

```
;*****;
;*          F F . A S M      ( FileFind )          *;
;*-----*
;* Task      : Searches the specified drive for files. *;
;*-----*
;* Author    : Michael Tischer                      *;
;* Developed on : 09/12/90                          *;
;* Last update : 01/20/92                          *;
;*-----*
;* Assembly  : MASM FF;                             *;
;*           : LINK FF;                             *;
;*           : EXE2BIN FF FF.COM                     or *;
;*           :                                         *;
;*           : TASM FF                               *;
;*           : TLINK FF /T                           *;
;*-----*
;* Call      : FF [dr:]filename [+ | - | =date]      *;
;*****;
```

```
== Constants =====
```

```
CMD_OFS      equ      81h          ;Start system prompt in PSP
CR           equ      13          ;Carriage return
LF           equ      10          ;Linefeed
DELIMITER    equ      "-"        ;Date component delimiter
```

```
;-- DOS functions -----
```

```
GET_DR_NO    equ      19h          ;Get current drive number
WRT_CHAR     equ      02h          ;Display character on STDOUT
SEARCH_FIRST equ      4Eh          ;Search first file
SEARCH_NEXT  equ      4Fh          ;Search next file
SET_DTA      equ      1Ah          ;Set new DTA range
PRINT_STR    equ      09h          ;Display string ended with $
```

```
ATTR_DIR     equ      10h          ;Search subdirectory
ATTR_NRM     equ      00h          ;Search for normal files
```

```
DATCOMP_NO   equ      0           ;No date comparison
DATCOMP_LS   equ      1           ;Files before date
DATCOMP_EQ   equ      2           ;Files on date
DATCOMP_GR   equ      3           ;Files after date
```

```
== Structures =====
```

```
PSP          struc              ;Base PSP structure on
    intcall   dw (?)            ;interrupt call 20H
    endadd    dw (?)            ;End address
    db (?)    ;Reserved
    farcall   db 5 dup (?)      ;FAR call to interrupt 21h
    int22h    dd (?)            ;Copy of interrupt 22H
    int23h    dd (?)            ;Copy of interrupt 23H
    int24h    dd (?)            ;Copy of interrupt 24H
    db 22 dup (?) ;Reserved
    envseg    dw (?)            ;Segment address of environment
    db 46 dup (?) ;Reserved
```

```
;-- First FCB starts here -----
```

```
drvcode1 db (?)            ;Drive number
fi_name1 db 8 dup (?)      ;Filename
fi_ext1 db 3 dup (?)       ;File extension
blockno1 dw (?)            ;Current block number
fi_len1 dd (?)             ;File lengths
datmod1 dw (?)             ;Date of last modification
timemod1 dw (?)            ;Time of last modification
    dw 4 dup (?) ;Reserved
currecl db (?)             ;Current record number
setfree1 dd (?)            ;Record number for free access
```

```
;-- Second FCB starts here -----
```

```
drvcode2 db (?)            ;Drive number
fi_name2 db 8 dup (?)      ;Filename
fi_ext2 db 3 dup (?)       ;File extension
blockno2 dw (?)            ;Current block number
```

```

        fi_len2 dd (?)      ;File lengths
        datmod2 dw (?)      ;Date of last modification
        timemod2 dw (?)      ;Time of last modification
                dw 4 dup (?) ;Reserved
        currec2 db (?)      ;Current record number
        setfree2 dd (?)     ;Record number for free access

                db 128 dup (?) ;Command line
PSP        ends

;-----

DTA        struc                ;DTA structure for file search
                db 21 dup (?) ;Reserved
        attr      db (?)      ;Found file's attributes
        timemod   dw (?)      ;Time of last modification
        datemod   dw (?)      ;Date of last modification
        fi_len    dd (?)      ;File length
        fi_n      db 13 dup (?) ;Filename
DTA        ends

;-----

DATES       struc                ;Get date information
        month     db (?)      ;Month (1-12)
        day       db (?)      ;Day of the month (1-31)
        year      dw (?)      ;Year, including century
DATES       ends

;== Code segment definition and start =====
code        segment para 'CODE'      ;Definition of code segment

                org 100h                ;Start after PSP

                assume cs:code, ds:code, es:code, ss:code

start:      jmp startff                ;Jump to program start

;== Data =====

datcomp     db  DATCOMP_NO            ;Default = no date search
sdate       DATES < 0, 0, 0>          ;Get date from system prompt
tdate       DATES < 0, 0, 0>          ;Get date of file for comparison

dir_name     db  64, ":\\"            ;Name of current directory
firstdir     db  79 dup (0)           ;in search
srchname     db  "*.*", 0             ;Directory to be searched
                db  9 dup (0)

wcnname      db  "*.*", 0             ;Subdirectory wildcard
found        dw  0                   ;Number of files found

mes          db  CR, LF, "FF: 0 file(s) found."
crlf         db  CR, LF, "$"

date_buff    db  "dddddd-dd-dddd ",0 ;Buffer for date conversion
datbend      equ this byte           ;into ASCII characters

errmes       db  "FF - (c) 1991, 92 by MICHAEL TISCHER", 13,10,10
                db  "Syntax: FF [dr:]Filename [+Date | -Date | =Date]",13,10
                db  "$"

;== Start of program =====

startff:     ;-- Get arguments from system prompt -----

                cld                    ;Decrement on string instructions
                mov     di,offset srchname
                mov     si,CMD_OFS      ;SI at beginning of command line

arg1:        ;-- Get next drive specifier -----

                lodsb                    ;Get character from system prompt
                cmp     al,CR            ;End of command reached?
                je      arg6            ;Yes --> End process

