```
**********************************************************
    2524 Wed Feb 28 17:34:17 2018
new/usr/src/cmd/mdb/Makefile.kmdb.files
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
#
# 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 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

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

KMDBSRCS += \
        ffs.c \
        kaif_start.c \
        mdb.c \
        mdb_addrvec.c \
        mdb_argvec.c \
        mdb_callb.c \
        mdb_cmdbuf.c \
        mdb_cmds.c \
        kvm_cpu.c \
        kmdb_conf.c \
        kmdb_context.c \
        kmdb_create.c \
        mdb_ctf.c \
        kmdb_ctf_open.c \
        mdb_debug.c \
        kmdb_demangle.c \
        mdb_disasm.c \
        kmdb_dl.c \
        kmdb_dpi.c \
        mdb_dump.c \
        mdb_err.c \
        mdb_evset.c \
        kmdb_fault.c \
        kmdb_fdio.c \
        mdb_fmt.c \
        mdb_frame.c \
```



```
        mdb_gelf.c \
        mdb_help.c \
        mdb_io.c \
        kmdb_kdi.c \
        kmdb_kvm.c \
        mdb_logio.c \
        mdb_list.c \
        mdb_macalias.c \
        kmdb_main.c \
        mdb_modapi.c \
        mdb_module.c \
        kmdb_module.c \
        kmdb_module_load.c \
        mdb_nm.c \
        mdb_nv.c \
        mdb_pipeio.c \
        mdb_print.c \
        kmdb_promio.c \
        kmdb_promif.c \
        mdb_set.c \
        kmdb_shell.c \
        mdb_signal.c \
        mdb_string.c \
        mdb_strio.c \
        kmdb_stubs.c \
        mdb_tab.c \
        mdb_target.c \
        kmdb_terminfo.c \
        mdb_termio.c \
        mdb_typedef.c \
        mdb_umem.c \
        kmdb_umemglue.c \
        mdb_value.c \
        mdb_vcb.c \
        mdb_wcb.c \
        mdb_whatis.c \
        kmdb_wr.c

KMDBML +=

KMDBOBJS = $(KMDBSRCS:%.c=%.o) $(KMDBML:%.s=%.o)

PROMSRCS +=

PROMOBJS = $(PROMSRCS:%.c=%.o)

KCTLSRCS += \
        kctl_auxv.c \
        kctl_dmod.c \
        kctl_err.c \
        kctl_main.c \
        kctl_mod.c \
        kctl_string.c \
        kctl_wr.c

KCTLML +=

KCTLOBJS = $(KCTLSRCS:%.c=%.o) $(KCTLML:%.s=%.o)

SRCS += $(KMDBSRCS) $(PROMSRCS)
MLSRCS += $(KMDBML)
OBJS = $(SRCS:%.c=%.o) $(KMDBML:%.s=%.o)

ALLOBJS = $(OBJS) $(KCTLOBJS)
ALLLINTFILES = $(ALLOBJS:%.o=%.ln)
```

```
123 # files that need KMDB_VERSION defined
124 VERSFILES = \
125         kmdb_conf.c \
126         kctl_main.c
127 VERSOBJS = $(VERSFILES:%.c=%.o)
```

```
**********************************************************
    3496 Wed Feb 28 17:34:17 2018
new/usr/src/cmd/mdb/common/kmdb/kmdb_dpi_impl.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#ifndef _KMDB_DPI_IMPL_H
#define _KMDB_DPI_IMPL_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <setjmp.h>
#ifdef  __sparc
#include <sys/regset.h>
#endif  /* __sparc */
#include <sys/types.h>

#include <kmdb/kmdb_auxv.h>
#include <kmdb/kmdb_dpi.h>
#include <mdb/mdb_kreg.h>
#include <mdb/mdb_target.h>

#ifdef __cplusplus
extern "C" {
#endif

extern jmp_buf *kmdb_dpi_fault_pcb;

/*
 * The routines used by the kmdb side of the DPI to access the saved state
 * of the current kernel instance, and to control that instance.  A populated
 * version of this vector is provided by the DPI backend used to control the
 * machine.  General use of the kmdb DPI is not via direct invocation of the
 * functions in this ops vector, but rather flows through the convenience
 * wrappers in kmdb_dpi.c.
 */
struct dpi_ops {
```



```
        int (*dpo_init)(kmdb_auxv_t *);

        void (*dpo_debugger_activate)(kdi_debugvec_t **, uint_t);
        void (*dpo_debugger_deactivate)(void);

        void (*dpo_enter_mon)(void);

        void (*dpo_modchg_register)(void (*)(struct modctl *, int));
        void (*dpo_modchg_cancel)(void);

        int (*dpo_get_cpu_state)(int);
        int (*dpo_get_master_cpuid)(void);

        const mdb_tgt_gregset_t *(*dpo_get_gregs)(int);
        int (*dpo_get_register)(const char *, kreg_t *);
        int (*dpo_set_register)(const char *, kreg_t);
#ifdef __sparc
        int (*dpo_get_rwin)(int, int, struct rwindow *);
        int (*dpo_get_nwin)(int);
#endif

        int (*dpo_brkpt_arm)(uintptr_t, mdb_instr_t *);
        int (*dpo_brkpt_disarm)(uintptr_t, mdb_instr_t);

        int (*dpo_wapt_validate)(kmdb_wapt_t *);
        int (*dpo_wapt_reserve)(kmdb_wapt_t *);
        void (*dpo_wapt_release)(kmdb_wapt_t *);
        void (*dpo_wapt_arm)(kmdb_wapt_t *);
        void (*dpo_wapt_disarm)(kmdb_wapt_t *);
        int (*dpo_wapt_match)(kmdb_wapt_t *);

        int (*dpo_step)(void);
#if defined(__i386) || defined(__amd64)
        void (*dpo_step_branch)(void);
#endif

        uintptr_t (*dpo_call)(uintptr_t, uint_t, const uintptr_t *);

        void (*dpo_dump_crumbs)(uintptr_t, int);

#if defined(__i386) || defined(__amd64)
        void (*dpo_msr_add)(const kdi_msr_t *);
        uint64_t (*dpo_msr_get)(int, uint_t);
#endif

#ifdef __sparc
        void (*dpo_kernpanic)(int);
#endif
};
______unchanged_portion_omitted_
```

```
**********************************************************
   63016 Wed Feb 28 17:34:18 2018
new/usr/src/cmd/mdb/common/kmdb/kmdb_kvm.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2013 by Delphix. All rights reserved.
 *
 * Copyright 2018 Joyent, Inc.
 */

#include <kmdb/kmdb_kvm.h>
#include <kmdb/kvm.h>
#include <kmdb/kmdb_kdi.h>
#include <kmdb/kmdb_promif.h>
#include <kmdb/kmdb_module.h>
#include <kmdb/kmdb_asmutil.h>
#include <mdb/mdb_types.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_string.h>
#include <mdb/mdb_ctf.h>
#include <mdb/mdb_kreg_impl.h>
#include <mdb/mdb_ks.h>
#include <mdb/mdb.h>

#include <strings.h>
#include <dlfcn.h>
#include <sys/isa_defs.h>
#include <sys/kobj.h>
#include <sys/kobj_impl.h>
#include <sys/bitmap.h>
#include <vm/as.h>

static const char KMT_RTLD_NAME[] = "krtld";
static const char KMT_MODULE[] = "mdb_ks";
static const char KMT_CTFPARENT[] = "genunix";

static mdb_list_t kmt_defbp_list;       /* List of current deferred bp's */
```



```
static int kmt_defbp_lock;              /* For list, running kernel holds */
static uint_t kmt_defbp_modchg_isload;  /* Whether mod change is load/unload */
static struct modctl *kmt_defbp_modchg_modctl; /* modctl for defbp checking */
static uint_t kmt_defbp_num;            /* Number of referenced def'd bp's */
static int kmt_defbp_bpspec;            /* vespec for def'd bp activation bp */

static const mdb_se_ops_t kmt_brkpt_ops;
static const mdb_se_ops_t kmt_wapt_ops;

static void kmt_sync(mdb_tgt_t *);

typedef struct kmt_symarg {
        mdb_tgt_sym_f *sym_cb;          /* Caller's callback function */
        void *sym_data;                 /* Callback function argument */
        uint_t sym_type;                /* Symbol type/binding filter */
        mdb_syminfo_t sym_info;         /* Symbol id and table id */
        const char *sym_obj;            /* Containing object */
} kmt_symarg_t;
```

*unchanged_portion_omitted_*

```
/*ARGSUSED*/
static int
kmt_status_dcmd(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
{
        kmt_data_t *kmt = mdb.m_target->t_data;
        struct utsname uts;
        char uuid[37];
        kreg_t tt;

        if (mdb_tgt_readsym(mdb.m_target, MDB_TGT_AS_VIRT, &uts, sizeof (uts),
            "unix", "utsname") != sizeof (uts)) {
                warn("failed to read 'utsname' struct from kernel\n");
                bzero(&uts, sizeof (uts));
                (void) strcpy(uts.nodename, "unknown machine");
        }

        mdb_printf("debugging live kernel (%d-bit) on %s\n",
            (int)(sizeof (void *) * NBBY),
            (*uts.nodename == '\0' ? "(not set)" : uts.nodename));
        mdb_printf("operating system: %s %s (%s)\n",
            uts.release, uts.version, uts.machine);

        if (mdb_tgt_readsym(mdb.m_target, MDB_TGT_AS_VIRT, uuid, sizeof (uuid),
            "genunix", "dump_osimage_uuid") != sizeof (uuid)) {
                warn("failed to read 'dump_osimage_uuid' string from kernel\n");
                (void) strcpy(uuid, "(error)");
        } else if (*uuid == '\0') {
                (void) strcpy(uuid, "(not set)");
        } else if (uuid[36] != '\0') {
                (void) strcpy(uuid, "(invalid)");
        }
        mdb_printf("image uuid: %s\n", uuid);

        if (kmt->kmt_cpu != NULL) {
                mdb_printf("CPU-specific support: %s\n",
                    kmt_cpu_name(kmt->kmt_cpu));
        }

        mdb_printf("DTrace state: %s\n", (kmdb_kdi_dtrace_get_state() ==
            KDI_DTSTATE_DTRACE_ACTIVE ? "active (debugger breakpoints cannot "
            "be armed)" : "inactive"));

        (void) kmdb_dpi_get_register("tt", &tt);
        mdb_printf("stopped on: %s\n", kmt_trapname(tt));

        (void) kmt_dmod_status("pending dmod loads:", KMDB_MC_STATE_LOADING);
```

```
 590         (void) kmt_dmod_status("pending dmod unloads:",
 591             KMDB_MC_STATE_UNLOADING);

 593         return (DCMD_OK);
 594 }
_____unchanged_portion_omitted_

2373 static void
2374 kmt_destroy(mdb_tgt_t *t)
2375 {
2376         kmt_data_t *kmt = t->t_data;
2377         kmt_module_t *km, *pkm;

2379         mdb_nv_destroy(&kmt->kmt_modules);
2380         for (km = mdb_list_prev(&kmt->kmt_modlist); km != NULL; km = pkm) {
2381                 pkm = mdb_list_prev(km);
2382                 mdb_free(km, sizeof (kmt_module_t));
2383         }

2385         if (!kmt_defbp_lock)
2386                 kmt_defbp_destroy_all();

2388         if (kmt->kmt_trapmap != NULL)
2389                 mdb_free(kmt->kmt_trapmap, BT_SIZEOFMAP(kmt->kmt_trapmax));

2395         if (kmt->kmt_cpu != NULL)
2396                 kmt_cpu_destroy(kmt->kmt_cpu);

2391         if (kmt != NULL)
2392                 mdb_free(kmt, sizeof (kmt_data_t));
2393 }

2395 static const mdb_tgt_ops_t kmt_ops = {
2396         kmt_setflags,                           /* t_setflags */
2397         (int (*)()) mdb_tgt_notsup,             /* t_setcontext */
2398         kmt_activate,                           /* t_activate */
2399         (void (*)()) mdb_tgt_nop,               /* t_deactivate */
2400         kmt_periodic,                           /* t_periodic */
2401         kmt_destroy,                            /* t_destroy */
2402         kmt_name,                               /* t_name */
2403         (const char *(*)()) mdb_conf_isa,       /* t_isa */
2404         kmt_platform,                           /* t_platform */
2405         kmt_uname,                              /* t_uname */
2406         kmt_dmodel,                             /* t_dmodel */
2407         (ssize_t (*)()) mdb_tgt_notsup,         /* t_aread */
2408         (ssize_t (*)()) mdb_tgt_notsup,         /* t_awrite */
2409         kmt_read,                               /* t_vread */
2410         kmt_write,                              /* t_vwrite */
2411         kmt_pread,                              /* t_pread */
2412         kmt_pwrite,                             /* t_pwrite */
2413         kmt_read,                               /* t_fread */
2414         kmt_write,                              /* t_fwrite */
2415         kmt_ioread,                             /* t_ioread */
2416         kmt_iowrite,                            /* t_iowrite */
2417         kmt_vtop,                               /* t_vtop */
2418         kmt_lookup_by_name,                     /* t_lookup_by_name */
2419         kmt_lookup_by_addr,                     /* t_lookup_by_addr */
2420         kmt_symbol_iter,                        /* t_symbol_iter */
2421         kmt_mapping_iter,                       /* t_mapping_iter */
2422         kmt_object_iter,                        /* t_object_iter */
2423         kmt_addr_to_map,                        /* t_addr_to_map */
2424         kmt_name_to_map,                        /* t_name_to_map */
2425         kmt_addr_to_ctf,                        /* t_addr_to_ctf */
2426         kmt_name_to_ctf,                        /* t_name_to_ctf */
2427         kmt_status,                             /* t_status */
2428         (int (*)()) mdb_tgt_notsup,             /* t_run */
```



```
2429         kmt_step,                               /* t_step */
2430         kmt_step_out,                           /* t_step_out */
2438         kmt_step_branch,                        /* t_step_branch */
2431         kmt_next,                               /* t_next */
2432         kmt_continue,                           /* t_cont */
2433         (int (*)()) mdb_tgt_notsup,             /* t_signal */
2434         kmt_add_vbrkpt,                         /* t_add_vbrkpt */
2435         kmt_add_sbrkpt,                         /* t_add_sbrkpt */
2436         kmt_add_pwapt,                          /* t_add_pwapt */
2437         kmt_add_vwapt,                          /* t_add_vwapt */
2438         kmt_add_iowapt,                         /* t_add_iowapt */
2439         (int (*)()) mdb_tgt_null,               /* t_add_sysenter */
2440         (int (*)()) mdb_tgt_null,               /* t_add_sysexit */
2441         (int (*)()) mdb_tgt_null,               /* t_add_signal */
2442         kmt_add_trap,                           /* t_add_fault */
2443         kmt_getareg,                            /* t_getareg */
2444         kmt_putareg,                            /* t_putareg */
2445         (int (*)()) mdb_tgt_nop,                /* XXX t_stack_iter */
2446         (int (*)()) mdb_tgt_notsup              /* t_auxv */
2447 };

2449 /*
2450  * Called immediately upon resumption of the system after a step or continue.
2451  * Allows us to synchronize kmt's view of the world with reality.
2452  */
2453 /*ARGSUSED*/
2454 static void
2455 kmt_sync(mdb_tgt_t *t)
2456 {
2457         kmt_data_t *kmt = t->t_data;
2458         int symavail;

2460         mdb_dprintf(MDB_DBG_KMOD, "synchronizing with kernel\n");

2462         symavail = kmt->kmt_symavail;
2463         kmt->kmt_symavail = FALSE;

2465         /*
2466          * Resync our view of the world if the modules have changed, or if we
2467          * didn't have any symbols coming into this function.  The latter will
2468          * only happen on startup.
2469          */
2470         if (kmdb_kdi_mods_changed() || !symavail)
2471                 kmt_modlist_update(t);

2473         /*
2474          * It would be nice if we could run this less frequently, perhaps
2475          * after a dvec-initiated trigger.
2476          */
2477         kmdb_module_sync();

2479         kmt->kmt_symavail = TRUE;

2481         mdb_dprintf(MDB_DBG_KMOD, "synchronization complete\n");

2483         kmt_defbp_prune();

2485         if (kmt_defbp_num > 0 && kmt_defbp_bpspec == 0 &&
2486             kmdb_kdi_dtrace_get_state() != KDI_DTSTATE_DTRACE_ACTIVE) {
2487                 /*
2488                  * Deferred breakpoints were created while DTrace was active,
2489                  * and consequently the deferred breakpoint enabling mechanism
2490                  * wasn't activated.  Activate it now, and then try to activate
2491                  * the deferred breakpoints.  We do this so that we can catch
2492                  * the ones which may apply to modules that have been loaded
2493                  * while they were waiting for DTrace to deactivate.
```

```
2494                  */
2495                 (void) kmt_defbp_activate(t);
2496                 (void) mdb_tgt_sespec_activate_all(t);
2497         }

2507         if (kmt->kmt_cpu_retry && ((kmt->kmt_cpu = kmt_cpu_create(t)) !=
2508             NULL || errno != EAGAIN))
2509                 kmt->kmt_cpu_retry = FALSE;

2499         (void) mdb_tgt_status(t, &t->t_status);
2500 }

2502 /*
2503  * This routine executes while the kernel is running.
2504  */
2505 /*ARGSUSED*/
2506 int
2507 kmdb_kvm_create(mdb_tgt_t *t, int argc, const char *argv[])
2508 {
2509         kmt_data_t *kmt;

2511         if (argc != 0)
2512                 return (set_errno(EINVAL));

2514         kmt = mdb_zalloc(sizeof (kmt_data_t), UM_SLEEP);
2515         t->t_data = kmt;
2516         t->t_ops = &kmt_ops;
2517         t->t_flags |= MDB_TGT_F_RDWR;   /* kmdb is always r/w */

2519         (void) mdb_nv_insert(&mdb.m_nv, "cpuid", &kmt_cpuid_disc, 0,
2520             MDB_NV_PERSIST | MDB_NV_RDONLY);

2522         (void) mdb_nv_create(&kmt->kmt_modules, UM_SLEEP);

2524         kmt_init_isadep(t);

2526         kmt->kmt_symavail = FALSE;
2539         kmt->kmt_cpu_retry = TRUE;

2528         bzero(&kmt_defbp_list, sizeof (mdb_list_t));

2530         return (0);

2532 create_err:
2533         kmt_destroy(t);

2535         return (-1);
2536 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    4411 Wed Feb 28 17:34:18 2018
new/usr/src/cmd/mdb/common/kmdb/kvm.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#ifndef _KVM_H
#define _KVM_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * The kmdb target
 */

#include <mdb/mdb_modapi.h>
#include <mdb/mdb_target.h>
#include <kmdb/kmdb_dpi.h>
#include <kmdb/kvm_isadep.h>
#include <kmdb/kvm_cpu.h>

#include <sys/kobj.h>

#ifdef __cplusplus
extern "C" {
#endif

#define KM_F_PRIMARY            1

#define KMT_TRAP_NOTENUM        -1      /* Glob for unnamed traps */
#define KMT_TRAP_ALL            -2      /* Glob for all traps */

typedef struct kmt_module {
        mdb_list_t      km_list;        /* List forward/back pointers */
        char            *km_name;       /* Module name */
        char            km_seen;
        GElf_Ehdr       km_ehdr;
```



```
        mdb_gelf_symtab_t *km_symtab;
        Shdr            km_symtab_hdr;
        Shdr            km_strtab_hdr;
        const void      *km_symtab_va;
        const void      *km_strtab_va;

        uintptr_t       km_text_va;
        size_t          km_text_size;
        uintptr_t       km_data_va;
        size_t          km_data_size;
        uintptr_t       km_bss_va;
        size_t          km_bss_size;
        const void      *km_ctf_va;
        size_t          km_ctf_size;

        ctf_file_t      *km_ctfp;
        struct modctl   km_modctl;
        struct module   km_module;
        int             km_flags;
} kmt_module_t;

typedef struct kmt_data {
        const mdb_tgt_regdesc_t *kmt_rds;       /* Register description table */
        mdb_nv_t        kmt_modules;            /* Hash table of modules */
        mdb_list_t      kmt_modlist;            /* List of mods in load order */
        const char      *kmt_rtld_name;         /* Module containing krtld */
        caddr_t         kmt_writemap;           /* Used to map PAs for writes */
        size_t          kmt_writemapsz;         /* Size of same */
        mdb_map_t       kmt_map;                /* Persistant map for callers */
        ulong_t         *kmt_trapmap;
        size_t          kmt_trapmax;
        kmt_cpu_t       *kmt_cpu;               /* CPU-specific plugin */
        int             kmt_cpu_retry;          /* Try CPU detect again? */
        int             kmt_symavail;           /* Symbol resolution allowed */
        uint_t          kmt_narmedbpts;         /* Number of armed brkpts */
#if defined(__i386) || defined(__amd64)
        struct {
                GElf_Sym        _kmt_cmnint;
                GElf_Sym        _kmt_cmntrap;
                GElf_Sym        _kmt_sysenter;
                GElf_Sym        _kmt_brand_sysenter;
#if defined(__amd64)
                GElf_Sym        _kmt_syscall;
                GElf_Sym        _kmt_brand_syscall;
#endif
        } kmt_intrsyms;
#endif
} kmt_data_t;
______unchanged_portion_omitted_

extern void kmt_printregs(const mdb_tgt_gregset_t *gregs);

extern const char *kmt_def_dismode(void);

extern void kmt_init_isadep(mdb_tgt_t *);
extern void kmt_startup_isadep(mdb_tgt_t *);

extern ssize_t kmt_write(mdb_tgt_t *, const void *, size_t, uintptr_t);
extern ssize_t kmt_pwrite(mdb_tgt_t *, const void *, size_t, physaddr_t);
extern ssize_t kmt_rw(mdb_tgt_t *, void *, size_t, uint64_t,
    ssize_t (*)(void *, size_t, uint64_t));
extern ssize_t kmt_writer(void *, size_t, uint64_t);
extern ssize_t kmt_ioread(mdb_tgt_t *, void *, size_t, uintptr_t);
extern ssize_t kmt_iowrite(mdb_tgt_t *, const void *, size_t, uintptr_t);
```

```
142 extern int kmt_step_out(mdb_tgt_t *, uintptr_t *);
146 extern int kmt_step_branch(mdb_tgt_t *);
143 extern int kmt_next(mdb_tgt_t *, uintptr_t *);

145 extern int kmt_stack(uintptr_t, uint_t, int, const mdb_arg_t *);
146 extern int kmt_stackv(uintptr_t, uint_t, int, const mdb_arg_t *);
147 extern int kmt_stackr(uintptr_t, uint_t, int, const mdb_arg_t *);
148 extern int kmt_cpustack(uintptr_t, uint_t, int, const mdb_arg_t *, int, int);

150 extern const char *kmt_trapname(int);

152 #ifdef __cplusplus
153 }
```

