```
**********************************************************
    7961 Mon Mar 27 00:11:46 2017
new/usr/src/uts/i86pc/os/biosdisk.c
XXX nobios
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/controlregs.h>
#include <sys/bootconf.h>
#include <sys/bootvfs.h>
#include <sys/bootregs.h>
#include <sys/bootconf.h>
#include <sys/conf.h>
#include <sys/promif.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/sunndi.h>
#include <sys/biosdisk.h>
#include <sys/psw.h>
#include <sys/machsystm.h>
#if defined(__xpv)
#include <sys/hypervisor.h>
#endif

extern int prom_debug;

/* hard code realmode memory address for now */
#define BIOS_RES_BUFFER_ADDR            0x7000

#define BIOSDEV_NUM     8
#define STARTING_DRVNUM 0x80
#define FP_OFF(fp) (((uintptr_t)(fp)) & 0xFFFF)
#define FP_SEG(fp) ((((uintptr_t)(fp)) >> 16) & 0xFFFF)

#ifdef DEBUG
int biosdebug = 0;
#define dprintf(fmt) \
        if (biosdebug) \
                prom_printf fmt
#else
#define dprintf(fmt)
```



```
#endif

biosdev_data_t biosdev_info[BIOSDEV_NUM]; /* from 0x80 to 0x87 */
int dobiosdev = 1;


static int bios_check_extension_present(uchar_t);
static int get_dev_params(uchar_t);
static int read_firstblock(uchar_t drivenum);
static int drive_present(uchar_t drivenum);
static void reset_disk(uchar_t drivenum);
static int is_eltorito(uchar_t drivenum);

#if !defined(__xpv)
void
startup_bios_disk()
{
        uchar_t drivenum;
        int got_devparams = 0;
        int got_first_block = 0;
        uchar_t name[20];
        dev_info_t      *devi;
        int extensions;

        if (dobiosdev == 0 || !bios_calls_available) {
                /*
                 * If BIOS calls have been disabled, or are not supported on
                 * this system, we cannot probe for the startup disk.
                 */
        if (dobiosdev == 0)
                return;
        }

        for (drivenum = 0x80; drivenum < (0x80 + BIOSDEV_NUM); drivenum++) {

                if (!drive_present(drivenum))
                        continue;

                extensions = bios_check_extension_present(drivenum);

                /*
                 * If we're booting from an Eltorito CD/DVD image, there's
                 * no need to get the device parameters or read the first block
                 * because we'll never install onto this device.
                 */
                if (extensions && is_eltorito(drivenum))
                        continue;

                if (extensions && get_dev_params(drivenum))
                        got_devparams = 1;
                else
                        got_devparams = 0;

                if ((got_first_block = read_firstblock(drivenum)) == 0) {
                        /* retry */
                        got_first_block = read_firstblock(drivenum);
                }

                if (got_devparams || got_first_block) {
                        (void) sprintf((char *)name, "biosdev-0x%x", drivenum);
                        devi = ddi_root_node();
                        (void) e_ddi_prop_update_byte_array(DDI_DEV_T_NONE,
                            devi, (char *)name,
                            (uchar_t *)&biosdev_info[drivenum - 0x80],
                            sizeof (biosdev_data_t));
                }
```

```
 127         }
 128 }
______unchanged_portion_omitted_
```

```
**********************************************************
   63138 Mon Mar 27 00:11:47 2017
new/usr/src/uts/i86pc/os/fakebop.c
XXX nobios
**********************************************************
______unchanged_portion_omitted_

 120 static bootprop_t *bprops = NULL;
 121 static char *curr_page = NULL;          /* ptr to avail bprop memory */
 122 static int curr_space = 0;              /* amount of memory at curr_page */

 124 #ifdef __xpv
 125 start_info_t *xen_info;
 126 shared_info_t *HYPERVISOR_shared_info;
 127 #endif

 129 /*
 130  * some allocator statistics
 131  */
 132 static ulong_t total_bop_alloc_scratch = 0;
 133 static ulong_t total_bop_alloc_kernel = 0;

 135 static void build_firmware_properties(void);

 137 static int early_allocation = 1;

 139 int force_fastreboot = 0;
 140 volatile int fastreboot_onpanic = 0;
 141 int post_fastreboot = 0;
 142 #ifdef  __xpv
 143 volatile int fastreboot_capable = 0;
 144 boolean_t bios_calls_available = B_FALSE;
 145 #else
 146 volatile int fastreboot_capable = 1;
 147 boolean_t bios_calls_available = B_TRUE;
 148 #endif

 150 /*
 151  * Information saved from current boot for fast reboot.