```

```

    cmp     al," "           ;No --> Are there spaces?
    jbe     arg1             ;No --> Tabs, continue

    ;-- Character found -----

    cmp     ds:[0].drvcode1,0 ;Is there a drive specifier?
    je      arg2             ;No --> Otherwise it's already in FCB
    lodsb                   ;Load colon
    lodsb                   ;Load next character
    cmp     al," "           ;<= SPACE?
    jbe     arg3             ;Yes --> End processing

arg2:    ;-- Place filenames in srchname buffer -----

    stosb                   ;Place characters in search name
    lodsb                   ;Load next character
    cmp     al," "           ;<= SPACE?
    ja      arg2             ;No --> Character - continue loading

arg3:    ;-- End filename processing -----

    cmp     al,CR            ;End of line found?
    je      arg6

    ;-- Check and process date expression -----

arg4:    lodsb               ;Load next character
    cmp     al," "           ;SPACE and TAB are valid
    je      arg4
    cmp     al,9
    je      arg4
    cmp     al,CR            ;End of line found?
    je      arg6

    cmp     al,"="           ;Same date?
    jne     arg4a            ;No --> Continue check
    mov     al,DATCOMP_EQ    ;Date comparison flag
    jmp     short arg5

arg4a:    cmp     al,"-"      ;Before date ...
    jne     arg4b            ;No --> Continue check
    mov     al,DATCOMP_LS    ;Date comparison flag
    jmp     short arg5

arg4b:    cmp     al,"+"      ;After date...
    jne     argerr           ;No --> Syntax error in command line
    mov     al,DATCOMP_GR    ;Date comparison flag

arg5:    mov     datcomp,al    ;Set date comparison flag
    mov     di,offset sdate ;Pointer to date structure
    call    getdat           ;Get date from command syntax
    jc      argerr           ;Syntax error --> End program

arg6:    ;-- Finish processing command, everything is O.K. -----

    mov     al,ds:[0].drvcode1 ;Get current drive number
    or      al,al            ;Does one already exist?
    jne     arg7             ;Yes --> Don't use current

    ;-- No drive specifier stated, use current -----

    mov     ah,GET_DR_NO     ;Get current drive number
    int     21h              ;(0=A)
    inc     al                ;Increment drive number (1 = A)

arg7:    add     dir_name,al   ;Convert specifier to ASCII
    mov     dx,offset startdta ;First DTA address
    mov     bx,offset firstdir ;Append directory name
    call    go                ;Begin search

    mov     ax,found          ;Load number of files found
    or      ax,ax             ;Files found?
    je      arg8              ;No --> display unchanged string
    mov     si,offset mes+6 ;Pointer to buffer
    call    toint

