                            3513+$SL      OPSYN SL
                                            3514+$SLR     OPSYN SLR
                                            3515+$SR      OPSYN SR
                                            3516+$ST      OPSYN ST
                                            3517+$STM     OPSYN STM
                                            3518+$X       OPSYN X
 LOC       OBJECT CODE       ADDR1   ADDR2   STMT

                                             3520 ********************************************************************
                                             3521 *        Initiate the CDSG CSECT in the CODE region                *
                                             3522 *        with the location counter at 0                            *
                                             3523 ********************************************************************

                                             3525 CDSGTEST ASALOAD  REGION=CODE
                            000000  0409CB   3526+CDSGTEST START 0,CODE
000000  000A0000 00000008                    3528+         PSW   0,0,2,0,X'008'      64-bit Restart ISR Trap New PSW
000008                      000008  000058   3529+         ORG   CDSGTEST+X'058'
000058  000A0000 00000018                    3531+         PSW   0,0,2,0,X'018'      64-bit External ISR Trap New PSW
000060  000A0000 00000020                    3532+         PSW   0,0,2,0,X'020'      64-bit Supervisor Call ISR Trap New PSW
000068  000A0000 00000028                    3533+         PSW   0,0,2,0,X'028'      64-bit Program ISR Trap New PSW
000070  000A0000 00000030                    3534+         PSW   0,0,2,0,X'030'      64-bit Machine Check Trap New PSW
000078  000A0000 00000038                    3535+         PSW   0,0,2,0,X'038'      64-bit Input/Output Trap New PSW
000080                      000080  000200   3536+         ORG   CDSGTEST+512


                                             3538 ********************************************************************
                                             3539 *        Define the z/Arch RESTART PSW                             *
                                             3540 ********************************************************************

                            000200  000001   3542 PREVORG  EQU   *
000200                      000200  0001A0   3543          ORG   CDSGTEST+X'1A0'
                                             3544 *        PSWZ  <sys>,<key>,<mwp>,<prog>,<addr>[,amode]
0001A0  00000001 80000000                    3545          PSWZ  0,0,0,0,X'200',64
0001B0                      0001B0  000200   3546          ORG   PREVORG


                                             3548 ********************************************************************
                                             3549 *        Create IPL (restart) PSW                                  *
                                             3550 ********************************************************************

                                             3552          ASAIPL   IA=BEGIN
                            000000  0409CB   3553+CDSGTEST CSECT
000200                      000200  000000   3554+         ORG   CDSGTEST
000000  00080000 00000200                    3555+         PSW   0,0,0,0,BEGIN,24
000008                      000008  000200   3556+         ORG   CDSGTEST+512     Reset CSECT to end of assigned storage area
                            000000  0409CB   3557+CDSGTEST CSECT
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                            3559 **********************************************************************
                                            3560 *        The actual CDSG program itself...                           *
                                            3561 **********************************************************************

000200                      000970          3563          USING CDSG,R0                    No base registers needed

000200  1F00                                3565 BEGIN    SLR   R0,R0                      Start clean
000202  4110 0001                   000001  3566          LA    R1,1                       Request z/Arch mode
000206  1F22                                3567          SLR   R2,R2                      Start clean
000208  1F33                                3568          SLR   R3,R3                      Start clean
00020A  AE02 0012                   000012  3569          SIGP  R0,R2,X'12'                Request z/Arch mode

00020E  1F11                                3571          SLR   R1,R1                      Start clean
000210  4120 0000                   000000  3572          LA    R2,0                       Get our CPU number
000214  4140 0224                   000224  3573          LA    R4,BEGIN2                  Our restart entry point
000218  4040 01AE                   0001AE  3574          STH   R4,X'1AE'                  Update restart PSW
00021C  AE02 0006                   000006  3575          SIGP  R0,R2,X'06'                Restart our CPU
000220  47F0 03E8                   0003E8  3576          B     SIG1FAIL                   WTF?! How did we get here?!

