

```

;*****;
;*                                P R O C P A                                *;
;*-----*;
;* Task          : Determines the type of processor installed in *;
;*               : a PC.                                         *;
;*-----*;
;* Author       : Michael Tischer                                *;
;* Developed on  : 08/22/88                                       *;
;* Last update   : 01/14/92                                       *;
;*-----*;
;* assembly     : MASM PROCPA;   or TASM PROCPA                   *;
;*               : Use $L compiler directive to link to Pascal   *;
;*               : programs                                         *;
;*****;

```

```

;== Constants =====

```

```

p_80486 equ 8           ;Codes for different types of
p_80386 equ 7           ;processors
p_80286 equ 6
p_80186 equ 5
p_80188 equ 4
p_v30   equ 3
p_v20   equ 2
p_8086  equ 1
p_8088  equ 0

```

```

co_80387 equ 3           ;Math coprocessor codes
co_80287 equ 2
co_8087  equ 1
co_none  equ 0

```

```

NOP_CODE    equ    90h                ;Code for NOP assembler instruction
DEC_DX_C    equ    4Ah                ;Code for DEC DX

;== Data segment =====

DATA        segment word public

cpz         dw ?                      ;For coprocessor test

DATA        ends

;== Program =====

code        segment byte public      ;Definition of CODE segment

          assume cs:code, ds:data

public      getproz                  ;Function for accessing
public      getco                    ;other programs

;-- GETPROZ: Describes processor type on board the PC -----
;-- Call from Pascal: function getproz : integer;            -----
;-- Output:  Processor type number (see constants)           -----

getproz     proc near

          pushf                      ;Get contents of flag registers

          ;== Determine whether model is 80286 or lower =====

          xor  ax,ax                  ;Set AX to 0

```

```

push ax          ;and push onto stack
popf             ;Pop into flag register from stack
pushf           ;Return to stack
pop ax          ;And pop back into AX
and ax,0f000h   ;Avoid clearing the 4 bits
cmp ax,0f000h   ;Are bits 12-15 all equal to 1?
je not_286_386  ;YES->Not an 80386 or an 80286

```

!-- Test whether to handle it as an 80406, 80386 or 80286 ---

```

mov dl,p_80286   ;This narrows it down to one of
mov ax,07000h    ;the two processors
push ax         ;Push value 7000(h) onto the stack
popf            ;Pop off as flag register
pushf           ;and push it back onto the stack
pop ax          ;Pop off and return to AX register
and ax,07000h    ;Avoid masking bits 12-14
je pende        ;Are bits 12-14 all equal to 0?
                ;YES->Handle it as an 80286

```

```

inc dl          ;NO->Handle it as an 80386 or 80486

```

!-- The following tests for 80386/80486 status through -----
 !-- the EFlags register at bit position 18 (found on -----
 !-- 80486 chips only). -----

```

cli            ;No interrupts right now

```

db 066h,08Bh,0DCh	;mov ebx,esp	Mark current SP
db 066h,083h,0E4h,0FCh	;and esp,0FFFCh	Round to DWORD
db 066h,09Ch	;pushfd	Put Flag reg. on
db 066h,058h	;pop eax	stack from AX,

```

db 066h,08Bh,0C8h          ;mov     ecx,eax      mark in CX
db 066h,035h,000h,0h,4h,0h ;xor      eax,1 shl 18 Shift alignment
db 066h,050h                ;push     eax      bit, pass it
db 066h,09Dh                ;popfd      to flag register
db 066h,09Ch                ;pushfd     Get flag from
db 066h,058h                ;pop      eax      stack, AX
db 066h,051h                ;push     ecx      Old flag data
db 066h,09Dh                ;popfd      Restore
db 066h,033h,0C1h          ;xor      eax,ecx    Test AL bit
db 066h,0C1h,0E8h,012h     ;xor      eax,12h    AL bit to bit 0
db 066h,083h,0E0h,001h     ;and      eax,1h    Mask all others
db 066h,08Bh,0E3h          ;mov      esp,ebx    Restore SP

        sti                    ;Allow interrupts again
        add     dl,al          ;If AL = 1 then 486
        jmp     pende          ;End of test

;-- Test for 80186 or 80188 -----
not_286_386 label near

        mov     dl,p_80188     ;Load code for 80188
        mov     al,0ffh        ;Set all bits in AL register to 1
        mov     cl,021h        ;Number of shift operations after CL
        shr     al,cl          ;Shift AL CL times to the right
        jne     t88_86         ;If AL is unequal to 0 it must be
                                ;handled as an 80188 or 80186

;-- Test for NEC V20 or V30 -----

        mov     dl,p_v20       ;Load code for NEC V20
        sti                    ;Interrupts should be enabled starting

