****
554 Wed Mar 18 09:16:44 2015
new/usr/src/uts/armv7/bcm2836/os/bcm2836_bsmdep.c
cpuid for ARMv7
****

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2014 Joyent, Inc.  All rights reserved.
 */

/*
 * Broadcom 2836 board specific functions.
 */

#include <vm/vm_dep.h>

void
armv7_bsmdep_l2cacheinfo(void)
{
        /*
         * Per the BCM 2836 ARM peripherals manual, the L2 cache on the BCM
         * 2836 is actually used by the GPU, and from the CPU point of view,
         * we don't have one.
         *
         * This can be toggled on the rPi, but it appears that that toggling
         * can't be probed for.
         *
         * At present, we set these variables as if we owned the l2,
         * regardless of whether we in fact do.  This might be a terrible
         * idea.
         *
         * XXX: It might be reasonable to demand that we (the CPU) have the l2
         * cache, although since it is off-chip this may actually hinder
         * performance.
         */

        armv6_l2cache_linesz = 32;
        /* 128K per the BCM2836 manual, though we by default see none of it */
        armv6_l2cache_size = 0x20000;
}
```

****
5278 Wed Mar 18 09:16:44 2015
new/usr/src/uts/armv7/ml/cache.s
cpuid for ARMv7
****
______unchanged_portion_omitted_

```
#endif  /* __lint */

#if defined(lint) || defined(__lint)

/*
 * The ARM architecture uses a modified Harvard Architecture which means that we
/* The ARM architecture uses a modified Harvard Architecture which means that we
 * get the joys of fixing up this mess. Primarily this means that when we update
 * data, it gets written to do the data cache. That needs to be flushed to main
 * memory and then the instruction cache needs to be invalidated. This is
 * particularly important for things like krtld and DTrace. While the data cache
 * does write itself out over time, we cannot rely on it having written itself
 * out to the state that we care about by the time that we'd like it to. As
 * such, we need to ensure that it's been flushed out ourselves. This also means
 * that we could accidentally flush a region of the icache that's already
 * updated itself, but that's just what we have to do to keep Von Neumann's
 * spirt and great gift alive.
 *
 * The controllers for the caches have a few different options for invalidation.
 * One may:
 *
 *   o Invalidate or flush the entire cache
 *   o Invalidate or flush a cache line
 *   o Invalidate or flush a cache range
 *
 * We opt to take the third option here for the general case of making sure that
 * text has been synchronized. While the data cache allows us to both invalidate
 * and flush the cache line, we don't currently have a need to do the
 * invalidation.
 *
 * Note that all of these operations should be aligned on an 8-byte boundary.
 * The instructions actually only end up using bits [31:5] of an address.
 * Callers are required to ensure that this is the case.
 */

void
armv7_icache_disable(void)
{}

void
armv7_icache_enable(void)
{}

void
armv7_dcache_disable(void)
{}

void
armv7_dcache_enable(void)
{}

void
armv7_icache_inval(void)
{}

void
armv7_dcache_inval(void)
{}
```

```
void
armv7_dcache_flush(void)
{}

void
armv7_text_flush_range(caddr_t start, size_t len)
{}

void
armv7_text_flush(void)
{}

#else   /* __lint */

        ENTRY(armv7_icache_enable)
        mrc     p15, 0, r0, c1, c0, 0
        orr     r0, #0x1000
        mcr     p15, 0, r0, c1, c0, 0
        SET_SIZE(armv7_icache_enable)
```

______unchanged_portion_omitted_

```
***********************************************************
    2922 Wed Mar 18 09:16:44 2015
new/usr/src/uts/armv7/ml/cpuid_ml.s
cpuid for ARMv7
***********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2014 Joyent, Inc.  All rights reserved.
 */

        .file   "cpuid.s"

/*
 * Read cpuid values from coprocessors
 */

#include <sys/asm_linkage.h>

#if defined(lint) || defined(__lint)