 152  * If the information size exceeds what we have allocated, fast reboot
 153  * will not be supported.
 154  */
 155 multiboot_info_t saved_mbi;
 156 mb_memory_map_t saved_mmap[FASTBOOT_SAVED_MMAP_COUNT];
 157 uint8_t saved_drives[FASTBOOT_SAVED_DRIVES_SIZE];
 158 char saved_cmdline[FASTBOOT_SAVED_CMDLINE_LEN];
 159 int saved_cmdline_len = 0;
 160 size_t saved_file_size[FASTBOOT_MAX_FILES_MAP];

 162 /*
 163  * Turn off fastreboot_onpanic to avoid panic loop.
 164  */
 165 char fastreboot_onpanic_cmdline[FASTBOOT_SAVED_CMDLINE_LEN];
 166 static const char fastreboot_onpanic_args[] = " -B fastreboot_onpanic=0";

 168 /*
 169  * Pointers to where System Resource Affinity Table (SRAT), System Locality
 170  * Information Table (SLIT) and Maximum System Capability Table (MSCT)
 171  * are mapped into virtual memory
 172  */
 173 ACPI_TABLE_SRAT *srat_ptr = NULL;
 174 ACPI_TABLE_SLIT *slit_ptr = NULL;
 175 ACPI_TABLE_MSCT *msct_ptr = NULL;

 177 /*
 178  * Arbitrary limit on number of localities we handle; if
```



```
 179  * this limit is raised to more than UINT16_MAX, make sure
 180  * process_slit() knows how to handle it.
 181  */
 182 #define SLIT_LOCALITIES_MAX     (4096)

 184 #define SLIT_NUM_PROPNAME       "acpi-slit-localities"
 185 #define SLIT_PROPNAME           "acpi-slit"

 187 /*
 188  * Allocate aligned physical memory at boot time. This allocator allocates
 189  * from the highest possible addresses. This avoids exhausting memory that
 190  * would be useful for DMA buffers.
 191  */
 192 paddr_t
 193 do_bop_phys_alloc(uint64_t size, uint64_t align)
 194 {
 195         paddr_t pa = 0;
 196         paddr_t start;
 197         paddr_t end;
 198         struct memlist  *ml = (struct memlist *)xbootp->bi_phys_install;

 200         /*
 201          * Be careful if high memory usage is limited in startup.c
 202          * Since there are holes in the low part of the physical address
 203          * space we can treat physmem as a pfn (not just a pgcnt) and
 204          * get a conservative upper limit.
 205          */
 206         if (physmem != 0 && high_phys > pfn_to_pa(physmem))
 207                 high_phys = pfn_to_pa(physmem);

 209         /*
 210          * find the lowest or highest available memory in physinstalled
 211          * On 32 bit avoid physmem above 4Gig if PAE isn't enabled
 212          */
 213 #if defined(__i386)
 214         if (xbootp->bi_use_pae == 0 && high_phys > FOUR_GIG)
 215                 high_phys = FOUR_GIG;
 216 #endif

 218         /*
 219          * find the highest available memory in physinstalled
 220          */
 221         size = P2ROUNDUP(size, align);
 222         for (; ml; ml = ml->ml_next) {
 223                 start = P2ROUNDUP(ml->ml_address, align);
 224                 end = P2ALIGN(ml->ml_address + ml->ml_size, align);
 225                 if (start < next_phys)
 226                         start = P2ROUNDUP(next_phys, align);
 227                 if (end > high_phys)
 228                         end = P2ALIGN(high_phys, align);

 230                 if (end <= start)
 231                         continue;
 232                 if (end - start < size)
 233                         continue;

 235                 /*
 236                  * Early allocations need to use low memory, since
 237                  * physmem might be further limited by bootenv.rc
 238                  */
 239                 if (early_allocation) {
 240                         if (pa == 0 || start < pa)
 241                                 pa = start;
 242                 } else {
 243                         if (end - size > pa)
 244                                 pa = end - size;
```

```
 245                 }
 246         }
 247         if (pa != 0) {
 248                 if (early_allocation)
 249                         next_phys = pa + size;
 250                 else
 251                         high_phys = pa;
 252                 return (pa);
 253         }
 254         bop_panic("do_bop_phys_alloc(0x%" PRIx64 ", 0x%" PRIx64
 255             ") Out of memory\n", size, align);
 256         /*NOTREACHED*/
 257 }
```

*unchanged_portion_omitted_*

```
2514 /*
2515  * If this system has a PC-compatible BIOS, it will have handlers for
2516  * various well-known BIOS calls.  These calls take the form of INT
2517  * instructions, revectoring to the nominated entry in the real mode
2518  * Interrupt Vector Table (IVT).  If all of the commonly used entries (from
2519  * INT 10h up to INT 1Ah) are zero, we almost certainly don't want to make
2520  * use of BOP_DOINT() later.
