```
/* -*- linux-c -*- ------------------------------------------------------- *
 *
 *   Copyright (C) 1991, 1992 Linus Torvalds
 *   Copyright 2007 rPath, Inc. - All Rights Reserved
 *
 *   This file is part of the Linux kernel, and is made available under
 *   the terms of the GNU General Public License version 2.
 *
 * ----------------------------------------------------------------------- */

/*
 * Prepare the machine for transition to protected mode.
 */

#include "boot.h"
#include <asm/segment.h>

/*
 * Invoke the realmode switch hook if present; otherwise
 * disable all interrupts.
 */
static void realmode switch hook(void)
{
          if (boot_params.hdr.realmode_swtch) {
                    asm volatile("lcallw *%0"
                                 : : "m" (boot_params.hdr.realmode_swtch)
                                 : "eax", "ebx", "ecx", "edx");
          } else {
                    asm volatile("cli");
                    outb(0x80, 0x70); /* Disable NMI */
                    io_delay();
          }
}

/*
 * Disable all interrupts at the legacy PIC.
 */
static void mask_all_interrupts(void)
{
          outb(0xff, 0xa1);  /* Mask all interrupts on the secondary PIC */
          io_delay();
          outb(0xfb, 0x21);  /* Mask all but cascade on the primary PIC */
          io_delay();
}

/*
 * Reset IGNNE# if asserted in the FPU.
 */
static void reset_coprocessor(void)
{
          outb(0, 0xf0);
          io_delay();
          outb(0, 0xf1);
          io_delay();
}

/*
 * Set up the GDT
 */

struct gdt ptr {
          u16 len;
          u32 ptr;
} __attribute__((packed));

```

```
66  static void setup_gdt(void)
67  {
68            /* There are machines which are known to not boot with the GDT
69               being 8-byte unaligned.  Intel recommends 16 byte alignment. */
70            static const u64 boot_gdt[] __attribute__((aligned(16))) = {
71                      /* CS: code, read/execute, 4 GB, base 0 */
72                      [GDT_ENTRY_BOOT_CS] = GDT_ENTRY(0xc09b, 0, 0xfffff),
73                      /* DS: data, read/write, 4 GB, base 0 */
74                      [GDT_ENTRY_BOOT_DS] = GDT_ENTRY(0xc093, 0, 0xfffff),
75                      /* TSS: 32-bit tss, 104 bytes, base 4096 */
76                      /* We only have a TSS here to keep Intel VT happy;
77                         we don't actually use it for anything. */
78                      [GDT_ENTRY_BOOT_TSS] = GDT_ENTRY(0x0089, 4096, 103),
79            };
80            /* Xen HVM incorrectly stores a pointer to the gdt_ptr, instead
81               of the gdt_ptr contents.  Thus, make it static so it will
82               stay in memory, at least long enough that we switch to the
83               proper kernel GDT. */
84            static struct gdt_ptr gdt;
85
86            gdt.len = sizeof(boot_gdt)-1;
87            gdt.ptr = (u32)&boot_gdt + (ds() << 4);
88
89            asm volatile("lgdtl %0" : : "m" (gdt));
90  }
91
92  /*
93   * Set up the IDT
94   */
95  static void setup_idt(void)
96  {
97            static const struct gdt_ptr null_idt = {0, 0};
98            asm volatile("lidtl %0" : : "m" (null_idt));
99  }
100
101 /*
102  * Actual invocation sequence
103  */
104 void go_to_protected_mode(void)
105 {
106           /* Hook before leaving real mode, also disables interrupts */
107           realmode_switch_hook();
108
109           /* Enable the A20 gate */
110           if (enable_a20()) {
111                     puts("A20 gate not responding, unable to boot...\n");
112                     die();
113           }
114
115           /* Reset coprocessor (IGNNE#) */
116           reset_coprocessor();
117
118           /* Mask all interrupts in the PIC */
119           mask_all_interrupts();
120
121           /* Actual transition to protected mode... */
122           setup_idt();
123           setup_gdt();
124           protected_mode_jump(boot_params.hdr.code32_start,
125                               (u32)&boot_params + (ds() << 4));
126 }
```