uint32_t
arm_cpuid_midr()
arm_cpuid_idreg()
{}

uint32_t
arm_cpuid_pfr0()
{}

uint32_t
arm_cpuid_pfr1()
{}

uint32_t
arm_cpuid_dfr0()
{}

uint32_t
arm_cpuid_mmfr0()
{}

uint32_t
arm_cpuid_mmfr1()
{}

uint32_t
arm_cpuid_mmfr2()
{}

uint32_t
arm_cpuid_mmfr3()
{}

uint32_t
arm_cpuid_isar0()
{}
```

```
uint32_t
arm_cpuid_isar1()
{}

uint32_t
arm_cpuid_isar2()
{}

uint32_t
arm_cpuid_isar3()
{}

uint32_t
arm_cpuid_isar4()
{}

uint32_t
arm_cpuid_isar5()
{}

uint32_t
arm_cpuid_vfpidreg()
{}

uint32_t
arm_cpuid_mvfr0()
{}

uint32_t
arm_cpuid_mvfr1()
{}

uint32_t
arm_cpuid_ctr()
{}

#else   /* __lint */

        ENTRY(arm_cpuid_midr)
        ENTRY(arm_cpuid_idreg)
        mrc     p15, 0, r0, c0, c0, 0
        bx      lr
        SET_SIZE(arm_cpuid_midr)
        SET_SIZE(arm_cpuid_idreg)

        ENTRY(arm_cpuid_pfr0)
        mrc     p15, 0, r0, c0, c1, 0
        bx      lr
        SET_SIZE(arm_cpuid_pfr0)
______unchanged_portion_omitted_
#endif /* __lint */
#endif /* ! codereview */

        ENTRY(arm_cpuid_clidr)
        mrc     p15, 1, r0, c0, c0, 1
        ENTRY(arm_cpuid_ctr)
        mrc     p15, 0, r0, c0, c0, 1
        bx      lr
        SET_SIZE(arm_cpuid_clidr)

        ENTRY(arm_cpuid_ccsidr)
        lsl     r0, r0, #1
        cmp     r1, #0                          /* icache == B_FALSE */
        orrne   r0, r0, #1
        mcr     p15, 2, r0, c0, c0, 0           /* write CSSELR */
        mrc     p15, 1, r0, c0, c0, 0           /* read selected CCSIDR */
        bx      lr
        SET_SIZE(arm_cpuid_ccsidr)
        SET_SIZE(arm_cpuid_ctr)
#endif /* __lint */
```

```
**********************************************************
    1079 Wed Mar 18 09:16:44 2015
new/usr/src/uts/armv7/os/bsmdep.c
cpuid for ARMv7
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2014 Joyent, Inc.  All rights reserved.
 */
#include <sys/types.h>
#include <sys/time.h>

/*
 * Board Specific Module dependencies.
 */

/*
 * In addition to the entry points defined below, a board is also required to
 * implement the following functions:
 *
 * void armv6_bsmdep_l2cacheinfo(void);
 *
 *      The board should set the value of 'armv6_l2cache_linesz'
 *
 * XXX Some day we should make all of this into modules that can be loaded early
 * by unix so that way we can have one kernel for all boards...
 */

/*
 * While we would like to have a single consistent hrtime function across all of
 * the ARMv7 implementations, the chip itself leaves us rather lacking. As such,
 * the ARMv6 implementations, the chip itself leaves us rather lacking. As such,
 * we have to rely on each ARM board or implementation to do the work for us,
 * alas.
 */
static hrtime_t
dummy_hrtime(void)
{
        return (0);
}
______unchanged_portion_omitted_
```

```
**********************************************************
    7198 Wed Mar 18 09:16:45 2015
new/usr/src/uts/armv7/os/cpuid.c
cpuid for ARMv7
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2014 Joyent, Inc.  All rights reserved.
 * Copyright (c) 2015 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
#endif /* ! codereview */
 */

#include <sys/cpuid_impl.h>
#include <sys/param.h>
#include <sys/bootconf.h>
#include <vm/vm_dep.h>
#include <sys/armv7_bsmf.h>

/*
 * Handle classification and identification of ARM processors.
 *
 * Currently we do a single pass which reads in information and asserts that the
 * basic information which we receive here matches what we'd expect and are able
 * to do everything that we need with this ARM CPU.
 *
 * TODO We'll eventually do another pass to make sure that we properly determine
 * the feature set to expose to userland.
 */

static arm_cpuid_t cpuid_data0;

static uint32_t
extract(uint32_t line, uint32_t mask, uint32_t shift)
static void
cpuid_parse_stage(uint32_t line, uint32_t mask, uint32_t shift, int *out)
{
        return ((line & mask) >> shift);
        *out = (line & mask) >> shift;
}
______unchanged_portion_omitted_

#define CCSIDR_WT               0x80000000
#define CCSIDR_WB               0x40000000
#define CCSIDR_RA               0x20000000
#define CCSIDR_WA               0x10000000
#define CCSIDR_NUMSETS_MASK     0x0fffe000
#define CCSIDR_NUMSETS_SHIFT    13
#define CCSIDR_ASSOC_MASK       0x00001ff8
#define CCSIDR_ASSOC_SHIFT      3
#define CCSIDR_LINESIZE_MASK    0x00000007
#define CCSIDR_LINESIZE_SHIFT   0

static void
cpuid_fill_onecache(arm_cpuid_t *cpd, int level, boolean_t icache)
{
        arm_cpuid_cache_t *cache = &cpd->ac_caches[icache][level];
```

