SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE SDPSUP
OBJECT MODULE PLACED IN :F1:SDPSUP.OBJ
ASSEMBLER INVOKED BY: ASM86.86 :F1:SDPSUP.A86 XREF PRINT(:F5:SDPSUP.LST)

```
LOC  OBJ                  SOURCE

                          ;$title('IOS Double Precision Support')
                                   name    sdpsup
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;
                          ;        TITLE:              sdpsup.a86
                          ;                            Defines a set of double precision support routines.
                          ;
                          ;        DATE:               7-2-80
                          ;                                      3-8-82
                          ;                                                Lifted from Second Stage bootstrap for the iTPS System
                          ;                                                Bootstrap Loader
                          ;
                          ;
                          ;        ABSTRACT:           Double precision integer (32 bit) operations for use
                          ;                            by the bootstrap loader.  This module copied directly
                          ;                            from Basic I/O System idpsup.a86, but with all
                          ;                            procedures not used by the bootstrap loader commented
                          ;                            out.
                          ;
                          ;                            Double precision numbers are passed around as PL/M 32-bit pointers.
                          ;                            Thus, when returned from procedures here, are returned in es:bx.
                          ;                            Note that this could be nasty on the 8096.
                          ;
                          ;        LANGUAGE DEPENDENCIES: The procedures defined in this module may only be
                          ;                            called from COMPACT procedures.
                          ;
                     +1   $include(:f1:bprop.asm)
               =1         ;/*
               =1         ; *      INTEL CORPORATION PROPRIETARY INFORMATION.  THIS LISTING IS
               =1         ; *      SUPPLIED UNDER THE TERMS OF A LICENSE AGREEMENT WITH INTEL
               =1         ; *      CORPORATION AND MAY NOT BE COPIED NOR DISCLOSED EXCEPT IN
               =1         ; *      ACCORDANCE WITH THE TERMS OF THAT AGREEMENT.
               =1         ; */
                          ;
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 0004                     arg_off          equ     4                ; set args for COMPACT

----                      code             segment word public 'CODE'
----                      code             ends

                          cgroup           group   code
                          ;;$subtitle('Double to Single Conversion')
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;;
                          ;; sfd
                          ;;       Return low-order part of a double precision (unsigned 32-bit)
```

```
LOC  OBJ              LINE     SOURCE

                               ;;      number.  No overflow detection occurs.
                               ;;
                               ;;      sfd: PROCEDURE(dp) WORD;
                               ;;          DECLARE
                               ;;              dp      DWORD;
                               ;;      END sfd;
                               ;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;
                               ;code           segment
                               ;               assume  cs: cgroup
                               ;
                               ;sfd            proc    near
                               ;               public  sfd
                               ;               push    bp
                               ;               mov     bp, sp          ; save and mark stack
                               ;
                               ;dp_low         equ     word ptr [bp + arg_off + 0]
                               ;dp_high                equ     word ptr [bp + arg_off + 2]
                               ;argbytes       equ     4
                               ;;
                               ;               mov     ax, dp_low
                               ;
                               ;;              mov     sp, bp
                               ;               pop     bp
                               ;               ret     argbytes
                               ;sfd            endp
                               ;
                               ;               purge   dp_low, dp_high
                               ;               purge   argbytes
                               ;
                               ;code           ends
                               ;               assume  cs: nothing
                               ;;$subtitle('Single to Double Conversion')
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;
                               ;; dfs
                               ;;      Turn a single precision (unsigned 16-bit) number into a double
                               ;;      precision number (unsigned 32-bit).
