Startup Support for SMP#

procnto-smp relies on the startup program to:

- manage the initialisation of each cpu
- provide support for inter-processor communication via inter-processor interrupts (IPIs)

This support is split into two areas:

- generic support in libstartup
- board-specific support in the board startup

---

libstartup SMP support#

---

libstartup.a provides the basic framework for SMP initialisation:

- init_smp.c provides cpu-independent code to drive the SMP initialisation sequence:
  - init_smp() sets up the hooks for the initialisation sequencing
  - start_aps() handles the initial execution of each secondary cpu
  - transfer_aps() handles the execution of each secondary cpu before the boot cpu jumps to the kernel
- cpu-dependent code provides the following:
  - _smp_start is the initial entry point for secondary cpus
  - smp_spin a callout that implements the entry into the kernel for secondary cpus

_smp_start#

This is the cpu-specific entry point for secondary processors. Its basic operation is:

```
void
_smp_start(void)
{
 set stack pointer
 perform any other cpu-specific setup required for the C calling environment

 // normalise our cpu number. start_aps() sets cpu_starting
 cpu_num = board_cpu_adjust_num(cpu_starting)
 // perform any cpu-dependent initialistion
 cpu_startup() // or cpu_one_startup(cpu_num) depending on the libstartup code

 // set up our syspage cpuinfo entry
 init_one_cpuinfo(cpu_num)

 // tell start_aps() it can start the next cpu
 cpu_starting = 0

 // wait for the boot cpu to finish initialising the system page
 while (syspage_available != cpu_num)
   ;

 // jump to smp_spin callout
 cpu_startnext(smp_spin_vaddr, cpu_num)

 // cpu_startnext should not return...
}
```

See the following for the actual implementations:

- arm/smp_start.S
- mips/smp_start.S
- ppc/smp_start.s
- sh/smp_start.s

smp_spin#

This is a cpu-specific callout used to hold secondary cpus in a spin-loop until the kernel is ready for them to take part in the kernel initialisation sequence. Its basic operation is:

```
smp_spin(struct syspage_entry *syspage, unsigned cpu)
{
   struct smp_entry *smp = _SYSPAGE_ENTRY(syspage, smp);
   unsigned         entry;

   // tell transfer_aps() we have started
   smp->pending = 0;

   // wait for smp->start_addr to be non-zero, then atomically exchange with 0
   do {
      while (smp->start_addr == 0)
         ;
   } while ((entry = _smp_xchg(&smp->start_addr, 0)) == 0);

   if (entry != -1) {
      void (*call_entry)() = (void (*)())entry;
      // jump to entry
      call_entry();
   } else {
      // kernel wants us to shut down. Put -1 back so next cpu will also shut down
      smp->start_addr = -1;
      // halt cpu
   }
}
```

See the following for the actual implementation:

- arm/callout_smp_spin.S
- mips/callout_smp_spin.S
- ppc/callout_smp_spin.s
- sh/callout_smp_spin.s

Board Startup SMP Support#

The board specific startup program is responsible for providing the following:

- board_smp.c implements a number of support functions for the generic libstartup code
  - board_smp_num_cpu()
  - board_smp_init()
  - board_smp_start()
  - board_smp_adjust_num()
- a send_ipi callout that is used by procnto-smp to send interprocessor interrupts

board_smp_num_cpu()#

This function is called by smp_init() to find out the number of cpus physically present in the system:

```
unsigned
board_smp_num_cpu()
{
   unsigned num;

   // board specific operation to determine number of cpus in the system
   return num;
}
```

board_smp_init()#

This function is called by smp_init() to perform any board specific SMP initialisation.

At a minimum, it needs to specify the board-specific send_ipi callout:

```
void
board_smp_init(struct smp_entry *smp, unsigned num_cpus)
{
   smp->send_ipi = (void *)&my_send_ipi;
}
```

board_smp_start()#

This function is called by start_aps() to perform whatever board/cpu specific actions are needed to start the specified cpu.

board_smp_adjust_num()#

This function is called by _smp_start to perform any board/cpu-specific actions required to adjust the cpu number:

```
unsigned
board_smp_adjust_num(unsigned cpu)
{
   // board or cpu-specific actions to set cpu id to cpu
   return cpu;
}
```

send_ipi callout#

This is a board-specific callout routine used by the kernel to send an inter-processor interrupt (IPI) to a specific cpu.

As with all callout routines, it must be hand-coded in assembler to ensure that it is position independent (the code is copied into the system page).

The kernel IPI protocol uses a bitmask of pending commands for each cpu, and this send_ipi callout is used to atomically set the command bit for the target cpu and to perform the board specific operations required to trigger an IPI interrupt on the target cpu:

```
void
send_ipi(struct syspage_entry *sysp, unsigned cpu, unsigned cmd, unsigned *cmd_bits)
{
```

```
    if (atomic_set_value(cmd_bits, cmd) == 0) {
        // trigger IPI interrupt on target cpu
    }
}
```