_____unchanged_portion_omitted_

```
**********************************************************
   84154 Wed Feb 28 17:34:18 2018
new/usr/src/cmd/mdb/common/mdb/mdb_cmds.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2012 by Delphix. All rights reserved.
 * Copyright (c) 2018 Joyent, Inc. All rights reserved.
 * Copyright (c) 2015 Joyent, Inc. All rights reserved.
 * Copyright (c) 2013 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 * Copyright (c) 2015, 2017 by Delphix. All rights reserved.
 */

#include <sys/elf.h>
#include <sys/elf_SPARC.h>

#include <libproc.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <alloca.h>
#include <libctf.h>
#include <ctype.h>

#include <mdb/mdb_string.h>
#include <mdb/mdb_argvec.h>
#include <mdb/mdb_nv.h>
#include <mdb/mdb_fmt.h>
#include <mdb/mdb_target.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_module.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_stdlib.h>
#include <mdb/mdb_lex.h>
```



```
#include <mdb/mdb_io_impl.h>
#include <mdb/mdb_help.h>
#include <mdb/mdb_disasm.h>
#include <mdb/mdb_frame.h>
#include <mdb/mdb_evset.h>
#include <mdb/mdb_print.h>
#include <mdb/mdb_nm.h>
#include <mdb/mdb_set.h>
#include <mdb/mdb_demangle.h>
#include <mdb/mdb_ctf.h>
#include <mdb/mdb_whatis.h>
#include <mdb/mdb_whatis_impl.h>
#include <mdb/mdb_macalias.h>
#include <mdb/mdb_tab.h>
#include <mdb/mdb_typedef.h>
#ifdef _KMDB
#include <kmdb/kmdb_kdi.h>
#endif
#include <mdb/mdb.h>

#ifdef __sparc
#define SETHI_MASK      0xc1c00000
#define SETHI_VALUE     0x01000000

#define IS_SETHI(machcode)      (((machcode) & SETHI_MASK) == SETHI_VALUE)

#define OP(machcode)    ((machcode) >> 30)
#define OP3(machcode)   (((machcode) >> 19) & 0x3f)
#define RD(machcode)    (((machcode) >> 25) & 0x1f)
#define RS1(machcode)   (((machcode) >> 14) & 0x1f)
#define I(machcode)     (((machcode) >> 13) & 0x01)

#define IMM13(machcode) ((machcode) & 0x1fff)
#define IMM22(machcode) ((machcode) & 0x3fffff)

#define OP_ARITH_MEM_MASK       0x2
#define OP_ARITH                0x2
#define OP_MEM                  0x3

#define OP3_CC_MASK             0x10
#define OP3_COMPLEX_MASK        0x20

#define OP3_ADD                 0x00
#define OP3_OR                  0x02
#define OP3_XOR                 0x03

#ifndef R_O7
#define R_O7    0xf
#endif
#endif /* __sparc */

static mdb_tgt_addr_t
write_uint8(mdb_tgt_as_t as, mdb_tgt_addr_t addr, uint64_t ull, uint_t rdback)
{
        uint8_t o, n = (uint8_t)ull;

        if (rdback && mdb_tgt_aread(mdb.m_target, as, &o, sizeof (o),
            addr) == -1)
                return (addr);

        if (mdb_tgt_awrite(mdb.m_target, as, &n, sizeof (n), addr) == -1)
                return (addr);

        if (rdback) {
                if (mdb_tgt_aread(mdb.m_target, as, &n, sizeof (n), addr) == -1)
                        return (addr);
```

```
 125                 mdb_iob_printf(mdb.m_out, "%-#*lla%16T%-#8x=%8T0x%x\n",
 126                     mdb_iob_getmargin(mdb.m_out), addr, o, n);
 127         }

 129         return (addr + sizeof (n));
 130 }
______unchanged_portion_omitted_

2734 static int
2735 cmd_step(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
2736 {
2737         int (*func)(mdb_tgt_t *, mdb_tgt_status_t *) = &mdb_tgt_step;
2738         const char *name = "single-step";

2740         if (argc > 0 && argv->a_type == MDB_TYPE_STRING) {
2741                 if (strcmp(argv->a_un.a_str, "out") == 0) {
2742                         func = &mdb_tgt_step_out;
2743                         name = "step (out)";
2744                         argv++;
2745                         argc--;
2746                 } else if (strcmp(argv->a_un.a_str, "branch") == 0) {
2747                         func = &mdb_tgt_step_branch;
2748                         name = "step (branch)";
2749                         argv++;
2750                         argc--;
2746                 } else if (strcmp(argv->a_un.a_str, "over") == 0) {
2747                         func = &mdb_tgt_next;
2748                         name = "step (over)";
2749                         argv++;
2750                         argc--;
2751                 }
2752         }

2754         return (cmd_cont_common(addr, flags, argc, argv, func, name));
2755 }
______unchanged_portion_omitted_
```

```
**********************************************************
   30681 Wed Feb 28 17:34:19 2018
new/usr/src/cmd/mdb/common/mdb/mdb_kproc.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

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

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Kernel Process View Target
 *
 * The kproc target is activated when the user is debugging a kernel using the
 * kvm target and executes a ::context dcmd to change the debugger view to one
 * of the running processes.  The kvm target's t_setcontext operation will
 * create and activate a kproc target in response to this call.  The kproc
 * target itself is built upon the kvm target's libkvm cookie and the ability
 * to read information from the kernel itself and the ability to read the
 * address space of a particular user process with kvm_aread().  It also relies
 * on a special set of functions provided by the kvm target's mdb_ks support
 * module in order to bootstrap: specifically, given the initial proc pointer,
 * mdb_ks provides functions to return the set of address space mappings, the
 * address space pointer itself, the aux vector vector saved in the u-area,
 * and the process data model.  The kproc target maintains a list of address
 * space mappings (kp_map_t) and load objects (kp_file_t), and for each load
 * object will attempt to read the corresponding dynamic symbol table.  In
 * order to bootstrap, the target uses the AT_BASE and AT_ENTRY aux vector
 * elements to locate the dynamic linker and executable mappings.  With these
 * mappings in place, we initialize a librtld_db agent on the target (see
 * mdb_pservice.c for how this is done), and then process each load object
 * found in the link-map chain.  In order to simplify the construction of
 * symbol tables for each load object, we would like make use of our existing
 * library of GElf processing code.  Since the MDB GElf code uses mdb_io
 * objects to read in an ELF file, we simply define a new type of mdb_io object
 * where each read operation is translated into a call to kproc's t_vread
 * function to read from the range of the address space defined by the mapping
 * as if it were a file.
 */
```



```
#include <sys/types.h>
#include <sys/proc.h>
#include <sys/auxv.h>

#include <strings.h>
#include <limits.h>
#include <rtld_db.h>
#include <procfs.h>
#include <dlfcn.h>
#include <kvm.h>

#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_string.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_ks.h>
#include <mdb/mdb_gelf.h>
#include <mdb/mdb_io_impl.h>
#include <mdb/mdb.h>

typedef struct kp_symarg {
        mdb_tgt_sym_f *sym_cb;          /* Caller's callback function */
        void *sym_data;                 /* Callback function argument */
        uint_t sym_type;                /* Symbol type/binding filter */
        uintptr_t sym_adjust;           /* Symbol value adjustment */
        mdb_syminfo_t sym_info;         /* Symbol id and table id */
        const char *sym_obj;            /* Containing object */
} kp_symarg_t;
```

*unchanged_portion_omitted_*

```
static const mdb_tgt_ops_t kproc_ops = {
        (int (*)()) mdb_tgt_notsup,             /* t_setflags */
        kp_setcontext,                          /* t_setcontext */
        kp_activate,                            /* t_activate */
        kp_deactivate,                          /* t_deactivate */
        (void (*)()) mdb_tgt_nop,               /* t_periodic */
        kp_destroy,                             /* t_destroy */
        kp_name,                                /* t_name */
        kp_isa,                                 /* t_isa */
        kp_platform,                            /* t_platform */
        kp_uname,                               /* t_uname */
        kp_dmodel,                              /* t_dmodel */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_aread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_awrite */
        kp_vread,                               /* t_vread */
        kp_vwrite,                              /* t_vwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_pread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_pwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_fread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_fwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_ioread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_iowrite */
        kp_vtop,                                /* t_vtop */
        kp_lookup_by_name,                      /* t_lookup_by_name */
        kp_lookup_by_addr,                      /* t_lookup_by_addr */
        kp_symbol_iter,                         /* t_symbol_iter */
        kp_mapping_iter,                        /* t_mapping_iter */
        kp_object_iter,                         /* t_object_iter */
        kp_addr_to_map,                         /* t_addr_to_map */
        kp_name_to_map,                         /* t_name_to_map */
        (struct ctf_file *(*)()) mdb_tgt_null,  /* t_addr_to_ctf */
        (struct ctf_file *(*)()) mdb_tgt_null,  /* t_name_to_ctf */
        kp_status,                              /* t_status */
        (int (*)()) mdb_tgt_notsup,             /* t_run */
        (int (*)()) mdb_tgt_notsup,             /* t_step */
```

```
 921         (int (*)()) mdb_tgt_notsup,             /* t_step_out */
 922         (int (*)()) mdb_tgt_notsup,             /* t_step_branch */
 922         (int (*)()) mdb_tgt_notsup,             /* t_next */
 923         (int (*)()) mdb_tgt_notsup,             /* t_cont */
 924         (int (*)()) mdb_tgt_notsup,             /* t_signal */
 925         (int (*)()) mdb_tgt_null,               /* t_add_sbrkpt */
 926         (int (*)()) mdb_tgt_null,               /* t_add_vbrkpt */
 927         (int (*)()) mdb_tgt_null,               /* t_add_pwapt */
 928         (int (*)()) mdb_tgt_null,               /* t_add_vwapt */
 929         (int (*)()) mdb_tgt_null,               /* t_add_iowapt */
 930         (int (*)()) mdb_tgt_null,               /* t_add_sysenter */
 931         (int (*)()) mdb_tgt_null,               /* t_add_sysexit */
 932         (int (*)()) mdb_tgt_null,               /* t_add_signal */
 933         (int (*)()) mdb_tgt_null,               /* t_add_fault */
 934         (int (*)()) mdb_tgt_notsup,             /* t_getareg XXX */
 935         (int (*)()) mdb_tgt_notsup,             /* t_putareg XXX */
 936         (int (*)()) mdb_tgt_notsup,             /* t_stack_iter XXX */
 937         kp_auxv                                 /* t_auxv */
 938 };
_____unchanged_portion_omitted_
```

```
**********************************************************
  148244 Wed Feb 28 17:34:19 2018
new/usr/src/cmd/mdb/common/mdb/mdb_proc.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright 2018 Joyent, Inc.
 * Copyright 2015 Joyent, Inc.
 * Copyright (c) 2014 by Delphix. All rights reserved.
 */

/*
 * User Process Target
 *
 * The user process target is invoked when the -u or -p command-line options
 * are used, or when an ELF executable file or ELF core file is specified on
 * the command-line.  This target is also selected by default when no target
 * options are present.  In this case, it defaults the executable name to
 * "a.out".  If no process or core file is currently attached, the target
 * functions as a kind of virtual /dev/zero (in accordance with adb(1)
 * semantics); reads from the virtual address space return zeroes and writes
 * fail silently.  The proc target itself is designed as a wrapper around the
 * services provided by libproc.so: t->t_pshandle is set to the struct
 * ps_prochandle pointer returned as a handle by libproc.  The target also
 * opens the executable file itself using the MDB GElf services, for
 * interpreting the .symtab and .dynsym if no libproc handle has been
 * initialized, and for handling i/o to and from the object file.  Currently,
 * the only ISA-dependent portions of the proc target are the $r and ::fpregs
 * dcmds, the callbacks for t_next() and t_step_out(), and the list of named
 * registers; these are linked in from the proc_isadep.c file for each ISA and
 * called from the common code in this file.
 *
 * The user process target implements complete user process control using the
 * facilities provided by libproc.so.  The MDB execution control model and
 * an overview of software event management is described in mdb_target.c.  The
 * proc target implements breakpoints by replacing the instruction of interest
 * with a trap instruction, and then restoring the original instruction to step
 * over the breakpoint.  The idea of replacing program text with instructions
```



```
 * that transfer control to the debugger dates back as far as 1951 [1].  When
 * the target stops, we replace each breakpoint with the original instruction
 * as part of the disarm operation.  This means that no special processing is
 * required for t_vread() because the instrumented instructions will never be
 * seen by the debugger once the target stops.  Some debuggers have improved
 * start/stop performance by leaving breakpoint traps in place and then
 * handling a read from a breakpoint address as a special case.  Although this
 * improves efficiency for a source-level debugger, it runs somewhat contrary
 * to the philosophy of the low-level debugger.  Since we remove the
 * instructions, users can apply other external debugging tools to the process
 * once it has stopped (e.g. the proc(1) tools) and not be misled by MDB
 * instrumentation.  The tracing of faults, signals, system calls, and
 * watchpoints and general process inspection is implemented directly using
 * the mechanisms provided by /proc, as described originally in [2] and [3].
 *
 * References
 *
 * [1] S. Gill, "The Diagnosis Of Mistakes In Programmes on the EDSAC",
 *     Proceedings of the Royal Society Series A Mathematical and Physical
 *     Sciences, Cambridge University Press, 206(1087), May 1951, pp. 538-554.
 *
 * [2] T.J. Killian, "Processes as Files", Proceedings of the USENIX Association
 *     Summer Conference, Salt Lake City, June 1984, pp. 203-207.
 *
 * [3] Roger Faulkner and Ron Gomes, "The Process File System and Process
 *     Model in UNIX System V", Proceedings of the USENIX Association
 *     Winter Conference, Dallas, January 1991, pp. 243-252.
 */

#include <mdb/mdb_proc.h>
#include <mdb/mdb_disasm.h>
#include <mdb/mdb_signal.h>
#include <mdb/mdb_string.h>
#include <mdb/mdb_module.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_types.h>
#include <mdb/mdb.h>

#include <sys/utsname.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <termio.h>
#include <signal.h>
#include <stdio_ext.h>
#include <stdlib.h>
#include <string.h>

#define PC_FAKE         -1UL                    /* illegal pc value unequal 0 */
#define PANIC_BUFSIZE   1024

static const char PT_EXEC_PATH[] = "a.out";     /* Default executable */
static const char PT_CORE_PATH[] = "core";      /* Default core file */

static const pt_ptl_ops_t proc_lwp_ops;
static const pt_ptl_ops_t proc_tdb_ops;
static const mdb_se_ops_t proc_brkpt_ops;
static const mdb_se_ops_t proc_wapt_ops;

static int pt_setrun(mdb_tgt_t *, mdb_tgt_status_t *, int);
static void pt_activate_common(mdb_tgt_t *);
static mdb_tgt_vespec_f pt_ignore_sig;
static mdb_tgt_se_f pt_fork;
static mdb_tgt_se_f pt_exec;
```

```
 124 static int pt_lookup_by_name_thr(mdb_tgt_t *, const char *,
 125     const char *, GElf_Sym *, mdb_syminfo_t *, mdb_tgt_tid_t);
 126 static int tlsbase(mdb_tgt_t *, mdb_tgt_tid_t, Lmid_t, const char *,
 127     psaddr_t *);

 129 /*
 130  * When debugging postmortem, we don't resolve names as we may very well not
 131  * be on a system on which those names resolve.
 132  */
 133 #define PT_LIBPROC_RESOLVE(P) \
 134         (!(mdb.m_flags & MDB_FL_LMRAW) && Pstate(P) != PS_DEAD)

 136 /*
 137  * The Perror_printf() function interposes on the default, empty libproc
 138  * definition.  It will be called to report additional information on complex
 139  * errors, such as a corrupt core file.  We just pass the args to vwarn.
 140  */
 141 /*ARGSUSED*/
 142 void
 143 Perror_printf(struct ps_prochandle *P, const char *format, ...)
 144 {
 145         va_list alist;

 147         va_start(alist, format);
 148         vwarn(format, alist);
 149         va_end(alist);
 150 }
______unchanged_portion_omitted_


4662 static const mdb_tgt_ops_t proc_ops = {
4663         pt_setflags,                            /* t_setflags */
4664         (int (*)()) mdb_tgt_notsup,             /* t_setcontext */
4665         pt_activate,                            /* t_activate */
4666         pt_deactivate,                          /* t_deactivate */
4667         pt_periodic,                            /* t_periodic */
4668         pt_destroy,                             /* t_destroy */
4669         pt_name,                                /* t_name */
4670         (const char *(*)()) mdb_conf_isa,       /* t_isa */
4671         pt_platform,                            /* t_platform */
4672         pt_uname,                               /* t_uname */
4673         pt_dmodel,                              /* t_dmodel */
4674         (ssize_t (*)()) mdb_tgt_notsup,         /* t_aread */
4675         (ssize_t (*)()) mdb_tgt_notsup,         /* t_awrite */
4676         pt_vread,                               /* t_vread */
4677         pt_vwrite,                              /* t_vwrite */
4678         (ssize_t (*)()) mdb_tgt_notsup,         /* t_pread */
4679         (ssize_t (*)()) mdb_tgt_notsup,         /* t_pwrite */
4680         pt_fread,                               /* t_fread */
4681         pt_fwrite,                              /* t_fwrite */
4682         (ssize_t (*)()) mdb_tgt_notsup,         /* t_ioread */
4683         (ssize_t (*)()) mdb_tgt_notsup,         /* t_iowrite */
4684         (int (*)()) mdb_tgt_notsup,             /* t_vtop */
4685         pt_lookup_by_name,                      /* t_lookup_by_name */
4686         pt_lookup_by_addr,                      /* t_lookup_by_addr */
4687         pt_symbol_iter,                         /* t_symbol_iter */
4688         pt_mapping_iter,                        /* t_mapping_iter */
4689         pt_object_iter,                         /* t_object_iter */
4690         pt_addr_to_map,                         /* t_addr_to_map */
4691         pt_name_to_map,                         /* t_name_to_map */
4692         pt_addr_to_ctf,                         /* t_addr_to_ctf */
4693         pt_name_to_ctf,                         /* t_name_to_ctf */
4694         pt_status,                              /* t_status */
4695         pt_run,                                 /* t_run */
4696         pt_step,                                /* t_step */
4697         pt_step_out,                            /* t_step_out */
```



```
4698         (int (*)()) mdb_tgt_notsup,             /* t_step_branch */
4698         pt_next,                                /* t_next */
4699         pt_continue,                            /* t_cont */
4700         pt_signal,                              /* t_signal */
4701         pt_add_vbrkpt,                          /* t_add_vbrkpt */
4702         pt_add_sbrkpt,                          /* t_add_sbrkpt */
4703         (int (*)()) mdb_tgt_null,               /* t_add_pwapt */
4704         pt_add_vwapt,                           /* t_add_vwapt */
4705         (int (*)()) mdb_tgt_null,               /* t_add_iowapt */
4706         pt_add_sysenter,                        /* t_add_sysenter */
4707         pt_add_sysexit,                         /* t_add_sysexit */
4708         pt_add_signal,                          /* t_add_signal */
4709         pt_add_fault,                           /* t_add_fault */
4710         pt_getareg,                             /* t_getareg */
4711         pt_putareg,                             /* t_putareg */
4712         pt_stack_iter,                          /* t_stack_iter */
4713         pt_auxv                                 /* t_auxv */
4714 };
______unchanged_portion_omitted_
```

```
**********************************************************
   11832 Wed Feb 28 17:34:19 2018
new/usr/src/cmd/mdb/common/mdb/mdb_rawfile.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

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

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Raw File Target
 *
 * The raw file target is invoked whenever a file of unrecognizable type is
 * specified on the command line, or when raw file examination is forced using
 * the -f option.  If one file is specified, that file will be opened as the
 * "object" file.  If two files are specified, the second one will be opened
 * as the "core" file.  Each file is opened using the fdio backend, which
 * internally supports both byte-oriented i/o and block-oriented i/o as needed.
 */

#include <mdb/mdb_modapi.h>
#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_io_impl.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb.h>

#include <sys/dtrace.h>
#include <fcntl.h>

typedef struct rf_data {
        mdb_io_t *r_object_fio;
        mdb_io_t *r_core_fio;
} rf_data_t;
```

