

SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE SDPSUP
 OBJECT MODULE PLACED IN :F1:SDPSUP.OBJ
 ASSEMBLER INVOKED BY: ASMS6.86 :F1:SDPSUP.A86 XREF PRINT(:F5:SDPSUP.LST)

```

LOC  OBJ          LINE      SOURCE
                                1      ;Stitle('IOS Double Precision Support')
                                2          name      sdpsup
                                3      ;;;;;;;;;;;;;;
                                4      ;
                                5      ;      TITLE:          sdpsup.a86
                                6      ;      Defines a set of double precision support routines.
                                7      ;
                                8      ;      DATE:           7-2-80
                                9      ;                               3-8-82
                               10      ;                               Lifted from Second Stage bootstrap for the iT
                               11      ;                               PS System                               Bootstrap Loader
                               12      ;
                               13      ;
                               14      ;      ABSTRACT:       Double precision integer (32 bit) operations for use
                               15      ;                               by the bootstrap loader. This module copied directly
                               16      ;                               from Basic I/O System idpsup.a86, but with all
                               17      ;                               procedures not used by the bootstrap loader commented
                               18      ;                               out.
                               19      ;
                               20      ;                               Double precision numbers are passed around as PL/M 32-bit poi
                               21      ;                               nters. Thus, when returned from procedures here, are returned in es:
                               22      ;                               bx. Note that this could be nasty on the 8096.
                               23      ;
                               24      ;      LANGUAGE DEPENDENCIES: The procedures defined in this module may only be
                               25      ;                               called from COMPACT procedures.
                               26      ;
                               27      +1 $include(:f1:bprop.asm)
                               28      ;/*
                               29      ; *      INTEL CORPORATION PROPRIETARY INFORMATION. THIS LISTING IS
                               30      ; *      SUPPLIED UNDER THE TERMS OF A LICENSE AGREEMENT WITH INTEL
                               31      ; *      CORPORATION AND MAY NOT BE COPIED NOR DISCLOSED EXCEPT IN
                               32      ; *      ACCORDANCE WITH THE TERMS OF THAT AGREEMENT.
                               33      ; */
                               34      ;
                               35      ;;;;;;;;;;;;;;
                               36      ;
                               37      arg_off      equ      4      ; set args for COMPACT
                               38      ;
                               39      code          segment word public 'CODE'
                               40      code          ends
                               41      ;
                               42      cgroup        group      code
                               43      ;$subtitle('Double to Single Conversion')
                               44      ;;;;;;;;;;;;;;
                               45      ;;
                               46      ;; sfd-
                               47      ;;      Return low-order part of a double precision (unsigned 32-bit)