```
        uint32_t ccsidr;

        ccsidr = arm_cpuid_ccsidr(level, icache);
        cpd->ac_ccsidr[icache][level] = ccsidr;

        cache->acc_exists = B_TRUE;
        cache->acc_wt = (ccsidr & CCSIDR_WT) == CCSIDR_WT;
        cache->acc_wb = (ccsidr & CCSIDR_WB) == CCSIDR_WB;
        cache->acc_ra = (ccsidr & CCSIDR_RA) == CCSIDR_RA;
        cache->acc_wa = (ccsidr & CCSIDR_WA) == CCSIDR_WA;
        cache->acc_sets = extract(ccsidr, CCSIDR_NUMSETS_MASK,
            CCSIDR_NUMSETS_SHIFT) + 1;
        cache->acc_assoc = extract(ccsidr, CCSIDR_ASSOC_MASK,
            CCSIDR_ASSOC_SHIFT) + 1;
        cache->acc_linesz = sizeof (uint32_t) << (extract(ccsidr,
            CCSIDR_LINESIZE_MASK, CCSIDR_LINESIZE_SHIFT) + 2);
#define CACHE_LEN_MASK          0x003
#define CACHE_M_BIT             0x004
#define CACHE_ASSOC_MASK        0x038
#define CACHE_ASSOC_SHIFT       3
#define CACHE_SIZE_MASK         0x3c0
#define CACHE_SIZE_SHIFT        6
#define CACHE_COLOR_BIT         0x800
#define CACHE_MASK              0xfff
#define CACHE_DCACHE_SHIFT      12
#define CACHE_SEPARATE          0x1000000

/*
 * XXX?
#warning "set acc_size?"
 * On ARMv6 the value of the cache size and the cache associativity depends on
 * the value of the M bit, which modifies the value that's in the actual index.
 */
static uint32_t armv6_cpuid_cache_sizes[2][9] = {
        { 0x200, 0x400, 0x800, 0x1000, 0x2000, 0x4000, 0x8000, 0x100000,
                0x20000 },
        { 0x300, 0x600, 0xc00, 0x1800, 0x3000, 0x6000, 0xc000, 0x18000,
                0x30000 }
};

static int8_t armv6_cpuid_cache_assoc[2][9] = {
        { 1, 2, 4, 8, 16, 32, 64, 128 },
        { -1, 3, 6, 12, 24, 48, 96, 192 }
};

static uint8_t armv6_cpuid_cache_linesz[] = {
        8,
        16,
        32,
        64
};

static void
cpuid_fill_onecache(arm_cpuid_cache_t *accp, uint32_t val)
{
        int mbit, index, assoc;

        mbit = (val & CACHE_M_BIT) != 0 ? 1 : 0;
        index = (val & CACHE_ASSOC_MASK) >> CACHE_ASSOC_SHIFT;
        assoc = armv6_cpuid_cache_assoc[mbit][index];
        if (assoc == -1) {
                accp->acc_exists = B_FALSE;
                return;
        }
        ASSERT(assoc > 0);
        accp->acc_assoc = assoc;
        accp->acc_rcolor = (val & CACHE_COLOR_BIT) == 0 ?
            B_FALSE : B_TRUE;
        index = val & CACHE_LEN_MASK;
        accp->acc_linesz = armv6_cpuid_cache_linesz[index];
        index = (val & CACHE_SIZE_MASK) >> CACHE_SIZE_SHIFT;
        accp->acc_size = armv6_cpuid_cache_sizes[mbit][index];
        accp->acc_exists = B_TRUE;
}

static void
cpuid_fill_caches(arm_cpuid_t *cpd)
{
        uint32_t erg, cwg;
        uint32_t l1ip;
        uint32_t ctr;
        uint32_t clidr;
        int level;

        clidr = arm_cpuid_clidr();
        cpd->ac_clidr = clidr;

        /* default all caches to not existing, and not unified */
        for (level = 0; level < 7; level++) {
                cpd->ac_caches[B_TRUE][level].acc_exists = B_FALSE;
                cpd->ac_caches[B_FALSE][level].acc_exists = B_FALSE;
                cpd->ac_caches[B_TRUE][level].acc_unified = B_FALSE;
                cpd->ac_caches[B_FALSE][level].acc_unified = B_FALSE;
        }

