```
**********************************************************
   25774 Fri Aug 10 16:06:33 2018
new/usr/src/cmd/mdb/i86pc/modules/unix/unix.c
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
______unchanged_portion_omitted_

 904 #ifdef _KMDB
 905 /* ARGSUSED */
 906 static int
 907 sysregs_dcmd(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
 907 crregs_dcmd(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
 908 {
 909         ulong_t cr0, cr2, cr3, cr4;
 910         desctbr_t gdtr;

 912         static const mdb_bitmask_t cr0_flag_bits[] = {
 913                 { "PE",         CR0_PE,         CR0_PE },
 914                 { "MP",         CR0_MP,         CR0_MP },
 915                 { "EM",         CR0_EM,         CR0_EM },
 916                 { "TS",         CR0_TS,         CR0_TS },
 917                 { "ET",         CR0_ET,         CR0_ET },
 918                 { "NE",         CR0_NE,         CR0_NE },
 919                 { "WP",         CR0_WP,         CR0_WP },
 920                 { "AM",         CR0_AM,         CR0_AM },
 921                 { "NW",         CR0_NW,         CR0_NW },
 922                 { "CD",         CR0_CD,         CR0_CD },
 923                 { "PG",         CR0_PG,         CR0_PG },
 924                 { NULL,         0,              0 }
 925         };

 927         static const mdb_bitmask_t cr3_flag_bits[] = {
 928                 { "PCD",        CR3_PCD,        CR3_PCD },
 929                 { "PWT",        CR3_PWT,        CR3_PWT },
 930                 { NULL,         0,              0, }
 931         };

 933         static const mdb_bitmask_t cr4_flag_bits[] = {
 934                 { "VME",        CR4_VME,        CR4_VME },
 935                 { "PVI",        CR4_PVI,        CR4_PVI },
 936                 { "TSD",        CR4_TSD,        CR4_TSD },
 937                 { "DE",         CR4_DE,         CR4_DE },
 938                 { "PSE",        CR4_PSE,        CR4_PSE },
 939                 { "PAE",        CR4_PAE,        CR4_PAE },
 940                 { "MCE",        CR4_MCE,        CR4_MCE },
 941                 { "PGE",        CR4_PGE,        CR4_PGE },
 942                 { "PCE",        CR4_PCE,        CR4_PCE },
 943                 { "OSFXSR",     CR4_OSFXSR,     CR4_OSFXSR },
 944                 { "OSXMMEXCPT", CR4_OSXMMEXCPT, CR4_OSXMMEXCPT },
 945                 { "VMXE",       CR4_VMXE,       CR4_VMXE },
 946                 { "SMXE",       CR4_SMXE,       CR4_SMXE },
 947                 { "PCIDE",      CR4_PCIDE,      CR4_PCIDE },
 948                 { "OSXSAVE",    CR4_OSXSAVE,    CR4_OSXSAVE },
 949                 { "SMEP",       CR4_SMEP,       CR4_SMEP },
 950                 { "SMAP",       CR4_SMAP,       CR4_SMAP },
 951                 { NULL,         0,              0 }
 952         };

 954         cr0 = kmdb_unix_getcr0();
 955         cr2 = kmdb_unix_getcr2();
 956         cr3 = kmdb_unix_getcr3();
 957         cr4 = kmdb_unix_getcr4();

 959         kmdb_unix_getgdtr(&gdtr);
```



```
 961         mdb_printf("%%cr0 = 0x%lx <%b>\n", cr0, cr0, cr0_flag_bits);
 962         mdb_printf("%%cr2 = 0x%lx <%a>\n", cr2, cr2);

 964         if ((cr4 & CR4_PCIDE)) {
 965                 mdb_printf("%%cr3 = 0x%lx <pfn:0x%lx pcid:%lu>\n", cr3,
 966                     cr3 >> MMU_PAGESHIFT, cr3 & MMU_PAGEOFFSET);
 967         } else {
 968                 mdb_printf("%%cr3 = 0x%lx <pfn:0x%lx flags:%b>\n", cr3,
 969                     cr3 >> MMU_PAGESHIFT, cr3, cr3_flag_bits);
 970         }

 972         mdb_printf("%%cr4 = 0x%lx <%b>\n", cr4, cr4, cr4_flag_bits);

 974         mdb_printf("%%gdtr.base = 0x%lx, %%gdtr.limit = 0x%hx\n",
 975             gdtr.dtr_base, gdtr.dtr_limit);

 977         return (DCMD_OK);
 978 }
 979 #endif