```

0004


```
LOC  OBJ          LINE  SOURCE
      48          ;;      number.  No overflow detection occurs.
      49          ;;
      50          ;;      sfd: PROCEDURE(dp) WORD;
      51          ;;      DECLARE
      52          ;;          dp      DWORD;
      53          ;;      END sfd;
      54          ;;
      55          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      56          ;
      57          ;code      segment
      58          ;          assume  cs: cgroup
      59          ;
      60          ;sfd      proc      near
      61          ;          public  sfd
      62          ;          push   bp
      63          ;          mov    bp, sp          ; save and mark stack
      64          ;
      65          ;dp_low   equ      word ptr [bp + arg_off + 0]
      66          ;dp_high  equ      word ptr [bp + arg_off + 2]
      67          ;argbytes equ      4
      68          ;;
      69          ;          mov    ax, dp_low
      70          ;
      71          ;;          mov    sp, bp
      72          ;          pop    bp
      73          ;          ret   argbytes
      74          ;sfd      endp
      75          ;
      76          ;          purge  dp_low, dp_high
      77          ;          purge  argbytes
      78          ;
      79          ;code      ends
      80          ;          assume  cs: nothing
      81          ;; $subtitle('Single to Double Conversion')
      82          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      83          ;;
      84          ;; dfs
      85          ;;      Turn a single precision (unsigned 16-bit) number into a double
      86          ;;      precision number (unsigned 32-bit).
      87          ;;
      88          ;;      dfs: PROCEDURE(sw) WORD;
      89          ;;      DECLARE
      90          ;;          sw      WORD;
      91          ;;      END dfs;
      92          ;;
      93          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      94          ;
      95          ;code      segment
      96          ;          assume  cs: cgroup
      97          ;
      98          ;dfs      proc      near
      99          ;          public  dfs
100          ;          push   bp
101          ;          mov    bp, sp          ; save and mark stack
102          ;
```

```

LOC  OBJ          LINE   SOURCE
                                103   ;sw          equ    word ptr [bp + arg_off + 0]
                                104   ;argbytes   equ    2
                                105   ;;
                                106   ;          mov    bx, sw
                                107   ;          xor    ax, ax
                                108   ;          mov    es, ax          ; es:bx = double(sp)
                                109   ;
                                110   ;;          mov    sp, bp
                                111   ;          pop    bp
                                112   ;          ret    argbytes
                                113   ;dfs       endp
                                114   ;
                                115   ;          purge  sw
                                116   ;          purge  argbytes
                                117   ;
                                118   ;code      ends
                                119   ;          assume cs: nothing
                                120   ;$subtitle('Double Precision Compare')
                                121   ;;;;;;;;;;;;;;
                                122   ;
                                123   ; dpcmp
                                124   ;          Compare dp1 and dp2 (un-signed, double precision (32-bit) numbers).
                                125   ;          Return -1, 0, +1 depending on dp1 <, =, > dp2.
                                126   ;
                                127   ;          dpcmp: PROCEDURE(dp1, dp2) INTEGER;
                                128   ;          DECLARE
                                129   ;              dp1      DWORD,
                                130   ;              dp2      DWORD;
                                131   ;          END dpcmp;
                                132   ;
                                133   ;;;;;;;;;;;;;;
                                134   ;
-----
                                135   code          segment
                                136   ;          assume cs: cgroup
                                137   ;
0000                                138   dpcmp         proc    near
                                139   ;          public dpcmp
0000 55                                140   ;          push   bp
0001 83EC                                141   ;          mov    bp, sp          ; save and mark stack
                                142   ;
                                143   ;          dp2_low equ    word ptr [bp + arg_off + 0]
0004[]                                144   ;          dp2_high equ    word ptr [bp + arg_off + 2]
                                145   ;          dp1_low  equ    word ptr [bp + arg_off + 4]
0008[]                                146   ;          dp1_high  equ    word ptr [bp + arg_off + 6]
000A[]                                147   ;          argbytes equ    8
0008                                148   ;
                                149   ;          Assume dp1 < dp2.
                                150   ;
0003 B8FFFF                                151   ;          mov    ax, 0FFFFH
                                152   ;
                                153   ;          Check High words.
                                154   ;
                                155   ;          mov    bx, dp2_high
0006 8B5E06                                156   ;          cmp    dp1_high, bx          ; high(dp1) <, =, > high(dp2) ??
0009 395E0A                                157   ;          ja    dpc_dp1_grtr
000C 770C

