Note: This API calls are shared between DOS and Win16 personality.

DPMI is a shared interface for DOS applications to access Intel 80286+ CPUs services. DOS DMPI host provides core services for protected mode applications. Multitasking OS with DOS support also provides DMPI in most cases. Windows standard and extended mode kernel is a DPMI client app. Standard and extended mode kernel differs minimally and shares common codebase. Standard Windows kernel works under DOSX extender. DOSX is a specialized version of 16-bit DPMI Extender (but it is standard DPMI host). Standard mode is just DPMI client, exnhanced mode is DPMI client running under Virtual Machime Manager (really, multitasker which allow to run many DOS sessions). Both modes shares DPMI interface for kernel communication. The OS/2 virtual DOS Protected Mode Interface (VDPMI) device driver provides Version 0.9 DPMI support for virtual DOS machines. Win16 (up to Windows ME) provides Version 0.9 DPMI support. Windows in Standard Mode provides DPMI services only for Windows Applications, not DOS sessions.

DPMI host often merged with DPMI extender. Usually DPMI extender provide DPMI host standard services and DOS translation or True DPMI services.

2021/08/05 10:15 · prokushev · 0 Comments

Int 31H, AH=03H, AL=00H

Version

0.9

Brief

Simulate Real Mode Interrupt

Input

```
AX = 0300H
BL = interrupt number
BH = flags
```

Bit	Significance
0	reserved for historical reason, must be zero
1-7	reserved, must be zero

```
CX = number of words to copy from protected mode to real mode stack
ES:(E)DI = selector:offset of real mode register data structure in the
```

```
following format:
```

Offset	Length	Contents
00H	4	DI or EDI
04H	4	SI or ESI
08H	4	BP or EBP
0CH	4	reserved, should be zero
10H	4	BX or EBX
14H	4	DX or EDX
18H	4	CX or ECX
1CH	4	AX or EAX
20H	2	CPU status flags
22H	2	ES
24H	2	DS
26H	2	FS
28H	2	GS
2AH	2	IP (reserved, ignored)
2CH	2	CS (reserved, ignored)
2EH	2	SP
30H	2	SS

```
if function successful
Carry flag = clear
ES:(E)DI = selector:offset of modified real mode register data structure

if function unsuccessful
Carry flag = set
AX = error code
8012H   linear memory unavailable (stack)
8013H   physical memory unavailable (stack)
8014H   backing store unavailable (stack)
8021H   invalid value (CX too large)
```

Notes

Simulates an interrupt in real mode. The function transfers control to the address specified by the real mode interrupt vector. The real mode handler must return by executing an IRET.

32-bit programs must use ES:EDI to point to the real mode register data structure. 16-bit programs should use ES:DI.

The CS:IP in the real mode register data structure is ignored by this function. The appropriate interrupt handler will be called based on the value passed in BL.

If the SS:SP fields in the real mode register data structure are zero, a real mode stack will be provided by the DPMI host. Otherwise, the real mode SS:SP will be set to the specified values before the interrupt handler is called.

The flags specified in the real mode register data structure will be pushed on the real mode stack's IRET frame. The interrupt handler will be called with the interrupt and trace flags clear.

Values placed in the segment register positions of the data structure must be valid for real mode; i.e. the values must be paragraph addresses and not selectors.

All general register fields in the data structure are DWORDs so that 32-bit registers can be passed to real mode. Note, however, that 16-bit hosts are not required to pass the high word of 32-bit general registers or the FS and GS segment registers to real mode even when running on an 80386 or later CPU.

The target real mode handler must return with the stack in the same state as when it was called. This means that the real mode code may switch stacks while it is running, but must return on the same stack that it was called on and must return with an IRET.

When this function returns, the real mode register data structure will contain the values that were returned by the real mode interrupt handler.

It is the caller's responsibility to remove any parameters that were pushed on the protected mode stack.

See also

Note

Text based on http://www.delorie.com/djgpp/doc/dpmi/

DPMI	
Process manager	**INT 2FH** 1680H, 1687H
Signals	
Memory manager	
Misc	**INT 2FH** 1686H, 168AH
Devices	

2021/08/13 14:23 · prokushev · 0 Comments