000224                                      3578 BEGIN2   DS    0H
000224  E340 0920 008F              000920  3579          LPQ   R4,INIT1                   Load INIT1+INIT2 using LPQ
00022A  E340 0920 0020              000920  3580          CG    R4,INIT1                   Did LPQ high DW work ...
000230  4770 0408                   000408  3581          BNE   LPQFAIL1                   ... or not ?
000234  E350 0928 0020              000928  3582          CG    R5,INIT2                   Did LPQ low DW work ...
00023A  4770 0420                   000420  3583          BNE   LPQFAIL2                   ... or not ?
00023E  E340 0900 008E              000900  3584          STPQ  R4,DEST1                   Store DEST1+DEST2 for CDSG use
                                            3585 *                                         R4+R5 for CDSGLOOP to use
000244  E340 0900 0020              000900  3586          CG    R4,DEST1                   Did STPQ high DW work ...
00024A  4770 0438                   000438  3587          BNE   STPQFAL1                   ... or not ?
00024E  E350 0908 0020              000908  3588          CG    R5,DEST2                   Did STPQ low DW work ...
000254  4770 0450                   000450  3589          BNE   STPQFAL2                   ... or not ?
000258  41E0 09D0                   0009D0  3590          LA    R14,TRACE                  Initialize CDSG trace pointer

00025C  B904 0065                           3592          LGR   R6,R5                      R6+R7 for CSG_LOOP to use
000260  E370 0930 0004              000930  3593          LG    R7,INIT3                   Load INIT3 to initialize ...
000266  E370 0910 0024              000910  3594          STG   R7,DEST3                   DEST3 so CSG_CPU can use it

00026C  4120 0001                   000001  3596          LA    R2,1                       Second CPU number
000270  4180 02EA                   0002EA  3597          LA    R8,CSG_CPU                 Point to its entry point
000274  4080 01AE                   0001AE  3598          STH   R8,X'1AE'                  Update restart PSW
000278  AE02 0006                   000006  3599          SIGP  R0,R2,X'06'                Restart second CPU
00027C  4770 03F8                   0003F8  3600          BNZ   SIG2FAIL                   WTF?! (SIGP failed!)
```

```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                            3602 ***********************************************************************
                                            3603 *  The first loop incremenets the rightmost halfwords of DEST1 and    *
                                            3604 *  DEST2 with a single atomic CDSG instruction.  But the 2nd CPU will *
                                            3605 *  attempt to do the same thing against DEST2 with a CSG instruction, *
                                            3606 *  thus causing collisions now and then.  We keep track of the total  *
                                            3607 *  CDSG's, which is limited to CNTRMAX.  The loop is expected to end  *
                                            3608 *  before that, when exactly LOOPMAX successfule increments (i.e.     *
                                            3609 *  SWAP operations) have taken place.                                 *
                                            3610 ***********************************************************************


000280  B982 00CC                           3612 CDSG_CPU XGR   R12,R12                  Initialise CDSG counter
000284                                      3613 CDSGLOOP DS    0H
000284  9201 E003                   000003  3614          MVI   3(R14),X'1'              Trace CC=1 from previous CDSG
000288                                      3615 CDSG_CC0 DS    0H
000288  9500 09C6                   0009C6  3616          CLI   STOPFLAG,X'00'           Are we being asked to stop?
00028C  4770 0364                   000364  3617          BNE   GOODEOJ                  Yes, then do so.
000290  41CC 0001                   000001  3618          LA    R12,1(R12)               Increment the CDSG counter
000294  50C0 099C                   00099C  3619          ST    R12,CDSGCNTR             Update CDSG counter
000298  59C0 09B0                   0009B0  3620          C     R12,CNTRMAX              CDSG counter overrun
00029C  4720 0468                   000468  3621          BH    CDSGCNTO                  Yes, that should never happen
0002A0  4184 0001                   000001  3622          LA    R8,1(R4)                 Increment DEST1
0002A4  4195 0001                   000001  3623          LA    R9,1(R5)                 Increment DEST1
0002A8  41EE 0010                   000010  3624          LA    R14,16(R14)              Point to the next TRACE entry
0002AC  409E 0000                   000000  3625          STH   R9,0(R14)                Trace the CDSG DEST2 update
0002B0  408E 0004                   000004  3626          STH   R8,4(R14)                Trace the CDSG DEST1 update
0002B4  EB48 0900 003E              000900  3627          CDSG  R4,R8,DEST1              CDSG to attempt doing it
0002BA  4770 0284                   000284  3628          BNE   CDSGLOOP                 The CSG_CPU came in between
0002BE  BD83 09B4                   0009B4  3629          CLM   R8,B'0011',LOOPMAX       End value reached ?
0002C2  4780 02D2                   0002D2  3630          BE    CDSGEND                  Yes, CDSG incrementing ended
0002C6  B904 0048                           3631          LGR   R4,R8                    Copy the incremented DEST1
0002CA  B904 0059                           3632          LGR   R5,R9                    Copy the incremented DEST2
0002CE  47F0 0288                   000288  3633          B     CDSG_CC0                 Go try the next CDSG
0002D2                                      3634 CDSGEND  DS    0H
0002D2  D501 0916 09B4      000916  0009B4  3635          CLC   DEST3+6(2),LOOPMAX       Is also CSG_LOOP ended yet ?
0002D8  4770 02D2                   0002D2  3636          BNE   CDSGEND                  Spin-loop style waiting for it
0002DC                                      3637 FINALTST DS    0H                       OK, both loops are finished
0002DC  D501 090E 09B6      00090E  0009B6  3638          CLC   DEST2+6(2),LOOPMAX2      DEST2 must be =2*LOOPMAX
0002E2  4780 0364                   000364  3639          BE    GOODEOJ                  Yes, then we have success !
0002E6  47F0 03D8                   0003D8  3640          B     FAILEOJ                  No, the test failed !!
```

