

Re: Floating point in kernel mode, Win2003 64-bit

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2004-04/1235.html>

nospam_at_cristalink.com

Date: 04/22/04

Date: Fri, 23 Apr 2004 10:55:43 +1200

I'm interested in this because I use SSE2 in kernel mode, and your problem is closely related to my own projects. It looks like only a few guys in MS know exactly what happens with FPU during context switches and interrupts.

So you are saying that PSR.dfh becomes 1 somewhere in the middle of your code. I can see a few possibilities.

(1) Some other driver messes up FPU, though I think it's unlikely.

(2) Upper registers are supported in the kernel, but you incorrectly use them. I would expect KeSave/RestoreFloatingPointState() to be enough. As per the Itanium manual, it's ok to have dfh=1 before using upper registers, assuming the OS is properly designed to handle deferred FPU fills/spills. The first instruction using upper registers causes an exception that is handled by the kernel to save or restore FPU state. With this approach, you must neither touch PSR.dfh nor manually save/restore upper registers, esp. because you're in a dedicated thread.

(3) There is a bug or feature in the kernel that doesn't allow using upper registers in the kernel mode on Itanium, and the kernel does not handle the exception on a fh-instruction with dfh=1. You can probably try wrapping your code in __try/__leave, reset dfh in the exception handler and continue. I am not sure this will work. Besides, I suspect other FPU registers get changed during a context switch, so your calculations might produce wrong results anyway.

"Sergey Plotnikov" <svplotnik@yahoo.com> wrote in message
news:EA6AC636-27EC-485D-875C-AC1A659CC0C3@microsoft.com...

>I certainly don't mix up .mfh and .dfh.

>

> You approach is correct. More precisely it looks like:

>

> StartDevice(){

> PsCreateSystemThread(..., WorkingThread,...);

> }

>

> Save(){

```
> PSR.dfh = 0;
> save fh;
> }
>
> Restore(){
> Restore fh;
> PSR.dfh = 1;
> }
>
> InterruptHundler(){
> RequestDpc();
> }
>
> DPC(){
> KeSetEvent(event) ;
> }
>
> WorkingThread(){
> for(;;){
> KeWaitForSingleObject(event);
> Save();
> a_lot_of_integer_computation(); //it may take up to several milliseconds
> float_computation(); //crash is always here
> Restore();
> }
> }
>
> As you can see my function is called only from my thread in system
> context, always on PASSIVE level. But still something changes .dfh. The
> probability of crash is about 1/1000.
>
> Sergey
```