NAME

fpu_kern - facility to use the FPU in the kernel

SYNOPSIS

#include <machine/fpu.h>

*struct fpu_kern_ctx **
fpu_kern_alloc_ctx(*u_int flags*);

void
fpu_kern_free_ctx(*struct fpu_kern_ctx *ctx*);

void
fpu_kern_enter(*struct thread *td*, *struct fpu_kern_ctx *ctx*, *u_int flags*);

int
fpu_kern_leave(*struct thread *td*, *struct fpu_kern_ctx *ctx*);

int
fpu_kern_thread(*u_int flags*);

int
is_fpu_kern_thread(*u_int flags*);

DESCRIPTION

The **fpu_kern** family of functions allows the use of FPU hardware in kernel code. Modern FPUs are not limited to providing hardware implementation for floating point arithmetic; they offer advanced accelerators for cryptography and other computational-intensive algorithms. These facilities share registers with the FPU hardware.

Typical kernel code does not need access to the FPU. Saving a large register file on each entry to the kernel would waste time. When kernel code uses the FPU, the current FPU state must be saved to avoid corrupting the user-mode state, and vice versa.

The management of the save and restore is automatic. The processor catches accesses to the FPU registers when the non-current context tries to access them. Explicit calls are required for the allocation of the save area and the notification of the start and end of the code using the FPU.

The **fpu_kern_alloc_ctx**() function allocates the memory used by **fpu_kern** to track the use of the FPU hardware state and the related software state. The **fpu_kern_alloc_ctx**() function requires the *flags*

argument, which currently accepts the following flags:

FPU_KERN_NOWAIT Do not wait for the available memory if the request could not be satisfied without sleep.

0 No special handling is required.

The function returns the allocated context area, or *NULL* if the allocation failed.

The **fpu_kern_free_ctx**() function frees the context previously allocated by **fpu_kern_alloc_ctx**().

The **fpu_kern_enter**() function designates the start of the region of kernel code where the use of the FPU is allowed. Its arguments are:

td Currently must be *curthread*.

ctx The context save area previously allocated by **fpu_kern_alloc_ctx**() and not currently in use by another call to **fpu_kern_enter**().

flags
This argument currently accepts the following flags:

FPU_KERN_NORMAL Indicates that the caller intends to access the full FPU state. Must be specified currently.

FPU_KERN_KTHR Indicates that no saving of the current FPU state should be performed, if the thread called fpu_kern_thread(9) function. This is intended to minimize code duplication in callers which could be used from both kernel thread and syscall contexts. The **fpu_kern_leave**() function correctly handles such contexts.

FPU_KERN_NOCTX Avoid nesting save area. If the flag is specified, the *ctx* must be passed as *NULL*. The flag should only be used for really short code blocks which can be executed in a critical section. It avoids the need to allocate the FPU context by the cost of increased system latency.

The function does not sleep or block. It could cause an FPU trap during execution, and on the first FPU access after the function returns, as well as after each context switch. On i386 and amd64 this will be

the **Device Not Available** exception (see Intel Software Developer Manual for the reference).

The **fpu_kern_leave**() function ends the region started by **fpu_kern_enter**(). It is erroneous to use the FPU in the kernel before **fpu_kern_enter**() or after **fpu_kern_leave**(). The function takes the *td* thread argument, which currently must be *curthread*, and the *ctx* context pointer, previously passed to **fpu_kern_enter**(). After the function returns, the context may be freed or reused by another invocation of **fpu_kern_enter**(). The function always returns 0.

The **fpu_kern_thread**() function enables an optimization for threads which never leave to the usermode. The current thread will reuse the usermode save area for the kernel FPU state instead of requiring an explicitly allocated context. There are no flags defined for the function, and no error states that the function returns. Once this function has been called, neither **fpu_kern_enter**() nor **fpu_kern_leave**() is required to be called and the fpu is available for use in the calling thread.

The **is_fpu_kern_thread**() function returns the boolean indicating whether the current thread entered the mode enabled by **fpu_kern_thread**(). There is currently no flags defined for the function, the return value is true if the current thread have the permanent FPU save area, and false otherwise.

NOTES

The **fpu_kern** is currently implemented only for the i386, amd64, and arm64 architectures.

There is no way to handle floating point exceptions raised from kernel mode.

The unused *flags* arguments to the **fpu_kern** functions are to be extended to allow specification of the set of the FPU hardware state used by the code region. This would allow optimizations of saving and restoring the state.

AUTHORS

The **fpu_kern** facitily and this manual page were written by Konstantin Belousov <*kib@FreeBSD.org*>.
The arm64 support was added by
Andrew Turner <*andrew@FreeBSD.org*>.

BUGS

fpu_kern_leave() should probably have type *void* (like **fpu_kern_enter**()).