*unchanged_portion_omitted_*

```
static const mdb_tgt_ops_t rawfile_ops = {
        rf_setflags,                            /* t_setflags */
```



```
        (int (*)()) mdb_tgt_notsup,             /* t_setcontext */
        rf_activate,                            /* t_activate */
        rf_deactivate,                          /* t_deactivate */
        (void (*)()) mdb_tgt_nop,               /* t_periodic */
        rf_destroy,                             /* t_destroy */
        rf_name,                                /* t_name */
        (const char *(*)()) mdb_conf_isa,       /* t_isa */
        (const char *(*)()) mdb_conf_platform,  /* t_platform */
        (int (*)()) mdb_tgt_notsup,             /* t_uname */
        (int (*)()) mdb_tgt_notsup,             /* t_dmodel */
        rf_aread,                               /* t_aread */
        rf_awrite,                              /* t_awrite */
        rf_vread,                               /* t_vread */
        rf_vwrite,                              /* t_vwrite */
        rf_pread,                               /* t_pread */
        rf_pwrite,                              /* t_pwrite */
        rf_fread,                               /* t_fread */
        rf_fwrite,                              /* t_fwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_ioread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_iowrite */
        (int (*)()) mdb_tgt_notsup,             /* t_vtop */
        (int (*)()) mdb_tgt_notsup,             /* t_lookup_by_name */
        (int (*)()) mdb_tgt_notsup,             /* t_lookup_by_addr */
        (int (*)()) mdb_tgt_notsup,             /* t_symbol_iter */
        rf_mapping_iter,                        /* t_mapping_iter */
        rf_mapping_iter,                        /* t_object_iter */
        (const mdb_map_t *(*)()) mdb_tgt_null,  /* t_addr_to_map */
        (const mdb_map_t *(*)()) mdb_tgt_null,  /* t_name_to_map */
        (struct ctf_file *(*)()) mdb_tgt_null,  /* t_addr_to_ctf */
        (struct ctf_file *(*)()) mdb_tgt_null,  /* t_name_to_ctf */
        rf_status,                              /* t_status */
        (int (*)()) mdb_tgt_notsup,             /* t_run */
        (int (*)()) mdb_tgt_notsup,             /* t_step */
        (int (*)()) mdb_tgt_notsup,             /* t_step_out */
        (int (*)()) mdb_tgt_notsup,             /* t_step_branch */
        (int (*)()) mdb_tgt_notsup,             /* t_next */
        (int (*)()) mdb_tgt_notsup,             /* t_cont */
        (int (*)()) mdb_tgt_notsup,             /* t_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_vbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_sbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_pwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_vwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_iowapt */
        (int (*)()) mdb_tgt_null,               /* t_add_sysenter */
        (int (*)()) mdb_tgt_null,               /* t_add_sysexit */
        (int (*)()) mdb_tgt_null,               /* t_add_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_fault */
        (int (*)()) mdb_tgt_notsup,             /* t_getareg */
        (int (*)()) mdb_tgt_notsup,             /* t_putareg */
        (int (*)()) mdb_tgt_notsup,             /* t_stack_iter */
        (int (*)()) mdb_tgt_notsup              /* t_auxv */
};
```