```

```

LOC  OBJ          LINE    SOURCE
000E 720C          158                jb      dpc_dp1_less
                               159                ;
                               160                ; High words equal. Check low words.
                               161                ;
0010 8B5E04        162                mov     bx, dp2_low
0013 395E08        163                cmp     dp1_low, bx          ; low(dp1) <, =, > low(dp2) ??
0016 7403          164                je     dpc_dp1_eq
0018 7202          165                jb     dpc_dp1_less
                               166                ;
                               167                ; Get correct value in ax, by successive increments.
                               168                ;
001A 40            169    dpc_dp1_grtr:    inc     ax
001B 40            170    dpc_dp1_eq:    inc     ax
001C                171    dpc_dp1_less:
                               172                ;
                               173                mov     sp, bp
                               174                pop     bp
                               175                ret     argbytes
001C 5D            173                dpcmp
001D C20800        174                endp
                               175
                               176
                               177                purge  dp1_low, dp1_high
                               178                purge  dp2_low, dp2_high
                               179                purge  argbytes
                               180
-----
                               181    code                ends
                               182                assume  cs: nothing
                               183                ;$subtitle('Double Precision Compare with Single Presision')
                               184                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               185                ;
                               186                ; dscmp
                               187                ; Compare dp1 (unsigned 32-bit) and sp2 (unsigned 16-bit), converting
                               188                ; sp2 to double-precision. Return -1, 0, +1 depending on dp1 <, =, >
                               189                ; sp2.
                               190                ;
                               191                ; dscmp: PROCEDURE(dp1, sp2) INTEGER;
                               192                ; DECLARE
                               193                ; dp1      DWORD,
                               194                ; sp2      WORD;
                               195                ; END dscmp;
                               196                ;
                               197                ; This code could be folded with dpcmp, but for now it stays this way.
                               198                ;
                               199                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
200
-----
201    code                segment
202                assume  cs: cgroup
203
0020          204    dscmp            proc    near
                               205                public  dscmp
                               206                push   bp
0020 55          206                mov     bp, sp          ; save and mark stack
0021 8BEC        207
                               208
                               209                sp2      equ     word ptr [bp + arg_off + 0]
                               210                dp1_low  equ     word ptr [bp + arg_off + 2]
                               211                dp1_high  equ     word ptr [bp + arg_off + 4]
                               212                argbytes  equ     6

```

```

LOC  OBJ          LINE   SOURCE
                                213   ;
                                214   ; Assume dp1 < sp2.
                                215   ;
0023  B8FFFF      216       mov     ax, 0FFFFH
                                217   ;
                                218   ; Check High words.
                                219   ;
0026  837E0800    220       cmp     dp1_high, 0           ; high(dp1) <, =, > high(sp2) ??
002A  770C        221       ja     dsc_dp1_grtr
002C  720C        222       jb     dsc_dp1_less
                                223   ;
                                224   ; High words equal. Check low words.
                                225   ;
002E  8B5E04      226       mov     bx, sp2
0031  395E06      227       cmp     dp1_low, bx         ; low(dp1) <, =, > low(sp2) ??
0034  7403        228       je     dsc_dp1_eq
0036  7202        229       jb     dsc_dp1_less
                                230   ;
                                231   ; Get correct value in ax, by successive increments.
                                232   ;
0038  40          233   dsc_dp1_grtr:  inc     ax
0039  40          234   dsc_dp1_eq:   inc     ax
003A                                235   dsc_dp1_less:
                                236   ;
003A  5D          237       mov     sp, bp
                                238       pop     bp
                                239       ret     argbytes
003B  C20600      238       dscmp
                                239       endp
                                240
                                241       purge  dp1_low, dp1_high, sp2
                                242       purge  argbytes
                                243
-----
                                244   code      ends
                                245       assume cs: nothing
                                246   ;$subtitle('Double Precision Add')
                                247   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                248   ;
                                249   ; dpadd
                                250   ; Add dp1 and dp2 (un-signed, double precision (32-bit) numbers).
                                251   ; No detection of overflow is performed.
                                252   ;
                                253   ; dpadd: PROCEDURE(dp1, dp2) DWORD;
                                254   ; DECLARE
                                255   ;     dp1     DWORD,
                                256   ;     dp2     DWORD;
                                257   ; END dpadd;
                                258   ;
                                259   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
-----
                                260
                                261   code      segment
                                262       assume cs: cgroup
                                263
003E                                264   dpadd     proc  near
                                265       public dpadd
003E  55          266       push   bp
003F  8BEC        267       mov     bp, sp           ; save and mark stack