                               ;;
                               ;;      dfs: PROCEDURE(sw) WORD;
                               ;;          DECLARE
                               ;;              sw      WORD;
                               ;;      END dfs;
                               ;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;
                               ;code           segment
                               ;               assume  cs: cgroup
                               ;
                               ;dfs            proc    near
                               ;               public  dfs
                               ;               push    bp
                               ;               mov     bp, sp          ; save and mark stack
                               ;
```

```
LOC  OBJ            LINE     SOURCE

                             ;sw             equ     word ptr [bp + arg_off + 0]
                             ;argbytes       equ     2
                             ;;
                             ;               mov     bx, sw
                             ;               xor     ax, ax
                             ;               mov     es, ax              ; es:bx = double(sp)
                             ;
                             ;;              mov     sp, bp
                             ;               pop     bp
                             ;               ret     argbytes
                             ;dfs            endp
                             ;
                             ;               purge   sw
                             ;               purge   argbytes
                             ;
                             ;code           ends
                             ;               assume  cs: nothing
                             ;$subtitle('Double Precision Compare')
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;
                             ; dpcmp
                             ;       Compare dp1 and dp2 (un-signed, double precision (32-bit) numbers).
                             ;       Return -1, 0, +1 depending on dp1 <, =, > dp2.
                             ;
                             ;       dpcmp: PROCEDURE(dp1, dp2) INTEGER;
                             ;           DECLARE
                             ;               dp1     DWORD,
                             ;               dp2     DWORD;
                             ;       END dpcmp;
                             ;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----                         code            segment
                                             assume  cs: cgroup

0000                         dpcmp           proc    near
                                             public  dpcmp
0000 55                                      push    bp
0001 8BEC                                    mov     bp, sp          ; save and mark stack

  0004[]                     dp2_low         equ     word ptr [bp + arg_off + 0]
  0006[]                     dp2_high        equ     word ptr [bp + arg_off + 2]
  0008[]                     dp1_low         equ     word ptr [bp + arg_off + 4]
  000A[]                     dp1_high        equ     word ptr [bp + arg_off + 6]
  0008                       argbytes        equ     8
                             ;
                             ; Assume dp1 < dp2.
                             ;
0003 B8FFFF                                  mov     ax, 0FFFFH
                             ;
                             ; Check High words.
                             ;
0006 8B5E06                                  mov     bx, dp2_high
0009 395E0A                                  cmp     dp1_high, bx        ; high(dp1) <, =, > high(dp2) ??
000C 770C                                    ja      dpc_dp1_grtr
```

```
LOC  OBJ        SOURCE

000E 720C                           jb      dpc_dp1_less
                ;
                ; High words equal. Check low words.
                ;
0010 8B5E04                         mov     bx, dp2_low
0013 395E08                         cmp     dp1_low, bx             ; low(dp1) <, =, > low(dp2) ??
0016 7403                           je      dpc_dp1_eq
0018 7202                           jb      dpc_dp1_less
                ;
                ; Get correct value in ax, by successive increments.
                ;
001A 40         dpc_dp1_grtr:       inc     ax
001B 40         dpc_dp1_eq:         inc     ax
001C            dpc_dp1_less:
                ;                   mov     sp, bp
001C 5D                             pop     bp
001D C20800                         ret     argbytes
                dpcmp               endp

                                    purge   dp1_low, dp1_high
                                    purge   dp2_low, dp2_high
                                    purge   argbytes

----            code                ends
                                    assume  cs: nothing
                ;$subtitle('Double Precision Compare with Single Presision')
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;
                ; dscmp
                ;       Compare dp1 (unsigned 32-bit) and sp2 (unsigned 16-bit), converting
                ;       sp2 to double-precision.  Return -1, 0, +1 depending on dp1 <, =, >
                ;       sp2.
                ;
                ;       dscmp: PROCEDURE(dp1, sp2) INTEGER;
                ;           DECLARE
                ;               dp1     DWORD,
                ;               sp2     WORD;
                ;       END dscmp;
                ;
                ; This code could be folded with dpcmp, but for mow it stays this way.
                ;
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----            code                segment
                                    assume  cs: cgroup

0020            dscmp               proc    near
                                    public  dscmp
0020 55                             push    bp
0021 8BEC                           mov     bp, sp          ; save and mark stack

  0004[]        sp2                 equ     word ptr [bp + arg_off + 0]
  0006[]        dp1_low             equ     word ptr [bp + arg_off + 2]
  0008[]        dp1_high            equ     word ptr [bp + arg_off + 4]
  0006          argbytes            equ     6
```

```
LOC  OBJ              SOURCE

                      ;
                      ; Assume dp1 < sp2.