```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                            3642 ***********************************************************************
                                            3643 *  The second loop incremenets the rightmost halfwords of DEST2 and   *
                                            3644 *  DEST3, both of which use the atomic CSG instruction.  But the one  *
                                            3645 *  against DEST2 will collide now and then with the CDSG atomic       *
                                            3646 *  increments of DEST1+DEST2.  We keep track of the total number of   *
                                            3647 *  CSG's against DEST2, which is limited to CNTRMAX.  The loop is     *
                                            3648 *  expected to end before that, when DEST2 (and DEST3 also) have been *
                                            3649 *  able to increment (i.e. SWAP) exactly LOOPMAX times.               *
                                            3650 ***********************************************************************


0002EA  B982 00DD                           3652 CSG_CPU  XGR   R13,R13                  Initialise CSG counter
0002EE  E360 0908 0004              000908  3653          LG    R6,DEST2                 Initialise R6 for CSG
0002F4  E370 0910 0004              000910  3654          LG    R7,DEST3                 Initialise R7 for CSG
0002FA  41F0 09D8                   0009D8  3655          LA    R15,TRACE+8              Initialize CSG trace pointer
0002FE                                      3656 CSG_LOOP DS    0H
0002FE  9201 F003                   000003  3657          MVI   3(R15),X'1'              Trace CC=1 from previous CSG
000302                                      3658 CSG_CC0  DS    0H
000302  9500 09C6                   0009C6  3659          CLI   STOPFLAG,X'00'           Are we being asked to stop?
000306  4770 03C8                   0003C8  3660          BNE   ENDNOTOK                 Yes, then do so.
00030A  41DD 0001                   000001  3661          LA    R13,1(R13)               Increment the CSG counter
00030E  50D0 09AC                   0009AC  3662          ST    R13,CSG_CNTR             Update CSG counter
000312  59D0 09B0                   0009B0  3663          C     R13,CNTRMAX              CSG counter overrun
000316  4720 0480                   000480  3664          BH    CSGCNTO                  Yes, that should never happen
00031A  41A6 0001                   000001  3665          LA    R10,1(R6)                Increment DEST2
00031E  41FF 0010                   000010  3666          LA    R15,16(R15)              Point to the next TRACE entry
000322  40AF 0000                   000000  3667          STH   R10,0(R15)               Trace the CSG DEST2 update
000326  40BF 0004                   000004  3668          STH   R11,4(R15)               Trace the previous DEST3 update
00032A  EB6A 0908 0030              000908  3669          CSG   R6,R10,DEST2             CSG to attempt doing it
000330  4770 02FE                   0002FE  3670          BNE   CSG_LOOP                 The CDSG_CPU came in between
000334  41B7 0001                   000001  3671          LA    R11,1(R7)                Increments DEST3
000338                                      3672 CSGLOOP2 DS    0H
000338  EB7B 0910 0030              000910  3673          CSG   R7,R11,DEST3             CSG to attempt doing it
00033E  4770 0338                   000338  3674          BNE   CSGLOOP2                 CDSGEND read came in between
000342  BDB3 09B4                   0009B4  3675          CLM   R11,B'0011',LOOPMAX      End value reached ?
000346  4780 0356                   000356  3676          BE    CSG_END                  Yes, CSG incrementing ended
00034A  B904 006A                           3677          LGR   R6,R10                   Copy the incremented DEST2
00034E  B904 007B                           3678          LGR   R7,R11                   Copy the incremented DEST3
000352  47F0 0302                   000302  3679          B     CSG_CC0                  Go try the next CSG
000356                                      3680 CSG_END  DS    0H
000356  D501 0906 09B4      000906  0009B4  3681          CLC   DEST1+6(2),LOOPMAX       Is also CDSGLOOP ended yet ?
00035C  4770 0356                   000356  3682          BNE   CSG_END                  Spin-loop style waiting for it
000360  47F0 03BA                   0003BA  3683          B     ENDOK                    OK, both loops are finished
```