 981 static const mdb_dcmd_t dcmds[] = {
 982         { "gate_desc", ":", "dump a gate descriptor", gate_desc },
 983         { "idt", ":[-v]", "dump an IDT", idt },
 984         { "ttrace", "[-x] [-t kthread]", "dump trap trace buffers", ttrace },
 985         { "vatopfn", ":[-a as]", "translate address to physical page",
 986           va2pfn_dcmd },
 987         { "report_maps", ":[-m]",
 988           "Given PFN, report mappings / page table usage",
 989           report_maps_dcmd, report_maps_help },
 990         { "htables", "", "Given hat_t *, lists all its htable_t * values",
 991           htables_dcmd, htables_help },
 992         { "ptable", ":[-lm]", "Given PFN, dump contents of a page table",
 993           ptable_dcmd, ptable_help },
 994         { "ptmap", ":", "Given a cr3 value, dump all mappings",
 995           ptmap_dcmd, ptmap_help },
 996         { "pte", ":[-l N]", "print human readable page table entry",
 997           pte_dcmd },
 998         { "pfntomfn", ":", "convert physical page to hypervisor machine page",
 999           pfntomfn_dcmd },
1000         { "mfntopfn", ":", "convert hypervisor machine page to physical page",
1001           mfntopfn_dcmd },
1002         { "memseg_list", ":", "show memseg list", memseg_list },
1003         { "scalehrtime", ":",
1004           "scale an unscaled high-res time", scalehrtime_cmd },
1005         { "x86_featureset", NULL, "dump the x86_featureset vector",
1006                 x86_featureset_cmd },
1007 #ifdef _KMDB
1008         { "sysregs", NULL, "dump system registers", sysregs_dcmd },
1000         { "crregs", NULL, "dump control registers", crregs_dcmd },
1009 #endif
1010         { NULL }
1011 };
______unchanged_portion_omitted_
```

****
877 Fri Aug 10 16:06:33 2018
new/usr/src/cmd/mdb/i86pc/modules/unix/unix_sup.h
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
****

```
/*
 * 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 2018 Joyent, Inc.
 */

#ifndef _UNIX_SUP_H
#define _UNIX_SUP_H

/*
 * Support routines for unix.
 */

#include <sys/types.h>
#include <sys/segments.h>

#ifdef __cplusplus
extern "C" {
#endif

extern ulong_t kmdb_unix_getcr0(void);
extern ulong_t kmdb_unix_getcr2(void);
extern ulong_t kmdb_unix_getcr3(void);
extern ulong_t kmdb_unix_getcr4(void);
extern void kmdb_unix_getgdtr(desctbr_t *);

#ifdef __cplusplus
}
```

*_____unchanged_portion_omitted_*

```
**********************************************************
    1008 Fri Aug 10 16:06:33 2018
new/usr/src/cmd/mdb/i86pc/modules/unix/unix_sup.s
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2018 Joyent, Inc.
 */

#if !defined(__lint)
        .file   "unix_sup.s"
#endif /* __lint */

/*
 * Support routines for the unix kmdb module
 */

#include <sys/asm_linkage.h>

#if defined(__lint)

#include <sys/types.h>

#else
ulong_t
kmdb_unix_getcr0(void)
{ return (0); }

#include <sys/asm_linkage.h>
ulong_t
kmdb_unix_getcr3(void)
{ return (0); }

        .file   "unix_sup.s"
ulong_t
kmdb_unix_getcr4(void)
{ return (0); }

#else   /* __lint */

#if defined(__amd64)
        ENTRY(kmdb_unix_getcr0)
        movq %cr0, %rax
        ret
        SET_SIZE(kmdb_unix_getcr0)
______unchanged_portion_omitted_

        ENTRY(kmdb_unix_getgdtr)
        sgdt (%rdi)
#elif defined (__i386)
        ENTRY(kmdb_unix_getcr0)
        movl %cr0, %eax
        ret
```



```
        SET_SIZE(kmdb_unix_getgdtr)
        SET_SIZE(kmdb_unix_getcr0)

#endif /* !__lint */
        ENTRY(kmdb_unix_getcr2)
        movl %cr2, %eax
        ret
        SET_SIZE(kmdb_unix_getcr2)

        ENTRY(kmdb_unix_getcr3)
        movl %cr3, %eax
        ret
        SET_SIZE(kmdb_unix_getcr3)

        ENTRY(kmdb_unix_getcr4)
        movl %cr4, %eax
        ret
        SET_SIZE(kmdb_unix_getcr4)

#endif  /* __i386 */

#endif /* __lint */
```