                      ;
0023 B8FFFF                           mov     ax, 0FFFFH
                      ;
                      ; Check High words.
                      ;
0026 837E0800                         cmp     dp1_high, 0              ; high(dp1) <, =, > high(sp2) ??
002A 770C                             ja      dsc_dp1_grtr
002C 720C                             jb      dsc_dp1_less
                      ;
                      ; High words equal. Check low words.
                      ;
002E 8B5E04                           mov     bx, sp2
0031 395E06                           cmp     dp1_low, bx              ; low(dp1) <, =, > low(sp2) ??
0034 7403                             je      dsc_dp1_eq
0036 7202                             jb      dsc_dp1_less
                      ;
                      ; Get correct value in ax, by successive increments.
                      ;
0038 40               dsc_dp1_grtr:   inc     ax
0039 40               dsc_dp1_eq:     inc     ax
003A                  dsc_dp1_less:
                      ;               mov     sp, bp
003A 5D                               pop     bp
003B C20600                           ret     argbytes
                      dscmp           endp

                                      purge   dp1_low, dp1_high, sp2
                                      purge   argbytes

----                  code            ends
                                      assume  cs: nothing
                      ;$subtitle('Double Precision Add')
                      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                      ;
                      ; dpadd
                      ;       Add dp1 and dp2 (un-signed, double precision (32-bit) numbers).
                      ;       No dectection of overflow is preformed.
                      ;
                      ;       dpadd: PROCEDURE(dp1, dp2) DWORD;
                      ;           DECLARE
                      ;               dp1     DWORD,
                      ;               dp2     DWORD;
                      ;       END dpadd;
                      ;
                      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----                  code            segment
                                      assume  cs: cgroup

003E                  dpadd           proc    near
                                      public  dpadd
003E 55                               push    bp
003F 8BEC                             mov     bp, sp          ; save and mark stack
```

```
LOC  OBJ          SOURCE

  0004[]          dp2_low         equ     word ptr [bp + arg_off + 0]
  0006[]          dp2_high        equ     word ptr [bp + arg_off + 2]
  0008[]          dp1_low         equ     word ptr [bp + arg_off + 4]
  000A[]          dp1_high        equ     word ptr [bp + arg_off + 6]
  0008            argbytes        equ     8
                  ;
                  ; Add low parts, then high parts.
                  ;
0041 8B5E08                       mov     bx, dp1_low
0044 035E04                       add     bx, dp2_low

0047 8B460A                       mov     ax, dp1_high
004A 134606                       adc     ax, dp2_high

004D 8EC0                         mov     es, ax                  ; es:bx now has result

                  ;               mov     sp, bp
004F 5D                           pop     bp
0050 C20800                       ret     argbytes
                  dpadd           endp

                                  purge   dp1_low, dp1_high
                                  purge   dp2_low, dp2_high
                                  purge   argbytes

----              code            ends
                                  assume  cs: nothing
                  ;$subtitle('Double Precision Add with Single Precision')
                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                  ;
                  ; dsadd
                  ;       Add dp1 (unsigned 32-bit), sp2 (unsigned 16-bit) by first
                  ;       converting sp2 to double-precision.
                  ;
                  ; No dectection of overflow is preformed.
                  ;
                  ;       dsadd: PROCEDURE(dp1, sp2) DWORD;
                  ;           DECLARE
                  ;               dp1     DWORD,
                  ;               sp2     WORD;
                  ;       END dsadd;
                  ;
                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----              code            segment
                                  assume  cs: cgroup

0053              dsadd           proc    near
                                  public  dsadd
0053 55                           push    bp
0054 8BEC                         mov     bp, sp          ; save and mark stack

  0004[]          sp2             equ     word ptr [bp + arg_off + 0]
  0006[]          dp1_low         equ     word ptr [bp + arg_off + 2]
```

```
LOC  OBJ             SOURCE

  0008[]             dp1_high        equ     word ptr [bp + arg_off + 4]
  0006               argbytes        equ     6
                     ;
                     ; Add low parts, then high parts.