```



```

LOC  OBJ                LINE    SOURCE
0008[] 323    dp1_high      equ    word ptr [bp + arg_off + 4]
0006   324    argbytes     equ    6
325    ;
326    ; Add low parts, then high parts.
327    ;
0056 835E06 328          mov     bx, dp1_low
0059 035E04 329          add     bx, sp2
330
005C 8B4608 331          mov     ax, dp1_high
005F 150000 332          adc     ax, 0                ; high(sp2) = 0
333
0062 8E00   334          mov     es, ax                ; es:bx now has result
335
336    ;
0064 5D     337          mov     sp, bp
0065 C20600 338          pop     bp
339          ret     argbytes
dsadd  340          endp
341          purge  dp1_low, dp1_high, sp2
342          purge  argbytes
343
----  344    code      ends
345          assume cs: nothing
346    ;$subtitle('Double Precision Subtract')
347    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
348    ;
349    ; dpsub
350    ; Subtract dp2 from dp1 (un-signed, double precision (32-bit) numbers).
351    ; No detection of overflow is preformed.
352    ;
353    ; dpsub: PROCEDURE(dp1, dp2) DWORD;
354    ; DECLARE
355    ;     dp1     DWORD,
356    ;     dp2     DWORD;
357    ; END dpsub;
358    ;
359    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
360
----  361    code      segment
362          assume  cs: cgroup
363
0068   364    dpsub     proc    near
365          public  dpsub
0068 55     366          push   bp
0069 8BEC   367          mov     bp, sp            ; save and mark stack
368
0004[] 369    dp2_low   equ    word ptr [bp + arg_off + 0]
0006[] 370    dp2_high  equ    word ptr [bp + arg_off + 2]
0008[] 371    dp1_low   equ    word ptr [bp + arg_off + 4]
000A[] 372    dp1_high  equ    word ptr [bp + arg_off + 6]
0008   373    argbytes  equ    8
374    ;
375    ; Subtract low parts, then high parts.
376    ;
0068 835E08 377          mov     bx, dp1_low

```

```

LOC  OBJ                LINE    SOURCE
006E 2B5E04            378      sub    bx, dp2_low
                                379
0071 8B460A            380      mov    ax, dp1_high
0074 1B4606            381      sbb   ax, dp2_high
                                382
0077 8ECO              383      mov    es, ax                ; es:bx now has result
                                384
                                385      ;
0079 5D                386      mov    sp, bp
007A C20800            387      pop   bp
                                388      ret   argbytes
                                389      dpsub
                                390      endp
                                391      purge dp1_low, dp1_high
                                392      purge dp2_low, dp2_high
                                393      purge argbytes
-----
                                394      code      ends
                                395      assume  cs: nothing
007A C20800            396      ;; $subtitle('Double Precision Subtract with Single Precision Result')
                                397      ;;
                                398      ;;
                                399      ;; sdpsub
007A C20800            400      ;; Subtract dp2 from dp1 (un-signed, double precision (32-bit) numbers),
007A C20800            401      ;; returning a single precision result (the low order part).
007A C20800            402      ;; Procedure only needs subtract the low order words.
007A C20800            403      ;;
007A C20800            404      ;; sdpsub: PROCEDURE(dp1, dp2) WORD;
007A C20800            405      ;; DECLARE
007A C20800            406      ;; dp1      DWORD;
007A C20800            407      ;; dp2      DWORD;
007A C20800            408      ;; .END sdpsub;
007A C20800            409      ;;
                                410      ;;
                                411      ;
007A C20800            412      ;code      segment
007A C20800            413      ;          assume  cs: cgroup
007A C20800            414      ;
007A C20800            415      ;sdpsub      proc    near
007A C20800            416      ;          public  sdpsub
007A C20800            417      ;          push   bp
007A C20800            418      ;          mov    bp, sp          ; save and mark stack
007A C20800            419      ;
007A C20800            420      ;dp2_low      equ    word ptr [bp + arg_off + 0]
007A C20800            421      ;dp2_high      equ    word ptr [bp + arg_off + 2]
007A C20800            422      ;dp1_low      equ    word ptr [bp + arg_off + 4]
007A C20800            423      ;dp1_high      equ    word ptr [bp + arg_off + 6]
007A C20800            424      ;argbytes      equ    8
007A C20800            425      ;;
007A C20800            426      ;; Just subtract low(dp2) from low(dp1).
007A C20800            427      ;;
007A C20800            428      ;          mov    ax, dp1_low
007A C20800            429      ;          sub    ax, dp2_low
007A C20800            430      ;
007A C20800            431      ;;          mov    sp, bp
007A C20800            432      ;          pop   bp