        /* retrieve cache info for each level */
        for (level = 0; level < 7; level++) {
                arm_cpuid_cache_t *icache = &cpd->ac_caches[B_TRUE][level];
                arm_cpuid_cache_t *dcache = &cpd->ac_caches[B_FALSE][level];
                uint32_t ctype = (cpd->ac_clidr >> (3 * level)) & 0x7;

                /* stop looking we find the first non-existent level */
                if (!ctype)
                        break;

                switch (ctype) {
                case 1:
                        cpuid_fill_onecache(cpd, level, B_TRUE);
                        break;
                case 2:
                        cpuid_fill_onecache(cpd, level, B_FALSE);
                        break;
                case 3:
                        cpuid_fill_onecache(cpd, level, B_TRUE);
                        cpuid_fill_onecache(cpd, level, B_FALSE);
                        break;
                case 4:
                        cpuid_fill_onecache(cpd, level, B_FALSE);
                        dcache->acc_unified = B_TRUE;
                        break;
                default:
                        bop_panic("unsupported cache type");
                }
        }
        uint32_t val, icache, dcache;

        /*
         * We require L1-I/D & L2-D.  Unified caches are OK as well.
         */
        if (!cpd->ac_caches[B_TRUE][0].acc_exists &&
            (!cpd->ac_caches[B_FALSE][0].acc_exists ||
            !cpd->ac_caches[B_FALSE][0].acc_unified))
```

```
                bop_panic("no L1 instructian cache detected");

        if (!cpd->ac_caches[B_FALSE][1].acc_exists)
                bop_panic("no L2 data cache detected");

        /*
         * set globals with cache size info
         */
        l2cache_sz = cpd->ac_caches[B_FALSE][1].acc_size;
        l2cache_linesz = cpd->ac_caches[B_FALSE][1].acc_linesz;
        l2cache_assoc = cpd->ac_caches[B_FALSE][1].acc_assoc;
        val = arm_cpuid_ctr();
        icache = val & CACHE_MASK;
        cpuid_fill_onecache(&cpd->ac_icache, icache);
        dcache = (val >> CACHE_DCACHE_SHIFT) & CACHE_MASK;
        cpuid_fill_onecache(&cpd->ac_dcache, dcache);

        if (val & CACHE_SEPARATE) {
                cpd->ac_unifiedl1 = B_FALSE;
        } else {
                cpd->ac_unifiedl1 = B_TRUE;
        }

        armv7_bsmdep_l2cacheinfo();
        armv6_cachesz = cpd->ac_dcache.acc_size;
        armv6_cache_assoc = cpd->ac_dcache.acc_assoc;
}


/*
 * We need to do is go through and check for a few features that we know
 * we're going to need.
 * There isn't a specific way to indicate that we're on ARMv6k. Instead what we
 * need to do is go through and check for a few features that we know we're
 * going to need.
 *
 * TODO This will have to be revisited with ARMv7 support
 */
static void
cpuid_verify(void)
{
        arm_cpuid_mem_vmsa_t vmsa;
        arm_cpuid_mem_barrier_t barrier;
        int sync, syncf;

        arm_cpuid_t *cpd = &cpuid_data0;

        /* v7 vmsa */
        vmsa = extract(cpd->ac_mmfr[0], ARM_CPUID_MMFR0_STATE0_MASK,
            ARM_CPUID_MMFR0_STATE0_SHIFT);
        /* v6 vmsa */
        cpuid_parse_stage(cpd->ac_mmfr[0], ARM_CPUID_MMFR0_STATE0_MASK,
            ARM_CPUID_MMFR0_STATE0_SHIFT, (int *)&vmsa);
        /* TODO We might be able to support v6, but bcm2835+qvpb are this */
        if (vmsa != ARM_CPUID_MEM_VMSA_V7) {
                bop_printf(NULL, "invalid vmsa setting, found 0x%x\n", vmsa);
                bop_panic("unsupported cpu");
        }

        /* check for ISB, DSB, etc. in cp15 */
        barrier = extract(cpd->ac_mmfr[2], ARM_CPUID_MMFR2_STATE5_MASK,
            ARM_CPUID_MMFR2_STATE5_SHIFT);
        if (barrier != ARM_CPUID_MEM_BARRIER_INSTR) {
                bop_printf(NULL, "missing support for memory barrier "
                    "instructions\n");
        cpuid_parse_stage(cpd->ac_mmfr[2], ARM_CPUID_MMFR2_STATE5_MASK,
            ARM_CPUID_MMFR2_STATE5_SHIFT, (int *)&barrier);
        if (barrier != ARM_CPUID_MEM_BARRIER_CP15 &&
            barrier != ARM_CPUID_MEM_BARRIER_INSTR) {
                bop_printf(NULL, "missing support for CP15 memory barriers\n");
                bop_panic("unsupported CPU");
        }