                     ;
0056 8B5E06                          mov     bx, dp1_low
0059 035E04                          add     bx, sp2

005C 8B4608                          mov     ax, dp1_high
005F 150000                          adc     ax, 0                   ; high(sp2) = 0

0062 8EC0                            mov     es, ax                  ; es:bx now has result

                     ;               mov     sp, bp
0064 5D                              pop     bp
0065 C20600                          ret     argbytes
                     dsadd           endp

                                     purge   dp1_low, dp1_high, sp2
                                     purge   argbytes

----                 code            ends
                                     assume  cs: nothing
                     ;$subtitle('Double Precision Subtract')
                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                     ;
                     ; dpsub
                     ;       Subtract dp2 from dp1 (un-signed, double precision (32-bit) numbers).
                     ;       No dectection of overflow is preformed.
                     ;
                     ;       dpsub: PROCEDURE(dp1, dp2) DWORD;
                     ;           DECLARE
                     ;               dp1     DWORD,
                     ;               dp2     DWORD;
                     ;       END dpsub;
                     ;
                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----                 code            segment
                                     assume  cs: cgroup

0068                 dpsub           proc    near
                                     public  dpsub
0068 55                              push    bp
0069 8BEC                            mov     bp, sp          ; save and mark stack

  0004[]             dp2_low         equ     word ptr [bp + arg_off + 0]
  0006[]             dp2_high        equ     word ptr [bp + arg_off + 2]
  0008[]             dp1_low         equ     word ptr [bp + arg_off + 4]
  000A[]             dp1_high        equ     word ptr [bp + arg_off + 6]
  0008               argbytes        equ     8
                     ;
                     ; Subtract low parts, then high parts.
                     ;
006B 8B5E08                          mov     bx, dp1_low
```

```
LOC  OBJ           SOURCE

006E 2B5E04                          sub     bx, dp2_low

0071 8B460A                          mov     ax, dp1_high
0074 1B4606                          sbb     ax, dp2_high

0077 8EC0                            mov     es, ax                 ; es:bx now has result

                     ;               mov     sp, bp
0079 5D                              pop     bp
007A C20800                          ret     argbytes
                     dpsub           endp

                                     purge   dp1_low, dp1_high
                                     purge   dp2_low, dp2_high
                                     purge   argbytes

----                 code            ends
                                     assume  cs: nothing
                     ;;$subtitle('Double Precision Subtract with Single Presicion Result')
                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                     ;;
                     ;; sdpsub
                     ;;      Subtract dp2 from dp1 (un-signed, double precision (32-bit) numbers),
                     ;;      returning a single precision result (the low order part).
                     ;; Procedure only needs subtract the low order words.
                     ;;
                     ;;      sdpsub: PROCDEURE(dp1, dp2) WORD;
                     ;;          DECLARE
                     ;;              dp1     DWORD,
                     ;;              dp2     DWORD;
                     ;;      END sdpsub;
                     ;;
                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                     ;
                     ;code           segment
                     ;               assume  cs: cgroup
                     ;
                     ;sdpsub         proc    near
                     ;               public  sdpsub
                     ;               push    bp
                     ;               mov     bp, sp          ; save and mark stack
                     ;
                     ;dp2_low                equ     word ptr [bp + arg_off + 0]
                     ;dp2_high       equ     word ptr [bp + arg_off + 2]
                     ;dp1_low                equ     word ptr [bp + arg_off + 4]
                     ;dp1_high       equ     word ptr [bp + arg_off + 6]
                     ;argbytes       equ     8
                     ;;
                     ;; Just subtract low(dp2) from low(dp1).