```



```

LOC  OBJ                LINE    SOURCE
                                433      ;
                                434      ;sdpsub      ret      argbytes
                                435      ;           endp
                                436      ;           ;
                                437      ;           purge   dp1_low, dp1_high
                                438      ;           purge   dp2_low, dp2_high
                                439      ;           purge   argbytes
                                440      ;code       ends
                                441      ;           assume  cs: nothing
                                442      ;$subtitle('Double Precision Subtract with Single Precision')
                                443      ;;;;;;;;;;;;;;
                                444      ;
                                445      ; dssub
                                446      ;           Subtract sp2 (unsigned 16-bit) from dp1 (unsigned 32-bit) by first
                                447      ;           converting sp2 to double-precision.
                                448      ;
                                449      ; No detection of overflow is performed.
                                450      ;
                                451      ;           dssub: PROCEDURE(dp1, sp2) DWORD;
                                452      ;           DECLARE
                                453      ;           dp1      DWORD,
                                454      ;           sp2      WORD;
                                455      ;           END dssub;
                                456      ;
                                457      ;;;;;;;;;;;;;;
                                458      ;
-----
                                459      code      segment
                                460      ;           assume  cs: cgroup
                                461      ;
007D  007D 55            462      dssub      proc      near
                                463      ;           public  dssub
                                464      ;           push   bp
007E  8BEC            465      ;           mov     bp, sp      ; save and mark stack
                                466      ;
                                467      ;           sp2     equ    word ptr [bp + arg_off + 0]
                                468      ;           dp1_low  equ    word ptr [bp + arg_off + 2]
                                469      ;           dp1_high equ    word ptr [bp + arg_off + 4]
                                470      ;           argbytes equ    6
                                471      ;
                                472      ; Subtract low parts, then high parts.
                                473      ;
0080  8B5E06          474      ;           mov     bx, dp1_low
0083  2B5E04          475      ;           sub     bx, sp2
                                476      ;
0086  8B4608          477      ;           mov     ax, dp1_high
0089  1D0000          478      ;           sbb    ax, 0      ; high(sp2) = 0
                                479      ;
008C  8EC0            480      ;           mov     es, ax      ; es:bx now has result
                                481      ;
                                482      ;           mov     sp, bp
008E  5D              483      ;           pop     bp
                                484      ;           ret     argbytes
008F  C20600          485      ;           dssub     endp
                                486      ;
                                487      ;           purge   dp1_low, dp1_high, sp2