        /* synch prims */
        sync = extract(cpd->ac_isar[3], ARM_CPUID_ISAR3_STATE3_SHIFT,
            ARM_CPUID_ISAR3_STATE3_SHIFT);
        syncf = extract(cpd->ac_isar[4], ARM_CPUID_ISAR4_STATE5_SHIFT,
            ARM_CPUID_ISAR4_STATE5_SHIFT);
        cpuid_parse_stage(cpd->ac_isar[4], ARM_CPUID_ISAR3_STATE3_SHIFT,
            ARM_CPUID_ISAR4_STATE5_SHIFT, (int *)&sync);
        cpuid_parse_stage(cpd->ac_isar[4], ARM_CPUID_ISAR4_STATE3_SHIFT,
            ARM_CPUID_ISAR4_STATE5_SHIFT, (int *)&syncf);
        if (sync != 0x2 && syncf != 0x0) {
                bop_printf(NULL, "unsupported synch primitives: sync,frac: "
                    "%x,%x\n", sync, syncf);
                bop_panic("unsupported CPU");
        }

        if (cpd->ac_icache.acc_exists == B_FALSE) {
                bop_printf(NULL, "icache not defined to exist\n");
                bop_panic("unsupported CPU");
        }

        if (cpd->ac_dcache.acc_exists == B_FALSE) {
                bop_printf(NULL, "dcache not defined to exist\n");
                bop_panic("unsupported CPU");
        }
}

static void
cpuid_valid_ident(uint32_t ident)
{
        arm_cpuid_ident_arch_t arch;

        /*
         * We don’t support anything older than ARMv7.
         * We don’t support stock ARMv6 or older.
         */
        arch = (ident & ARM_CPUID_IDENT_ARCH_MASK) >>
            ARM_CPUID_IDENT_ARCH_SHIFT;
        if (arch != ARM_CPUID_IDENT_ARCH_CPUID) {
                bop_printf(NULL, "encountered unsupported CPU arch: 0x%x",
                    arch);
                bop_panic("unsupported CPU");
        }
}

static void
cpuid_valid_fpident(uint32_t ident)
{
        arm_cpuid_vfp_arch_t vfp;

        vfp = extract(ident, ARM_CPUID_VFP_ARCH_MASK, ARM_CPUID_VFP_ARCH_SHIFT);
        // XXX: _V3_V2BASE? _V3_NOBASE? _V3_V3BASE?
        cpuid_parse_stage(ident, ARM_CPUID_VFP_ARCH_MASK,
            ARM_CPUID_VFP_ARCH_SHIFT, (int *)&vfp);
        if (vfp != ARM_CPUID_VFP_ARCH_V2) {
                bop_printf(NULL, "unsupported vfp version: %x\n", vfp);
                bop_panic("unsupported CPU");
        }

        if ((ident & ARM_CPUID_VFP_SW_MASK) != 0) {
```

```
                bop_printf(NULL, "encountered software-only vfp\n");
                bop_panic("unsuppored CPU");
        }
}
void
cpuid_setup(void)
{
        arm_cpuid_t *cpd = &cpuid_data0;

        cpd->ac_ident = arm_cpuid_midr();
        cpd->ac_ident = arm_cpuid_idreg();
        cpuid_valid_ident(cpd->ac_ident);
        cpuid_fill_main(cpd);

        cpd->ac_fpident = arm_cpuid_vfpidreg();
        cpuid_valid_fpident(cpd->ac_fpident);
        cpuid_fill_fpu(cpd);

#endif /* ! codereview */
        cpuid_fill_caches(cpd);

        cpuid_verify();
}
```

```
**********************************************************
    4146 Wed Mar 18 09:16:45 2015
new/usr/src/uts/armv7/os/startup.c
cpuid for ARMv7
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2014 Joyent, Inc.  All rights reserved.
 */

#include <sys/types.h>
#include <sys/bootconf.h>
#include <sys/obpdefs.h>
#include <sys/promif.h>

