Interface for registering and calling firmware-specific operations for ARM

Written by Tomasz Figa <t.figa@samsung.com>

Some boards are running with secure firmware running in TrustZone secure world, which changes the way some things have to be initialized. This makes a need to provide an interface for such platforms to specify available firmware operations and call them when needed.

Firmware operations can be specified by filling in a struct firmware_ops with appropriate callbacks and then registering it with register_firmware_ops() function:

```
void register_firmware_ops(const struct firmware_ops *ops)
```

The ops pointer must be non-NULL. More information about struct firmware_ops and its members can be found in arch/arm/include/asm/firmware.h header.

There is a default, empty set of operations provided, so there is no need to set anything if platform does not require firmware operations.

To call a firmware operation, a helper macro is provided:

```
#define call_firmware_op(op, ...)                                       \
        ((firmware_ops->op) ? firmware_ops->op(__VA_ARGS__) : (-ENOSYS))
```

the macro checks if the operation is provided and calls it or otherwise returns -ENOSYS to signal that given operation is not available (for example, to allow fallback to legacy operation).

Example of registering firmware operations:

```
/* board file */

static int platformX_do_idle(void)
{
        /* tell platformX firmware to enter idle */
        return 0;
}

static int platformX_cpu_boot(int i)
{
        /* tell platformX firmware to boot CPU i */
        return 0;
}

static const struct firmware_ops platformX_firmware_ops = {
        .do_idle        = exynos_do_idle,
        .cpu_boot       = exynos_cpu_boot,
        /* other operations not available on platformX */
};

/* init_early callback of machine descriptor */
static void __init board_init_early(void)
{
        register_firmware_ops(&platformX_firmware_ops);
}
```

Example of using a firmware operation:

```
/* some platform code, e.g. SMP initialization */

__raw_writel(__pa_symbol(exynos4_secondary_startup),
        CPU1_BOOT_REG);

/* Call Exynos specific smc call */
if (call_firmware_op(cpu_boot, cpu) == -ENOSYS)
        cpu_boot_legacy(...); /* Try legacy way */

gic_raise_softirq(cpumask_of(cpu), 1);
```