2521  *
2522  * The IVT begins at linear address 0 on the 8086.  Though later CPUs
2523  * allowed it to be moved, it seems that most BIOS implementations choose
2524  * not to do so for compatibility reasons.  Our BIOS call trampoline (see
2525  * "idt_info" in "uts/i86pc/ml/bios_call_src.s") also assumes this address.
2526  */
2527 static int
2528 system_has_bios(void)
2529 {
2530         uint32_t all_ivts = 0;

2532         DBG_MSG("\nBIOS IVT Entries:\n");
2533         for (uint32_t intnum = 0x10; intnum <= 0x1a; intnum++) {
2534                 /*
2535                  * The first software interrupt number (i.e. INT 0h) maps to
2536                  * vector number 32 in the IVT.  Each entry in the IVT is
2537                  * four bytes, describing a 16 bit far call address.
2538                  */
2539                 uintptr_t slot = 4 * (32 + intnum);
2540                 uint32_t ivte = *((uint32_t *)slot);

2542                 if (ivte != 0) {
2543                         DBG(intnum);
2544                         DBG(ivte);
2545                 }
2546                 all_ivts |= ivte;
2547         }
2548         if (all_ivts == 0) {
2549                 DBG_MSG("System has no BIOS IVT entries\n");
2550         }
2551         DBG_MSG("\n");

2553         return (all_ivts != 0);
2554 }

2556 #else /* __xpv */
2557 static void
2558 enumerate_xen_cpus()
2559 {
2560         processorid_t   id, max_id;

2562         /*
2563          * User-set boot-ncpus overrides enumeration
2564          */
```



```
2565         if (do_bsys_getproplen(NULL, BOOT_NCPUS_NAME) >= 0)
2566                 return;

2568         /*
2569          * Probe every possible virtual CPU id and remember the
2570          * highest id present; the count of CPUs is one greater
2571          * than this.  This tacitly assumes at least cpu 0 is present.
2572          */
2573         max_id = 0;
2574         for (id = 0; id < MAX_VIRT_CPUS; id++)
2575                 if (HYPERVISOR_vcpu_op(VCPUOP_is_up, id, NULL) == 0)
2576                         max_id = id;

2578         bsetpropsi(BOOT_NCPUS_NAME, max_id+1);

2580 }
2581 #endif /* __xpv */

2583 static void
2584 build_firmware_properties(void)
2585 {
2586         ACPI_TABLE_HEADER *tp = NULL;

2588 #ifndef __xpv
2589         if (do_bsys_getproplen(NULL, "no-bios") > 0 || !system_has_bios())
2590                 bios_calls_available = B_FALSE;

2592         if ((tp = find_fw_table(ACPI_SIG_MSCT)) != NULL)
2593                 msct_ptr = process_msct((ACPI_TABLE_MSCT *)tp);
2594         else
2595                 msct_ptr = NULL;

2597         if ((tp = find_fw_table(ACPI_SIG_MADT)) != NULL)
2598                 process_madt((ACPI_TABLE_MADT *)tp);

2600         if ((srat_ptr = (ACPI_TABLE_SRAT *)
2601             find_fw_table(ACPI_SIG_SRAT)) != NULL)
2602                 process_srat(srat_ptr);

2604         if (slit_ptr = (ACPI_TABLE_SLIT *)find_fw_table(ACPI_SIG_SLIT))
2605                 process_slit(slit_ptr);

2607         tp = find_fw_table(ACPI_SIG_MCFG);
2608 #else /* __xpv */
2609         enumerate_xen_cpus();
2610         if (DOMAIN_IS_INITDOMAIN(xen_info))
2611                 tp = find_fw_table(ACPI_SIG_MCFG);
2612 #endif /* __xpv */
2613         if (tp != NULL)
2614                 process_mcfg((ACPI_TABLE_MCFG *)tp);
2615 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   14254 Mon Mar 27 00:11:48 2017
new/usr/src/uts/i86pc/os/mlsetup.c
XXX nobios
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright (c) 1993, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2011 by Delphix. All rights reserved.