/*
 *      32-bit Kernel's Virtual memory layout.
 *               +-----------------------+
 *               |    exception table    |
 * 0xFFFF0000  -|-----------------------|- EXCEPTION_ADDRESS
 *               |                       |
 * 0xFFC00000  -|-----------------------|- ARGSBASE
 *               |   XXX  debugger?      |
 * 0xFF800000  -|-----------------------|- XXX SEGDEBUBBASE?+
 *               |      Kernel Data      |
 * 0xFEC00000  -|-----------------------|
 *               |      Kernel Text      |
 * 0xFE800000  -|-----------------------|- KERNEL_TEXT
 *               |                       |
 *               |    XXX No idea yet    |
 *               |                       |
 * 0xC8002000  -|-----------------------|- XXX segmap_start?
 *               |       Red Zone        |
 * 0xC8000000  -|-----------------------|- kernelbase / userlimit (floating)
 *               |      User Stack       |
 *               |                       |
 *               |                       |
 *
 *               :                       :
 *               |    shared objects     |
 *               :                       :
 *
 *               :                       :
 *               |       user data       |
 *              -|-----------------------|-
 *               |       user text       |
 * 0x00002000  -|-----------------------|-  XXX Not necessairily truetoday
 *               |       invalid         |
 * 0x00000000  -|-----------------------|-
 *
 * + Item does not exist at this time.
 */

struct bootops          *bootops = 0;   /* passed in from boot */
struct bootops          **bootopsp;
struct boot_syscalls    *sysp;          /* passed in from boot */
```

```
char kern_bootargs[OBP_MAXPATHLEN];
char kern_bootfile[OBP_MAXPATHLEN];

caddr_t s_text;         /* start of kernel text segment */
caddr_t e_text;         /* end of kernel text segment */
caddr_t s_data;         /* start of kernel data segment */
caddr_t e_data;         /* end of kernel data segment */
caddr_t modtext;        /* start of loadable module text reserved */
caddr_t e_modtext;      /* end of loadable module text reserved */
caddr_t moddata;        /* start of loadable module data reserved */
caddr_t e_moddata;      /* end of loadable module data reserved */

/*
 * Some CPUs have holes in the middle of the 64-bit virtual address range.
 */
uintptr_t hole_start, hole_end;

/*
 * PROM debugging facilities
 */
int prom_debug = 1;

/*
 * VM related data
 */
long page_hashsz;               /* Size of page hash table (power of two) */
unsigned int page_hashsz_shift; /* log2(page_hashsz) */
struct page *pp_base;           /* Base of initial system page struct array */
struct page **page_hash;        /* Page hash table */
pad_mutex_t *pse_mutex;         /* Locks protecting pp->p_selock */
size_t pse_table_size;          /* Number of mutexes in pse_mutex[] */
int pse_shift;                  /* log2(pse_table_size) */

/*
 * Cache size information filled in via cpuid.
 * Cache size information filled in via cpuid and startup_cache()
 */
int l2cache_sz;
int l2cache_linesz ;
int l2cache_assoc;
int armv6_cachesz;              /* Total size of the l1 cache */
int armv6_cache_assoc;          /* L1 cache associativity */
int armv6_l2cache_linesz;       /* Size of a line in the l2 cache */
int armv6_l2cache_size;         /* Total size of the l2 cache */