                     ;;
                     ;               mov     ax, dp1_low
                     ;               sub     ax, dp2_low
                     ;
                     ;;              mov     sp, bp
                     ;               pop     bp
```

```
LOC  OBJ            SOURCE

                    ;                ret      argbytes
                    ;sdpsub          endp
                    ;
                    ;                purge    dp1_low, dp1_high
                    ;                purge    dp2_low, dp2_high
                    ;                purge    argbytes
                    ;
                    ;code            ends
                    ;                assume   cs: nothing
                    ;$subtitle('Double Precision Subtract with Single Precision')
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;
                    ; dssub
                    ;        Subtract sp2 (unsigned 16-bit) from dp1 (unsigned 32-bit) by first
                    ;        converting sp2 to double-precision.
                    ;
                    ; No dectection of overflow is preformed.
                    ;
                    ;        dssub: PROCEDURE(dp1, sp2) DWORD;
                    ;            DECLARE
                    ;                dp1      DWORD,
                    ;                sp2      WORD;
                    ;        END dssub;
                    ;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----                code             segment
                                     assume   cs: cgroup

007D                dssub            proc     near
                                     public   dssub
007D 55                              push     bp
007E 8BEC                            mov      bp, sp            ; save and mark stack

  0004[]            sp2              equ      word ptr [bp + arg_off + 0]
  0006[]            dp1_low          equ      word ptr [bp + arg_off + 2]
  0008[]            dp1_high         equ      word ptr [bp + arg_off + 4]
  0006              argbytes         equ      6
                    ;
                    ; Subtract low parts, then high parts.
                    ;
0080 8B5E06                          mov      bx, dp1_low
0083 2B5E04                          sub      bx, sp2

0086 8B4608                          mov      ax, dp1_high
0089 1D0000                          sbb      ax, 0                    ; high(sp2) = 0

008C 8EC0                            mov      es, ax                   ; es:bx now has result

                    ;                mov      sp, bp
008E 5D                              pop      bp
008F C20600                          ret      argbytes
                    dssub            endp

                                     purge    dp1_low, dp1_high, sp2
```

```
LOC  OBJ           SOURCE

                                       purge    argbytes

----               code                ends
                                       assume   cs: nothing
                   ;$subtitle('Double Precision Multiply with Single Precision')
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                   ;
                   ; dsmul
                   ;       Multiply dp1 (unsigned 32-bit), sp2 (unsigned 16-bit).
                   ;
                   ; No dectection of overflow is preformed.
                   ;
                   ;       dsmul: PROCEDURE(dp1, sp2) DWORD;
                   ;           DECLARE
                   ;               dp1      DWORD,
                   ;               sp2      WORD;
                   ;       END dsmul;
                   ;
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----               code                segment
                                       assume   cs: cgroup

0092               dsmul               proc     near
                                       public   dsmul
0092 55                                push     bp
0093 8BEC                              mov      bp, sp           ; save and mark stack

  0004[]           sp2                 equ      word ptr [bp + arg_off + 0]
  0006[]           dp1_low             equ      word ptr [bp + arg_off + 2]
  0008[]           dp1_high            equ      word ptr [bp + arg_off + 4]
  0006             argbytes            equ      6
                   ;
                   ; Multiple dp1_low and sp2, getting a 32-bit result.
                   ;
0095 8B4606                            mov      ax, dp1_low
0098 F76604                            mul      sp2                        ; dx:ax is result
009B 8BD8                              mov      bx, ax
009D 8BCA                              mov      cx, dx                     ; cx:bx = dp1_low * sp2
                   ;
                   ; Multiply dp1_high by sp2, and add the low order result to the
                   ; high order result of dp1_low * sp2.
                   ;
                   ; Thus result =
                   ;       low(dp1_high * sp2) + high(dp1_low * sp2) : low(dp1_low * sp2)
                   ;
009F 8B4608                            mov      ax, dp1_high
00A2 F76604                            mul      sp2                        ; dx:ax = dp1_high * sp2

00A5 03C1                              add      ax, cx
00A7 8EC0                              mov      es, ax                     ; es:bx is it!