```

```

arg8:      mov     ah,09h          ;Display message
           mov     dx,offset mes
           int     21h

           mov     ax,4C00h        ;End program using
           int     21h             ;function 4CH

argerr:    ;-- Error occurred while reading command line -----

           mov     ah,09h          ;Display error string
           mov     dx,offset errmes
           int     21h

           mov     ax,4C01h        ;Display error code, end program
           int     21h

;-- GO: Controls search for specified files -----
;-- Input      : BX = Pointer to search string, including
;--              directory and filenames
;--              DX = Offset of previous DTA
;-- Output      : None
;-- Registers  : AX, SI, CX and FLAGS are affected

go         proc     near

           push     dx              ;Set pointer to current DTA

           mov     si,offset srchname ;Copy name of file to be
           call    strcpy           ;searched to buffer

           mov     cx,ATTR_NRM      ;Search for normal files only
           call    startsearch      ;Begin search
           jc      go2              ;None found --> Search directories

           call    printname        ;File found

gol:       ;-- Execute search for all other files -----

           call    nextfile         ;Search next file
           jc      go2              ;No more files found
           call    printname        ;File found
           jmp     short gol        ;

go2:       ;-- Search current directory again, after -----
           ;-- searching subdirectories -----

           pop      dx              ;Release pointer to DTA
           push     dx              ;Restore pointer to DTA
           mov     si,offset wcname ;Add *.* to directory name
           call    strcpy

           mov     cx,ATTR_DIR      ;Now search subdirectories
           call    startsearch      ;Search first subdirectory
           jc      go_end           ;None found --> End routine

           mov     si,dx            ;Found, but it's a directory
           test     [si].attr,ATTR_DIR
           jne     go4              ;Yes --> Process

go3:       ;-- No directory, continue search -----

           call    nextfile         ;Search next file
           jc      go_end           ;None found --> End routine
           mov     si,dx            ;Found, but it's a directory
           test     [si].attr,ATTR_DIR
           je      go3              ;Jump if zero

go4:       ;-- Subdirectory found -----

           cmp     [si].fi_n, "."   ;"." or ".."
           je      go3              ;Yes --> No processing

           push     di              ;Push DX and
           push     bx              ;DX onto stack
           mov     si,dx            ;Set SI to filenames in DTA in
           add     si,offset fi_n   ;which directory names are set

```

```

        mov     di,bx                ;Set DI to previous directory names

go5:    ;-- Add directory names to previous names -----

        lodsb                     ;Load character from directory name
        stosb                     ;and append to previous names
        or      al,al              ;Last byte found?
        jne     go5               ;No --> Continue

        mov     bx,di              ;Set new end of buffer in BX
        stosb                     ;Write another NULL byte
        mov     byte ptr [bx-1],"\\" ;Precede with a backslash
        call    go                 ;ReCursive call

        pop     bx                ;Pop BX and
        pop     di                ;DI from stack
        mov     byte ptr [bx],0    ;Append last directory names
                                   ;from previous calls

        mov     ah,SET_DTA        ;Reset DTA to old
        int     21h               ;address in DX

        jmp     short go3

go_end:  pop     dx                ;Release pointer to old DTA
        ret                     ;Return to caller

go       endp

;-- STARTSEARCH: Start search for new file -----
;-- Input      : CX = Attribute of file to be searched for
;--            : DX = Offset of previous DTA
;-- Output     : DX = New DTA address
;--            : Carry flag=0: O.K.
;--            : Carry flag=1: No file found
;-- Registers  : AX, DX, BP and FLAGS are affected
;-- Info       : The dir_name buffer specifies the search filename

startsearch proc    near

        push    cx                ;Push CX onto stack

        ;-- Set new DTA after current DTA -----

        add     dx,2ch            ;One DTA = 42 bytes
        mov     ah,SET_DTA        ;Set new DTA
        int     21h

        ;-- Search for first filename listed in dir_name buffer ----

        mov     bp,dx             ;Store DX
        mov     ah,SEARCH_FIRST    ;Search first file
        mov     dx,offset dir_name ;DS:DX = Pointer to filename
        int     21h

        mov     dx,bp             ;Reset DX
        pop     cx                ;Pop CX from stack
        ret                     ;Return to caller

startsearch        endp

;-- NEXTFILE: Search for next file -----
;-- Input      : None
;-- Output     : Carry flag=0: O.K.
;--            : Carry flag=1: No file found
;-- Registers  : AY and FLAGS are affected

nextfile  proc    near

        mov     bp,dx
        mov     ah,SEARCH_NEXT     ;Search for next file
        mov     dx,offset dir_name ;Path in dir_name buffer
        int     21h
        mov     dx,bp
        ret                     ;Return to caller