```

```

LOC  OBJ                LINE    SOURCE
-----
                                488                purge    argbytes
                                489
                                490                code      ends
                                491                assume   cs: nothing
                                492                ;$subtitle('Double Precision Multiply with Single Precision')
                                493                ;;;;;;;;;;;;;;
                                494                ;
                                495                ; dsmul
                                496                ;        Multiply dp1 (unsigned 32-bit), sp2 (unsigned 16-bit).
                                497                ;
                                498                ; No detection of overflow is preformed.
                                499                ;
                                500                ;        dsmul: PROCEDURE(dp1, sp2) DWORD;
                                501                ;        DECLARE
                                502                ;                dp1        DWORD,
                                503                ;                sp2        WORD;
                                504                ;        END dsmul;
                                505                ;
                                506                ;;;;;;;;;;;;;;
                                507
                                508                code      segment
                                509                assume   cs: cgroup
                                510
0092   511                dsmul   proc    near
                                512                public   dsmul
0092 55  513                push    bp
0093 8BEC 514                mov     bp, sp        ; save and mark stack
                                515
                                516                sp2      equ     word ptr [bp + arg_off + 0]
                                517                dp1_low  equ     word ptr [bp + arg_off + 2]
                                518                dp1_high equ     word ptr [bp + arg_off + 4]
                                519                argbytes equ     6
                                520                ;
                                521                ; Multiple dp1_low and sp2, getting a 32-bit result.
                                522                ;
0095 884606 523                mov     ax, dp1_low
0098 F76604 524                mul    sp2           ; dx:ax is result
009B 8BD8    525                mov    bx, ax
009D 8BCA    526                mov    cx, dx         ; cx:bx = dp1_low * sp2
                                527                ;
                                528                ; Multiply dp1_high by sp2, and add the low order result to the
                                529                ; high order result of dp1_low * sp2.
                                530                ;
                                531                ; Thus result =
                                532                ;        low(dp1_high * sp2) + high(dp1_low * sp2) : low(dp1_low * sp2)
                                533                ;
009F 884608 534                mov    ax, dp1_high
00A2 F76604 535                mul    sp2           ; dx:ax = dp1_high * sp2
                                536
00A5 03C1   537                add    ax, cx
00A7 8ECO    538                mov    es, ax         ; es:bx is it!
                                539
                                540                ;
                                541                mov    sp, bp
00A9 5D      541                pop    bp
00AA C20600 542                ret     argbytes

```



```

LOC 03J          LINE    SOURCE
                    598    ;
                    599    ; No overflow can occur, since a double precision result is returned
                    600    ; (see below proof). Divide by zero is still a problem.
                    601    ;
                    602    ; dsdiv: PROCEDURE(dp1, sp2) DWORD;
                    603    ; DECLARE
                    604    ;         dp1      DWORD,
                    605    ;         sp2      WORD;
                    606    ; END dsdiv;
                    607    ;
                    608    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    609
-----
                    610    code          segment
                    611    assume   cs: cgroup
                    612
OOBE              613    dsdiv          proc      near
                    614    public   dsdiv
OOBE 55           615    push      bp
OOBF 8BEC         616    mov       bp, sp          ; save and mark stack
                    617
                    618    sp2          equ      word ptr [bp + arg_off + 0]
                    619    dp1_low       equ      word ptr [bp + arg_off + 2]
                    620    dp1_high      equ      word ptr [bp + arg_off + 4]
                    621    argbytes     equ      6
                    622    ;
                    623    ; dp1_high = Q * sp2 + R.
                    624    ;
                    625    ; Thus, dp1 / sp2 = (2**16 * (Q * sp2 + R) + dp1_low) / sp2
                    626    ;                   = (2**16 * Q * sp2 + 2**16 * R + dp1_low) / sp2
                    627    ;                   = 2**16 * Q + (2**16 * R + dp1_low) / sp2
                    628    ;                   = Q : (R : dp1_low) / sp2
                    629    ;
                    630    ; (R : dp1_low) / sp2 "can't" overflow, since 0 <= R < sp2, by the
                    631    ; first division.
                    632    ;
                    633    ; (Is it obvious that I've never done multiple-precision division before?)
                    634    ;
OO01 8B4608       635    mov       ax, dp1_high
OO04 33D2         636    xor      dx, dx
OO06 F77604       637    div     sp2          ; dx = R, ax = Q
OO09 8EC0         638    mov     es, ax       ; es = Q
                    639
OO03 884606       640    mov     ax, dp1_low
OO0E F77604       641    div     sp2          ; (R : dp1_low) / sp2
                    642
OO01 88D8         643    mov     bx, ax       ; bx = (R : dp1_low) / sp1
                    644
                    645    ;
OO03 5D           646    mov     sp, bp
OO04 C20600       647    pop     bp
                    648    ret     argbytes
                    649    dsdiv      endp
                    650    purge   dp1_low, dp1_high, sp2
                    651    purge   argbytes
                    652