```
**********************************************************
    3797 Fri Aug 10 16:06:33 2018
new/usr/src/pkg/manifests/system-test-ostest.mf
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
#
# 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) 2012, 2016 by Delphix. All rights reserved.
# Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2018 Joyent, Inc.
#

set name=pkg.fmri value=pkg:/system/test/ostest@$(PKGVERS)
set name=pkg.description value="Miscellaneous OS Unit Tests"
set name=pkg.summary value="OS Unit Test Suite"
set name=info.classification \
    value=org.opensolaris.category.2008:Development/System
set name=variant.arch value=$(ARCH)
dir path=opt/os-tests
dir path=opt/os-tests/bin
dir path=opt/os-tests/runfiles
dir path=opt/os-tests/tests
dir path=opt/os-tests/tests/file-locking
$(i386_ONLY)dir path=opt/os-tests/tests/i386
dir path=opt/os-tests/tests/pf_key
dir path=opt/os-tests/tests/sdevfs
dir path=opt/os-tests/tests/secflags
dir path=opt/os-tests/tests/sigqueue
dir path=opt/os-tests/tests/sockfs
dir path=opt/os-tests/tests/stress
file path=opt/os-tests/README mode=0444
file path=opt/os-tests/bin/ostest mode=0555
file path=opt/os-tests/runfiles/default.run mode=0444
file path=opt/os-tests/tests/epoll_test mode=0555
file path=opt/os-tests/tests/file-locking/acquire-lock.32 mode=0555
file path=opt/os-tests/tests/file-locking/acquire-lock.64 mode=0555
file path=opt/os-tests/tests/file-locking/runtests.32 mode=0555
file path=opt/os-tests/tests/file-locking/runtests.64 mode=0555
$(i386_ONLY)file path=opt/os-tests/tests/i386/badseg mode=0555
$(i386_ONLY)file path=opt/os-tests/tests/i386/ldt mode=0555
file path=opt/os-tests/tests/pf_key/acquire-compare mode=0555
file path=opt/os-tests/tests/pf_key/acquire-spray mode=0555
file path=opt/os-tests/tests/pf_key/eacq-enabler mode=0555
file path=opt/os-tests/tests/pf_key/kmc-update mode=0555
file path=opt/os-tests/tests/pf_key/kmc-updater mode=0555
file path=opt/os-tests/tests/poll_test mode=0555
file path=opt/os-tests/tests/sdevfs/sdevfs_eisdir mode=0555
file path=opt/os-tests/tests/secflags/addrs-32 mode=0555
file path=opt/os-tests/tests/secflags/addrs-64 mode=0555
file path=opt/os-tests/tests/secflags/secflags_aslr mode=0555
file path=opt/os-tests/tests/secflags/secflags_core mode=0555
file path=opt/os-tests/tests/secflags/secflags_dts mode=0555
file path=opt/os-tests/tests/secflags/secflags_elfdump mode=0555
file path=opt/os-tests/tests/secflags/secflags_forbidnullmap mode=0555
```



```
file path=opt/os-tests/tests/secflags/secflags_limits mode=0555
file path=opt/os-tests/tests/secflags/secflags_noexecstack mode=0555
file path=opt/os-tests/tests/secflags/secflags_proc mode=0555
file path=opt/os-tests/tests/secflags/secflags_psecflags mode=0555
file path=opt/os-tests/tests/secflags/secflags_syscall mode=0555
file path=opt/os-tests/tests/secflags/secflags_truss mode=0555
file path=opt/os-tests/tests/secflags/secflags_zonecfg mode=0555
file path=opt/os-tests/tests/secflags/stacky mode=0555
file path=opt/os-tests/tests/sigqueue/sigqueue_queue_size mode=0555
file path=opt/os-tests/tests/sockfs/conn mode=0555
file path=opt/os-tests/tests/sockfs/dgram mode=0555
file path=opt/os-tests/tests/sockfs/drop_priv mode=0555
file path=opt/os-tests/tests/sockfs/nosignal mode=0555
file path=opt/os-tests/tests/sockfs/sockpair mode=0555
file path=opt/os-tests/tests/spoof-ras mode=0555
file path=opt/os-tests/tests/stress/dladm-kstat mode=0555
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
depend fmri=pkg:/network/telnet type=require
depend fmri=system/test/testrunner type=require
```

```
***********************************************************
    1605 Fri Aug 10 16:06:33 2018
new/usr/src/test/os-tests/runfiles/default.run
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
***********************************************************
```