```

```

nextfile    endp

;-- STRCOPY: Copies an ASCII string ended with a NULL byte -----
;-- Input      : SI = Source string offset
;--            BX = Target string offset
;--            DS = Source string segment
;--            ES = Target string segment
;-- Output     : None
;-- Registers  : AL, DI, SI and FLAGS are affected

strcpy      proc    near

        mov     di,bx          ;Set DI to target string

scl:      ;-- Copy loop -----

        lodsb             ;Load character from source string
        stosb             ;Store in target string
        or      al,al        ;Last character found?
        jne     scl         ;No --> Continue

        ret              ;Return to caller

strcpy      endp

;-- PRINTNAME: Display found filenames -----
;-- Input      : DX = Pointer to current DTA with names
;--            BX = Target string offset
;--            DS = Source string segment
;--            ES = Target string segment
;-- Output     : None
;-- Registers  : AL, DI, SI and FLAGS are affected
;-- Info      : The filename will only be displayed if it is not
;--            handled as a subdirectory, and if the file matches the
;--            date parameter specified (if any)

printname   proc    near

        mov     bp,dx          ;Store pointer to DTA in BP
        cmp     [bp].fi_n, "." ;"." or ".."
        je      prnend        ;Yes --> No output

        ;-- Execute date comparison if requested -----

        cmp     datcomp,DATCOMP_NO ;Date check?
        je      pnl           ;No --> Display names

        mov     ax,[bp].datemod    ;Yes --> Get date and format
        mov     si,offset tdate
        call    unpackdate

        mov     di,offset sdate    ;Compare date with command
        call    cmpdat             ;parameters
        cmp     al,datcomp         ;Do files match parameter?
        jne     prnend            ;No --> Do not display

        call    printdat          ;Date O.K., display

        ;-- Display first directory names -----

pnl:      mov     dx,offset dir_name ;Pointer to directory names
        xor     al,al             ;Get last character of directory
        xchg    al,[bx]           ;and reset to 0
        mov     di,ax             ;Place character in DI
        call    printasciic       ;Display directory names

        xchg    ax,di             ;Place old value in AX
        mov     [bx],al           ;Return old character to its place

        ;-- Add filenames -----

        mov     dx,bp             ;Restore pointer to DTA
        add     dx,offset fi_n    ;Address filenames in DTA
        call    printasciic       ;and display these names

        mov     ah,PRINT_STR      ;Send linefeed (get linefeed

```

```

        mov     dx,offset crlf ;from DOS)
        int     21h

        inc     found           ;Another file found
        mov     dx,bp          ;Release old DX again

prnend:   ret                   ;Return to caller

printname endp

;-- PRINTASCIIC: Displays an ASCII string ended with a -----
;--                NULL byte, on the standard output device -----
;-- Input          : DX = Pointer to start of string
;--                DS = String segment
;-- Output         : None
;-- Registers      : AX, DX, SI and FLAGS are affected

printasciic proc    near

        mov     si,dx          ;Get string from DX
        mov     ah,WRT_CHAR    ;Function number for character output
        lodsb                 ;Load first character

pal:      ;-- Output loop -----

        mov     dl,al          ;Get DOS function character
        int     21h           ;from DL and display

        lodsb                 ;Load next character
        or      al,al          ;End of string found?
        jne     pal            ;No --> Display character

        ret                   ;Return to caller

printasciic endp

;-- TOINT: Converts a binary number into ASCII and -----
;--                places this number in the caller's buffer -----
;-- Input          : DS:SI = Address for start of buffer
;--                AX      = Binary number to be converted
;-- Output         : DS:SI = Address of first number
;-- Registers      : AX, SI and FLAGS are affected
;-- Info          : - Buffer must have a minimum of five characters free
;--                - Number will be right-justified in the buffer

toint     proc near

        push dx                ;Place changed registers
        push bx                ;on the stack

        ;-- Fill buffer with spaces -----

        mov     word ptr [si], 32 shl 8 + 32
        mov     word ptr [si+2], 32 shl 8 + 32
        mov     byte ptr [si+4], 32

        ;-- Divide number by 10, convert least significant portion --
        ;-- into ASCII and store this information in the buffer    --

        add     si,5           ;Set SI to end of buffer
        mov     bx,10          ;Always divide by 10

til:      dec     si           ;Set SI to previous character
        xor     dx,dx          ;Dividend is DX:AX
        div     bx             ;Divide DX:AX by 10
        or      dl,'0'         ;Convert DL to ASCII format
        mov     [si],dl        ;Place in buffer
        or      ax,ax          ;Is there a remainder?
        jne     til            ;Yes --> Next number

        pop     bx             ;Restore registers
        pop     dx

        ret                   ;Return to caller

toint     endp