                   ;                   mov      sp, bp
00A9 5D                                pop      bp
00AA C20600                            ret      argbytes
```

```
LOC  OBJ            SOURCE

                    dsmul           endp

                                    purge   dp1_low, dp1_high, sp2
                                    purge   argbytes

----                code            ends
                                    assume  cs: nothing
                    ;$subtitle('Single Precision Multiply with Double Precision Result')
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;
                    ; dssmul
                    ;       Multiply sp1 (unsigned 16-bit), sp2 (unsigned 16-bit).
                    ;
                    ;       dssmul: PROCEDURE(sp1, sp2) DWORD;
                    ;           DECLARE
                    ;               sp1     WORD,
                    ;               sp2     WORD;
                    ;       END dssmul;
                    ;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----                code            segment
                                    assume  cs: cgroup

00AD                dssmul          proc    near
                                    public  dssmul
00AD 55                             push    bp
00AE 8BEC                           mov     bp, sp          ; save and mark stack

  0004[]            sp2             equ     word ptr [bp + arg_off + 0]
  0006[]            sp1             equ     word ptr [bp + arg_off + 2]
  0004              argbytes        equ     4
                    ;
00B0 8B4606                         mov     ax, sp1
00B3 F76604                         mul     sp2             ; dx:ax = sp1 * sp2

00B6 8EC2                           mov     es, dx          ; es = high(sp1 * sp2)
00B8 8BD8                           mov     bx, ax          ; bx = low(sp1 * sp2)

                    ;               mov     sp, bp
00BA 5D                             pop     bp
00BB C20400                         ret     argbytes
                    dssmul          endp

                                    purge   sp1, sp2
                                    purge   argbytes

----                code            ends
                                    assume  cs: nothing
                    ;$subtitle('Double Precision Divide by Single Precision')
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;
                    ; dsdiv
                    ;       Divide dp1 (unsigned 32-bit) by sp2 (unsigned 16-bit) and return
                    ;       double precision result.
```

```
LOC  OBJ            SOURCE

                    ;
                    ; No overflow can occur, since a double precision result is returned
                    ; (see below proof).  Divide by zero is still a problem.
                    ;
                    ;       dsdiv: PROCEDURE(dp1, sp2) DWORD;
                    ;           DECLARE
                    ;               dp1     DWORD,
                    ;               sp2     WORD;
                    ;       END dsdiv;
                    ;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----                code            segment
                                    assume  cs: cgroup

00BE                dsdiv           proc    near
                                    public  dsdiv
00BE 55                             push    bp
00BF 8BEC                           mov     bp, sp          ; save and mark stack

  0004[]            sp2             equ     word ptr [bp + arg_off + 0]
  0006[]            dp1_low         equ     word ptr [bp + arg_off + 2]
  0008[]            dp1_high        equ     word ptr [bp + arg_off + 4]
  0006              argbytes        equ     6
                    ;
                    ; dp1_high = Q * sp2 + R.
                    ;
                    ; Thus, dp1 / sp2 = (2**16 * (Q * sp2 + R) + dp1_low) / sp2
                    ;                 = (2**16 * Q * sp2 + 2**16 * R + dp1_low) / sp2
                    ;                 = 2**16 * Q + (2**16 * R + dp1_low) / sp2
                    ;                 = Q : (R : dp1_low) / sp2
                    ;
                    ; (R : dp1_low) / sp2 "can't" overflow, since 0 <= R < sp2, by the
                    ; first division.
                    ;
                    ; (Is it obvious that I've never done multiple-precision division before?)