```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                            3685 **********************************************************************
                                            3686 *        Final counter updates.                                      *
                                            3687 **********************************************************************


000364                                      3689 GOODEOJ  DS       0H
000364  D21F 0940 0498      000940  000498  3690          MVC      STATUS,=CL32'Success ! CDSG, STPQ and LPQ: OK'
00036A  58C0 099C                   00099C  3691          L        R12,CDSGCNTR            Load CSDG counter
00036E  4EC0 09B8                   0009B8  3692          CVD      R12,PACKED              Convert CDSG counter to packed
000372  D205 0974 09C0      000974  0009C0  3693          MVC      CDSGCMPR,EDIT           Copy EDIT mask in CDSG counter
000378  DE05 0974 09BD      000974  0009BD  3694          ED       CDSGCMPR,PACKED+5       CDSG counter in zoned format
00037E  48C0 0906                   000906  3695          LH       R12,DEST1+6             Load CDSG SWAPs counter
000382  4EC0 09B8                   0009B8  3696          CVD      R12,PACKED              Convert it to packed
000386  D205 097A 09C0      00097A  0009C0  3697          MVC      CDSGSWAP,EDIT           Copy EDIT mask in CDSG SWAP ctr
00038C  DE05 097A 09BD      00097A  0009BD  3698          ED       CDSGSWAP,PACKED+5       CDSG SWAPS counter in zoned
000392  58D0 09AC                   0009AC  3699          L        R13,CSG_CNTR            Load CSG counter
000396  4ED0 09B8                   0009B8  3700          CVD      R13,PACKED              Convert CSG  counter to packed
00039A  D205 0984 09C0      000984  0009C0  3701          MVC      CSG_CMPR,EDIT           Copy EDIT mask in CSG  counter
0003A0  DE05 0984 09BD      000984  0009BD  3702          ED       CSG_CMPR,PACKED+5       CSG  counter in zoned format
0003A6  48D0 0916                   000916  3703          LH       R13,DEST3+6             Load CSG SWAPs counter
0003AA  4ED0 09B8                   0009B8  3704          CVD      R13,PACKED              Convert it to packed
0003AE  D205 098A 09C0      00098A  0009C0  3705          MVC      CSG_SWAP,EDIT           Copy EDIT mask in CSG SWAP cntr
0003B4  DE05 098A 09BD      00098A  0009BD  3706          ED       CSG_SWAP,PACKED+5       CSG SWAPS counter in zoned
```

```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                            3708 ***********************************************************************
                                            3709 *        PSWs and final status update.                                *
                                            3710 ***********************************************************************


0003BA                                      3712 ENDOK    DS       0H
                                            3713          DWAITEND LOAD=YES               Normal completion
0003BA  8200 03C0                   0003C0  3715+         LPSW  DWAT0008
0003C0  000A0000 00000000                   3716+DWAT0008 PSW   0,0,2,0,X'000000'



0003C8                                      3718 ENDNOTOK DS       0H
                                            3719          DWAIT    LOAD=YES,CODE=BAD     Abnormal termination
0003C8  8200 03D0                   0003D0  3720+         LPSW  DWAT0009
0003D0  000A0000 00010BAD                   3721+DWAT0009 PSW   0,0,2,0,X'010BAD'



0003D8  92FF 09C6                   0009C6  3723 FAILEOJ  MVI      STOPFLAG,X'FF'        Tell the other CPU to stop
                                            3724          DWAIT    LOAD=YES,CODE=BAD     Abnormal termination
0003DC  8200 03E0                   0003E0  3725+         LPSW  DWAT0010
0003E0  000A0000 00010BAD                   3726+DWAT0010 PSW   0,0,2,0,X'010BAD'