*unchanged_portion_omitted_*

```
**********************************************************
   64895 Wed Feb 28 17:34:19 2018
new/usr/src/cmd/mdb/common/mdb/mdb_target.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

/*
 * MDB Target Layer
 *
 * The *target* is the program being inspected by the debugger.  The MDB target
 * layer provides a set of functions that insulate common debugger code,
 * including the MDB Module API, from the implementation details of how the
 * debugger accesses information from a given target.  Each target exports a
 * standard set of properties, including one or more address  spaces, one or
 * more symbol tables, a set of load objects, and a set of threads that can be
 * examined using the interfaces in <mdb/mdb_target.h>.  This technique has
 * been employed successfully in other debuggers, including [1], primarily
 * to improve portability, although the term "target" often refers to the
 * encapsulation of architectural or operating system-specific details.  The
 * target abstraction is useful for MDB because it allows us to easily extend
 * the debugger to examine a variety of different program forms.  Primarily,
 * the target functions validate input arguments and then call an appropriate
 * function in the target ops vector, defined in <mdb/mdb_target_impl.h>.
 * However, this interface layer provides a very high level of flexibility for
 * separating the debugger interface from instrumentation details.  Experience
 * has shown this kind of design can facilitate separating out debugger
 * instrumentation into an external agent [2] and enable the development of
 * advanced instrumentation frameworks [3].  We want MDB to be an ideal
 * extensible framework for the development of such applications.
 *
 * Aside from a set of wrapper functions, the target layer also provides event
 * management for targets that represent live executing programs.  Our model of
 * events is also extensible, and is based upon work in [3] and [4].  We define
 * a *software event* as a state transition in the target program (for example,
 * the transition of the program counter to a location of interest) that is
 * observed by the debugger or its agent.  A *software event specifier* is a
 * description of a class of software events that is used by the debugger to
```



```
 * instrument the target so that the corresponding software events can be
 * observed.  In MDB, software event specifiers are represented by the
 * mdb_sespec_t structure, defined in <mdb/mdb_target_impl.h>.  As the user,
 * the internal debugger code, and MDB modules may all wish to observe software
 * events and receive appropriate notification and callbacks, we do not expose
 * software event specifiers directly as part of the user interface.  Instead,
 * clients of the target layer request that events be observed by creating
 * new *virtual event specifiers*.  Each virtual specifier is named by a unique
 * non-zero integer (the VID), and is represented by a mdb_vespec_t structure.
 * One or more virtual specifiers are then associated with each underlying
 * software event specifier.  This design enforces the constraint that the
 * target must only insert one set of instrumentation, regardless of how many
 * times the target layer was asked to trace a given event.  For example, if
 * multiple clients request a breakpoint at a particular address, the virtual
 * specifiers will map to the same sespec, ensuring that only one breakpoint
 * trap instruction is actually planted at the given target address.  When no
 * virtual specifiers refer to an sespec, it is no longer needed and can be
 * removed, along with the corresponding instrumentation.
 *
 * The following state transition diagram illustrates the life cycle of a
 * software event specifier and example transitions:
 *
 *                                          cont/
 *     +--------+   delete   +--------+    stop    +-------+
 *   (|( DEAD )|) <------- (  ACTIVE  ) <------> (  ARMED  )
 *     +--------+           +--------+            +-------+
 *          ^   load/unload  ^        ^   failure/     |
 *   delete |        object /          \  reset        | failure
 *          |             v            v              |
 *          |      +--------+         +-------+       |
 *          +---- (   IDLE   )        (   ERR   ) <----+
 *          |      +--------+         +-------+
 *          |                            |
 *          +----------------------------+
 *
 * The MDB execution control model is based upon the synchronous debugging
 * model exported by Solaris proc(4).  A target program is set running or the
 * debugger is attached to a running target.  On ISTOP (stop on event of
 * interest), one target thread is selected as the representative.  The
 * algorithm for selecting the representative is target-specific, but we assume
 * that if an observed software event has occurred, the target will select the
 * thread that triggered the state transition of interest.  The other threads
 * are stopped in sympathy with the representative as soon as possible.  Prior
 * to continuing the target, we plant our instrumentation, transitioning event
 * specifiers from the ACTIVE to the ARMED state, and then back again when the
 * target stops.  We then query each active event specifier to learn which ones
 * are matched, and then invoke the callbacks associated with their vespecs.
 * If an OS error occurs while attempting to arm or disarm a specifier, the
 * specifier is transitioned to the ERROR state; we will attempt to arm it
 * again at the next continue.  If no target process is under our control or
 * if an event is not currently applicable (e.g. a deferred breakpoint on an
 * object that is not yet loaded), it remains in the IDLE state.  The target
 * implementation should intercept object load events and then transition the
 * specifier to the ACTIVE state when the corresponding object is loaded.
 *
 * To simplify the debugger implementation and allow targets to easily provide
 * new types of observable events, most of the event specifier management is
 * done by the target layer.  Each software event specifier provides an ops
 * vector of subroutines that the target layer can call to perform the
 * various state transitions described above.  The target maintains two lists
 * of mdb_sespec_t's: the t_idle list (IDLE state) and the t_active list
 * (ACTIVE, ARMED, and ERROR states).  Each mdb_sespec_t maintains a list of
 * associated mdb_vespec_t's.  If an sespec is IDLE or ERROR, its se_errno
 * field will have an errno value specifying the reason for its inactivity.
 * The vespec stores the client's callback function and private data, and the
 * arguments used to construct the sespec.  All objects are reference counted
```

```
 125  * so we can destroy an object when it is no longer needed.  The mdb_sespec_t
 126  * invariants for the respective states are as follows:
 127  *
 128  *   IDLE: on t_idle list, se_data == NULL, se_errno != 0, se_ctor not called
 129  * ACTIVE: on t_active list, se_data valid, se_errno == 0, se_ctor called
 130  *  ARMED: on t_active list, se_data valid, se_errno == 0, se_ctor called
 131  *  ERROR: on t_active list, se_data valid, se_errno != 0, se_ctor called
 132  *
 133  * Additional commentary on specific state transitions and issues involving
 134  * event management can be found below near the target layer functions.
 135  *
 136  * References
 137  *
 138  * [1] John Gilmore, "Working in GDB", Technical Report, Cygnus Support,
 139  *     1.84 edition, 1994.
 140  *
 141  * [2] David R. Hanson and Mukund Raghavachari, "A Machine-Independent
 142  *     Debugger", Software--Practice and Experience, 26(11), 1277-1299(1996).
 143  *
 144  * [3] Michael W. Shapiro, "RDB: A System for Incremental Replay Debugging",
 145  *     Technical Report CS-97-12, Department of Computer Science,
 146  *     Brown University.
 147  *
 148  * [4] Daniel B. Price, "New Techniques for Replay Debugging", Technical
 149  *     Report CS-98-05, Department of Computer Science, Brown University.
 150  */

 152 #include <mdb/mdb_target_impl.h>
 153 #include <mdb/mdb_debug.h>
 154 #include <mdb/mdb_modapi.h>
 155 #include <mdb/mdb_err.h>
 156 #include <mdb/mdb_callb.h>
 157 #include <mdb/mdb_gelf.h>
 158 #include <mdb/mdb_io_impl.h>
 159 #include <mdb/mdb_string.h>
 160 #include <mdb/mdb_signal.h>
 161 #include <mdb/mdb_frame.h>
 162 #include <mdb/mdb.h>

 164 #include <sys/stat.h>
 165 #include <sys/param.h>
 166 #include <sys/signal.h>
 167 #include <strings.h>
 168 #include <stdlib.h>
 169 #include <errno.h>

 171 /*
 172  * Define convenience macros for referencing the set of vespec flag bits that
 173  * are preserved by the target implementation, and the set of bits that
 174  * determine automatic ve_hits == ve_limit behavior.
 175  */
 176 #define T_IMPL_BITS     \
 177         (MDB_TGT_SPEC_INTERNAL | MDB_TGT_SPEC_SILENT | MDB_TGT_SPEC_MATCHED | \
 178         MDB_TGT_SPEC_DELETED)

 180 #define T_AUTO_BITS     \
 181         (MDB_TGT_SPEC_AUTOSTOP | MDB_TGT_SPEC_AUTODEL | MDB_TGT_SPEC_AUTODIS)

 183 /*
 184  * Define convenience macro for referencing target flag pending continue bits.
 185  */
 186 #define T_CONT_BITS     \
 187         (MDB_TGT_F_STEP | MDB_TGT_F_STEP_OUT | MDB_TGT_F_NEXT | MDB_TGT_F_CONT)
 185         (MDB_TGT_F_STEP | MDB_TGT_F_STEP_OUT | MDB_TGT_F_STEP_BRANCH | \
 186         MDB_TGT_F_NEXT | MDB_TGT_F_CONT)
```



```
 189 mdb_tgt_t *
 190 mdb_tgt_create(mdb_tgt_ctor_f *ctor, int flags, int argc, const char *argv[])
 191 {
 192         mdb_module_t *mp;
 193         mdb_tgt_t *t;

 195         if (flags & ~MDB_TGT_F_ALL) {
 196                 (void) set_errno(EINVAL);
 197                 return (NULL);
 198         }

 200         t = mdb_zalloc(sizeof (mdb_tgt_t), UM_SLEEP);
 201         mdb_list_append(&mdb.m_tgtlist, t);

 203         t->t_module = &mdb.m_rmod;
 204         t->t_matched = T_SE_END;
 205         t->t_flags = flags;
 206         t->t_vepos = 1;
 207         t->t_veneg = 1;

 209         for (mp = mdb.m_mhead; mp != NULL; mp = mp->mod_next) {
 210                 if (ctor == mp->mod_tgt_ctor) {
 211                         t->t_module = mp;
 212                         break;
 213                 }
 214         }

 216         if (ctor(t, argc, argv) != 0) {
 217                 mdb_list_delete(&mdb.m_tgtlist, t);
 218                 mdb_free(t, sizeof (mdb_tgt_t));
 219                 return (NULL);
 220         }

 222         mdb_dprintf(MDB_DBG_TGT, "t_create %s (%p)\n",
 223             t->t_module->mod_name, (void *)t);

 225         (void) t->t_ops->t_status(t, &t->t_status);
 226         return (t);
 227 }
_____unchanged_portion_omitted_

1047 /*
1048  * This function provides the low-level target continue algorithm.  We proceed
1049  * in three phases: (1) we arm the active sespecs, except the specs matched at
1050  * the time we last stopped, (2) we call se_cont() on any matched sespecs to
1051  * step over these event transitions, and then arm the corresponding sespecs,
1052  * and (3) we call the appropriate low-level continue routine.  Once the
1053  * target stops again, we determine which sespecs were matched, and invoke the
1054  * appropriate vespec callbacks and perform other vespec maintenance.
1055  */
1056 static int
1057 tgt_continue(mdb_tgt_t *t, mdb_tgt_status_t *tsp,
1058     int (*t_cont)(mdb_tgt_t *, mdb_tgt_status_t *))
1059 {
1060         mdb_var_t *hitv = mdb_nv_lookup(&mdb.m_nv, "hits");
1061         uintptr_t pc = t->t_status.st_pc;
1062         int error = 0;

1064         mdb_sespec_t *sep, *nsep, *matched;
1065         mdb_vespec_t *vep, *nvep;
1066         uintptr_t addr;

1068         uint_t cbits = 0;       /* union of pending continue bits */
1069         uint_t ncont = 0;       /* # of callbacks that requested cont */
1070         uint_t n = 0;           /* # of callbacks */
```

```
1072         /*
1073          * If the target is undead, dead, or lost, we no longer allow continue.
1074          * This effectively forces the user to use ::kill or ::run after death.
1075          */
1076         if (t->t_status.st_state == MDB_TGT_UNDEAD)
1077                 return (set_errno(EMDB_TGTZOMB));
1078         if (t->t_status.st_state == MDB_TGT_DEAD)
1079                 return (set_errno(EMDB_TGTCORE));
1080         if (t->t_status.st_state == MDB_TGT_LOST)
1081                 return (set_errno(EMDB_TGTLOST));

1083         /*
1084          * If any of single-step, step-over, or step-out is pending, it takes
1085          * precedence over an explicit or pending continue, because these are
1086          * all different specialized forms of continue.
1087          */
1088         if (t->t_flags & MDB_TGT_F_STEP)
1089                 t_cont = t->t_ops->t_step;
1090         else if (t->t_flags & MDB_TGT_F_NEXT)
1091                 t_cont = t->t_ops->t_step;
1091         else if (t->t_flags & MDB_TGT_F_STEP_BRANCH)
1092                 t_cont = t->t_ops->t_cont;
1092         else if (t->t_flags & MDB_TGT_F_STEP_OUT)
1093                 t_cont = t->t_ops->t_cont;

1095         /*
1096          * To handle step-over, we ask the target to find the address past the
1097          * next control transfer instruction.  If an address is found, we plant
1098          * a temporary breakpoint there and continue; otherwise just step.
1099          */
1100         if ((t->t_flags & MDB_TGT_F_NEXT) && !(t->t_flags & MDB_TGT_F_STEP)) {
1101                 if (t->t_ops->t_next(t, &addr) == -1 || mdb_tgt_add_vbrkpt(t,
1102                     addr, MDB_TGT_SPEC_HIDDEN | MDB_TGT_SPEC_TEMPORARY,
1103                     no_se_f, NULL) == 0) {
1104                         mdb_dprintf(MDB_DBG_TGT, "next falling back to step: "
1105                             "%s\n", mdb_strerror(errno));
1106                 } else
1107                         t_cont = t->t_ops->t_cont;
1108         }

1110         /*
1111          * To handle step-out, we ask the target to find the return address of
1112          * the current frame, plant a temporary breakpoint there, and continue.
1113          */
1114         if (t->t_flags & MDB_TGT_F_STEP_OUT) {
1115                 if (t->t_ops->t_step_out(t, &addr) == -1)
1116                         return (-1); /* errno is set for us */

1118                 if (mdb_tgt_add_vbrkpt(t, addr, MDB_TGT_SPEC_HIDDEN |
1119                     MDB_TGT_SPEC_TEMPORARY, no_se_f, NULL) == 0)
1120                         return (-1); /* errno is set for us */
1121         }

1124         /*
1125          * To handle step-branch, we ask the target to enable it for the coming
1126          * continue.  Step-branch is incompatible with step, so don't enable it
1127          * if we're going to be stepping.
1128          */
1129         if (t->t_flags & MDB_TGT_F_STEP_BRANCH && t_cont == t->t_ops->t_cont) {
1130                 if (t->t_ops->t_step_branch(t) == -1)
1131                         return (-1); /* errno is set for us */
1132         }

1123         (void) mdb_signal_block(SIGHUP);
1124         (void) mdb_signal_block(SIGTERM);
1125         mdb_intr_disable();
```



```
1127         t->t_flags &= ~T_CONT_BITS;
1128         t->t_flags |= MDB_TGT_F_BUSY;
1129         mdb_tgt_sespec_arm_all(t);

1131         ASSERT(t->t_matched != NULL);
1132         matched = t->t_matched;
1133         t->t_matched = T_SE_END;

1135         if (mdb.m_term != NULL)
1136                 IOP_SUSPEND(mdb.m_term);

1138         /*
1139          * Iterate over the matched sespec list, performing autostop processing
1140          * and clearing the matched bit for each associated vespec.  We then
1141          * invoke each sespec's se_cont callback in order to continue past
1142          * the corresponding event.  If the matched list has more than one
1143          * sespec, we assume that the se_cont callbacks are non-interfering.
1144          */
1145         for (sep = matched; sep != T_SE_END; sep = sep->se_matched) {
1146                 for (vep = mdb_list_next(&sep->se_velist); vep != NULL; ) {
1147                         if ((vep->ve_flags & MDB_TGT_SPEC_AUTOSTOP) &&
1148                             (vep->ve_limit && vep->ve_hits == vep->ve_limit))
1149                                 vep->ve_hits = 0;

1151                         vep->ve_flags &= ~MDB_TGT_SPEC_MATCHED;
1152                         vep = mdb_list_next(vep);
1153                 }

1155                 if (sep->se_ops->se_cont(t, sep, &t->t_status) == -1) {
1156                         error = errno ? errno : -1;
1157                         tgt_disarm_sespecs(t);
1158                         break;
1159                 }

1161                 if (!(t->t_status.st_flags & MDB_TGT_ISTOP)) {
1162                         tgt_disarm_sespecs(t);
1163                         if (t->t_status.st_state == MDB_TGT_UNDEAD)
1164                                 mdb_tgt_sespec_idle_all(t, EMDB_TGTZOMB, TRUE);
1165                         else if (t->t_status.st_state == MDB_TGT_LOST)
1166                                 mdb_tgt_sespec_idle_all(t, EMDB_TGTLOST, TRUE);
1167                         break;
1168                 }
1169         }

1171         /*
1172          * Clear the se_matched field for each matched sespec, and drop the
1173          * reference count since the sespec is no longer on the matched list.
1174          */
1175         for (sep = matched; sep != T_SE_END; sep = nsep) {
1176                 nsep = sep->se_matched;
1177                 sep->se_matched = NULL;
1178                 mdb_tgt_sespec_rele(t, sep);
1179         }

1181         /*
1182          * If the matched list was non-empty, see if we hit another event while
1183          * performing se_cont() processing.  If so, don't bother continuing any
1184          * further.  If not, arm the sespecs on the old matched list by calling
1185          * mdb_tgt_sespec_arm_all() again and then continue by calling t_cont.
1186          */
1187         if (matched != T_SE_END) {
1188                 if (error != 0 || !(t->t_status.st_flags & MDB_TGT_ISTOP))
1189                         goto out; /* abort now if se_cont() failed */

1191                 if ((t->t_matched = tgt_match_sespecs(t, FALSE)) != T_SE_END) {
```

```
1192                        tgt_disarm_sespecs(t);
1193                        goto out;
1194                }

1196                mdb_tgt_sespec_arm_all(t);
1197        }

1199        if (t_cont != t->t_ops->t_step || pc == t->t_status.st_pc) {
1200                if (t_cont(t, &t->t_status) != 0)
1201                        error = errno ? errno : -1;
1202        }

1204        tgt_disarm_sespecs(t);

1206        if (t->t_flags & MDB_TGT_F_UNLOAD)
1207                longjmp(mdb.m_frame->f_pcb, MDB_ERR_QUIT);

1209        if (t->t_status.st_state == MDB_TGT_UNDEAD)
1210                mdb_tgt_sespec_idle_all(t, EMDB_TGTZOMB, TRUE);
1211        else if (t->t_status.st_state == MDB_TGT_LOST)
1212                mdb_tgt_sespec_idle_all(t, EMDB_TGTLOST, TRUE);
1213        else if (t->t_status.st_flags & MDB_TGT_ISTOP)
1214                t->t_matched = tgt_match_sespecs(t, TRUE);
1215 out:
1216        if (mdb.m_term != NULL)
1217                IOP_RESUME(mdb.m_term);

1219        (void) mdb_signal_unblock(SIGTERM);
1220        (void) mdb_signal_unblock(SIGHUP);
1221        mdb_intr_enable();

1223        for (sep = t->t_matched; sep != T_SE_END; sep = sep->se_matched) {
1224                /*
1225                 * When we invoke a ve_callback, it may in turn request that the
1226                 * target continue immediately after callback processing is
1227                 * complete.  We only allow this to occur if *all* callbacks
1228                 * agree to continue.  To implement this behavior, we keep a
1229                 * count (ncont) of such requests, and only apply the cumulative
1230                 * continue bits (cbits) to the target if ncont is equal to the
1231                 * total number of callbacks that are invoked (n).
1232                 */
1233                for (vep = mdb_list_next(&sep->se_velist);
1234                    vep != NULL; vep = nvep, n++) {
1235                        /*
1236                         * Place an extra hold on the current vespec and pick
1237                         * up the next pointer before invoking the callback: we
1238                         * must be prepared for the vespec to be deleted or
1239                         * moved to a different list by the callback.
1240                         */
1241                        mdb_tgt_vespec_hold(t, vep);
1242                        nvep = mdb_list_next(vep);

1244                        vep->ve_flags |= MDB_TGT_SPEC_MATCHED;
1245                        vep->ve_hits++;

1247                        mdb_nv_set_value(mdb.m_dot, t->t_status.st_pc);
1248                        mdb_nv_set_value(hitv, vep->ve_hits);

1250                        ASSERT((t->t_flags & T_CONT_BITS) == 0);
1251                        vep->ve_callback(t, vep->ve_id, vep->ve_data);

1253                        ncont += (t->t_flags & T_CONT_BITS) != 0;
1254                        cbits |= (t->t_flags & T_CONT_BITS);
1255                        t->t_flags &= ~T_CONT_BITS;

1257                        if (vep->ve_limit && vep->ve_hits == vep->ve_limit) {
```



```
1258                                if (vep->ve_flags & MDB_TGT_SPEC_AUTODEL)
1259                                        (void) mdb_tgt_vespec_delete(t,
1260                                            vep->ve_id);
1261                                else if (vep->ve_flags & MDB_TGT_SPEC_AUTODIS)
1262                                        (void) mdb_tgt_vespec_disable(t,
1263                                            vep->ve_id);
1264                        }

1266                        if (vep->ve_limit && vep->ve_hits < vep->ve_limit) {
1267                                if (vep->ve_flags & MDB_TGT_SPEC_AUTOSTOP)
1268                                        (void) mdb_tgt_continue(t, NULL);
1269                        }

1271                        mdb_tgt_vespec_rele(t, vep);
1272                }
1273        }

1275        if (t->t_matched != T_SE_END && ncont == n)
1276                t->t_flags |= cbits; /* apply continues (see above) */

1278        mdb_tgt_sespec_prune_all(t);

1280        t->t_status.st_flags &= ~MDB_TGT_BUSY;
1281        t->t_flags &= ~MDB_TGT_F_BUSY;

1283        if (tsp != NULL)
1284                bcopy(&t->t_status, tsp, sizeof (mdb_tgt_status_t));

1286        if (error != 0)
1287                return (set_errno(error));

1289        return (0);
1290 }
______unchanged_portion_omitted_

1398 int
1399 mdb_tgt_step_branch(mdb_tgt_t *t, mdb_tgt_status_t *tsp)
1400 {
1401        t->t_flags |= MDB_TGT_F_STEP_BRANCH; /* set flag even if tgt not busy */
1402        return (tgt_request_continue(t, tsp, 0, t->t_ops->t_cont));
1403 }

1387 int
1388 mdb_tgt_next(mdb_tgt_t *t, mdb_tgt_status_t *tsp)
1389 {
1390        t->t_flags |= MDB_TGT_F_NEXT; /* set flag even if tgt not busy */
1391        return (tgt_request_continue(t, tsp, 0, t->t_ops->t_step));
1392 }
______unchanged_portion_omitted_
```

```
**********************************************************
   22252 Wed Feb 28 17:34:20 2018
new/usr/src/cmd/mdb/common/mdb/mdb_target.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License, Version 1.0 only
   6  * (the "License").  You may not use this file except in compliance
   7  * with the License.
   8  *
   9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
  10  * or http://www.opensolaris.org/os/licensing.
  11  * See the License for the specific language governing permissions
  12  * and limitations under the License.
  13  *
  14  * When distributing Covered Code, include this CDDL HEADER in each
  15  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  16  * If applicable, add the following below this CDDL HEADER, with the
  17  * fields enclosed by brackets "[]" replaced with your own identifying
  18  * information: Portions Copyright [yyyy] [name of copyright owner]
  19  *
  20  * CDDL HEADER END
  21  */
  22 /*
  23  * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
  24  * Use is subject to license terms.
  25  *
  26  * Copyright 2018 Joyent, Inc.
  27  */

  29 #ifndef _MDB_TARGET_H
  30 #define _MDB_TARGET_H

  32 #include <sys/utsname.h>
  33 #include <sys/types.h>
  34 #include <gelf.h>

  36 #ifdef  __cplusplus
  37 extern "C" {
  38 #endif

  40 /*
  41  * Forward declaration of the target structure: the target itself is defined in
  42  * mdb_tgt_impl.h and is opaque with respect to callers of this interface.
  43  */

  45 struct mdb_tgt;
  46 struct mdb_arg;
  47 struct ctf_file;

  49 typedef struct mdb_tgt mdb_tgt_t;

  51 extern void mdb_create_builtin_tgts(void);
  52 extern void mdb_create_loadable_disasms(void);

  54 /*
  55  * Target Constructors
  56  *
  57  * These functions are used to create a complete debugger target.  The
  58  * constructor is passed as an argument to mdb_tgt_create().
```



```
  59  */

  61 extern int mdb_value_tgt_create(mdb_tgt_t *, int, const char *[]);
  62 #ifndef _KMDB
  63 extern int mdb_kvm_tgt_create(mdb_tgt_t *, int, const char *[]);
  64 extern int mdb_proc_tgt_create(mdb_tgt_t *, int, const char *[]);
  65 extern int mdb_kproc_tgt_create(mdb_tgt_t *, int, const char *[]);
  66 extern int mdb_rawfile_tgt_create(mdb_tgt_t *, int, const char *[]);
  67 #else
  68 extern int kmdb_kvm_create(mdb_tgt_t *, int, const char *[]);
  69 #endif

  71 /*
  72  * Targets are created by calling mdb_tgt_create() with an optional set of
  73  * target flags, an argument list, and a target constructor (see above):
  74  */

  76 #define MDB_TGT_F_RDWR          0x0001  /* Open for writing (else read-only) */
  77 #define MDB_TGT_F_ALLOWIO       0x0002  /* Allow I/O mem access (live only) */
  78 #define MDB_TGT_F_FORCE         0x0004  /* Force open (even if non-exclusive) */
  79 #define MDB_TGT_F_PRELOAD       0x0008  /* Preload all symbol tables */
  80 #define MDB_TGT_F_NOLOAD        0x0010  /* Do not do load-object processing */
  81 #define MDB_TGT_F_NOSTOP        0x0020  /* Do not stop target on attach */
  82 #define MDB_TGT_F_STEP          0x0040  /* Single-step is pending */
  83 #define MDB_TGT_F_STEP_OUT      0x0080  /* Step-out is pending */
  84 #define MDB_TGT_F_NEXT          0x0100  /* Step-over is pending */
  85 #define MDB_TGT_F_CONT          0x0200  /* Continue is pending */
  86 #define MDB_TGT_F_BUSY          0x0400  /* Target is busy executing */
  87 #define MDB_TGT_F_ASIO          0x0800  /* Use t_aread and t_awrite for i/o */
  88 #define MDB_TGT_F_UNLOAD        0x1000  /* Unload has been requested */
  89 #define MDB_TGT_F_ALL           0x1fff  /* Mask of all valid flags */
  82 #define MDB_TGT_F_STEP_BRANCH   0x0100  /* Step-branch is pending */
  83 #define MDB_TGT_F_NEXT          0x0200  /* Step-over is pending */
  84 #define MDB_TGT_F_CONT          0x0400  /* Continue is pending */
  85 #define MDB_TGT_F_BUSY          0x0800  /* Target is busy executing */
  86 #define MDB_TGT_F_ASIO          0x1000  /* Use t_aread and t_awrite for i/o */
  87 #define MDB_TGT_F_UNLOAD        0x2000  /* Unload has been requested */
  88 #define MDB_TGT_F_ALL           0x3fff  /* Mask of all valid flags */

  91 typedef int mdb_tgt_ctor_f(mdb_tgt_t *, int, const char *[]);

  93 extern mdb_tgt_t *mdb_tgt_create(mdb_tgt_ctor_f *, int, int, const char *[]);
  94 extern void mdb_tgt_destroy(mdb_tgt_t *);

  96 extern int mdb_tgt_getflags(mdb_tgt_t *);
  97 extern int mdb_tgt_setflags(mdb_tgt_t *, int);
  98 extern int mdb_tgt_setcontext(mdb_tgt_t *, void *);

 100 /*
 101  * Targets are activated and de-activated by the debugger framework.  An
 102  * activation occurs after construction when the target becomes the current
 103  * target in the debugger.  A target is de-activated prior to its destructor
 104  * being called by mdb_tgt_destroy, or when another target is activated.
 105  * These callbacks are suitable for loading support modules and other tasks.
 106  */
 107 extern void mdb_tgt_activate(mdb_tgt_t *);

 109 /*
 110  * Prior to issuing a new command prompt, the debugger framework calls the
 111  * target's periodic callback to allow it to load new modules or perform
 112  * other background tasks.
 113  */
 114 extern void mdb_tgt_periodic(mdb_tgt_t *);

 116 /*
 117  * Convenience functions for accessing miscellaneous target information.
```

```
 118  */
 119 extern const char *mdb_tgt_name(mdb_tgt_t *);
 120 extern const char *mdb_tgt_isa(mdb_tgt_t *);
 121 extern const char *mdb_tgt_platform(mdb_tgt_t *);
 122 extern int mdb_tgt_uname(mdb_tgt_t *, struct utsname *);
 123 extern int mdb_tgt_dmodel(mdb_tgt_t *);

 125 /*
 126  * Address Space Interface
 127  *
 128  * Each target can provide access to a set of address spaces, which may include
 129  * a primary virtual address space, a physical address space, an object file
 130  * address space (where virtual addresses are converted to file offsets in an
 131  * object file), and an I/O port address space.  Additionally, the target can
 132  * provide access to alternate address spaces, which are identified by the
 133  * opaque mdb_tgt_as_t type.  If the 'as' parameter to mdb_tgt_aread or
 134  * mdb_tgt_awrite is one of the listed constants, these calls are equivalent
 135  * to mdb_tgt_{v|p|f|io}read or write.
 136  */

 138 typedef void *          mdb_tgt_as_t;           /* Opaque address space id */
 139 typedef uint64_t        mdb_tgt_addr_t;         /* Generic unsigned address */
 140 typedef uint64_t        physaddr_t;             /* Physical memory address */

 142 #define MDB_TGT_AS_VIRT ((mdb_tgt_as_t)-1L)     /* Virtual address space */
 143 #define MDB_TGT_AS_PHYS ((mdb_tgt_as_t)-2L)     /* Physical address space */
 144 #define MDB_TGT_AS_FILE ((mdb_tgt_as_t)-3L)     /* Object file address space */
 145 #define MDB_TGT_AS_IO   ((mdb_tgt_as_t)-4L)     /* I/o address space */

 147 extern ssize_t mdb_tgt_aread(mdb_tgt_t *, mdb_tgt_as_t,
 148         void *, size_t, mdb_tgt_addr_t);

 150 extern ssize_t mdb_tgt_awrite(mdb_tgt_t *, mdb_tgt_as_t,
 151         const void *, size_t, mdb_tgt_addr_t);

 153 extern ssize_t mdb_tgt_vread(mdb_tgt_t *, void *, size_t, uintptr_t);
 154 extern ssize_t mdb_tgt_vwrite(mdb_tgt_t *, const void *, size_t, uintptr_t);
 155 extern ssize_t mdb_tgt_pread(mdb_tgt_t *, void *, size_t, physaddr_t);
 156 extern ssize_t mdb_tgt_pwrite(mdb_tgt_t *, const void *, size_t, physaddr_t);
 157 extern ssize_t mdb_tgt_fread(mdb_tgt_t *, void *, size_t, uintptr_t);
 158 extern ssize_t mdb_tgt_fwrite(mdb_tgt_t *, const void *, size_t, uintptr_t);
 159 extern ssize_t mdb_tgt_ioread(mdb_tgt_t *, void *, size_t, uintptr_t);
 160 extern ssize_t mdb_tgt_iowrite(mdb_tgt_t *, const void *, size_t, uintptr_t);

 162 /*
 163  * Convert an address-space's virtual address to the corresponding
 164  * physical address (only useful for kernel targets):
 165  */
 166 extern int mdb_tgt_vtop(mdb_tgt_t *, mdb_tgt_as_t, uintptr_t, physaddr_t *);

 168 /*
 169  * Convenience functions for reading and writing null-terminated
 170  * strings from any of the target address spaces:
 171  */
 172 extern ssize_t mdb_tgt_readstr(mdb_tgt_t *, mdb_tgt_as_t,
 173         char *, size_t, mdb_tgt_addr_t);

 175 extern ssize_t mdb_tgt_writestr(mdb_tgt_t *, mdb_tgt_as_t,
 176         const char *, mdb_tgt_addr_t);

 178 /*
 179  * Symbol Table Interface
 180  *
 181  * Each target can provide access to one or more symbol tables, which can be
 182  * iterated over, or used to lookup symbols by either name or address.  The
 183  * target can support a primary executable and primary dynamic symbol table,
```



```
 184  * a symbol table for its run-time link-editor, and symbol tables for one or
 185  * more loaded objects.  A symbol is uniquely identified by an object name,
 186  * a symbol table id, and a symbol id.  Symbols can be discovered by iterating
 187  * over them, looking them up by name, or looking them up by address.
 188  */

 190 typedef struct mdb_syminfo {
 191         uint_t sym_table;       /* Symbol table id (see symbol_iter, below) */
 192         uint_t sym_id;          /* Symbol identifier */
 193 } mdb_syminfo_t;
_____unchanged_portion_omitted_

 334 /*
 335  * Program state (st_state):
 336  * (MDB_STATE_* definitions in the module API need to be in sync with these)
 337  */
 338 #define MDB_TGT_IDLE            0       /* Target is idle (not running yet) */
 339 #define MDB_TGT_RUNNING         1       /* Target is currently executing */
 340 #define MDB_TGT_STOPPED         2       /* Target is stopped */
 341 #define MDB_TGT_UNDEAD          3       /* Target is undead (zombie) */
 342 #define MDB_TGT_DEAD            4       /* Target is dead (core dump) */
 343 #define MDB_TGT_LOST            5       /* Target lost by debugger */

 345 /*
 346  * Status flags (st_flags):
 347  */
 348 #define MDB_TGT_ISTOP           0x1     /* Stop on event of interest */
 349 #define MDB_TGT_DSTOP           0x2     /* Stop directive is pending */
 350 #define MDB_TGT_BUSY            0x4     /* Busy in debugger */

 352 extern int mdb_tgt_status(mdb_tgt_t *, mdb_tgt_status_t *);
 353 extern int mdb_tgt_run(mdb_tgt_t *, int, const struct mdb_arg *);
 354 extern int mdb_tgt_step(mdb_tgt_t *, mdb_tgt_status_t *);
 355 extern int mdb_tgt_step_out(mdb_tgt_t *, mdb_tgt_status_t *);
 355 extern int mdb_tgt_step_branch(mdb_tgt_t *, mdb_tgt_status_t *);
 356 extern int mdb_tgt_next(mdb_tgt_t *, mdb_tgt_status_t *);
 357 extern int mdb_tgt_continue(mdb_tgt_t *, mdb_tgt_status_t *);
 358 extern int mdb_tgt_signal(mdb_tgt_t *, int);

 360 /*
 361  * Iterating through the specifier list yields the integer id (VID) and private
 362  * data pointer for each specifier.
 363  */
 364 typedef int mdb_tgt_vespec_f(mdb_tgt_t *, void *, int, void *);

 366 /*
 367  * Each event specifier is defined to be in one of the following states.  The
 368  * state transitions are discussed in detail in the comments in mdb_target.c.
 369  */
 370 #define MDB_TGT_SPEC_IDLE       1       /* Inactive (e.g. object not loaded) */
 371 #define MDB_TGT_SPEC_ACTIVE     2       /* Active but not armed in target */
 372 #define MDB_TGT_SPEC_ARMED      3       /* Active and armed (e.g. bkpt set) */
 373 #define MDB_TGT_SPEC_ERROR      4       /* Failed to arm event */

 375 /*
 376  * Event specifiers may also have one or more of the following additional
 377  * properties (spec_flags bits):
 378  */
 379 #define MDB_TGT_SPEC_INTERNAL   0x0001  /* Internal to target implementation */
 380 #define MDB_TGT_SPEC_SILENT     0x0002  /* Do not describe when matched */
 381 #define MDB_TGT_SPEC_TEMPORARY  0x0004  /* Delete next time target stops */
 382 #define MDB_TGT_SPEC_MATCHED    0x0008  /* Specifier matched at last stop */
 383 #define MDB_TGT_SPEC_DISABLED   0x0010  /* Specifier cannot be armed */
 384 #define MDB_TGT_SPEC_DELETED    0x0020  /* Specifier has been deleted */
 385 #define MDB_TGT_SPEC_AUTODEL    0x0040  /* Delete when match limit reached */
 386 #define MDB_TGT_SPEC_AUTODIS    0x0080  /* Disable when match limit reached */
```

```
387 #define MDB_TGT_SPEC_AUTOSTOP   0x0100  /* Stop when match limit reached */
388 #define MDB_TGT_SPEC_STICKY     0x0200  /* Do not delete as part of :z */

390 #define MDB_TGT_SPEC_HIDDEN     (MDB_TGT_SPEC_INTERNAL | MDB_TGT_SPEC_SILENT)

392 typedef struct mdb_tgt_spec_desc {
393         int spec_id;                    /* Event specifier id (VID) */
394         uint_t spec_flags;              /* Flags (see above) */
395         uint_t spec_hits;               /* Count of number of times matched */
396         uint_t spec_limit;              /* Limit on number of times matched */
397         int spec_state;                 /* State (see above) */
398         int spec_errno;                 /* Last error code (if IDLE or ERROR) */
399         uintptr_t spec_base;            /* Start of affected memory region */
400         size_t spec_size;               /* Size of affected memory region */
401         void *spec_data;                /* Callback private data */
402 } mdb_tgt_spec_desc_t;
_____unchanged_portion_omitted_
```

```
**********************************************************
   14888 Wed Feb 28 17:34:20 2018
new/usr/src/cmd/mdb/common/mdb/mdb_target_impl.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2018, Joyent, Inc.  All rights reserved.
 * Copyright (c) 2012, Joyent, Inc.  All rights reserved.
 */

#ifndef _MDB_TARGET_IMPL_H
#define _MDB_TARGET_IMPL_H

#include <mdb/mdb_target.h>
#include <mdb/mdb_module.h>
#include <mdb/mdb_list.h>
#include <mdb/mdb_gelf.h>
#include <sys/auxv.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef _MDB

/*
 * Target Operations
 *
 * This ops vector implements the set of primitives which can be used by the
 * debugger to interact with the target, and encompasses most of the calls
 * found in <mdb/mdb_target.h>.  The remainder of the target interface is
 * implemented by common code that invokes these primitives or manipulates
 * the common target structures directly.
 */

typedef struct mdb_tgt_ops {
        int (*t_setflags)(mdb_tgt_t *, int);
        int (*t_setcontext)(mdb_tgt_t *, void *);
```



```
        void (*t_activate)(mdb_tgt_t *);
        void (*t_deactivate)(mdb_tgt_t *);
        void (*t_periodic)(mdb_tgt_t *);
        void (*t_destroy)(mdb_tgt_t *);

        const char *(*t_name)(mdb_tgt_t *);
        const char *(*t_isa)(mdb_tgt_t *);
        const char *(*t_platform)(mdb_tgt_t *);
        int (*t_uname)(mdb_tgt_t *, struct utsname *);
        int (*t_dmodel)(mdb_tgt_t *);

        ssize_t (*t_aread)(mdb_tgt_t *,
            mdb_tgt_as_t, void *, size_t, mdb_tgt_addr_t);

        ssize_t (*t_awrite)(mdb_tgt_t *,
            mdb_tgt_as_t, const void *, size_t, mdb_tgt_addr_t);

        ssize_t (*t_vread)(mdb_tgt_t *, void *, size_t, uintptr_t);
        ssize_t (*t_vwrite)(mdb_tgt_t *, const void *, size_t, uintptr_t);
        ssize_t (*t_pread)(mdb_tgt_t *, void *, size_t, physaddr_t);
        ssize_t (*t_pwrite)(mdb_tgt_t *, const void *, size_t, physaddr_t);
        ssize_t (*t_fread)(mdb_tgt_t *, void *, size_t, uintptr_t);
        ssize_t (*t_fwrite)(mdb_tgt_t *, const void *, size_t, uintptr_t);
        ssize_t (*t_ioread)(mdb_tgt_t *, void *, size_t, uintptr_t);
        ssize_t (*t_iowrite)(mdb_tgt_t *, const void *, size_t, uintptr_t);

        int (*t_vtop)(mdb_tgt_t *, mdb_tgt_as_t, uintptr_t, physaddr_t *);

        int (*t_lookup_by_name)(mdb_tgt_t *,
            const char *, const char *, GElf_Sym *, mdb_syminfo_t *);

        int (*t_lookup_by_addr)(mdb_tgt_t *,
            uintptr_t, uint_t, char *, size_t, GElf_Sym *, mdb_syminfo_t *);

        int (*t_symbol_iter)(mdb_tgt_t *,
            const char *, uint_t, uint_t, mdb_tgt_sym_f *, void *);

        int (*t_mapping_iter)(mdb_tgt_t *, mdb_tgt_map_f *, void *);
        int (*t_object_iter)(mdb_tgt_t *, mdb_tgt_map_f *, void *);

        const mdb_map_t *(*t_addr_to_map)(mdb_tgt_t *, uintptr_t);
        const mdb_map_t *(*t_name_to_map)(mdb_tgt_t *, const char *);
        struct ctf_file *(*t_addr_to_ctf)(mdb_tgt_t *, uintptr_t);
        struct ctf_file *(*t_name_to_ctf)(mdb_tgt_t *, const char *);

        int (*t_status)(mdb_tgt_t *, mdb_tgt_status_t *);
        int (*t_run)(mdb_tgt_t *, int, const struct mdb_arg *);
        int (*t_step)(mdb_tgt_t *, mdb_tgt_status_t *);
        int (*t_step_out)(mdb_tgt_t *, uintptr_t *);
        int (*t_step_branch)(mdb_tgt_t *);
        int (*t_next)(mdb_tgt_t *, uintptr_t *);
        int (*t_cont)(mdb_tgt_t *, mdb_tgt_status_t *);
        int (*t_signal)(mdb_tgt_t *, int);

        int (*t_add_vbrkpt)(mdb_tgt_t *, uintptr_t,
            int, mdb_tgt_se_f *, void *);
        int (*t_add_sbrkpt)(mdb_tgt_t *, const char *,
            int, mdb_tgt_se_f *, void *);

        int (*t_add_pwapt)(mdb_tgt_t *, physaddr_t, size_t, uint_t,
            int, mdb_tgt_se_f *, void *);
        int (*t_add_vwapt)(mdb_tgt_t *, uintptr_t, size_t, uint_t,
            int, mdb_tgt_se_f *, void *);
        int (*t_add_iowapt)(mdb_tgt_t *, uintptr_t, size_t, uint_t,
            int, mdb_tgt_se_f *, void *);
```

```
123         int (*t_add_sysenter)(mdb_tgt_t *, int, int, mdb_tgt_se_f *, void *);
124         int (*t_add_sysexit)(mdb_tgt_t *, int, int, mdb_tgt_se_f *, void *);
125         int (*t_add_signal)(mdb_tgt_t *, int, int, mdb_tgt_se_f *, void *);
126         int (*t_add_fault)(mdb_tgt_t *, int, int, mdb_tgt_se_f *, void *);

128         int (*t_getareg)(mdb_tgt_t *, mdb_tgt_tid_t, const char *,
129             mdb_tgt_reg_t *);
130         int (*t_putareg)(mdb_tgt_t *, mdb_tgt_tid_t, const char *,
131             mdb_tgt_reg_t);

133         int (*t_stack_iter)(mdb_tgt_t *, const mdb_tgt_gregset_t *,
134             mdb_tgt_stack_f *, void *);

136         int (*t_auxv)(mdb_tgt_t *, const auxv_t **auxvp);
137 } mdb_tgt_ops_t;
```