                    ;
00C1 8B4608                         mov     ax, dp1_high
00C4 33D2                           xor     dx, dx
00C6 F77604                         div     sp2                     ; dx = R, ax = Q
00C9 8EC0                           mov     es, ax                  ; es = Q

00CB 8B4606                         mov     ax, dp1_low
00CE F77604                         div     sp2                     ; (R : dp1_low) / sp2

00D1 8BD8                           mov     bx, ax                  ; bx = (R : dp1_low) / sp1

                    ;               mov     sp, bp
00D3 5D                             pop     bp
00D4 C20600                         ret     argbytes
                    dsdiv           endp

                                    purge   dp1_low, dp1_high, sp2
                                    purge   argbytes

```

```
LOC  OBJ                  SOURCE

----                      code            ends
                                          assume  cs: nothing
                          ;;$subtitle('Double Precision Modulo by Single Precision')
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;;
                          ;; sdsmod
                          ;;      Divide dp1 (unsigned 32-bit) by sp2 (unsighned 16-bit) and return
                          ;;      remainder after division.
                          ;;
                          ;; Uses same algorithm as dsdiv, only returns remainder.
                          ;;
                          ;;      sdsmod: PROCEDURE(dp1, sp2) WORD;
                          ;;          DECLARE
                          ;;              dp1     DWORD,
                          ;;              sp2     WORD;
                          ;;      END sdsmod;
                          ;;
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;
                          ;code           segment
                          ;               assume  cs: cgroup
                          ;
                          ;sdsmod         proc    near
                          ;               public  sdsmod
                          ;               push    bp
                          ;               mov     bp, sp          ; save and mark stack
                          ;
                          ;sp2            equ     word ptr [bp + arg_off + 0]
                          ;dp1_low                equ     word ptr [bp + arg_off + 2]
                          ;dp1_high       equ     word ptr [bp + arg_off + 4]
                          ;argbytes       equ     6
                          ;;
                          ;               mov     ax, dp1_high
                          ;               xor     dx, dx
                          ;               div     sp2                     ; dx = R, ax = Q
                          ;
                          ;               mov     ax, dp1_low
                          ;               div     sp2                     ; (R : dp1_low) / sp2
                          ;
                          ;               mov     ax, dx                  ; get final remainder.
                          ;
                          ;;              mov     sp, bp
                          ;               pop     bp
                          ;               ret     argbytes
                          ;sdsmod         endp
                          ;
                          ;               purge   dp1_low, dp1_high, sp2
                          ;               purge   argbytes
                          ;
                          ;code           ends
                          ;               assume  cs: nothing
                          ;$subtitle('Double Divide by Single with Single Result')
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;
                          ; sdsdiv
```

```
LOC  OBJ          SOURCE

                  ;       Divide dp1 (unsigned 32-bit) by sp2 (unsigned 16-bit) and return
                  ;       single precision result.
                  ;
                  ; Overflow can occur, and is detected by the hardware (int 0).
                  ;
                  ;       sdsdiv: PROCEDURE(dp1, sp2) WORD;
                  ;           DECLARE
                  ;               dp1     DWORD,
                  ;               sp2     WORD;
                  ;       END sdsdiv;
                  ;
                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

----              code            segment
                                  assume  cs: cgroup

0007              sdsdiv          proc    near
                                  public  sdsdiv
0007 55                           push    bp
0008 8BEC                         mov     bp, sp          ; save and mark stack

  0004[]          sp2             equ     word ptr [bp + arg_off + 0]
  0006[]          dp1_low         equ     word ptr [bp + arg_off + 2]
  0008[]          dp1_high        equ     word ptr [bp + arg_off + 4]
  0006            argbytes        equ     6
                  ;
000A 8B4606                       mov     ax, dp1_low
000D 8B5608                       mov     dx, dp1_high
00E0 F77604                       div     sp2             ; ax = Q, dx = R

                  ;               mov     sp, bp
00E3 5D                           pop     bp
00E4 C20600                       ret     argbytes
                  sdsdiv          endp

                                  purge   dp1_low, dp1_high, sp2
                                  purge   argbytes

----              code            ends
                                  assume  cs: nothing

                                  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   0038H   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   00BEH   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
```