0003E8  92FF 09C6                   0009C6  3728 SIG1FAIL MVI      STOPFLAG,X'FF'        Tell the other CPU to stop
                                            3729          DWAIT    LOAD=YES,CODE=111     First SIGP failed
0003EC  8200 03F0                   0003F0  3730+         LPSW  DWAT0011
0003F0  000A0000 00010111                   3731+DWAT0011 PSW   0,0,2,0,X'010111'



0003F8  92FF 09C6                   0009C6  3733 SIG2FAIL MVI      STOPFLAG,X'FF'        Tell the other CPU to stop
                                            3734          DWAIT    LOAD=YES,CODE=222     Second SIGP failed
0003FC  8200 0400                   000400  3735+         LPSW  DWAT0012
000400  000A0000 00010222                   3736+DWAT0012 PSW   0,0,2,0,X'010222'



000408  92FF 09C6                   0009C6  3738 LPQFAIL1 MVI      STOPFLAG,X'FF'        Tell the other CPU to stop
00040C  D21F 0940 04B8      000940  0004B8  3739          MVC      STATUS,=CL32'Failure! LPQ  Hi  does NOT match'
                                            3740          DWAIT    LOAD=YES,CODE=333     LPQ High part failed
000412  8200 0418                   000418  3741+         LPSW  DWAT0013
000418  000A0000 00010333                   3742+DWAT0013 PSW   0,0,2,0,X'010333'
```

```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

000420  92FF 09C6                   0009C6  3744 LPQFAIL2 MVI      STOPFLAG,X'FF'         Tell the other CPU to stop
000424  D21F 0940 04D8      000940  0004D8  3745          MVC      STATUS,=CL32'Failure! LPQ  Lo  does NOT match'
                                            3746          DWAIT    LOAD=YES,CODE=444     LPQ Low part failed
00042A  8200 0430                   000430  3747+         LPSW  DWAT0014
000430  000A0000 00010444                   3748+DWAT0014 PSW   0,0,2,0,X'010444'


000438  92FF 09C6                   0009C6  3750 STPQFAL1 MVI      STOPFLAG,X'FF'         Tell the other CPU to stop
00043C  D21F 0940 04F8      000940  0004F8  3751          MVC      STATUS,=CL32'Failure! STPQ Hi  does NOT match'
                                            3752          DWAIT    LOAD=YES,CODE=555     STPQ High part failes
000442  8200 0448                   000448  3753+         LPSW  DWAT0015
000448  000A0000 00010555                   3754+DWAT0015 PSW   0,0,2,0,X'010555'


000450  92FF 09C6                   0009C6  3756 STPQFAL2 MVI      STOPFLAG,X'FF'         Tell the other CPU to stop
000454  D21F 0940 0518      000940  000518  3757          MVC      STATUS,=CL32'Failure! STPQ Lo  does NOT match'
                                            3758          DWAIT    LOAD=YES,CODE=666     STPQ Low part failed
00045A  8200 0460                   000460  3759+         LPSW  DWAT0016
000460  000A0000 00010666                   3760+DWAT0016 PSW   0,0,2,0,X'010666'


000468  92FF 09C6                   0009C6  3762 CDSGCNTO MVI      STOPFLAG,X'FF'         Tell the other CPU to stop
00046C  D21F 0940 0538      000940  000538  3763          MVC      STATUS,=CL32'Failure! CDSG    Counter Overrun'
                                            3764          DWAIT    LOAD=YES,CODE=777     CDSG Counter Overrun
000472  8200 0478                   000478  3765+         LPSW  DWAT0017
000478  000A0000 00010777                   3766+DWAT0017 PSW   0,0,2,0,X'010777'


000480  92FF 09C6                   0009C6  3768 CSGCNTO  MVI      STOPFLAG,X'FF'         Tell the other CPU to stop
000484  D21F 0940 0558      000940  000558  3769          MVC      STATUS,=CL32'Failure! CBG     Counter Overrun'
                                            3770          DWAIT    LOAD=YES,CODE=888     CSG Counter Overrun
00048A  8200 0490                   000490  3771+         LPSW  DWAT0018
000490  000A0000 00010888                   3772+DWAT0018 PSW   0,0,2,0,X'010888'
```