*unchanged_portion_omitted_*

```
**********************************************************
    6175 Wed Feb 28 17:34:20 2018
new/usr/src/cmd/mdb/common/mdb/mdb_value.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

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

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Immediate Value Target
 *
 * The immediate value target is used when the '=' verb is used to
 * format an immediate value, or with ::print -i.  The target is
 * initialized with a specific value, and then simply copies bytes from
 * this integer in its read routine.  Two notes:
 *
 * (1) the address parameter of value_read is treated as an offset into
 * the immediate value.
 *
 * (2) on big-endian systems, we need to be careful about the place we
 * copy data from. If the caller specified a typesize in the argv array
 * we use that for offsetting, otherwise we use the read size.
 * If the user didn't specify the typesize, then 'addr' is ignored,
 * and all reads are at an offset of 0 into the immediate value. This
 * covers both the usage of ::print -i, and the semantics of adb
 * commands like "0x1234=X", which should produce 0x1234 as a result;
 * the adb model is for it to act like a cast down to the smaller
 * integer type; this is handled as mentioned.
 */

#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_types.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_err.h>

#include <sys/isa_defs.h>
#include <strings.h>
```



```
void mdb_value_tgt_destroy(mdb_tgt_t *);

typedef struct mdb_value_data {
        uintmax_t mvd_data;
        size_t mvd_typesize;
} mdb_value_data_t;
______unchanged_portion_omitted_

static const mdb_tgt_ops_t value_ops = {
        (int (*)()) mdb_tgt_notsup,             /* t_setflags */
        (int (*)()) mdb_tgt_notsup,             /* t_setcontext */
        (void (*)()) mdb_tgt_nop,               /* t_activate */
        (void (*)()) mdb_tgt_nop,               /* t_deactivate */
        (void (*)()) mdb_tgt_nop,               /* t_periodic */
        mdb_value_tgt_destroy,                  /* t_destroy */
        (const char *(*)()) mdb_tgt_null,       /* t_name */
        (const char *(*)()) mdb_conf_isa,       /* t_isa */
        (const char *(*)()) mdb_conf_platform,  /* t_platform */
        (int (*)()) mdb_tgt_notsup,             /* t_uname */
        (int (*)()) mdb_tgt_notsup,             /* t_dmodel */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_aread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_awrite */
        value_read,                             /* t_vread */
        value_write,                            /* t_vwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_pread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_pwrite */
        value_read,                             /* t_fread */
        value_write,                            /* t_fwrite */
        value_read,                             /* t_ioread */
        value_write,                            /* t_iowrite */
        (int (*)()) mdb_tgt_notsup,             /* t_vtop */
        (int (*)()) mdb_tgt_notsup,             /* t_lookup_by_name */
        (int (*)()) mdb_tgt_notsup,             /* t_lookup_by_addr */
        (int (*)()) mdb_tgt_notsup,             /* t_symbol_iter */
        (int (*)()) mdb_tgt_notsup,             /* t_mapping_iter */
        (int (*)()) mdb_tgt_notsup,             /* t_object_iter */
        (const mdb_map_t *(*)()) mdb_tgt_null,  /* t_addr_to_map */
        (const mdb_map_t *(*)()) mdb_tgt_null,  /* t_name_to_map */
        (struct ctf_file *(*)()) mdb_tgt_null,  /* t_addr_to_ctf */
        (struct ctf_file *(*)()) mdb_tgt_null,  /* t_name_to_ctf */
        (int (*)()) mdb_tgt_notsup,             /* t_status */
        (int (*)()) mdb_tgt_notsup,             /* t_run */
        (int (*)()) mdb_tgt_notsup,             /* t_step */
        (int (*)()) mdb_tgt_notsup,             /* t_step_out */
        (int (*)()) mdb_tgt_notsup,             /* t_step_branch */
        (int (*)()) mdb_tgt_notsup,             /* t_next */
        (int (*)()) mdb_tgt_notsup,             /* t_cont */
        (int (*)()) mdb_tgt_notsup,             /* t_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_vbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_sbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_pwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_vwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_iowapt */
        (int (*)()) mdb_tgt_null,               /* t_add_sysenter */
        (int (*)()) mdb_tgt_null,               /* t_add_sysexit */
        (int (*)()) mdb_tgt_null,               /* t_add_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_fault */
        (int (*)()) mdb_tgt_notsup,             /* t_getareg */
        (int (*)()) mdb_tgt_notsup,             /* t_putareg */
        (int (*)()) mdb_tgt_nop,                /* t_stack_iter */
        (int (*)()) mdb_tgt_notsup              /* t_auxv */
};
______unchanged_portion_omitted_
```

```
**********************************************************
   25151 Wed Feb 28 17:34:20 2018
new/usr/src/cmd/mdb/i86pc/modules/unix/unix.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2018 Joyent, Inc.
 * Copyright 2015 Joyent, Inc.
 */

#include <mdb/mdb_modapi.h>
#include <mdb/mdb_ctf.h>
#include <sys/cpuvar.h>
#include <sys/systm.h>
#include <sys/traptrace.h>
#include <sys/x_call.h>
#include <sys/xc_levels.h>
#include <sys/avintr.h>
#include <sys/systm.h>
#include <sys/trap.h>
#include <sys/mutex.h>
#include <sys/mutex_impl.h>
#include "i86mmu.h"
#include "unix_sup.h"
#include <sys/apix.h>
#include <sys/x86_archext.h>
#include <sys/bitmap.h>
#include <sys/controlregs.h>

#define TT_HDLR_WIDTH   17


/* apix only */
static apix_impl_t *d_apixs[NCPU];
static int use_apix = 0;

static int
ttrace_ttr_size_check(void)
{
        mdb_ctf_id_t ttrtid;
        ssize_t ttr_size;
```



```
        if (mdb_ctf_lookup_by_name("trap_trace_rec_t", &ttrtid) != 0 ||
            mdb_ctf_type_resolve(ttrtid, &ttrtid) != 0) {
                mdb_warn("failed to determine size of trap_trace_rec_t; "
                    "non-TRAPTRACE kernel?\n");
                return (0);
        }

        if ((ttr_size = mdb_ctf_type_size(ttrtid)) !=
            sizeof (trap_trace_rec_t)) {
                /*
                 * On Intel machines, this will happen when TTR_STACK_DEPTH
                 * is changed.  This code could be smarter, and could
                 * dynamically adapt to different depths, but not until a
                 * need for such adaptation is demonstrated.
                 */
                mdb_warn("size of trap_trace_rec_t (%d bytes) doesn't "
                    "match expected %d\n", ttr_size, sizeof (trap_trace_rec_t));
                return (0);
        }

        return (1);
}
```

*_____unchanged_portion_omitted_*

```
typedef struct ttrace_dcmd {
        processorid_t ttd_cpu;
        uint_t ttd_extended;
        uintptr_t ttd_kthread;
        trap_trace_ctl_t ttd_ttc[NCPU];
} ttrace_dcmd_t;
```