```
#
# 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) 2012 by Delphix. All rights reserved.
# Copyright 2018 Joyent, Inc.
#

[DEFAULT]
pre =
verbose = False
quiet = False
timeout = 60
post =
outputdir = /var/tmp/test_results

[/opt/os-tests/tests/poll_test]
user = root
tests = ['poll_test', 'epoll_test']

[/opt/os-tests/tests/secflags]
user = root
tests = ['secflags_aslr',
         'secflags_core',
         'secflags_dts',
         'secflags_elfdump',
         'secflags_forbidnullmap',
         'secflags_limits',
         'secflags_noexecstack',
         'secflags_proc',
         'secflags_psecflags',
         'secflags_syscall',
         'secflags_truss',
         'secflags_zonecfg']

[/opt/os-tests/tests/sigqueue]
tests = ['sigqueue_queue_size']

[/opt/os-tests/tests/sdevfs]
user = root
tests = ['sdevfs_eisdir']

[/opt/os-tests/tests/stress]
user = root
tests = ['dladm-kstat']

[/opt/os-tests/tests/file-locking]
tests = ['runtests.32', 'runtests.64']

[/opt/os-tests/tests/sockfs]
user = root
```



```
tests = ['conn', 'dgram', 'drop_priv', 'nosignal', 'sockpair']

[/opt/os-tests/tests/pf_key]
user = root
tests = ['acquire-compare', 'acquire-spray']

[/opt/os-tests/tests/i386]
user = root
arch = i86pc
tests = ['ldt', 'badseg']
tests = ['ldt']
```

```
**********************************************************
     866 Fri Aug 10 16:06:33 2018
new/usr/src/test/os-tests/tests/i386/Makefile
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
#
# 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 2018 Joyent, Inc.
#

include $(SRC)/cmd/Makefile.cmd
include $(SRC)/test/Makefile.com

PROGS += ldt badseg
PROG += ldt

ROOTOPTPKG = $(ROOT)/opt/os-tests
TESTDIR = $(ROOTOPTPKG)/tests/i386
ROOTOPTPROGS = $(PROGS:%=$(TESTDIR)/%)

CSTD = $(CSTD_GNU99)

# for badseg
COPTFLAG =
CMDS = $(PROG:%=$(TESTDIR)/%)
$(CMDS) := FILEMODE = 0555

all: $(PROGS)
all: $(PROG)

install: all $(ROOTOPTPROGS)
install: all $(CMDS)

lint:

clobber: clean
        -$(RM) $(PROGS)
        -$(RM) $(PROG)

clean:

$(ROOTOPTPROGS): $(TESTDIR) $(PROGS)
$(CMDS): $(TESTDIR) $(PROG)

$(TESTDIR):
        $(INS.dir)

$(TESTDIR)/%: %
        $(INS.file)
```

```
**********************************************************
    2282 Fri Aug 10 16:06:34 2018
new/usr/src/test/os-tests/tests/i386/badseg.c
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2018 Joyent, Inc.
 */

#include <stdlib.h>
#include <ucontext.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sys/regset.h>

/*
 * Load a bunch of bad selectors into the seg regs: this will typically cause
 * the child process to core dump, but it shouldn't panic the kernel...
 *
 * It's especially interesting to run this on CPU0.
