CPU Features

Hollis Blanchard <hollis@austin.ibm.com> 5 Jun 2002

This document describes the system (including self-modifying code) used in the PPC Linux kernel to support a variety of PowerPC CPUs without requiring compile-time selection.

Early in the boot process the ppc32 kernel detects the current CPU type and chooses a set of features accordingly. Some examples include Altivec support, split instruction and data caches, and if the CPU supports the DOZE and NAP sleep modes.

Detection of the feature set is simple. A list of processors can be found in arch/powerpc/kernel/cputable.c. The PVR register is masked and compared with each value in the list. If a match is found, the cpu_features of cur_cpu_spec is assigned to the feature bitmask for this processor and a __setup_cpu function is called.

C code may test 'cur_cpu_spec[smp_processor_id()]->cpu_features' for a particular feature bit. This is done in quite a few places, for example in ppc_setup_l2cr().

Implementing cpufeatures in assembly is a little more involved. There are several paths that are performance-critical and would suffer if an array index, structure dereference, and conditional branch were added. To avoid the performance penalty but still allow for runtime (rather than compile-time) CPU selection, unused code is replaced by 'nop' instructions. This nop'ing is based on CPU 0's capabilities, so a multi-processor system with non-identical processors will not work (but such a system would likely have other problems anyways).

After detecting the processor type, the kernel patches out sections of code that shouldn't be used by writing nop's over it. Using cpufeatures requires just 2 macros (found in arch/powerpc/include/asm/cputable.h), as seen in head.S transfer_to_handler:

```
#ifdef CONFIG_ALTIVEC
BEGIN_FTR_SECTION
        mfspr   r22,SPRN_VRSAVE         /* if G4, save vrsave register value */
        stw     r22,THREAD_VRSAVE(r23)
END_FTR_SECTION_IFSET(CPU_FTR_ALTIVEC)
#endif /* CONFIG_ALTIVEC */
```

If CPU 0 supports Altivec, the code is left untouched. If it doesn't, both instructions are replaced with nop's.

The END_FTR_SECTION macro has two simpler variations: END_FTR_SECTION_IFSET and END_FTR_SECTION_IFCLR. These simply test if a flag is set (in cur_cpu_spec[0]->cpu_features) or is cleared, respectively. These two macros should be used in the majority of cases.

The END_FTR_SECTION macros are implemented by storing information about this code in the '__ftr_fixup' ELF section. When do_cpu_ftr_fixups (arch/powerpc/kernel/misc.S) is invoked, it will iterate over the records in __ftr_fixup, and if the required feature is not present it will loop writing nop's from each BEGIN_FTR_SECTION to END_FTR_SECTION.