```

```

;-- GETINT: Reads a positive decimal number from the command -----
;--          line and converts this number into binary format -----
;-- Input    : SI = Pointer to next character to be read
;--          :           in command line buffer
;-- Output   : Carry flag=0: Number O.K.
;--          :           AX = Number
;--          :           Carry flag=1: Error
;--          :           SI = Pointer following the last character read
;-- Registers : AX, CX, SI and FLAGS are affected

```

```

getint    proc near

        push  bx                ;Set changed registers
        push  dx
        push  di

        mov   di,10             ;Factor is always 10
        xor   bx,bx             ;BX gets binary number
        mov   ah,bh             ;High byte of AH is always 0

        ;-- Start processing from the beginning of the number -----

gi1:     lodsb                  ;Load next character
        cmp   al,'0'            ;Test for a number
        jnb   gi2
        cmp   al,'9'
        jbe   gi5               ;Number found --> GI5
gi2:     cmp   al,' '            ;No number --> SPACES and
        je    gi1               ;TABS allowed only
        cmp   al,9
        je    gi1

        ;-- Read number digit by digit, and convert to binary -----

gi4:     lodsb                  ;Load next character
        cmp   al,'0'            ;Is it a number?
        jnb   gi7               ;No --> Process next character
        cmp   al,'9'
        ja    gi7               ;No --> Process next character

        ;-- Yes, digit is a number -----

gi5:     xchg  ax,bx             ;Get char from BX and number from AX
        mul   di                ;Multiply AX by 10
        or    dx,dx             ;Product greater than 65536?
        jne   gierr            ;Yes --> Number too large
        and   bl,0Fh            ;Logical AND of lowest 4 bits
        add   ax,bx             ;Add to number
        xchg  ax,bx             ;Exchange AX and BX
        jmp   gi4              ;Read next character

gi6:     cld                    ;Read correct number
        mov   ax,bx             ;Move number to AX
        jmp   short giend       ;Return to caller

gi7:     dec   si                ;One character too many read,
        jmp   gi6              ;but the number is O.K.

gierr:   stc                    ;Display error

giend:   pop   di                ;Restore altered
        pop   dx                ;registers
        pop   bx
        ret                    ;Return to caller

getint    endp

```

```

;-- GETDAT: Reads date from the command line in "MM-DD-YY" format, ----
;--          converts the date into binary and places the result in ----
;--          the specified structure                                     ----
;-- Input    : SI = Pointer to next character to be
;--          :           read in command line buffer
;--          :           DI = Pointer to the data structure into which
;--          :           the converted date should be placed
;-- Output   : Carry flag=0: Number O.K.

```



```

;--          Carry flag=1: Error
;--          SI = Pointer following the last character read
;-- Registers : AX, CX, SI and FLAGS are affected

getdat      proc near

    ;-- Process month -----
    call getint          ;Get month
    jc  gderr            ;Error? ---> Abort routine

    or  ax,ax            ;Month O.K., but is it valid?
    je  gderr            ;No NULLs allowed --> Error
    cmp ax,12            ;12 is the maximum
    ja  gderr            ;AX > 12? --> Error

    mov [di].month,ax    ;Month O.K., store and continue

    lodsb                ;Hyphen delimiter should follow
    cmp al,DELIMITER
    jne gderr            ;No hyphen --> Error

    ;-- Month O.K., now process day -----
    call getint          ;Get day
    jc  gderr            ;Error? --> Abort routine

    or  ax,ax            ;Day O.K., but is it valid?
    je  gderr            ;No NULLs allowed --> Error
    cmp ax,31            ;31 is the maximum
    ja  gderr            ;AX > 31? --> Error

    mov [di].day,ax      ;Day O.K., store and continue

    lodsb                ;Hyphen delimiter should follow
    cmp al,DELIMITER
    jne gderr            ;No hyphen --> Error

    ;-- Day O.K., now process year -----
    call getint          ;Get year
    jc  gderr            ;Error? ---> Abort routine

    cmp ax,100            ;Year include 19 prefix (e.g., 1999?)
    ja  gd1              ;Yes --> Continue

    add ax,1900           ;No --> Add 1900

gd1:        cmp ax,1980    ;1980 is earliest allowable year
    jb  gderr            ;AX < 1980? --> Error

    mov [di].year,ax     ;Year O.K., so everything's O.K.
    cld
    ret

gderr:      stc           ;Date could not be processed
    ret                ;Return to caller

getdat      endp

;-- UNPACKDATE: Unpacks a date stored in DOS format, and places -----
;--              the unpacked date in a structure of type DATES -----
;-- Input       : AX = The date in DOS format to be unpacked
;--              SI = Pointer to the date block into which the
;--                  information should be stored
;-- Output      : None
;-- Registers   : AX, CX and FLAGS are affected

unpackdate  proc near

    push bx              ;Push BX onto stack

    mov bx,ax            ;Get entire date from BX
    mov cl,5             ;Shift five bits to the right
    shr bx,cl
    and bl,15            ;AND upper four bits