 * Copyright 2016 Joyent, Inc.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

#include <sys/types.h>
#include <sys/sysmacros.h>
#include <sys/disp.h>
#include <sys/promif.h>
#include <sys/clock.h>
#include <sys/cpuvar.h>
#include <sys/stack.h>
#include <vm/as.h>
#include <vm/hat.h>
#include <sys/reboot.h>
#include <sys/avintr.h>
#include <sys/vtrace.h>
#include <sys/proc.h>
#include <sys/thread.h>
#include <sys/cpupart.h>
#include <sys/pset.h>
#include <sys/copyops.h>
#include <sys/pg.h>
#include <sys/disp.h>
#include <sys/debug.h>
#include <sys/sunddi.h>
#include <sys/x86_archext.h>
#include <sys/privregs.h>
#include <sys/machsystm.h>
#include <sys/ontrap.h>
#include <sys/bootconf.h>
#include <sys/boot_console.h>
#include <sys/kdi_machimpl.h>
#include <sys/archsystm.h>
```



```
#include <sys/promif.h>
#include <sys/pci_cfgspace.h>
#include <sys/bootvfs.h>
#include <sys/tsc.h>
#ifdef __xpv
#include <sys/hypervisor.h>
#else
#include <sys/xpv_support.h>
#endif

/*
 * some globals for patching the result of cpuid
 * to solve problems w/ creative cpu vendors
 */

extern uint32_t cpuid_feature_ecx_include;
extern uint32_t cpuid_feature_ecx_exclude;
extern uint32_t cpuid_feature_edx_include;
extern uint32_t cpuid_feature_edx_exclude;

/*
 * Set console mode
 */
static void
set_console_mode(uint8_t val)
{
        struct bop_regs rp = {0};

        if (!bios_calls_available)
                return;

        rp.eax.byte.ah = 0x0;
        rp.eax.byte.al = val;
        rp.ebx.word.bx = 0x0;

        BOP_DOINT(bootops, 0x10, &rp);
}
```

*unchanged_portion_omitted_*

```
**********************************************************
    5862 Mon Mar 27 00:11:48 2017
new/usr/src/uts/i86pc/os/pci_bios.c
XXX nobios
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/sunndi.h>
#include <sys/pci.h>
#include <sys/pci_impl.h>
#include <sys/pci_cfgspace.h>
#include <sys/pci_cfgspace_impl.h>
#include <sys/memlist.h>
#include <sys/bootconf.h>
#include <sys/psw.h>
#include <sys/machsystm.h>

/*
 * pci irq routing information table
 */
int                             pci_irq_nroutes;
static pci_irq_route_t          *pci_irq_routes;


static int pci_bios_get_irq_routing(pci_irq_route_t *, int, int *);
static void pci_get_irq_routing_table(void);


/*
 * Retrieve information from the bios needed for system
 * configuration early during startup.
 */
void
startup_pci_bios(void)
{
        pci_get_irq_routing_table();
}


/*
 * Issue the bios get irq routing information table interrupt
 *
```



```
 * Despite the name, the information in the table is only
 * used to derive slot names for some named pci hot-plug slots.
 *
 * Returns the number of irq routing table entries returned
 * by the bios, or 0 and optionally, the number of entries required.
 */
static int
pci_bios_get_irq_routing(pci_irq_route_t *routes, int nroutes, int *nneededp)
{
        struct bop_regs regs;
        uchar_t         *hdrp;
        uchar_t         *bufp;
        int             i, n;
        int             rval = 0;

        if (nneededp)
                *nneededp = 0;

        /*
         * If this system does not support BIOS calls, we can't use this
         * mechanism.
         */
        if (!bios_calls_available)
                return (0);

        /*
         * Set up irq routing header with the size and address
         * of some useable low-memory data addresses.  Initalize
         * data area to zero, avoiding memcpy/bzero.