 */

unsigned short selector;

static void badds(void)
{
        __asm__ volatile("movw %0, %%ds" : : "r" (selector));
}

static void bades(void)
{
        __asm__ volatile("movw %0, %%es" : : "r" (selector));
}

static void badfs(void)
{
        __asm__ volatile("movw %0, %%fs" : : "r" (selector));
}

static void badgs(void)
{
        __asm__ volatile("movw %0, %%gs" : : "r" (selector));
}

static void badss(void)
{
        __asm__ volatile("movw %0, %%ss" : : "r" (selector));
}

static void
resetseg(uint_t seg)
{
        ucontext_t ucp;
```



```
        int done = 0;

        int rc = getcontext(&ucp);
        if (done) {
                rc = getcontext(&ucp);
                return;
        }

        done = 1;
        ucp.uc_mcontext.gregs[seg] = selector;
        setcontext(&ucp);
        abort();
}

static void
resetcs(void)
{
        return (resetseg(CS));
}

static void
resetds(void)
{
        return (resetseg(DS));
}

static void
resetes(void)
{
        return (resetseg(ES));
}

static void
resetfs(void)
{
        return (resetseg(FS));
}

static void
resetgs(void)
{
        return (resetseg(GS));
}

static void
resetss(void)
{
        return (resetseg(SS));
}

static void
inchild(void (*func)())
{
        pid_t pid;

        switch ((pid = fork())) {
        case 0:
                func();
                exit(0);
        case -1:
                exit(1);
        default:
                (void) waitpid(pid, NULL, 0);
                return;
        }
```

```
126 }

128 int
129 main(int argc, char *argv[])
130 {
131         for (selector = 0; selector < 8194; selector++) {
132                 inchild(resetcs);
133                 inchild(resetds);
134                 inchild(resetes);
135                 inchild(resetfs);
136                 inchild(resetgs);
137                 inchild(resetss);
138                 inchild(badds);
139                 inchild(bades);
140                 inchild(badfs);
141                 inchild(badgs);
142                 inchild(badss);
143         }

145         exit(0);
146 }
```

```
**********************************************************
   38668 Fri Aug 10 16:06:34 2018
new/usr/src/uts/intel/ia32/os/desctbls.c
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
______unchanged_portion_omitted_

1291 #endif  /* __xpv */

1293 #ifndef __xpv
1294 /*
1295  * As per Intel Vol 3 27.5.2, the GDTR limit is reset to 64Kb on a VM exit, so
1296  * we have to manually fix it up ourselves.
1297  *
1298  * The caller may still need to make sure that it can't go off-CPU with the
1299  * incorrect limit, before calling this (such as disabling pre-emption).
1300  */
1301 void
1302 reset_gdtr_limit(void)
1303 {
1304         ulong_t flags = intr_clear();
1305         desctbr_t gdtr;

1307         rd_gdtr(&gdtr);
1308         gdtr.dtr_limit = (sizeof (user_desc_t) * NGDT) - 1;
1309         wr_gdtr(&gdtr);

1311         intr_restore(flags);
1312 }
1313 #endif /* __xpv */

1315 /*
1316  * In the early kernel, we need to set up a simple GDT to run on.
1317  *
1318  * XXPV Can dboot use this too?  See dboot_gdt.s
1319  */
1320 void
1321 init_boot_gdt(user_desc_t *bgdt)
1322 {
1323 #if defined(__amd64)
1324         set_usegd(&bgdt[GDT_B32DATA], SDP_LONG, NULL, -1, SDT_MEMRWA, SEL_KPL,
1325             SDP_PAGES, SDP_OP32);
1326         set_usegd(&bgdt[GDT_B64CODE], SDP_LONG, NULL, -1, SDT_MEMERA, SEL_KPL,
1327             SDP_PAGES, SDP_OP32);
1328 #elif defined(__i386)
1329         set_usegd(&bgdt[GDT_B32DATA], NULL, -1, SDT_MEMRWA, SEL_KPL,
1330             SDP_PAGES, SDP_OP32);
1331         set_usegd(&bgdt[GDT_B32CODE], NULL, -1, SDT_MEMERA, SEL_KPL,
1332             SDP_PAGES, SDP_OP32);
1333 #endif  /* __i386 */
1334 }
______unchanged_portion_omitted_
```

```
**********************************************************
   33198 Fri Aug 10 16:06:34 2018
new/usr/src/uts/intel/sys/x86_archext.h
9723 provide support for VMM's GDT handling
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
______unchanged_portion_omitted_

 742 extern int x86_use_pcid;
 743 extern int x86_use_invpcid;

 745 /*
 746  * Utility functions to get/set extended control registers (XCR)
 747  * Initial use is to get/set the contents of the XFEATURE_ENABLED_MASK.