*_____unchanged_portion_omitted_*

```
#endif  /* __amd64 */

int
ttrace_walk(uintptr_t addr, trap_trace_rec_t *rec, ttrace_dcmd_t *dcmd)
{
        struct regs *regs = &rec->ttr_regs;
        processorid_t cpu = -1, i;

        for (i = 0; i < NCPU; i++) {
                if (addr >= dcmd->ttd_ttc[i].ttc_first &&
                    addr < dcmd->ttd_ttc[i].ttc_limit) {
                        cpu = i;
                        break;
                }
        }

        if (cpu == -1) {
                mdb_warn("couldn't find %p in any trap trace ctl\n", addr);
                return (WALK_ERR);
        }

        if (dcmd->ttd_cpu != -1 && cpu != dcmd->ttd_cpu)
                return (WALK_NEXT);

        if (dcmd->ttd_kthread != 0 &&
            dcmd->ttd_kthread != rec->ttr_curthread)
                return (WALK_NEXT);

        mdb_printf("%3d %15llx ", cpu, rec->ttr_stamp);

        for (i = 0; ttrace_hdlr[i].t_hdlr != NULL; i++) {
                if (rec->ttr_marker != ttrace_hdlr[i].t_marker)
                        continue;
                mdb_printf("%4s ", ttrace_hdlr[i].t_name);
```

```
492                 if (ttrace_hdlr[i].t_hdlr(rec) == -1)
493                         return (WALK_ERR);
494         }

496         mdb_printf(" %a\n", regs->r_pc);

498         if (dcmd->ttd_extended == FALSE)
499                 return (WALK_NEXT);

501         if (rec->ttr_marker == TT_INTERRUPT)
502                 ttrace_intr_detail(rec);
503         else
504                 ttrace_dumpregs(rec);

506         if (rec->ttr_sdepth > 0) {
507                 for (i = 0; i < rec->ttr_sdepth; i++) {
508                         if (i >= TTR_STACK_DEPTH) {
509                                 mdb_printf("%17s*** invalid ttr_sdepth (is %d, "
510                                     "should be <= %d)\n", " ", rec->ttr_sdepth,
511                                     TTR_STACK_DEPTH);
512                                 break;
513                         }

515                         mdb_printf("%17s %a()\n", " ", rec->ttr_stack[i]);
516                 }
517                 mdb_printf("\n");
518         }

520         return (WALK_NEXT);
521 }

523 int
524 ttrace(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
525 {
526         ttrace_dcmd_t dcmd;
527         trap_trace_ctl_t *ttc = dcmd.ttd_ttc;
528         trap_trace_rec_t rec;
529         size_t ttc_size = sizeof (trap_trace_ctl_t) * NCPU;

531         if (!ttrace_ttr_size_check())
532                 return (WALK_ERR);

534         bzero(&dcmd, sizeof (dcmd));
535         dcmd.ttd_cpu = -1;
536         dcmd.ttd_extended = FALSE;

538         if (mdb_readsym(ttc, ttc_size, "trap_trace_ctl") == -1) {
539                 mdb_warn("symbol 'trap_trace_ctl' not found; "
540                     "non-TRAPTRACE kernel?\n");
541                 return (DCMD_ERR);
542         }

544         if (mdb_getopts(argc, argv,
545             'x', MDB_OPT_SETBITS, TRUE, &dcmd.ttd_extended,
546             't', MDB_OPT_UINTPTR, &dcmd.ttd_kthread, NULL) != argc)
540             'x', MDB_OPT_SETBITS, TRUE, &dcmd.ttd_extended, NULL) != argc)
547                 return (DCMD_USAGE);

549         if (DCMD_HDRSPEC(flags)) {
550                 mdb_printf("%3s %15s %4s %2s %-*s%s\n", "CPU",
551                     "TIMESTAMP", "TYPE", "Vec", TT_HDLR_WIDTH, "HANDLER",
552                     " EIP");
553         }

555         if (flags & DCMD_ADDRSPEC) {
556                 if (addr >= NCPU) {
```



```
557                         if (mdb_vread(&rec, sizeof (rec), addr) == -1) {
558                                 mdb_warn("couldn't read trap trace record "
559                                     "at %p", addr);
560                                 return (DCMD_ERR);
561                         }

563                         if (ttrace_walk(addr, &rec, &dcmd) == WALK_ERR)
564                                 return (DCMD_ERR);

566                         return (DCMD_OK);
567                 }
568                 dcmd.ttd_cpu = addr;
569         }

571         if (mdb_readvar(&use_apix, "apix_enable") == -1) {
572                 mdb_warn("failed to read apix_enable");
573                 use_apix = 0;
574         }

576         if (use_apix) {
577                 if (mdb_readvar(&d_apixs, "apixs") == -1) {
578                         mdb_warn("\nfailed to read apixs.");
579                         return (DCMD_ERR);
580                 }
581                 /* change to apix ttrace interrupt handler */
582                 ttrace_hdlr[4].t_hdlr = ttrace_apix_interrupt;
583         }

585         if (mdb_walk("ttrace", (mdb_walk_cb_t)ttrace_walk, &dcmd) == -1) {
586                 mdb_warn("couldn't walk 'ttrace'");
587                 return (DCMD_ERR);
588         }

590         return (DCMD_OK);
591 }
```

*unchanged_portion_omitted_*

```
890 #ifdef _KMDB
891 /* ARGSUSED */
892 static int
893 crregs_dcmd(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
894 {
895         ulong_t cr0, cr2, cr3, cr4;
889         ulong_t cr0, cr4;
896         static const mdb_bitmask_t cr0_flag_bits[] = {
897                 { "PE",         CR0_PE,         CR0_PE },
898                 { "MP",         CR0_MP,         CR0_MP },
899                 { "EM",         CR0_EM,         CR0_EM },
900                 { "TS",         CR0_TS,         CR0_TS },
901                 { "ET",         CR0_ET,         CR0_ET },
902                 { "NE",         CR0_NE,         CR0_NE },
903                 { "WP",         CR0_WP,         CR0_WP },
904                 { "AM",         CR0_AM,         CR0_AM },
905                 { "NW",         CR0_NW,         CR0_NW },
906                 { "CD",         CR0_CD,         CR0_CD },
907                 { "PG",         CR0_PG,         CR0_PG },
908                 { NULL,         0,              0 }
909         };

911         static const mdb_bitmask_t cr3_flag_bits[] = {
912                 { "PCD",        CR3_PCD,        CR3_PCD },
913                 { "PWT",        CR3_PWT,        CR3_PWT },
914                 { NULL,         0,              0, }
915         };

917         static const mdb_bitmask_t cr4_flag_bits[] = {
```

```
                { "VME",        CR4_VME,        CR4_VME },
                { "PVI",        CR4_PVI,        CR4_PVI },
                { "TSD",        CR4_TSD,        CR4_TSD },
                { "DE",         CR4_DE,         CR4_DE },
                { "PSE",        CR4_PSE,        CR4_PSE },
                { "PAE",        CR4_PAE,        CR4_PAE },
                { "MCE",        CR4_MCE,        CR4_MCE },
                { "PGE",        CR4_PGE,        CR4_PGE },
                { "PCE",        CR4_PCE,        CR4_PCE },
                { "OSFXSR",     CR4_OSFXSR,     CR4_OSFXSR },
                { "OSXMMEXCPT", CR4_OSXMMEXCPT, CR4_OSXMMEXCPT },
                { "VMXE",       CR4_VMXE,       CR4_VMXE },
                { "SMXE",       CR4_SMXE,       CR4_SMXE },
                { "PCIDE",      CR4_PCIDE,      CR4_PCIDE },
                { "OSXSAVE",    CR4_OSXSAVE,    CR4_OSXSAVE },
                { "SMEP",       CR4_SMEP,       CR4_SMEP },
                { "SMAP",       CR4_SMAP,       CR4_SMAP },
                { NULL,         0,              0 }
        };

        cr0 = kmdb_unix_getcr0();
        cr2 = kmdb_unix_getcr2();
        cr3 = kmdb_unix_getcr3();
        cr4 = kmdb_unix_getcr4();
        mdb_printf("%%cr0 = 0x%08x <%b>\n", cr0, cr0, cr0_flag_bits);
        mdb_printf("%%cr2 = 0x%08x <%a>\n", cr2, cr2);

        if ((cr4 & CR4_PCIDE)) {
                mdb_printf("%%cr3 = 0x%08x <pfn:%lu pcid:%u>\n",
                    cr3 >> MMU_PAGESHIFT, cr3 & MMU_PAGEOFFSET);
        } else {
                mdb_printf("%%cr3 = 0x%08x <pfn:%lu flags:%b>\n", cr3,
                    cr3 >> MMU_PAGESHIFT, cr3, cr3_flag_bits);
        }

        mdb_printf("%%cr4 = 0x%08x <%b>\n", cr4, cr4, cr4_flag_bits);

        return (DCMD_OK);
}
#endif

static const mdb_dcmd_t dcmds[] = {
        { "gate_desc", ":", "dump a gate descriptor", gate_desc },
        { "idt", ":[-v]", "dump an IDT", idt },
        { "ttrace", "[-x] [-t kthread]", "dump trap trace buffers", ttrace },
        { "ttrace", "[-x]", "dump trap trace buffers", ttrace },
        { "vatopfn", ":[-a as]", "translate address to physical page",
            va2pfn_dcmd },
        { "report_maps", ":[-m]",
            "Given PFN, report mappings / page table usage",
            report_maps_dcmd, report_maps_help },
        { "htables", "", "Given hat_t *, lists all its htable_t * values",
            htables_dcmd, htables_help },
        { "ptable", ":[-m]", "Given PFN, dump contents of a page table",
            ptable_dcmd, ptable_help },
        { "pte", ":[-p XXXXX] [-l N]", "print human readable page table entry",
            pte_dcmd },
        { "pfntomfn", ":", "convert physical page to hypervisor machine page",
            pfntomfn_dcmd },
        { "mfntopfn", ":", "convert hypervisor machine page to physical page",
            mfntopfn_dcmd },
        { "memseg_list", ":", "show memseg list", memseg_list },
        { "scalehrtime", ":",
            "scale an unscaled high-res time", scalehrtime_cmd },
        { "x86_featureset", NULL, "dump the x86_featureset vector",
                x86_featureset_cmd },
```



```
#ifdef _KMDB
        { "crregs", NULL, "dump control registers", crregs_dcmd },
#endif
        { NULL }
};
```

*unchanged_portion_omitted_*

```
**********************************************************
     807 Wed Feb 28 17:34:20 2018
new/usr/src/cmd/mdb/i86pc/modules/unix/unix_sup.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
   1 /*
   2  * This file and its contents are supplied under the terms of the
   3  * Common Development and Distribution License ("CDDL"), version 1.0.
   4  * You may only use this file in accordance with the terms of version
   5  * 1.0 of the CDDL.
   6  *
   7  * A full copy of the text of the CDDL should have accompanied this
   8  * source.  A copy of the CDDL is also available via the Internet at
   9  * http://www.illumos.org/license/CDDL.
  10  */

  12 /*
  13  * Copyright 2018 Joyent, Inc.
  13  * Copyright 2015 Joyent, Inc.
  14  */

  16 #ifndef _UNIX_SUP_H
  17 #define _UNIX_SUP_H

  19 /*
  20  * Support routines for unix.
  21  */

  23 #include <sys/types.h>

  25 #ifdef __cplusplus
  26 extern "C" {
  27 #endif

  29 extern ulong_t kmdb_unix_getcr0(void);
  30 extern ulong_t kmdb_unix_getcr2(void);
  31 extern ulong_t kmdb_unix_getcr3(void);
  32 extern ulong_t kmdb_unix_getcr4(void);

  34 #ifdef __cplusplus
  35 }
```

*_____unchanged_portion_omitted_*

```
**********************************************************
    1452 Wed Feb 28 17:34:20 2018
new/usr/src/cmd/mdb/i86pc/modules/unix/unix_sup.s
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
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.
 * Copyright 2015 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>

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

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)

        ENTRY(kmdb_unix_getcr2)
        movq %cr2, %rax
        ret
        SET_SIZE(kmdb_unix_getcr2)

        ENTRY(kmdb_unix_getcr3)
        movq %cr3, %rax
        ret
        SET_SIZE(kmdb_unix_getcr3)

        ENTRY(kmdb_unix_getcr4)
        movq %cr4, %rax
```



```
        ret
        SET_SIZE(kmdb_unix_getcr4)
______unchanged_portion_omitted_

        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)
______unchanged_portion_omitted_
```

```
**********************************************************
    1842 Wed Feb 28 17:34:21 2018
new/usr/src/cmd/mdb/intel/Makefile.kmdb
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
#
# 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 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright 2018 Joyent, Inc.
#

PROMSRCS += \
        prom_env.c \
        prom_getchar.c \
        prom_init.c \
        prom_printf.c \
        prom_putchar.c

KMDBSRCS += \
        kaif.c \
        kmdb_dpi_isadep.c \
        kmdb_fault_isadep.c \
        kmdb_kdi_isadep.c \
        kmdb_promif_isadep.c \
        kvm_cpu_amd.c \
        kvm_cpu_p4.c \
        kvm_isadep.c

KMDBML += \
        kmdb_asmutil.s \
        kmdb_setcontext.s

KCTLSRCS += \
        kctl_isadep.c

CTXOFFUSERS = \
        kmdb_setcontext.o

CPPFLAGS += -DDIS_TEXT

$(CTXOFFUSERS) $(CTXOFFUSERS:%.o=%.ln): kmdb_context_off.h
```



```
kaif_activate.o kaif_activate.ln := CPPFLAGS += -D_MACHDEP -D_KMEMUSER

STANDLIBS += \
        ../libstandctf/libstandctf.so \
        $(SRC)/lib/libumem/$(MACHDIR)/libstandumem.so \
        ../libstand/libstand.a

KMDBLIBS = $(STANDLIBS) ../mdb_ks/kmod/mdb_ks

MAPFILE_SOURCES = \
        $(MAPFILE_SOURCES_COMMON) \
        ../../kmdb/kmdb_dpi_isadep.h \
        $(MAPFILE_SOURCES_$(MACH))

%.o: ../../../../../uts/intel/promif/%.c
        $(COMPILE.c) $<
        $(CTFCONVERT_O)

%.ln: ../../../../../uts/intel/promif/%.c
        $(LINT.c) -c $<
```

```
**********************************************************
    1053 Wed Feb 28 17:34:21 2018
new/usr/src/cmd/mdb/intel/ia32/Makefile.kmdb
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
#
# 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 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright 2018 Joyent, Inc.
#ident  "%Z%%M% %I%     %E% SMI"
#

KMDBML += \
        kaif_invoke.s \
        kmdb_start.s

KMDBSRCS += \
        mdb_ia32util.c \
        kmdb_makecontext.c
        kmdb_makecontext.c \
        kvm_cpu_p6.c

SACPPFLAGS = -D__$(MACH)
```

```
**********************************************************
   18868 Wed Feb 28 17:34:21 2018
new/usr/src/cmd/mdb/intel/kmdb/kaif.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * The debugger/"PROM" interface layer
 *
 * It makes more sense on SPARC. In reality, these interfaces deal with three
 * things: setting break/watchpoints, stepping, and interfacing with the KDI to
 * set up kmdb's IDT handlers.
 */

#include <kmdb/kmdb_dpi_impl.h>
#include <kmdb/kmdb_kdi.h>
#include <kmdb/kmdb_umemglue.h>
#include <kmdb/kaif.h>
#include <kmdb/kmdb_io.h>
#include <kmdb/kaif_start.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_isautil.h>
#include <mdb/mdb_io_impl.h>
#include <mdb/mdb_kreg_impl.h>
#include <mdb/mdb.h>

#include <sys/types.h>
#include <sys/bitmap.h>
#include <sys/termios.h>
#include <sys/kdi_impl.h>

/*
 * This is the area containing the saved state when we enter
 * via kmdb's IDT entries.
```



```
 */
kdi_cpusave_t   *kaif_cpusave;
int             kaif_ncpusave;
kdi_drreg_t     kaif_drreg;

uint32_t        kaif_waptmap;

int             kaif_trap_switch;

void (*kaif_modchg_cb)(struct modctl *, int);

enum {
        M_SYSRET        = 0x07, /* after M_ESC */
        M_ESC           = 0x0f,
        M_SYSEXIT       = 0x35, /* after M_ESC */
        M_REX_LO        = 0x40, /* first REX prefix */
        M_REX_HI        = 0x4f, /* last REX prefix */
        M_PUSHF         = 0x9c, /* pushfl and pushfq */
        M_POPF          = 0x9d, /* popfl and popfq */
        M_INT3          = 0xcc,
        M_INTX          = 0xcd,
        M_INTO          = 0xce,
        M_IRET          = 0xcf,
        M_CLI           = 0xfa,
        M_STI           = 0xfb
};
______unchanged_portion_omitted_
```

```
/*
 * The target has already configured the chip for branch step, leaving us to
 * actually make the machine go.  Due to a number of issues involving
 * the potential alteration of system state via instructions like sti, cli,
 * pushfl, and popfl, we're going to treat this like a normal system resume.
 * All CPUs will be released, on the kernel's IDT.  Our primary concern is
 * the alteration/storage of our TF'd EFLAGS via pushfl and popfl.  There's no
 * real workaround - we don't have opcode breakpoints - so the best we can do is
 * to ensure that the world won't end if someone does bad things to EFLAGS.
 *
 * Two things can happen:
 *  1. EFLAGS.TF may be cleared, either maliciously or via a popfl from saved
 *     state.  The CPU will continue execution beyond the branch, and will not
 *     reenter the debugger unless brought/sent in by other means.
 *  2. Someone may pushlf the TF'd EFLAGS, and may stash a copy of it somewhere.
 *     When the saved version is popfl'd back into place, the debugger will be
 *     re-entered on a single-step trap.
 */
static void
kaif_step_branch(void)
{
        kreg_t fl;

        (void) kmdb_dpi_get_register(FLAGS_REG_NAME, &fl);
        (void) kmdb_dpi_set_register(FLAGS_REG_NAME,
            (fl | (1 << KREG_EFLAGS_TF_SHIFT)));

        kmdb_dpi_resume_master();

        (void) kmdb_dpi_set_register(FLAGS_REG_NAME, fl);
}
```

```
/*ARGSUSED*/
static uintptr_t
kaif_call(uintptr_t funcva, uint_t argc, const uintptr_t argv[])
{
        return (kaif_invoke(funcva, argc, argv));
}
______unchanged_portion_omitted_
```

```
727 static void
728 kaif_msr_add(const kdi_msr_t *msrs)
729 {
730         kdi_msr_t *save;
731         size_t nr_msrs = 0;
732         size_t i;
733
734         while (msrs[nr_msrs].msr_num != 0)
735                 nr_msrs++;
736         /* we want to copy the terminating kdi_msr_t too */
737         nr_msrs++;
738
739         save = mdb_zalloc(sizeof (kdi_msr_t) * nr_msrs * kaif_ncpusave,
740             UM_SLEEP);
741
742         for (i = 0; i < kaif_ncpusave; i++)
743                 bcopy(msrs, &save[nr_msrs * i], sizeof (kdi_msr_t) * nr_msrs);
744
745         kmdb_kdi_set_debug_msrs(save);
746 }
747
748 static uint64_t
749 kaif_msr_get(int cpuid, uint_t num)
750 {
751         kdi_cpusave_t *save;
752         kdi_msr_t *msr;
753         int i;
754
755         if ((save = kaif_cpuid2save(cpuid)) == NULL)
756                 return (-1); /* errno is set for us */
757
758         msr = save->krs_msr;
759
760         for (i = 0; msr[i].msr_num != 0; i++) {
761                 if (msr[i].msr_num == num && (msr[i].msr_type & KDI_MSR_READ))
762                         return (msr[i].kdi_msr_val);
763         }
764
765         return (0);
766 }
767
768 void
769 kaif_trap_set_debugger(void)
770 {
771         kmdb_kdi_idt_switch(NULL);
772 }
```