```

```

mov     si,0                ;with the first byte in ES
mov     cx,0ffffh          ;Read a complete segment
rep     lodsb byte ptr es:[si] ;REP w/ segment override only
                                ;works with NEC V20 and V30 processors
or      cx,cx              ;Has complete segment been read?
je      t88_86             ;YES-> V20 or V30

mov     dl,p_8088           ;NO-> Must be an 8088 or 8086

;-- Test for 8088 or 8086/V20 or V30 -----
;-- Use queue to test for chip -----

```

t88_86 label near

```

push cs                ;Push CS onto stack
pop  es               ;Pop off to ES
std                     ;Using string inst. count backwards
mov  di,offset q2_end ;Set DI to end of queue
mov  al,0fbh          ;Instruction code for "STI"
mov  cx,3              ;Execute string instruction 3 times
cli                     ;Suppress interrupts
rep  stosb            ;Overwrite INC DX instruction
cld                     ;Using string inst. count backwards
nop                     ;Fill queue with dummy instruction
nop
nop

inc  dx                ;Increment processor code
nop
q2_end: sti            ;Re-enable interrupts

```

```

;-----
pende    label near          ;End testing

        popf                ;Pop flag register from stack
        xor  dh,dh           ;Processor code high byte = 0
        mov  ax,dx           ;Processor code is return value of

        ret                 ;Return to caller

getproz  endp                ;End procedure

;-- GETCO Returns the type of coprocessor (if applicable) -----
;-- Call from Pascal: function getco : integer; -----
;-- Output: Number of the coprocessor type (see constants) -----

getco    proc near

        mov  dx,co_none      ;First assume no coproc. exists

        mov  byte ptr cs:wait1,NOP_CODE ;Replace 8087 WAIT with
        mov  byte ptr cs:wait2,NOP_CODE ;NOP

wait1:    finit              ;Copro. initialization
        mov  byte ptr cpz+1,0  ;High byte control word to 0
wait2:    fstcw cpz           ;Store control word
        cmp  byte ptr cpz+1,3  ;High-byte control word = 3?
        jne  gcende           ;No --> No coprocessor

        ;-- Coprocessor exists: Test for 8087 -----

        inc  dx

```

```

and    cpz,0FF7Fh      ;Mask interrupt enable mask flag
fldcw  cpz              ;Pass to control word
fdisi                      ;Set IEM flag
fstcw  cpz              ;Reload control word
test   cpz,80h          ;IEM flag set?
jne     gcende          ;Yes --> 8087, end test

;-- Coprocessor exists: Test for 80287/80387 -----

inc     dx
finit                      ;Coproc. initialization
fldl                      ;Number 1 to Cop stack
fldz                      ;Number 0 to Cop stack
fdiv                      ;Reload control word
fld     st                ;IEM flag set?
fchs                      ;Yes --> 8087, end test
fcompp                      ;Pop return address off of stack
fstsw   cpz               ;Take first 9 bytes from there
mov     ah,byte ptr cpz+1 ;Return to the testing routine
sahf                      ;
je       gcende           ;Zero flag = 1: 80287

inc     dx                ;No 80287? Must be 80387 or coproc.
                        ;integrated in 80486

gcende:  mov     ax,dx      ;AX gets result
         ret              ;Return to caller

getco    endp

;== End =====

```

code

ends

end

;End of code segment

;End program