```

```

LOC  OBJ          LINE    SOURCE
-----
653      code      ends
654      assume    cs: nothing
655      ;;$subtitle('Double Precision Modulo by Single Precision')
656      ;;;;;;;;;;;;;;
657      ;;
658      ;; sdsmod
659      ;;      Divide dp1 (unsigned 32-bit) by sp2 (unsigned 16-bit) and return
660      ;;      remainder after division.
661      ;;
662      ;; Uses same algorithm as dsdiv, only returns remainder.
663      ;;
664      ;;      sdsmod: PROCEDURE(dp1, sp2) WORD;
665      ;;      DECLARE
666      ;;          dp1      DWORD,
667      ;;          sp2      WORD;
668      ;;      END sdsmod;
669      ;;
670      ;;;;;;;;;;;;;;
671      ;
672      ;code      segment
673      ;          assume cs: cgroup
674      ;
675      ;sdsmod    proc    near
676      ;          public  sdsmod
677      ;          push   bp
678      ;          mov    bp, sp          ; save and mark stack
679      ;
680      ;sp2      equ    word ptr [bp + arg_off + 0]
681      ;dp1_low  equ    word ptr [bp + arg_off + 2]
682      ;dp1_high equ    word ptr [bp + arg_off + 4]
683      ;argbytes equ    6
684      ;;
685      ;          mov    ax, dp1_high
686      ;          xor    dx, dx
687      ;          div   sp2              ; dx = R, ax = Q
688      ;
689      ;          mov    ax, dp1_low
690      ;          div   sp2              ; (R : dp1_low) / sp2
691      ;
692      ;          mov    ax, dx          ; get final remainder.
693      ;
694      ;;
695      ;          mov    sp, bp
696      ;          pop   bp
697      ;          ret   argbytes
698      ;sdsmod    endp
699      ;
700      ;          purge  dp1_low, dp1_high, sp2
701      ;          purge  argbytes
702      ;
703      ;code      ends
704      ;          assume cs: nothing
705      ;$subtitle('Double Divide by Single with Single Result')
706      ;;;;;;;;;;;;;;
707      ;          sdsdiv

```

```

LOC  OBJ                LINE    SOURCE
                                708      ;      Divide dp1 (unsigned 32-bit) by sp2 (unsigned 16-bit) and return
                                709      ;      single precision result.
                                710      ;
                                711      ;      Overflow can occur, and is detected by the hardware (int 0).
                                712      ;
                                713      ;      sdsdiv: PROCEDURE(dp1, sp2) WORD;
                                714      ;      DECLARE
                                715      ;          dp1      DWORD,
                                716      ;          sp2      WORD;
                                717      ;      END sdsdiv;
                                718      ;
                                719      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                720
-----
                                721      code          segment
                                722      assume      cs: cgroup
                                723
00D7      724      sdsdiv      proc      near
                                725      public      sdsdiv
00D7 55    726      push      bp
00D8 8BEC   727      mov       bp, sp          ; save and mark stack
                                728
                                729      sp2      equ      word ptr [bp + arg_off + 0]
                                730      dp1_low  equ      word ptr [bp + arg_off + 2]
                                731      dp1_high equ      word ptr [bp + arg_off + 4]
                                732      argbytes equ      6
                                733      ;
00DA 8B4606 734      mov     ax, dp1_low
00DD 8B5608 735      mov     dx, dp1_high
00E0 F77604 736      div     sp2          ; ax = Q, dx = R
                                737
                                738      ;
00E3 5D     739      mov     sp, bp
00E4 C20600 740      pop     bp
                                741      ret     argbytes
                                742      sdsdiv      endp
                                743      purge    dp1_low, dp1_high, sp2
                                744      purge    argbytes
                                745
-----
                                746      code          ends
                                747      assume      cs: nothing
                                748
                                749      end