_____unchanged_portion_omitted_

```
793 dpi_ops_t kmdb_dpi_ops = {
794         kaif_init,
795         kaif_activate,
796         kmdb_kdi_deactivate,
797         kaif_enter_mon,
798         kaif_modchg_register,
799         kaif_modchg_cancel,
800         kaif_get_cpu_state,
801         kaif_get_master_cpuid,
802         kaif_get_gregs,
803         kaif_get_register,
804         kaif_set_register,
805         kaif_brkpt_arm,
806         kaif_brkpt_disarm,
807         kaif_wapt_validate,
808         kaif_wapt_reserve,
809         kaif_wapt_release,
```



```
810         kaif_wapt_arm,
811         kaif_wapt_disarm,
812         kaif_wapt_match,
813         kaif_step,
887         kaif_step_branch,
814         kaif_call,
815         kaif_dump_crumbs,
890         kaif_msr_add,
891         kaif_msr_get,
816 };
```

_____unchanged_portion_omitted_

```
**********************************************************
    3622 Wed Feb 28 17:34:21 2018
new/usr/src/cmd/mdb/intel/kmdb/kmdb_dpi_isadep.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Intel-specific portions of the DPI
 */

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

#include <kmdb/kmdb_dpi_impl.h>
#include <kmdb/kmdb_fault.h>
#include <kmdb/kmdb_kdi.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_kreg.h>
#include <mdb/mdb.h>

void
kmdb_dpi_handle_fault(kreg_t trapno, kreg_t pc, kreg_t sp, int cpuid)
{
        kmdb_kdi_system_claim();

        mdb_dprintf(MDB_DBG_DPI, "\ndpi_handle_fault: trapno %u, pc 0x%0?p, "
            "sp 0x%0?p\n", (int)trapno, pc, sp);

        switch (trapno) {
        case T_GPFLT:
                errno = EACCES;
        default:
                errno = EMDB_NOMAP;
        }
```



```
        if (kmdb_dpi_fault_pcb != NULL) {
                longjmp(*kmdb_dpi_fault_pcb, 1);
                /*NOTREACHED*/
        }

        /* Debugger fault */
        kmdb_fault(trapno, pc, sp, cpuid);
}
```

_unchanged_portion_omitted_

```
void
kmdb_dpi_reboot(void)
{
        /*
         * We're going to skip all of the niceties we employ in resume_common,
         * as we don't plan to ever return.
         */
        longjmp(kmdb_dpi_entry_pcb, KMDB_DPI_CMD_REBOOT);
}

void
kmdb_dpi_msr_add(const kdi_msr_t *msrs)
{
        mdb.m_dpi->dpo_msr_add(msrs);
}

uint64_t
kmdb_dpi_msr_get(uint_t msr)
{
        return (mdb.m_dpi->dpo_msr_get(DPI_MASTER_CPUID, msr));
}

uint64_t
kmdb_dpi_msr_get_by_cpu(int cpuid, uint_t msr)
{
        return (mdb.m_dpi->dpo_msr_get(cpuid, msr));
}
```

_unchanged_portion_omitted_

```
**********************************************************
    1342 Wed Feb 28 17:34:21 2018
new/usr/src/cmd/mdb/intel/kmdb/kmdb_dpi_isadep.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#ifndef _KMDB_DPI_ISADEP_H
#define _KMDB_DPI_ISADEP_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#ifndef _ASM
#include <mdb/mdb_isautil.h>
#include <sys/kdi_machimpl.h>
#endif

#ifdef __cplusplus
extern "C" {
#endif

#ifndef _ASM

extern void kmdb_dpi_handle_fault(kreg_t, kreg_t, kreg_t, int);

extern void kmdb_dpi_reboot(void) __NORETURN;

extern void kmdb_dpi_msr_add(const kdi_msr_t *);
extern uint64_t kmdb_dpi_msr_get(uint_t);
extern uint64_t kmdb_dpi_msr_get_by_cpu(int, uint_t);

#endif /* _ASM */

#ifdef __cplusplus
}
#endif

#endif /* _KMDB_DPI_ISADEP_H */
```

```
**********************************************************
    2397 Wed Feb 28 17:34:21 2018
new/usr/src/cmd/mdb/intel/kmdb/kmdb_kdi_isadep.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

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

#include <mdb/mdb_debug.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_umem.h>
#include <kmdb/kmdb_dpi.h>
#include <mdb/mdb.h>

/*ARGSUSED*/
void
kmdb_kdi_stop_slaves(int my_cpuid, int doxc)
{
        /* Stop other CPUs if there are CPUs to stop */
        mdb.m_kdi->mkdi_stop_slaves(my_cpuid, doxc);
}
```

_____*unchanged_portion_omitted_*

```
void
kmdb_kdi_set_debug_msrs(kdi_msr_t *msrs)
{
        mdb.m_kdi->mkdi_set_debug_msrs(msrs);
}
```

```
void
kmdb_kdi_memrange_add(caddr_t base, size_t len)
{
```



```
        mdb.m_kdi->mkdi_memrange_add(base, len);
}
```

_____*unchanged_portion_omitted_*

```
**********************************************************
    1627 Wed Feb 28 17:34:21 2018
new/usr/src/cmd/mdb/intel/kmdb/kmdb_kdi_isadep.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  *
  25  * Copyright 2018 Joyent, Inc.
  26  */

  28 #ifndef _KMDB_KDI_ISADEP_H
  29 #define _KMDB_KDI_ISADEP_H

  29 #pragma ident   "%Z%%M% %I%     %E% SMI"

  31 #include <sys/types.h>
  32 #include <sys/kdi_machimpl.h>

  34 #include <mdb/mdb_target.h>

  36 #ifdef __cplusplus
  37 extern "C" {
  38 #endif

  40 struct gate_desc;

  42 extern void kmdb_kdi_activate(kdi_main_t, kdi_cpusave_t *, int);
  43 extern void kmdb_kdi_deactivate(void);

  45 extern void kmdb_kdi_idt_switch(kdi_cpusave_t *);

  47 extern void kmdb_kdi_update_drreg(kdi_drreg_t *);
  48 extern void kmdb_kdi_set_debug_msrs(kdi_msr_t *);

  49 extern uintptr_t kmdb_kdi_get_userlimit(void);

  51 extern int kmdb_kdi_get_cpuinfo(uint_t *, uint_t *, uint_t *);

  53 extern void kmdb_kdi_memrange_add(caddr_t, size_t);

  55 extern void kmdb_kdi_reboot(void);
```



```
  57 #ifdef __cplusplus
  58 }
______unchanged_portion_omitted_
```

```
**********************************************************
   14222 Wed Feb 28 17:34:22 2018
new/usr/src/cmd/mdb/intel/kmdb/kvm_isadep.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * isa-dependent portions of the kmdb target
 */

#include <kmdb/kvm.h>
#include <kmdb/kvm_cpu.h>
#include <kmdb/kmdb_kdi.h>
#include <kmdb/kmdb_asmutil.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_list.h>
#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_isautil.h>
#include <mdb/mdb_kreg_impl.h>
#include <mdb/mdb.h>

#include <sys/types.h>
#include <sys/frame.h>
#include <sys/trap.h>
#include <sys/bitmap.h>
#include <sys/pci_impl.h>

/* Higher than the highest trap number for which we have a defined specifier */
#define KMT_MAXTRAPNO   0x20

#define IOPORTLIMIT     0xffff  /* XXX find a new home for this */

const char *
kmt_def_dismode(void)
```



```
{
#ifdef  __amd64
        return ("amd64");
#else
        return ("ia32");
#endif
}
```

_____unchanged_portion_omitted_

```
int
kmt_step_branch(mdb_tgt_t *t)
{
        kmt_data_t *kmt = t->t_data;

        return (kmt_cpu_step_branch(t, kmt->kmt_cpu));
}
```

```
/*
 * Return the address of the next instruction following a call, or return -1
 * and set errno to EAGAIN if the target should just single-step.
 */
int
kmt_next(mdb_tgt_t *t, uintptr_t *p)
{
        kreg_t pc;
        mdb_instr_t instr;

        (void) kmdb_dpi_get_register("pc", &pc);

        if (mdb_tgt_vread(t, &instr, sizeof (mdb_instr_t), pc) !=
            sizeof (mdb_instr_t))
                return (-1); /* errno is set for us */

        return (mdb_isa_next(t, p, pc, instr));
}
```

_____unchanged_portion_omitted_

```
int
kmt_msr_validate(const kdi_msr_t *msr)
{
        uint64_t val;

        for (/* */; msr->msr_num != 0; msr++) {
                if (kmt_rwmsr(msr->msr_num, &val, rdmsr) < 0)
                        return (0);
        }

        return (1);
}
```

```
/*ARGSUSED*/
ssize_t
kmt_write(mdb_tgt_t *t, const void *buf, size_t nbytes, uintptr_t addr)
{
        if (!(t->t_flags & MDB_TGT_F_ALLOWIO) &&
            (nbytes = kmdb_kdi_range_is_nontoxic(addr, nbytes, 1)) == 0)
                return (set_errno(EMDB_NOMAP));

        /*
         * No writes to user space are allowed.  If we were to allow it, we'd
         * be in the unfortunate situation where kmdb could place a breakpoint
         * on a userspace executable page; this dirty page would end up being
         * flushed back to disk, incurring sadness when it's next executed.
         * Besides, we can't allow trapping in from userspace anyway.
         */
        if (addr < kmdb_kdi_get_userlimit())
```

```
 366                 return (set_errno(EMDB_TGTNOTSUP));

 368         return (kmt_rw(t, (void *)buf, nbytes, addr, kmt_writer));
 369 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    1670 Wed Feb 28 17:34:22 2018
new/usr/src/cmd/mdb/intel/kmdb/kvm_isadep.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#ifndef _KVM_ISADEP_H
#define _KVM_ISADEP_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#ifdef __cplusplus
extern "C" {
#endif

extern uintptr_t kmt_invoke(uintptr_t, uint_t, const uintptr_t *);

extern void kmt_in(void *, size_t, uintptr_t);
extern void kmt_out(void *, size_t, uintptr_t);

extern int kmt_in_dcmd(uintptr_t, uint_t, int, const mdb_arg_t *);
extern int kmt_out_dcmd(uintptr_t, uint_t, int, const mdb_arg_t *);

extern int kmt_rdmsr(uintptr_t, uint_t, int, const mdb_arg_t *);
extern int kmt_wrmsr(uintptr_t, uint_t, int, const mdb_arg_t *);

extern int kmt_rdpcicfg(uintptr_t, uint_t, int, const mdb_arg_t *);
extern int kmt_wrpcicfg(uintptr_t, uint_t, int, const mdb_arg_t *);
extern int kmt_msr_validate(const kdi_msr_t *);

#ifdef __cplusplus
}
```

_____unchanged_portion_omitted_

```
**********************************************************
    9127 Wed Feb 28 17:34:22 2018
new/usr/src/cmd/mdb/intel/mdb/kvm_amd64dep.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

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

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Libkvm Kernel Target Intel 64-bit component
 *
 * This file provides the ISA-dependent portion of the libkvm kernel target.
 * For more details on the implementation refer to mdb_kvm.c.
 */

#include <sys/types.h>
#include <sys/reg.h>
#include <sys/frame.h>
#include <sys/stack.h>
#include <sys/sysmacros.h>
#include <sys/panic.h>
#include <sys/privregs.h>
#include <strings.h>

#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_disasm.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_kreg_impl.h>
#include <mdb/mdb_amd64util.h>
#include <mdb/kvm_isadep.h>
#include <mdb/mdb_kvm.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb.h>

/*ARGSUSED*/
```



```
int
kt_regs(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
{
        mdb_amd64_printregs((const mdb_tgt_gregset_t *)addr);
        return (DCMD_OK);
}
```

*unchanged_portion_omitted_*

```
const mdb_tgt_ops_t kt_amd64_ops = {
        kt_setflags,                            /* t_setflags */
        kt_setcontext,                          /* t_setcontext */
        kt_activate,                            /* t_activate */
        kt_deactivate,                          /* t_deactivate */
        (void (*)()) mdb_tgt_nop,               /* t_periodic */
        kt_destroy,                             /* t_destroy */
        kt_name,                                /* t_name */
        (const char *(*)()) mdb_conf_isa,       /* t_isa */
        kt_platform,                            /* t_platform */
        kt_uname,                               /* t_uname */
        kt_dmodel,                              /* t_dmodel */
        kt_aread,                               /* t_aread */
        kt_awrite,                              /* t_awrite */
        kt_vread,                               /* t_vread */
        kt_vwrite,                              /* t_vwrite */
        kt_pread,                               /* t_pread */
        kt_pwrite,                              /* t_pwrite */
        kt_fread,                               /* t_fread */
        kt_fwrite,                              /* t_fwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_ioread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_iowrite */
        kt_vtop,                                /* t_vtop */
        kt_lookup_by_name,                      /* t_lookup_by_name */
        kt_lookup_by_addr,                      /* t_lookup_by_addr */
        kt_symbol_iter,                         /* t_symbol_iter */
        kt_mapping_iter,                        /* t_mapping_iter */
        kt_object_iter,                         /* t_object_iter */
        kt_addr_to_map,                         /* t_addr_to_map */
        kt_name_to_map,                         /* t_name_to_map */
        kt_addr_to_ctf,                         /* t_addr_to_ctf */
        kt_name_to_ctf,                         /* t_name_to_ctf */
        kt_status,                              /* t_status */
        (int (*)()) mdb_tgt_notsup,             /* t_run */
        (int (*)()) mdb_tgt_notsup,             /* t_step */
        (int (*)()) mdb_tgt_notsup,             /* t_step_out */
        (int (*)()) mdb_tgt_notsup,             /* t_step_branch */
        (int (*)()) mdb_tgt_notsup,             /* t_next */
        (int (*)()) mdb_tgt_notsup,             /* t_cont */
        (int (*)()) mdb_tgt_notsup,             /* t_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_vbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_sbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_pwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_vwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_iowapt */
        (int (*)()) mdb_tgt_null,               /* t_add_sysenter */
        (int (*)()) mdb_tgt_null,               /* t_add_sysexit */
        (int (*)()) mdb_tgt_null,               /* t_add_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_fault */
        kt_getareg,                             /* t_getareg */
        kt_putareg,                             /* t_putareg */
        mdb_amd64_kvm_stack_iter,               /* t_stack_iter */
        (int (*)()) mdb_tgt_notsup              /* t_auxv */
};
```

*unchanged_portion_omitted_*

```
**********************************************************
    9471 Wed Feb 28 17:34:22 2018
new/usr/src/cmd/mdb/intel/mdb/kvm_ia32dep.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

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

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Libkvm Kernel Target Intel 32-bit component
 *
 * This file provides the ISA-dependent portion of the libkvm kernel target.
 * For more details on the implementation refer to mdb_kvm.c.
 */

#include <sys/types.h>
#include <sys/regset.h>
#include <sys/frame.h>
#include <sys/stack.h>
#include <sys/sysmacros.h>
#include <sys/panic.h>
#include <strings.h>

#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_disasm.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_kreg_impl.h>
#include <mdb/mdb_ia32util.h>
#include <mdb/kvm_isadep.h>
#include <mdb/mdb_kvm.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb.h>


/*ARGSUSED*/
```



```
int
kt_regs(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
{
        mdb_ia32_printregs((const mdb_tgt_gregset_t *)addr);
        return (DCMD_OK);
}
```

*unchanged_portion_omitted_*

```
const mdb_tgt_ops_t kt_ia32_ops = {
        kt_setflags,                            /* t_setflags */
        kt_setcontext,                          /* t_setcontext */
        kt_activate,                            /* t_activate */
        kt_deactivate,                          /* t_deactivate */
        (void (*)()) mdb_tgt_nop,               /* t_periodic */
        kt_destroy,                             /* t_destroy */
        kt_name,                                /* t_name */
        (const char *(*)()) mdb_conf_isa,       /* t_isa */
        kt_platform,                            /* t_platform */
        kt_uname,                               /* t_uname */
        kt_dmodel,                              /* t_dmodel */
        kt_aread,                               /* t_aread */
        kt_awrite,                              /* t_awrite */
        kt_vread,                               /* t_vread */
        kt_vwrite,                              /* t_vwrite */
        kt_pread,                               /* t_pread */
        kt_pwrite,                              /* t_pwrite */
        kt_fread,                               /* t_fread */
        kt_fwrite,                              /* t_fwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_ioread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_iowrite */
        kt_vtop,                                /* t_vtop */
        kt_lookup_by_name,                      /* t_lookup_by_name */
        kt_lookup_by_addr,                      /* t_lookup_by_addr */
        kt_symbol_iter,                         /* t_symbol_iter */
        kt_mapping_iter,                        /* t_mapping_iter */
        kt_object_iter,                         /* t_object_iter */
        kt_addr_to_map,                         /* t_addr_to_map */
        kt_name_to_map,                         /* t_name_to_map */
        kt_addr_to_ctf,                         /* t_addr_to_ctf */
        kt_name_to_ctf,                         /* t_name_to_ctf */
        kt_status,                              /* t_status */
        (int (*)()) mdb_tgt_notsup,             /* t_run */
        (int (*)()) mdb_tgt_notsup,             /* t_step */
        (int (*)()) mdb_tgt_notsup,             /* t_step_out */
        (int (*)()) mdb_tgt_notsup,             /* t_step_branch */
        (int (*)()) mdb_tgt_notsup,             /* t_next */
        (int (*)()) mdb_tgt_notsup,             /* t_cont */
        (int (*)()) mdb_tgt_notsup,             /* t_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_vbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_sbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_pwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_vwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_iowapt */
        (int (*)()) mdb_tgt_null,               /* t_add_sysenter */
        (int (*)()) mdb_tgt_null,               /* t_add_sysexit */
        (int (*)()) mdb_tgt_null,               /* t_add_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_fault */
        kt_getareg,                             /* t_getareg */
        kt_putareg,                             /* t_putareg */
        mdb_ia32_kvm_stack_iter,                /* t_stack_iter */
        (int (*)()) mdb_tgt_notsup              /* t_auxv */
};
```