 748  */
 749 extern uint64_t get_xcr(uint_t);
 750 extern void set_xcr(uint_t, uint64_t);

 752 extern uint64_t rdmsr(uint_t);
 753 extern void wrmsr(uint_t, const uint64_t);
 754 extern uint64_t xrdmsr(uint_t);
 755 extern void xwrmsr(uint_t, const uint64_t);
 756 extern int checked_rdmsr(uint_t, uint64_t *);
 757 extern int checked_wrmsr(uint_t, uint64_t);

 759 extern void invalidate_cache(void);
 760 extern ulong_t getcr4(void);
 761 extern void setcr4(ulong_t);

 763 extern void mtrr_sync(void);

 765 extern void cpu_fast_syscall_enable(void);
 766 extern void cpu_fast_syscall_disable(void);

 768 struct cpu;

 770 extern int cpuid_checkpass(struct cpu *, int);
 771 extern uint32_t cpuid_insn(struct cpu *, struct cpuid_regs *);
 772 extern uint32_t __cpuid_insn(struct cpuid_regs *);
 773 extern int cpuid_getbrandstr(struct cpu *, char *, size_t);
 774 extern int cpuid_getidstr(struct cpu *, char *, size_t);
 775 extern const char *cpuid_getvendorstr(struct cpu *);
 776 extern uint_t cpuid_getvendor(struct cpu *);
 777 extern uint_t cpuid_getfamily(struct cpu *);
 778 extern uint_t cpuid_getmodel(struct cpu *);
 779 extern uint_t cpuid_getstep(struct cpu *);
 780 extern uint_t cpuid_getsig(struct cpu *);
 781 extern uint_t cpuid_get_ncpu_per_chip(struct cpu *);
 782 extern uint_t cpuid_get_ncore_per_chip(struct cpu *);
 783 extern uint_t cpuid_get_ncpu_sharing_last_cache(struct cpu *);
 784 extern id_t cpuid_get_last_lvl_cacheid(struct cpu *);
 785 extern int cpuid_get_chipid(struct cpu *);
 786 extern id_t cpuid_get_coreid(struct cpu *);
 787 extern int cpuid_get_pkgcoreid(struct cpu *);
 788 extern int cpuid_get_clogid(struct cpu *);
 789 extern int cpuid_get_cacheid(struct cpu *);
 790 extern uint32_t cpuid_get_apicid(struct cpu *);
 791 extern uint_t cpuid_get_procnodeid(struct cpu *cpu);
 792 extern uint_t cpuid_get_procnodes_per_pkg(struct cpu *cpu);
 793 extern uint_t cpuid_get_compunitid(struct cpu *cpu);
 794 extern uint_t cpuid_get_cores_per_compunit(struct cpu *cpu);
 795 extern size_t cpuid_get_xsave_size();
 796 extern boolean_t cpuid_need_fp_excp_handling();
 797 extern int cpuid_is_cmt(struct cpu *);
 798 extern int cpuid_syscall32_insn(struct cpu *);
```



```
 799 extern int getl2cacheinfo(struct cpu *, int *, int *, int *);

 801 extern uint32_t cpuid_getchiprev(struct cpu *);
 802 extern const char *cpuid_getchiprevstr(struct cpu *);
 803 extern uint32_t cpuid_getsockettype(struct cpu *);
 804 extern const char *cpuid_getsocketstr(struct cpu *);

 806 extern int cpuid_have_cr8access(struct cpu *);

 808 extern int cpuid_opteron_erratum(struct cpu *, uint_t);

 810 struct cpuid_info;

 812 extern void setx86isalist(void);
 813 extern void cpuid_alloc_space(struct cpu *);
 814 extern void cpuid_free_space(struct cpu *);
 815 extern void cpuid_pass1(struct cpu *, uchar_t *);
 816 extern void cpuid_pass2(struct cpu *);
 817 extern void cpuid_pass3(struct cpu *);
 818 extern void cpuid_pass4(struct cpu *, uint_t *);
 819 extern void cpuid_set_cpu_properties(void *, processorid_t,
 820     struct cpuid_info *);

 822 extern void cpuid_get_addrsize(struct cpu *, uint_t *, uint_t *);