```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                                            3774 ***********************************************************************
                                            3775 *        Working Storage                                              *
                                            3776 ***********************************************************************

000498                                      3778          LTORG ,                         Literals pool
000498  E2A48383 85A2A240                   3779                =CL32'Success ! CDSG, STPQ and LPQ: OK'
0004B8  C6818993 A499855A                   3780                =CL32'Failure! LPQ  Hi  does NOT match'
0004D8  C6818993 A499855A                   3781                =CL32'Failure! LPQ  Lo  does NOT match'
0004F8  C6818993 A499855A                   3782                =CL32'Failure! STPQ Hi  does NOT match'
000518  C6818993 A499855A                   3783                =CL32'Failure! STPQ Lo  does NOT match'
000538  C6818993 A499855A                   3784                =CL32'Failure! CDSG    Counter Overrun'
000558  C6818993 A499855A                   3785                =CL32'Failure! CBG     Counter Overrun'



000578                      000578  000900  3787          ORG   CDSGTEST+X'900'
000900                                      3788          CNOP  0,16                      MUST be quadword ALIGNED!
000900  00000000 00000000                   3789 DEST1    DC    XL8'0000000000000000'     DEST1+DEST2 updated using CDSG
000908  00000000 00000000                   3790 DEST2    DC    XL8'0000000000000000'     DEST2 updated using CSG
000910  00000000 00000000                   3791 DEST3    DC    XL8'0000000000000000'     DEST3 updated whenever CSG
                                            3792 *                                        successfully updates DEST
000918  07000700 07000700                   3793          CNOP  0,16                      MUST be quadword ALIGNED!
000920  1A1B2A2B 3A3B0000                   3794 INIT1    DC    XL8'1A1B2A2B3A3B0000'     Initial value for DEST1
000928  4A4B5A5B 6A6B0000                   3795 INIT2    DC    XL8'4A4B5A5B6A6B0000'     Initial value for DEST2
000930  7A7B8A8B 9A9B0000                   3796 INIT3    DC    XL8'7A7B8A8B9A9B0000'     Initial value for DEST3

000938  07000700 07000700                   3798          CNOP  0,16                      So that the output looks better
000940  C6818993 A499855A                   3799 STATUS   DC    CL32'Failure!'            Overall status message
000960  C995A2A3 994B40C3                   3800 COUNTERS DC    CL16'Instr. CMP  SWAP'    Title column
000970  C3C4E2C7                            3801 CDSG     DC    CL4'CDSG'                 CDSG Instruction
000974  40404040 4040                       3802 CDSGCMPR DC    CL6' '                    CDSG instructions attempted
00097A  40404040 4040                       3803 CDSGSWAP DC    CL6' '                    CDSG successful swaps done
000980  C3E2C740                            3804 CSG      DC    CL4'CSG '                 CSG  Instruction
000984  40404040 4040                       3805 CSG_CMPR DC    CL6' '                    CSG  DEST2 instructions done
00098A  40404040 4040                       3806 CSG_SWAP DC    CL6' '                    CSG  DEST2 successful swaps

000990                                      3808          CNOP  0,16                      So that the output looks better
000990  C3C4E2C7 C3D5E3D9                   3809 CDSGCNTL DC    CL8'CDSGCNTR'             CDSG counter label
000998  40404040                            3810          DC    CL4' '
00099C  00000000                            3811 CDSGCNTR DC    F'0'                      CDSG instructions attempted
0009A0  C3E2C76D C3D5E3D9                   3812 CSG_CNTL DC    CL8'CSG_CNTR'             CSG  counter label
0009A8  40404040                            3813          DC    CL4' '
0009AC  00000000                            3814 CSG_CNTR DC    F'0'                      CSG  instructions attempted

0009B0  0000EA60                            3816 CNTRMAX  DC    F'60000'                  Counter Overrun Maximum
0009B4  3A98                                3817 LOOPMAX  DC    H'15000'                  Maximum number of loops
0009B6  7530                                3818 LOOPMAX2 DC    H'30000'                  Maximum number of loops * 2

0009B8  00000000 0000000C                   3820 PACKED   DC    PL8'0'                    Packed decimal work area
0009C0  40202020 2020                       3821 EDIT     DC    XL6'402020202020'         EDIT mask with leading blank
0009C6  00                                  3822 STOPFLAG DC    X'00'                     Set to non-zero to stop test
```