```

XREF SYMBOL TABLE LISTING

```

-----
NAME          TYPE      VALUE  ATTRIBUTES, XREFS
??SEG . . . . SEGMENT      SIZE=0000H PARA PUBLIC
ARG_OFF . . . . NUMBER    0004H   37# 143 144 145 146 209 210 211 269 270 271 272 321 322 323 369 370 371 372 467 468 469
                                     516 517 518 572 573 618 619 620 729 730 731
ARGBYTES. . . . --PURGED-- 147# 174 179P 212# 238 242P 273# 287 292P 324# 338 342P 373# 387 392P 470# 484 488P 519#
                                     542 546P 574# 584 588P 621# 647 651P 732# 740 744P
CGROUP. . . . . GROUP      CODE   42# 136 202 262 314 362 460 509 565 611 722
CODE. . . . . SEGMENT      SIZE=00E7H WORD PUBLIC 'CODE' 39# 40 42 135 181 201 244 261 294 313 344 361 394 459 490
                                     508 548 564 590 610 653 721 746
DP1_HIGH. . . . --PURGED-- 146# 156 177P 211# 220 241P 272# 280 290P 323# 331 341P 372# 380 390P 469# 477 487P 518#
                                     534 545P 620# 635 650P 731# 735 743P
DP1_LOW . . . . --PURGED-- 145# 163 177P 210# 227 241P 271# 277 290P 322# 328 341P 371# 377 390P 468# 474 487P 517#
                                     523 545P 619# 640 650P 730# 734 743P
DP2_HIGH. . . . --PURGED-- 144# 155 178P 270# 281 291P 370# 381 391P
DP2_LOW . . . . --PURGED-- 143# 162 178P 269# 278 291P 369# 378 391P
DPADD . . . . . L NEAR    003EH   CODE PUBLIC 264# 265 288
DPC_DP1_EQ. . . . L NEAR    001BH   CODE 164 170#
DPC_DP1_GRTR. L NEAR    001AH   CODE 157 169#
DPC_DP1_LESS. L NEAR    001CH   CODE 158 165 171#
DPCMP . . . . . L NEAR    0000H   CODE PUBLIC 138# 139 175
DPSUB . . . . . L NEAR    0068H   CODE PUBLIC 364# 365 388
DSADD . . . . . L NEAR    0053H   CODE PUBLIC 316# 317 339
DSC_DP1_EQ. . . . L NEAR    0039H   CODE 228 234#
DSC_DP1_GRTR. L NEAR    0036H   CODE 221 233#
DSC_DP1_LESS. L NEAR    003AH   CODE 222 229 235#
DSCMP . . . . . L NEAR    0020H   CODE PUBLIC 204# 205 239
DSDIV . . . . . L NEAR    003EH   CODE PUBLIC 613# 614 648
DSMUL . . . . . L NEAR    0092H   CODE PUBLIC 511# 512 543
DSSMUL. . . . . L NEAR    00ADH   CODE PUBLIC 567# 568 585
DSSUB . . . . . L NEAR    007DH   CODE PUBLIC 462# 463 485
SDSDIV. . . . . L NEAR    00D7H   CODE PUBLIC 724# 725 741
SP1 . . . . . --PURGED-- 573# 576 587P
SP2 . . . . . --PURGED-- 209# 226 241P 321# 329 341P 467# 475 487P 516# 524 535 545P 572# 577 587P 618# 637 641
                                     650P 729# 736 743P

```

END OF SYMBOL TABLE LISTING

ASSEMBLY COMPLETE, NO ERRORS FOUND