/*
 * Do basic set up.
 */
static void
startup_init()
{
        if (BOP_GETPROPLEN(bootops, "prom_debug") >= 0) {
                ++prom_debug;
                prom_printf("prom_debug found in boot enviroment");
        }
}
______unchanged_portion_omitted_
```

`**********************************************************`
563 Wed Mar 18 09:16:45 2015
new/usr/src/uts/armv7/qve/os/qve_bsmdep.c
cpuid for ARMv7
`**********************************************************`

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2014 Joyent, Inc.  All rights reserved.
 */

/*
 * QEMU Versatile Express board specific functions.
 */

#include <vm/vm_dep.h>

void
armv7_bsmdep_l2cacheinfo(void)
{
	/* Per L220 Cache Controller Technical Reference Manual */
	armv6_l2cache_linesz = 32;
	/* 128 Kb l2 cache, per DUI0425F */
	armv6_l2cache_size = 0x20000;
}
```

```
**********************************************************
     749 Wed Mar 18 09:16:45 2015
new/usr/src/uts/armv7/sys/armv7_bsmf.h
cpuid for ARMv7
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2014 (c) Joyent, Inc.  All rights reserved.
 */

#ifndef _SYS_ARMV7_BSMF_H
#define _SYS_ARMV7_BSMF_H

/*
 * This describes interfaces that unix can expect each of the board specific
 * modules to have implemented.
 */

#ifdef __cplusplus
extern "C" {
#endif

/*
 * The platform should fill in the values for armv7_l2cache_linesz and
 * armv7_l2cache_size.
 */
extern void armv7_bsmdep_l2cacheinfo(void);

#ifdef __cplusplus
}
______unchanged_portion_omitted_
```

```
**********************************************************
    2358 Wed Mar 18 09:16:45 2015
new/usr/src/uts/armv7/sys/cpuid_impl.h
cpuid for ARMv7
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2014 Joyent, Inc.  All rights reserved.
 */

#ifndef _SYS_CPUID_IMPL_H
#define _SYS_CPUID_IMPL_H

#include <sys/stdint.h>
#include <sys/arm_archext.h>
#include <sys/types.h>

/*
 * Routines to read ARM cpuid co-processors
 */

#ifdef __cplusplus
extern "C" {
#endif

typedef struct arm_cpuid_cache {
        boolean_t acc_exists;
        boolean_t acc_unified;
        boolean_t acc_wt;
        boolean_t acc_wb;
        boolean_t acc_ra;
        boolean_t acc_wa;
        uint16_t acc_sets;
        uint8_t acc_linesz;
        uint16_t acc_assoc;

#endif /* ! codereview */
        boolean_t acc_rcolor;
        uint8_t acc_assoc;
        uint8_t acc_linesz;
        uint32_t acc_size;
} arm_cpuid_cache_t;

typedef struct arm_cpuid {
        uint32_t ac_ident;
        uint32_t ac_pfr[2];
        uint32_t ac_dfr;
        uint32_t ac_mmfr[4];
        uint32_t ac_isar[6];
        uint32_t ac_fpident;
        uint32_t ac_mvfr[2];
        uint32_t ac_clidr;

        /*
         * ARM supports 7 levels of caches.  Each level can have separate
         * I/D caches or a unified cache.  We keep track of all these as a
```

```
         * two dimensional array.  First, we select if we're dealing with a
         * I cache (B_TRUE) or a D/unified cache (B_FALSE), and then we
         * index on the level.  Note that L1 caches are at index 0.
         */
        uint32_t ac_ccsidr[2][7];
        arm_cpuid_cache_t ac_caches[2][7];
        boolean_t ac_unifiedl1;
        arm_cpuid_cache_t ac_icache;
        arm_cpuid_cache_t ac_dcache;
} arm_cpuid_t;

extern uint32_t arm_cpuid_midr();
extern uint32_t arm_cpuid_idreg();
extern uint32_t arm_cpuid_pfr0();
extern uint32_t arm_cpuid_pfr1();
extern uint32_t arm_cpuid_dfr0();
extern uint32_t arm_cpuid_mmfr0();
extern uint32_t arm_cpuid_mmfr1();
extern uint32_t arm_cpuid_mmfr2();
extern uint32_t arm_cpuid_mmfr3();
extern uint32_t arm_cpuid_isar0();
extern uint32_t arm_cpuid_isar1();
extern uint32_t arm_cpuid_isar2();
extern uint32_t arm_cpuid_isar3();
extern uint32_t arm_cpuid_isar4();
extern uint32_t arm_cpuid_isar5();

extern uint32_t arm_cpuid_vfpidreg();
extern uint32_t arm_cpuid_mvfr0();
extern uint32_t arm_cpuid_mvfr1();

extern uint32_t arm_cpuid_clidr();
extern uint32_t arm_cpuid_ccsidr(uint32_t level, boolean_t icache);
extern uint32_t arm_cpuid_ctr();

#ifdef __cplusplus
}
______unchanged_portion_omitted_
```

```
**********************************************************
   14706 Wed Mar 18 09:16:45 2015
new/usr/src/uts/armv7/vm/vm_dep.h
cpuid for ARMv7
**********************************************************
______unchanged_portion_omitted_

extern void page_list_walk_init(uchar_t szc, uint_t flags, uint_t bin,
    int can_split, int use_ceq, page_list_walker_t *plw);

extern struct cpu       cpus[];
#define CPU0            &cpus[0]

/*
 * XXX memory type initializaiton
 */