```

```

    mov     [si].month,b1        ;Store month

    mov     bl,al                ;Get low byte of the date in BX
    and     bl,31                ;AND upper three bits
    mov     [si].day,b1          ;Store day of month

    shr     ah,1                 ;Shift high byte one place to right
    mov     al,ah                ;Place year (rel. 1980) in low byte
    xor     ah,ah                ;Set high byte to 0
    add     ax,1980               ;Generate absolute year
    mov     [si].year,ax         ;Add to structure

    pop     bx                   ;Restore BX (pop from stack)
    ret                           ;Return to caller

```

unpackdate endp

```

;-- CMPDAT: Compares two date parameters -----
;-- Input      : SI = Pointer to first date block
;--            DI = Pointer to second date block
;-- Output     : None
;-- Registers  : AX and FLAGS are affected

```

cmpdat proc near

```

;-- Compare year numbers -----

mov     ax,[si].year            ;Load year 1, then
cmp     ax,[di].year            ;compare with year 2
jb      datels                  ;Date 1 < Date 2 --> DATELS
ja      dategr                  ;Date 1 > Date 2 --> DATEGR

;-- Year numbers are identical, now compare days -----

mov     al,[si].day             ;Load day 1, then
cmp     al,[di].day             ;compare with day 2
jb      datels                  ;Date 1 < Date 2 --> DATELS
ja      dategr                  ;Date 1 > Date 2 --> DATEGR

;-- Year numbers are identical, now compare months -----

mov     al,[si].month           ;Load month 1, then
cmp     al,[di].month           ;compare with month 2
jb      datels                  ;Date 1 < Date 2 --> DATELS
ja      dategr                  ;Date 1 > Date 2 --> DATEGR

mov     al,DATCOMP_EQ           ;Both dates are identical
ret                                   ;Return to caller

```

```

datels:  mov     al,DATCOMP_LS    ;Date 1 < Date 2
ret                                   ;Return to caller

```

```

dategr:  mov     al,DATCOMP_GR    ;Date 1 > Date 2
ret                                   ;Return to caller

```

cmpdat endp

```

;-- PRINTDAT: Displays date as a date structure on the screen -----
;-- Input      : SI = Pointer to date structure
;-- Output     : None
;-- Registers  : AX, DX, SI and FLAGS are affected

```

printdat proc near

```

    push    si                   ;Push SI, BX and DX onto the stack
    push    bx
    push    dx

    mov     bx,si                ;Set BX to the date structure
    mov     si,offset datbend-8 ;Allocate SI for year
    mov     ax,[bx].year         ;Get year and
    call    toint                ;convert to ASCII
    mov     byte ptr [si-1],DELIMITER ;Hyphen preceding year

    sub     si,6                  ;Allocate space for day
    mov     al,[bx].day          ;Pass day to AX

```

```

        xor  ah,ah
        call toint                ;and convert to ASCII

        cmp  [bx].day,10          ;First digit preceded by SPACE?
        jae  pd1                  ;Yes --> Process without changes

        dec  si                   ;No --> Make SI first digit
        mov  byte ptr [si],"0"    ;and add NULL

pd1:     mov  byte ptr [si-1],DELIMITER ;Hyphen preceding day
        sub  si,6                 ;Allocate SI space for month
        mov  al,[bx].month        ;Pass month to AX
        xor  ah,ah
        call toint                ;and convert to ASCII
        cmp  [bx].month,10        ;Any NULLs?
        jae  pd2                  ;No --> Continue

        dec  si
        mov  byte ptr [si],"0"

pd2:     mov  dx,si                ;Start at DX
        call printasciic          ;Display ASCII string

pd3:     pop  dx                   ;Pop DX, BX and SI from stack
        pop  bx
        pop  si

        ret

printdat endp

;-- DTAs for recursive calls should be specified here

startdta equ this byte

;== End =====

code     ends                    ;End of CODE segment
        end  start

```