

```

;*****;
;                                     E X E C                                     *;
;-----*
;* Task          :   Calls a program using the DOS EXEC function.  *;
;*               :   This demonstration program displays the      *;
;*               :   current device's directory contents.          *;
;-----*
;* Author        :   Michael Tischer                               *;
;* Developed on   :   08/01/87                                       *;
;* Last update    :   04/07/95                                       *;
;-----*
;* Assembly      :   MASM EXEC;                                       *;
;*               :   LINK EXEC;                                       *;
;*               :                                           or                                           *;
;*               :   TASM EXEC                                       *;
;*               :   TLINK EXEC                                       *;
;-----*
;* Call          :   EXEC                                             *;
;*****;

```

```

;== data =====

```

```

data      segment para 'DATA'      ;Definition of data segment

prgname    db "\command.com",0      ;Name of the program to be called
prgpara    db "/c dir",0           ;Parameters passed to program

;-- Messages used by this program -----
startmes   db " EXEC - (c) 1987,92 by Michael Tischer "
           db 13,10,"$"
mesok      db "OK",13,10,"$"
mesnotf    db "Error: COMMAND.COM not found",13,10,"$"
meselse    db "Error: DOS error code = "
mescode    db " ",13,10,"$"

data      ends                    ;end of data segment

```

```

;== code =====

```

```

code      segment para 'CODE'      ;Definition of CODE segment

          assume cs:code, ds:data, ss:stackseg

exec      proc far

          mov  ax,data              ;Load data segment's segment address
          mov  ds,ax                ;into the DS register

          mov  ah,09h               ;Display startup message
          mov  dx,offset startmes
          int  21h

          call setfree              ;release unused memory

          mov  dx,offset prgname     ;Offset address of program name
          mov  si,offset prgpara     ;Offset address of command line
          call exeprg               ;Call program

          mov  dx,offset mesok       ;Display if execution is error free
          jnc  ex_mes               ;No error --> Display message

          mov  dx,offset mesnotf     ;Program may not be available
          cmp  al,2                  ;Is this the case?
          je   ex_mes               ;Yes --> Display message

          xor  ah,ah                 ;High byte of error code
          mov  si,offset mescode+2   ;Error code in ASCII
          call toint

ex_mes:    mov  dx,offset meselse     ;Display string in DOS
          mov  ah,09h
          int  21h

          mov  ax,4C00h              ;End program with DOS function call

```

```

        int 21h                ;on return of error code 0
exec    endp

;-- SETFREE: Releases unused memory -----
;-- Input      : ES = address of PSP
;-- Output     : none
;-- Registers  : AX, BX, CL and FLAGS are affected
;-- Info      : Since the stack segment is always the last segment in
;              an EXE file, ES:0000 points to the beginning and SS:SP
;              to the end of the program in memory. Through this the
;              program's length can be calculated

setfree proc near

        mov  bx,ss              ;Subtract the segment addresses from
        mov  ax,es              ;each other (result = the number of
        sub  bx,ax              ;paragraphs from PSP to the beginning
                                ;of stack)
        mov  ax,sp              ;Since the stack pointer is at the
        mov  cl,4               ;end of the stack segment, its
        shr  ax,cl              ;contents indicate the stack length
        add  bx,ax              ;Add to current length
        inc  bx                 ;Add another paragraph to be safe

        mov  ah,4ah             ;Pass new length to DOS
        int  21h

        ret                     ;Return to caller

setfree endp

;-- EXEPRG: calls another program -----
;-- Input      : DS:DX = Address of the program name
;--            : DS:SI = Address of the command line
;-- Output     : Carry flag = 1: Error (AX = error code)
;-- Registers  : AX and FLAGS are affected
;-- Info      : Program name and command line must be ASCII strings
;--            : terminated with ASCII code 0

exeprg  proc near

        ;Send command line to its own buffer and count characters ----

        push bx                 ;Store all registers which are
        push cx                 ;destroyed by the call to the
        push dx                 ;DOS EXEC function
        push di
        push si
        push bp
        push ds
        push es

        mov  di,offset comline+1 ;Addresses of chars in command line
        push cs                 ;CS to stack
        pop  es                 ;Back as ES
        xor  bl,bl              ;Set character count to 0
copypara: lodsb                 ;Read a character
        or   al,al              ;Is it a null code (end)?
        je   copyend            ;Yes --> End copy process
        stosb                   ;Store in new buffer
        inc  bl                 ;Increment character count
        cmp  bl,126             ;Maximum reached?
        jne  copypara           ;No --> Continue

copyend: mov  cs:comline,bl      ;Store number of characters
        mov  byte ptr es:[di],13 ;Finish command line

        mov  cs:merkss,ss       ;SS and SP must be stored in
        mov  cs:merksp,sp       ;variables in code segment

        mov  bx,offset parblock ;ES:BX points to parameter block
        mov  ax,4B00h           ;Function number for EXEC function
        int  21h                ;Call DOS function

        cli                     ;Momentarily set stack segment and
        mov  ss,cs:merkss       ;stack pointer interrupts to their

```

```

        mov  sp,cs:merksp      ;original values
        sti                                ;Re-enable interrupt

        pop  es                ;Get all registers from stack again
        pop  ds
        pop  bp
        pop  si
        pop  di
        pop  dx
        pop  cx
        pop  bx

        jc   exeend            ;Errors? Yes --> End
        mov  ah,4dh            ;No errors, find end code of the
        int  21h              ;program which was executed

exeend:  ret                    ;back to caller

;-- Variables of this routine only addressable through CS ----

merkss  dw (?)                ;Accepts SS during program call
merksp  dw (?)                ;Accepts SP during program call
parblock equ this word       ;Parameter block for EXEC function
        dw 0                  ;Environment block
        dw offset comline     ;Offset and segment address of
        dw seg code           ;modified command line
        dd 0                  ;No data in PSP #1
        dd 0                  ;No data in PSP #2

comline db 128 dup (?)        ;Accepts modified command line

exeprg  endp

;-- TOINT: Converts a binary number into an ASCII number and places ---
;-- this number in the caller's buffer right-justified
;-- Input      : DS:DI = Pointer to the position at which the least
;--              significant number should be located
;--              AX      = Binary number to be converted
;-- Output     : DS:SI = Pointer to the most significant number
;-- Registers  : AX, BX, DX, SI and FLAGS are affected
;-- Info      : - Buffer requires a minimum of five characters
;--              - Number is right-justified in the buffer

toint   proc near

        ;-- A loop divides the number by 10, converts the least
        ;-- significant place into ASCII format and places the
        ;-- result in the buffer

        mov  bx,10            ;Divisor is always 10
        jmp  short ti2

ti1:    dec  si                ;SI to next character

ti2:    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  ti1              ;Yes --> Next digit

        ret                    ;Return to caller

toint   endp

;== stack =====

;--- Stack segment is placed here at the end of the file,
;--- to ensure configuration after data and code segments
;--- are placed in memory

stackseg segment para stack 'STACK' ;Definition of stack segment

        dw 256 dup (?)        ;Stack is comprised of 256 words

stackseg ends                  ;End stack segment

```

```
;== End =====
```

```
code      ends      ;End CODE segment  
end  exec      ;To re-execute start with EXEC
```