```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT


0009C8  07000700 07000700                   3824          CNOP  0,16                     Beautify trace table looks
0009D0  00000000 00000000                   3825 TRACE    DC    65535F'0'                Trace area for debugging
```

```
 LOC       OBJECT CODE      ADDR1   ADDR2   STMT

                            000000  000001  3828 R0        EQU   0
                            000001  000001  3829 R1        EQU   1
                            000002  000001  3830 R2        EQU   2
                            000003  000001  3831 R3        EQU   3
                            000004  000001  3832 R4        EQU   4
                            000005  000001  3833 R5        EQU   5
                            000006  000001  3834 R6        EQU   6
                            000007  000001  3835 R7        EQU   7
                            000008  000001  3836 R8        EQU   8
                            000009  000001  3837 R9        EQU   9
                            00000A  000001  3838 R10       EQU   10
                            00000B  000001  3839 R11       EQU   11
                            00000C  000001  3840 R12       EQU   12
                            00000D  000001  3841 R13       EQU   13
                            00000E  000001  3842 R14       EQU   14
                            00000F  000001  3843 R15       EQU   15


                                            3845           END
```

```
     SYMBOL        TYPE  VALUE   LENGTH  DEFN  REFERENCES

BEGIN               I    000200       2  3565  3555
BEGIN2              H    000224       2  3578  3573
CDSG                C    000970       4  3801  3563
CDSGCMPR            C    000974       6  3802  3693  3694
CDSGCNTL            C    000990       8  3809
CDSGCNTO            I    000468       4  3762  3621
CDSGCNTR            F    00099C       4  3811  3619  3691
CDSGEND             H    0002D2       2  3634  3630  3636
CDSGLOOP            H    000284       2  3613  3628
CDSGSWAP            C    00097A       6  3803  3697  3698
CDSGTEST            J    000000  264652  3526  3529  3536  3543  3554  3556  3787
CDSG_CC0            H    000288       2  3615  3633
CDSG_CPU            I    000280       4  3612
CNTRMAX             F    0009B0       4  3816  3620  3663
CODE                2    000000  264652  3526
COUNTERS            C    000960      16  3800
CSG                 C    000980       4  3804
CSGCNTO             I    000480       4  3768  3664
CSGLOOP2            H    000338       2  3672  3674
CSG_CC0             H    000302       2  3658  3679
CSG_CMPR            C    000984       6  3805  3701  3702
CSG_CNTL            C    0009A0       8  3812
CSG_CNTR            F    0009AC       4  3814  3662  3699
CSG_CPU             I    0002EA       4  3652  3597
CSG_END             H    000356       2  3680  3676  3682
CSG_LOOP            H    0002FE       2  3656  3670
CSG_SWAP            C    00098A       6  3806  3705  3706
DEST1               X    000900       8  3789  3584  3586  3627  3681  3695
DEST2               X    000908       8  3790  3588  3638  3653  3669
DEST3               X    000910       8  3791  3594  3635  3654  3673  3703
DWAT0008            3    0003C0       8  3716  3715
DWAT0009            3    0003D0       8  3721  3720
DWAT0010            3    0003E0       8  3726  3725
DWAT0011            3    0003F0       8  3731  3730
DWAT0012            3    000400       8  3736  3735
DWAT0013            3    000418       8  3742  3741
DWAT0014            3    000430       8  3748  3747
DWAT0015            3    000448       8  3754  3753
DWAT0016            3    000460       8  3760  3759
DWAT0017            3    000478       8  3766  3765
DWAT0018            3    000490       8  3772  3771
EDIT                X    0009C0       6  3821  3693  3697  3701  3705
ENDNOTOK            H    0003C8       2  3718  3660
ENDOK               H    0003BA       2  3712  3683
FAILEOJ             I    0003D8       4  3723  3640
FINALTST            H    0002DC       2  3637
GOODEOJ             H    000364       2  3689  3617  3639
IMAGE               1    000000  264652     0
INIT1               X    000920       8  3794  3579  3580
INIT2               X    000928       8  3795  3582
INIT3               X    000930       8  3796  3593
LOOPMAX             H    0009B4       2  3817  3629  3635  3675  3681
```