         */
        hdrp = (uchar_t *)BIOS_IRQ_ROUTING_HDR;
        bufp = (uchar_t *)BIOS_IRQ_ROUTING_DATA;

        n = nroutes * sizeof (pci_irq_route_t);
        for (i = 0; i < n; i++)
                bufp[i] = 0;
        ((pci_irq_route_hdr_t *)hdrp)->pir_size = n;
        ((pci_irq_route_hdr_t *)hdrp)->pir_addr = (uint32_t)(uintptr_t)bufp;

        bzero(&regs, sizeof (regs));
        regs.eax.word.ax = (PCI_FUNCTION_ID << 8) | PCI_GET_IRQ_ROUTING;

        regs.ds = 0xf000;
        regs.es = FP_SEG((uint_t)(uintptr_t)hdrp);
        regs.edi.word.di = FP_OFF((uint_t)(uintptr_t)hdrp);

        BOP_DOINT(bootops, 0x1a, &regs);

        n = (int)(((pci_irq_route_hdr_t *)hdrp)->pir_size /
            sizeof (pci_irq_route_t));

        if ((regs.eflags & PS_C) != 0) {
                if (nneededp)
                        *nneededp = n;
        } else {
                /*
                 * Copy resulting irq routing data from low memory up to
                 * the kernel address space, avoiding memcpy as usual.
                 */
                if (n <= nroutes) {
                        for (i = 0; i < n * sizeof (pci_irq_route_t); i++)
                                ((uchar_t *)routes)[i] = bufp[i];
                        rval = n;
                }
        }
        return (rval);
```

```
 128 }
______unchanged_portion_omitted_
```

```
**********************************************************
    8846 Mon Mar 27 00:11:49 2017
new/usr/src/uts/i86pc/os/pci_cfgspace.c
XXX nobios
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 * PCI configuration space access routines
 */

#include <sys/systm.h>
#include <sys/psw.h>
#include <sys/bootconf.h>
#include <sys/reboot.h>
#include <sys/pci_impl.h>
#include <sys/pci_cfgspace.h>
#include <sys/pci_cfgspace_impl.h>
#include <sys/pci_cfgacc.h>
#include <sys/machsystm.h>
#if defined(__xpv)
#include <sys/hypervisor.h>
#endif

#if defined(__xpv)
int pci_max_nbus = 0xFE;
#endif
int pci_bios_cfg_type = PCI_MECHANISM_UNKNOWN;
int pci_bios_maxbus;
int pci_bios_mech;
int pci_bios_vers;

/*
 * These two variables can be used to force a configuration mechanism or
 * to force which function is used to probe for the presence of the PCI bus.
 */
int     PCI_CFG_TYPE = 0;
int     PCI_PROBE_TYPE = 0;

/*
 * No valid mcfg_mem_base by default, and accessing pci config space
 * in mem-mapped way is disabled.
 */
```



```
uint64_t mcfg_mem_base = 0;
uint8_t mcfg_bus_start = 0;
uint8_t mcfg_bus_end = 0xff;

/*
 * Maximum offset in config space when not using MMIO
 */
uint_t pci_iocfg_max_offset = 0xff;

/*
 * These function pointers lead to the actual implementation routines
 * for configuration space access.  Normally they lead to either the
 * pci_mech1_* or pci_mech2_* routines, but they can also lead to
 * routines that work around chipset bugs.
 * These functions are accessing pci config space via I/O way.
 * Pci_cfgacc_get/put functions shoul be used as more common interfaces,
 * which also provide accessing pci config space via mem-mapped way.