#define MTYPE_INIT(mtype, vp, vaddr, flags, pgsz)       panic("mtype_init")
#define MTYPE_START(mnode, mtype, flags)        panic("mtype_start")
#define MTYPE_NEXT(mnode, mtype, flags) panic("mtype_next")
#define MTYPE_PGR_INIT(mtype, flags, pp, mnode, pgcnt)  panic("mtype_pgr_init")
#define MNODETYPE_2_PFN(mnode, mtype, pfnlo, pfnhi)     panic("mnodetype_2_pfn")

#ifdef DEBUG
#define CHK_LPG(pp, szc)        panic("chk_lpg")
#else
#define CHK_LPG(pp, szc)
#endif

#define FULL_REGION_CNT(rg_szc) \
        (PAGE_GET_SIZE(rg_szc) >> PAGE_GET_SHIFT(rg_szc - 1))

/* Return the leader for this mapping size */
#define PP_GROUPLEADER(pp, szc) \
        (&(pp)[-(int)((pp)->p_pagenum & (SZCPAGES(szc)-1))])

/* Return the root page for this page based on p_szc */
#define PP_PAGEROOT(pp) ((pp)->p_szc == 0 ? (pp) : \
        PP_GROUPLEADER((pp), (pp)->p_szc))

/*
 * The counter base must be per page_counter element to prevent
 * races when re-indexing, and the base page size element should
 * be aligned on a boundary of the given region size.
 *
 * We also round up the number of pages spanned by the counters
 * for a given region to PC_BASE_ALIGN in certain situations to simplify
 * the coding for some non-performance critical routines.
 */
#define PC_BASE_ALIGN           ((pfn_t)1 << PAGE_BSZS_SHIFT(mmu_page_sizes-1))
#define PC_BASE_ALIGN_MASK      (PC_BASE_ALIGN - 1)

/*
 * The following three constants describe the set of page sizes that are
 * supported by the hardware. Note that there is a notion of legacy page sizes
 * for certain applications. However, such applications don't exist on ARMv7, so
 * they'll always get the same data.
 */
extern uint_t mmu_page_sizes;
extern uint_t mmu_exported_page_sizes;
extern uint_t mmu_legacy_page_sizes;

/*
 * These macros are used for converting between userland page sizes and kernel
 * page sizes. However, these are the same on ARMv7 (just like i86pc).
 */
#define USERSZC_2_SZC(userszc)  userszc
```

```
#define SZC_2_USERSZC(szc)      szc

/*
 * for hw_page_map_t, sized to hold the ratio of large page to base
 * pagesize
 */
typedef short   hpmctr_t;

/*
 * get the setsize of the current cpu
 *
 * This is complicated by the fact that the I-cache and D-cache may be
 * separate.
 * On ARMv6 the layer two cache isn't architecturally defined. A given
 * implementation may or may not support it. The maximum size appears to be
 * 64-bytes; however, we end up having to defer to the individual platforms for
 * more information. Because of this, we also get and use the l1 cache
 * information. This is further complicated by the fact that the I-cache and
 * D-cache are separate usually; therefore we us the the l1 d-cache for
 * CPUSETSIZE().
 */
extern int l2cache_sz, l2cache_linesz, l2cache_assoc;
#define L2CACHE_ALIGN           l2cache_linesz
extern int      armv6_cachesz, armv6_cache_assoc;
extern int      armv6_l2cache_size, armv6_l2cache_linesz;
#define L2CACHE_ALIGN           armv6_l2cache_linesz
#define L2CACHE_ALIGN_MAX       64
#define CPUSETSIZE()            (l2cache_sz / l2cache_assoc)
#define CPUSETSIZE()            (armv6_cachesz / armv6_cache_assoc)

/*
 * Return the log2(pagesize(szc) / MMU_PAGESIZE) --- or the shift count
 * for the number of base pages in this pagesize
 */
#define PAGE_BSZS_SHIFT(szc) (PNUM_SHIFT(szc) - MMU_PAGESHIFT)

/*
 * Internal PG_ flags.
 */
#define PGI_RELOCONLY   0x010000        /* opposite of PG_NORELOC */
#define PGI_NOCAGE      0x020000        /* cage is disabled */
#define PGI_PGCPHIPRI   0x040000        /* page_get_contig_page pri alloc */
#define PGI_PGCPSZC0    0x080000        /* relocate base pagesize page */

/*
 * XXX Consider PGI flags for ourselves
 */

#define AS_2_BIN(as, seg, vp, addr, bin, szc)   panic("as_2_bin")

/*
 * XXX For the moment, we'll use the same value for VM_CPU_DATA_PADSIZE that
 * is used on other platforms. We don't use this at all, but it's required for
 * stuff like vm_pagelist.c to build. We should figure out what the right answer
 * looks like here.
 */
/*
 * cpu private vm data - accessed thru CPU->cpu_vm_data
 *      vc_pnum_memseg: tracks last memseg visited in page_numtopp_nolock()
 *      vc_pnext_memseg: tracks last memseg visited in page_nextn()
 *      vc_kmptr: orignal unaligned kmem pointer for this vm_cpu_data_t
 *      vc_kmsize: orignal kmem size for this vm_cpu_data_t
 */

typedef struct {
        struct memseg   *vc_pnum_memseg;
        struct memseg   *vc_pnext_memseg;
        void            *vc_kmptr;
        size_t          vc_kmsize;
} vm_cpu_data_t;
```
______***unchanged_portion_omitted***_