```
     SYMBOL        TYPE  VALUE   LENGTH  DEFN  REFERENCES

LOOPMAX2            H    0009B6       2  3818  3638
LPQFAIL1            I    000408       4  3738  3581
LPQFAIL2            I    000420       4  3744  3583
PACKED              P    0009B8       8  3820  3692  3694  3696  3698  3700  3702  3704  3706
PREVORG             U    000200       1  3542  3546
R0                  U    000000       1  3828  3563  3565  3569  3575  3599
R1                  U    000001       1  3829  3566  3571
R10                 U    00000A       1  3838  3665  3667  3669  3677
R11                 U    00000B       1  3839  3668  3671  3673  3675  3678
R12                 U    00000C       1  3840  3612  3618  3619  3620  3691  3692  3695  3696
R13                 U    00000D       1  3841  3652  3661  3662  3663  3699  3700  3703  3704
R14                 U    00000E       1  3842  3590  3614  3624  3625  3626
R15                 U    00000F       1  3843  3655  3657  3666  3667  3668
R2                  U    000002       1  3830  3567  3569  3572  3575  3596  3599
R3                  U    000003       1  3831  3568
R4                  U    000004       1  3832  3573  3574  3579  3580  3584  3586  3622  3627  3631
R5                  U    000005       1  3833  3582  3588  3592  3623  3632
R6                  U    000006       1  3834  3592  3653  3665  3669  3677
R7                  U    000007       1  3835  3593  3594  3654  3671  3673  3678
R8                  U    000008       1  3836  3597  3598  3622  3626  3627  3629  3631
R9                  U    000009       1  3837  3623  3625  3632
SIG1FAIL            I    0003E8       4  3728  3576
SIG2FAIL            I    0003F8       4  3733  3600
STATUS              C    000940      32  3799  3690  3739  3745  3751  3757  3763  3769
STOPFLAG            X    0009C6       1  3822  3616  3659  3723  3728  3733  3738  3744  3750  3756  3762  3768
STPQFAL1            I    000438       4  3750  3587
STPQFAL2            I    000450       4  3756  3589
TRACE               F    0009D0       4  3825  3590  3655
=CL32'Failure! CBG     Counter Overrun'
                    C    000558      32  3785  3769
=CL32'Failure! CDSG    Counter Overrun'
                    C    000538      32  3784  3763
=CL32'Failure! LPQ  Hi  does NOT match'
                    C    0004B8      32  3780  3739
=CL32'Failure! LPQ  Lo  does NOT match'
                    C    0004D8      32  3781  3745
=CL32'Failure! STPQ Hi  does NOT match'
                    C    0004F8      32  3782  3751
=CL32'Failure! STPQ Lo  does NOT match'
                    C    000518      32  3783  3757
=CL32'Success ! CDSG, STPQ and LPQ: OK'
                    C    000498      32  3779  3690
```

```
 MACRO      DEFN  REFERENCES

ANTR         160
APROB        292
ARCHIND      452   3482
ARCHLVL      593   3481
ASAIPL       719   3552
ASALOAD      799   3525
ASAREA       854
ASAZAREA    1039
CPUWAIT     1122
DSECTS      1448
DWAIT       1651   3714   3719   3724   3729   3734   3740   3746   3752   3758   3764   3770
DWAITEND    1708   3713
ENADEV      1716
ESA390      1816
IOCB        1827
IOCBDS      2003
IOFMT       2037
IOINIT      2375
IOTRFR      2416
ORB         2464
POINTER     2653
PSWFMT      2681
RAWAIT      2815
RAWIO       2911
SIGCPU      3069
SMMGR       3127
SMMGRB      3227
TRAP128     3276
TRAP64      3253   3527   3530
TRAPS       3289
ZARCH       3363
ZEROH       3375
ZEROL       3403
ZEROLH      3431
ZEROLL      3454
```

```
   DESC      SYMBOL    SIZE       POS          ADDR

Entry: 0

Image      IMAGE     264652  00000-409CB  00000-409CB
  Region   CODE      264652  00000-409CB  00000-409CB
    CSECT  CDSGTEST  264652  00000-409CB  00000-409CB
   STMT                           FILE NAME

1     F:\Qsync\Source\Changes\SDL-hyperion\tests\CDSG.asm
2     C:\Users\Peter\SATK\srcasm\satk.mac


** NO ERRORS FOUND **
```