 */
uint8_t (*pci_getb_func)(int bus, int dev, int func, int reg);
uint16_t (*pci_getw_func)(int bus, int dev, int func, int reg);
uint32_t (*pci_getl_func)(int bus, int dev, int func, int reg);
void (*pci_putb_func)(int bus, int dev, int func, int reg, uint8_t val);
void (*pci_putw_func)(int bus, int dev, int func, int reg, uint16_t val);
void (*pci_putl_func)(int bus, int dev, int func, int reg, uint32_t val);

extern void (*pci_cfgacc_acc_p)(pci_cfgacc_req_t *req);

/*
 * Internal routines
 */
static int pci_check(void);

#if !defined(__xpv)
static int pci_check_bios(void);
static int pci_get_cfg_type(void);
#endif

/* for legacy io-based config space access */
kmutex_t pcicfg_mutex;

/* for mmio-based config space access */
kmutex_t pcicfg_mmio_mutex;

/* ..except Orion and Neptune, which have to have their own */
kmutex_t pcicfg_chipset_mutex;

void
pci_cfgspace_init(void)
{
        mutex_init(&pcicfg_mutex, NULL, MUTEX_SPIN,
            (ddi_iblock_cookie_t)ipltospl(15));
        mutex_init(&pcicfg_mmio_mutex, NULL, MUTEX_SPIN,
            (ddi_iblock_cookie_t)ipltospl(DISP_LEVEL));
        mutex_init(&pcicfg_chipset_mutex, NULL, MUTEX_SPIN,
            (ddi_iblock_cookie_t)ipltospl(15));
        if (!pci_check()) {
                mutex_destroy(&pcicfg_mutex);
                mutex_destroy(&pcicfg_mmio_mutex);
                mutex_destroy(&pcicfg_chipset_mutex);
        }
}
______unchanged_portion_omitted_

#if !defined(__xpv)

static int
```

```
249 pci_check_bios(void)
250 {
251         struct bop_regs regs;
252         uint32_t        carryflag;
253         uint16_t        ax, dx;

255         if (!bios_calls_available) {
256                 /*
257                  * If this system does not support BIOS calls, we must fall
258                  * back to default values and a search of all of the possible
259                  * PCI buses.
260                  */
261                 pci_bios_mech = 1;
262                 pci_bios_vers = 0;
263                 pci_bios_maxbus = pci_max_nbus;
264                 return (PCI_MECHANISM_1);
265         }

267         bzero(&regs, sizeof (regs));
268         regs.eax.word.ax = (PCI_FUNCTION_ID << 8) | PCI_BIOS_PRESENT;

270         BOP_DOINT(bootops, 0x1a, &regs);
271         carryflag = regs.eflags & PS_C;
272         ax = regs.eax.word.ax;
273         dx = regs.edx.word.dx;

275         /* the carry flag must not be set */
276         if (carryflag != 0)
277                 return (PCI_MECHANISM_NONE);

279         if (dx != ('P' | 'C'<<8))
280                 return (PCI_MECHANISM_NONE);

282         /* ah (the high byte of ax) must be zero */
283         if ((ax & 0xff00) != 0)
284                 return (PCI_MECHANISM_NONE);

286         pci_bios_mech = (ax & 0x3);
287         pci_bios_vers = regs.ebx.word.bx;
288         pci_bios_maxbus = (regs.ecx.word.cx & 0xff);

290         switch (pci_bios_mech) {
291         default:        /* ?!? */
292         case 0:         /* supports neither? */
293                 return (PCI_MECHANISM_NONE);

295         case 1:
296         case 3:         /* supports both */
297                 return (PCI_MECHANISM_1);

299         case 2:
300                 return (PCI_MECHANISM_2);
301         }
302 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    6588 Mon Mar 27 00:11:49 2017
new/usr/src/uts/i86pc/sys/machsystm.h
XXX nobios
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 1993, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

#ifndef _SYS_MACHSYSTM_H
#define _SYS_MACHSYSTM_H

/*
 * Numerous platform-dependent interfaces that don't seem to belong
 * in any other header file.
 *
 * This file should not be included by code that purports to be
 * platform-independent.
 *
 */

#include <sys/machparam.h>
#include <sys/varargs.h>
#include <sys/thread.h>
#include <sys/cpuvar.h>
#include <sys/privregs.h>
#include <sys/systm.h>
#include <sys/traptrace.h>
#include <vm/page.h>

#ifdef __cplusplus
extern "C" {
#endif

#ifdef _KERNEL

typedef enum mach_cpu_add_arg_type {
        MACH_CPU_ARG_LOCAL_APIC,
        MACH_CPU_ARG_LOCAL_X2APIC,
} mach_cpu_add_arg_type_t;
______unchanged_portion_omitted_
```



```
/* Maximum physical page number (PFN) for memory DR operations. */
extern uint64_t plat_dr_physmax;

#ifdef __xpv
#include <sys/xen_mmu.h>
extern page_t *page_get_high_mfn(mfn_t);
#endif

extern hrtime_t tsc_gethrtime_tick_delta(void);

extern boolean_t bios_calls_available;

#endif /* _KERNEL */

#ifdef __cplusplus
}
______unchanged_portion_omitted_
```