 823 extern uint_t cpuid_get_dtlb_nent(struct cpu *, size_t);

 825 #if !defined(__xpv)
 826 extern uint32_t *cpuid_mwait_alloc(struct cpu *);
 827 extern void cpuid_mwait_free(struct cpu *);
 828 extern int cpuid_deep_cstates_supported(void);
 829 extern int cpuid_arat_supported(void);
 830 extern int cpuid_iepb_supported(struct cpu *);
 831 extern int cpuid_deadline_tsc_supported(void);
 832 extern void vmware_port(int, uint32_t *);
 833 #endif

 835 struct cpu_ucode_info;

 837 extern void ucode_alloc_space(struct cpu *);
 838 extern void ucode_free_space(struct cpu *);
 839 extern void ucode_check(struct cpu *);
 840 extern void ucode_cleanup();

 842 #if !defined(__xpv)
 843 extern  char _tsc_mfence_start;
 844 extern  char _tsc_mfence_end;
 845 extern  char _tscp_start;
 846 extern  char _tscp_end;
 847 extern  char _no_rdtsc_start;
 848 extern  char _no_rdtsc_end;
 849 extern  char _tsc_lfence_start;
 850 extern  char _tsc_lfence_end;
 851 #endif

 853 #if !defined(__xpv)
 854 extern  char bcopy_patch_start;
 855 extern  char bcopy_patch_end;
 856 extern  char bcopy_ck_size;
 857 #endif

 859 extern void post_startup_cpu_fixups(void);

 861 extern uint_t workaround_errata(struct cpu *);

 863 #if defined(OPTERON_ERRATUM_93)
 864 extern int opteron_erratum_93;
```

```
865 #endif

867 #if defined(OPTERON_ERRATUM_91)
868 extern int opteron_erratum_91;
869 #endif

871 #if defined(OPTERON_ERRATUM_100)
872 extern int opteron_erratum_100;
873 #endif

875 #if defined(OPTERON_ERRATUM_121)
876 extern int opteron_erratum_121;
877 #endif

879 #if defined(OPTERON_WORKAROUND_6323525)
880 extern int opteron_workaround_6323525;
881 extern void patch_workaround_6323525(void);
882 #endif

884 #if !defined(__xpv)
885 extern void determine_platform(void);
886 #endif
887 extern int get_hwenv(void);
888 extern int is_controldom(void);

890 extern void enable_pcid(void);

892 extern void xsave_setup_msr(struct cpu *);

894 #if !defined(__xpv)
895 extern void reset_gdtr_limit(void);
896 #endif

898 /*
899  * Hypervisor signatures
900  */
901 #define HVSIG_XEN_HVM   "XenVMMXenVMM"
902 #define HVSIG_VMWARE    "VMwareVMware"
903 #define HVSIG_KVM       "KVMKVMKVM"
904 #define HVSIG_MICROSOFT "Microsoft Hv"

906 /*
907  * Defined hardware environments
908  */
909 #define HW_NATIVE       (1 << 0)        /* Running on bare metal */
910 #define HW_XEN_PV       (1 << 1)        /* Running on Xen PVM */

912 #define HW_XEN_HVM      (1 << 2)        /* Running on Xen HVM */
913 #define HW_VMWARE       (1 << 3)        /* Running on VMware hypervisor */
914 #define HW_KVM          (1 << 4)        /* Running on KVM hypervisor */
915 #define HW_MICROSOFT    (1 << 5)        /* Running on Microsoft hypervisor */

917 #define HW_VIRTUAL      (HW_XEN_HVM | HW_VMWARE | HW_KVM | HW_MICROSOFT)

919 #endif  /* _KERNEL */

921 #endif  /* !_ASM */

923 /*
924  * VMware hypervisor related defines
925  */
926 #define VMWARE_HVMAGIC          0x564d5868
927 #define VMWARE_HVPORT           0x5658
928 #define VMWARE_HVCMD_GETVERSION 0x0a
929 #define VMWARE_HVCMD_GETTSCFREQ 0x2d
```



```
931 #ifdef  __cplusplus
932 }
```

*unchanged_portion_omitted_*