*unchanged_portion_omitted_*

```
**********************************************************
   13668 Wed Feb 28 17:34:22 2018
new/usr/src/cmd/mdb/sparc/kmdb/kvm_isadep.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (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 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * isa-dependent portions of the kmdb target
 */

#include <mdb/mdb_kreg_impl.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_v9util.h>
#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_umem.h>
#include <kmdb/kmdb_kdi.h>
#include <kmdb/kmdb_dpi.h>
#include <kmdb/kmdb_promif.h>
#include <kmdb/kmdb_asmutil.h>
#include <kmdb/kvm.h>
#include <mdb/mdb.h>

#include <sys/types.h>
#include <sys/stack.h>
#include <sys/regset.h>
#include <sys/sysmacros.h>
#include <sys/bitmap.h>
#include <sys/machtrap.h>
#include <sys/trap.h>

/* Higher than the highest trap number for which we have a specific specifier */
#define KMT_MAXTRAPNO   0x1ff
```



```
#define OP(x)           ((x) >> 30)
#define OP3(x)          (((x) >> 19) & 0x3f)
#define RD(x)           (((x) >> 25) & 0x1f)
#define RS1(x)          (((x) >> 14) & 0x1f)
#define RS2(x)          ((x) & 0x1f)

#define OP_ARITH        0x2

#define OP3_OR          0x02
#define OP3_SAVE        0x3c
#define OP3_RESTORE     0x3d

static int
kmt_stack_iter(mdb_tgt_t *t, const mdb_tgt_gregset_t *gsp,
    mdb_tgt_stack_f *func, void *arg, int cpuid)
{
        const mdb_tgt_gregset_t *grp;
        mdb_tgt_gregset_t gregs;
        kreg_t *kregs = &gregs.kregs[0];
        long nwin, stopwin, canrestore, wp, i, sp;
        long argv[6];

        /*
         * If gsp isn't null, we were asked to dump a trace from a
         * specific location.  The normal iterator can handle that.
         */
        if (gsp != NULL) {
                if (cpuid != DPI_MASTER_CPUID)
                        warn("register set provided - ignoring cpu argument\n");
                return (mdb_kvm_v9stack_iter(t, gsp, func, arg));
        }

        if (kmdb_dpi_get_cpu_state(cpuid) < 0) {
                warn("failed to iterate through stack for cpu %u", cpuid);
                return (DCMD_ERR);
        }

        /*
         * We're being asked to dump the trace for the current CPU.
         * To do that, we need to iterate first through the saved
         * register windors.  If there's more to the trace than that,
         * we'll hand off to the normal iterator.
         */
        if ((grp = kmdb_dpi_get_gregs(cpuid)) == NULL) {
                warn("failed to retrieve registers for cpu %d", cpuid);
                return (DCMD_ERR);
        }

        bcopy(grp, &gregs, sizeof (mdb_tgt_gregset_t));

        wp = kregs[KREG_CWP];
        canrestore = kregs[KREG_CANRESTORE];
        nwin = kmdb_dpi_get_nwin(cpuid);
        stopwin = ((wp + nwin) - canrestore - 1) % nwin;

        mdb_dprintf(MDB_DBG_KMOD, "dumping cwp = %lu, canrestore = %lu, "
            "stopwin = %lu\n", wp, canrestore, stopwin);

        for (;;) {
                struct rwindow rwin;

                for (i = 0; i < 6; i++)
                        argv[i] = kregs[KREG_I0 + i];

                if (kregs[KREG_PC] != 0 &&
```

```
123                     func(arg, kregs[KREG_PC], 6, argv, &gregs) != 0)
124                         return (0);

126                 kregs[KREG_PC] = kregs[KREG_I7];
127                 kregs[KREG_NPC] = kregs[KREG_PC] + 4;

129                 if ((sp = kregs[KREG_FP] + STACK_BIAS) == STACK_BIAS || sp == 0)
130                         return (0); /* Stop if we're at the end of stack */

132                 if (sp & (STACK_ALIGN - 1))
133                         return (set_errno(EMDB_STKALIGN));

135                 wp = (wp + nwin - 1) % nwin;

137                 if (wp == stopwin)
138                         break;

140                 bcopy(&kregs[KREG_I0], &kregs[KREG_O0], 8 * sizeof (kreg_t));

142                 if (kmdb_dpi_get_rwin(cpuid, wp, &rwin) < 0) {
143                         warn("unable to get registers from window %ld\n", wp);
144                         return (-1);
145                 }

147                 for (i = 0; i < 8; i++)
148                         kregs[KREG_L0 + i] = (uintptr_t)rwin.rw_local[i];
149                 for (i = 0; i < 8; i++)
150                         kregs[KREG_I0 + i] = (uintptr_t)rwin.rw_in[i];
151         }

153         mdb_dprintf(MDB_DBG_KMOD, "dumping wp %ld and beyond normally\n", wp);

155         /*
156          * hack - if we null out pc here, iterator won't print the frame
157          * that corresponds to the current set of registers.  That's what we
158          * want because we just printed them above.
159          */
160         kregs[KREG_PC] = 0;
161         return (mdb_kvm_v9stack_iter(t, &gregs, func, arg));
162 }
```

_____unchanged_portion_omitted_

```
380 /*ARGSUSED*/
381 int
382 kmt_step_branch(mdb_tgt_t *t)
383 {
384         return (set_errno(EMDB_TGTHWNOTSUP));
385 }
```

```
380 static const char *
381 regno2name(int idx)
382 {
383         const mdb_tgt_regdesc_t *rd;

385         for (rd = mdb_sparcv9_kregs; rd->rd_name != NULL; rd++) {
386                 if (idx == rd->rd_num)
387                         return (rd->rd_name);
388         }

390         ASSERT(rd->rd_name != NULL);

392         return ("unknown");
393 }
```

_____unchanged_portion_omitted_

```
**********************************************************
   15729 Wed Feb 28 17:34:23 2018
new/usr/src/cmd/mdb/sparc/mdb/kvm_v7dep.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

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

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Libkvm Kernel Target SPARC v7 component
 *
 * This file provides the ISA-dependent portion of the libkvm kernel target.
 * For more details on the implementation refer to mdb_kvm.c.
 */

#include <sys/types.h>
#include <sys/machtypes.h>
#include <sys/regset.h>
#include <sys/frame.h>
#include <sys/stack.h>
#include <sys/sysmacros.h>
#include <sys/panic.h>
#include <strings.h>

#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_disasm.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_kreg.h>
#include <mdb/mdb_kvm.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb.h>

/*
 * The mdb_tgt_gregset type is opaque to callers of the target interface
 * and to our own target common code.  We now can define it explicitly.
```



```
 */
struct mdb_tgt_gregset {
        kreg_t kregs[KREG_NGREG];
};
```

*unchanged_portion_omitted_*

```
const mdb_tgt_ops_t kt_sparcv7_ops = {
        kt_setflags,                            /* t_setflags */
        kt_setcontext,                          /* t_setcontext */
        kt_activate,                            /* t_activate */
        kt_deactivate,                          /* t_deactivate */
        (void (*)()) mdb_tgt_nop,               /* t_periodic */
        kt_destroy,                             /* t_destroy */
        kt_name,                                /* t_name */
        (const char *(*)()) mdb_conf_isa,       /* t_isa */
        kt_platform,                            /* t_platform */
        kt_uname,                               /* t_uname */
        kt_dmodel,                              /* t_dmodel */
        kt_aread,                               /* t_aread */
        kt_awrite,                              /* t_awrite */
        kt_vread,                               /* t_vread */
        kt_vwrite,                              /* t_vwrite */
        kt_pread,                               /* t_pread */
        kt_pwrite,                              /* t_pwrite */
        kt_fread,                               /* t_fread */
        kt_fwrite,                              /* t_fwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_ioread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_iowrite */
        kt_vtop,                                /* t_vtop */
        kt_lookup_by_name,                      /* t_lookup_by_name */
        kt_lookup_by_addr,                      /* t_lookup_by_addr */
        kt_symbol_iter,                         /* t_symbol_iter */
        kt_mapping_iter,                        /* t_mapping_iter */
        kt_object_iter,                         /* t_object_iter */
        kt_addr_to_map,                         /* t_addr_to_map */
        kt_name_to_map,                         /* t_name_to_map */
        kt_addr_to_ctf,                         /* t_addr_to_ctf */
        kt_name_to_ctf,                         /* t_name_to_ctf */
        kt_status,                              /* t_status */
        (int (*)()) mdb_tgt_notsup,             /* t_run */
        (int (*)()) mdb_tgt_notsup,             /* t_step */
        (int (*)()) mdb_tgt_notsup,             /* t_step_out */
        (int (*)()) mdb_tgt_notsup,             /* t_step_branch */
        (int (*)()) mdb_tgt_notsup,             /* t_next */
        (int (*)()) mdb_tgt_notsup,             /* t_cont */
        (int (*)()) mdb_tgt_notsup,             /* t_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_vbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_sbrkpt */
        (int (*)()) mdb_tgt_null,               /* t_add_pwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_vwapt */
        (int (*)()) mdb_tgt_null,               /* t_add_iowapt */
        (int (*)()) mdb_tgt_null,               /* t_add_sysenter */
        (int (*)()) mdb_tgt_null,               /* t_add_sysexit */
        (int (*)()) mdb_tgt_null,               /* t_add_signal */
        (int (*)()) mdb_tgt_null,               /* t_add_fault */
        kt_getareg,                             /* t_getareg */
        kt_putareg,                             /* t_putareg */
        kt_stack_iter,                          /* t_stack_iter */
        (int (*)()) mdb_tgt_notsup              /* t_auxv */
};
```

*unchanged_portion_omitted_*

```
**********************************************************
   10883 Wed Feb 28 17:34:23 2018
new/usr/src/cmd/mdb/sparc/mdb/kvm_v9dep.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

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

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Libkvm Kernel Target SPARC v9 component
 *
 * This file provides the ISA-dependent portion of the libkvm kernel target.
 * For more details on the implementation refer to mdb_kvm.c.  The SPARC v9
 * ISA code is actually compiled into *both* the sparcv7 and sparcv9 MDB
 * binaries because we need to deal with the sparcv9 CPU registers when
 * debugging a 32-bit crash dump from a kernel running on a sparcv9 CPU.
 */

#ifndef __sparcv9cpu
#define __sparcv9cpu
#endif

#include <sys/types.h>
#include <sys/machtypes.h>
#include <sys/regset.h>
#include <sys/frame.h>
#include <sys/stack.h>
#include <sys/sysmacros.h>
#include <sys/panic.h>
#include <strings.h>

#include <mdb/mdb_target_impl.h>
#include <mdb/mdb_disasm.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_kreg_impl.h>
#include <mdb/mdb_v9util.h>
```



```
#include <mdb/mdb_kvm.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb.h>

#ifndef STACK_BIAS
#define STACK_BIAS      0
#endif

static int
kt_getareg(mdb_tgt_t *t, mdb_tgt_tid_t tid,
    const char *rname, mdb_tgt_reg_t *rp)
{
        const mdb_tgt_regdesc_t *rdp;
        kt_data_t *kt = t->t_data;

        if (tid != kt->k_tid)
                return (set_errno(EMDB_NOREGS));

        for (rdp = kt->k_rds; rdp->rd_name != NULL; rdp++) {
                if (strcmp(rname, rdp->rd_name) == 0) {
                        *rp = kt->k_regs->kregs[rdp->rd_num];
                        return (0);
                }
        }

        return (set_errno(EMDB_BADREG));
}
```

*unchanged_portion_omitted_*

```
const mdb_tgt_ops_t kt_sparcv9_ops = {
        kt_setflags,                            /* t_setflags */
        kt_setcontext,                          /* t_setcontext */
        kt_activate,                            /* t_activate */
        kt_deactivate,                          /* t_deactivate */
        (void (*)()) mdb_tgt_nop,               /* t_periodic */
        kt_destroy,                             /* t_destroy */
        kt_name,                                /* t_name */
        (const char *(*)()) mdb_conf_isa,       /* t_isa */
        kt_platform,                            /* t_platform */
        kt_uname,                               /* t_uname */
        kt_dmodel,                              /* t_dmodel */
        kt_aread,                               /* t_aread */
        kt_awrite,                              /* t_awrite */
        kt_vread,                               /* t_vread */
        kt_vwrite,                              /* t_vwrite */
        kt_pread,                               /* t_pread */
        kt_pwrite,                              /* t_pwrite */
        kt_fread,                               /* t_fread */
        kt_fwrite,                              /* t_fwrite */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_ioread */
        (ssize_t (*)()) mdb_tgt_notsup,         /* t_iowrite */
        kt_vtop,                                /* t_vtop */
        kt_lookup_by_name,                      /* t_lookup_by_name */
        kt_lookup_by_addr,                      /* t_lookup_by_addr */
        kt_symbol_iter,                         /* t_symbol_iter */
        kt_mapping_iter,                        /* t_mapping_iter */
        kt_object_iter,                         /* t_object_iter */
        kt_addr_to_map,                         /* t_addr_to_map */
        kt_name_to_map,                         /* t_name_to_map */
        kt_addr_to_ctf,                         /* t_addr_to_ctf */
        kt_name_to_ctf,                         /* t_name_to_ctf */
        kt_status,                              /* t_status */
        (int (*)()) mdb_tgt_notsup,             /* t_run */
        (int (*)()) mdb_tgt_notsup,             /* t_step */
        (int (*)()) mdb_tgt_notsup,             /* t_step_out */
```

```
220         (int (*)()) mdb_tgt_notsup,             /* t_step_branch */
220         (int (*)()) mdb_tgt_notsup,             /* t_next */
221         (int (*)()) mdb_tgt_notsup,             /* t_cont */
222         (int (*)()) mdb_tgt_notsup,             /* t_signal */
223         (int (*)()) mdb_tgt_null,               /* t_add_vbrkpt */
224         (int (*)()) mdb_tgt_null,               /* t_add_sbrkpt */
225         (int (*)()) mdb_tgt_null,               /* t_add_pwapt */
226         (int (*)()) mdb_tgt_null,               /* t_add_iowapt */
227         (int (*)()) mdb_tgt_null,               /* t_add_vwapt */
228         (int (*)()) mdb_tgt_null,               /* t_add_sysenter */
229         (int (*)()) mdb_tgt_null,               /* t_add_sysexit */
230         (int (*)()) mdb_tgt_null,               /* t_add_signal */
231         (int (*)()) mdb_tgt_null,               /* t_add_fault */
232         kt_getareg,                             /* t_getareg */
233         kt_putareg,                             /* t_putareg */
234         mdb_kvm_v9stack_iter,                   /* t_stack_iter */
235         (int (*)()) mdb_tgt_notsup              /* t_auxv */
236 };
```

*unchanged_portion_omitted_*

```
**********************************************************
  107878 Wed Feb 28 17:34:23 2018
new/usr/src/uts/i86pc/vm/hat_i86.c
9208 hati_demap_func should take pagesize into account
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Tim Kordas <tim.kordas@joyent.com>
**********************************************************
/*
 * 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) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2017 Joyent, Inc.  All rights reserved.
 * Copyright (c) 2014, 2015 by Delphix. All rights reserved.
 */

/*
 * VM - Hardware Address Translation management for i386 and amd64
 *
 * Implementation of the interfaces described in <common/vm/hat.h>
 *
 * Nearly all the details of how the hardware is managed should not be
 * visible outside this layer except for misc. machine specific functions
 * that work in conjunction with this code.
 *
 * Routines used only inside of i86pc/vm start with hati_ for HAT Internal.
 */

#include <sys/machparam.h>
#include <sys/machsystm.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/systm.h>
#include <sys/cpuvar.h>
#include <sys/thread.h>
#include <sys/proc.h>
#include <sys/cpu.h>
#include <sys/kmem.h>
#include <sys/disp.h>
#include <sys/shm.h>
#include <sys/sysmacros.h>
#include <sys/machparam.h>
```



```
#include <sys/vmem.h>
#include <sys/vmsystm.h>
#include <sys/promif.h>
#include <sys/var.h>
#include <sys/x86_archext.h>
#include <sys/atomic.h>
#include <sys/bitmap.h>
#include <sys/controlregs.h>
#include <sys/bootconf.h>
#include <sys/bootsvcs.h>
#include <sys/bootinfo.h>
#include <sys/archsystm.h>

#include <vm/seg_kmem.h>
#include <vm/hat_i86.h>
#include <vm/as.h>
#include <vm/seg.h>
#include <vm/page.h>
#include <vm/seg_kp.h>
#include <vm/seg_kpm.h>
#include <vm/vm_dep.h>
#ifdef __xpv
#include <sys/hypervisor.h>
#endif
#include <vm/kboot_mmu.h>
#include <vm/seg_spt.h>

#include <sys/cmn_err.h>

/*
 * Basic parameters for hat operation.
 */
struct hat_mmu_info mmu;

/*
 * The page that is the kernel's top level pagetable.
 *
 * For 32 bit PAE support on i86pc, the kernel hat will use the 1st 4 entries
 * on this 4K page for its top level page table. The remaining groups of
 * 4 entries are used for per processor copies of user VLP pagetables for
 * running threads.  See hat_switch() and reload_pae32() for details.
 *
 * vlp_page[0..3] - level==2 PTEs for kernel HAT
 * vlp_page[4..7] - level==2 PTEs for user thread on cpu 0
 * vlp_page[8..11]  - level==2 PTE for user thread on cpu 1
 * etc...
 */
static x86pte_t *vlp_page;

/*
 * forward declaration of internal utility routines
 */
static x86pte_t hati_update_pte(htable_t *ht, uint_t entry, x86pte_t expected,
        x86pte_t new);

/*
 * The kernel address space exists in all HATs. To implement this the
 * kernel reserves a fixed number of entries in the topmost level(s) of page
 * tables. The values are setup during startup and then copied to every user
 * hat created by hat_alloc(). This means that kernelbase must be:
 *
 *        4Meg aligned for 32 bit kernels
 *      512Gig aligned for x86_64 64 bit kernel
 *
 * The hat_kernel_range_ts describe what needs to be copied from kernel hat
 * to each user hat.
```

```
 126  */
 127 typedef struct hat_kernel_range {
 128         level_t         hkr_level;
 129         uintptr_t       hkr_start_va;
 130         uintptr_t       hkr_end_va;     /* zero means to end of memory */
 131 } hat_kernel_range_t;
______unchanged_portion_omitted_

1918 /*
1919  * A range of virtual pages for purposes of demapping.
1920  */
1921 typedef struct range_info {
1922         uintptr_t       rng_va;         /* address of page */
1923         ulong_t         rng_cnt;        /* number of pages in range */
1924         level_t         rng_level;      /* page table level */
1925 } range_info_t;

1927 #if !defined(__xpv)
1928 /*
1929  * Cross call service routine to demap a range of virtual
1930  * pages on the current CPU or flush all mappings in TLB.
1919  * Cross call service routine to demap a virtual page on
1920  * the current CPU or flush all mappings in TLB.
1931  */
1932 /*ARGSUSED*/
1933 static int
1934 hati_demap_func(xc_arg_t a1, xc_arg_t a2, xc_arg_t a3)
1935 {
1936         hat_t           *hat = (hat_t *)a1;
1937         range_info_t    *range = (range_info_t *)a2;
1927         caddr_t addr = (caddr_t)a2;
1938         size_t          len = (size_t)a3;
1939         caddr_t         addr = (caddr_t)range->rng_va;
1940         size_t          pgsz = LEVEL_SIZE(range->rng_level);

1942         /*
1943          * If the target hat isn't the kernel and this CPU isn't operating
1944          * in the target hat, we can ignore the cross call.
1945          */
1946         if (hat != kas.a_hat && hat != CPU->cpu_current_hat)
1947                 return (0);

1949         /*
1950          * For a normal address, we flush a range of contiguous mappings
1951          */
1952         if ((uintptr_t)addr != DEMAP_ALL_ADDR) {
1953                 for (size_t i = 0; i < len; i += pgsz)
1941                 for (size_t i = 0; i < len; i += MMU_PAGESIZE)
1954                         mmu_tlbflush_entry(addr + i);
1955                 return (0);
1956         }

1958         /*
1959          * Otherwise we reload cr3 to effect a complete TLB flush.
1960          *
1961          * A reload of cr3 on a VLP process also means we must also recopy in
1962          * the pte values from the struct hat
1963          */
1964         if (hat->hat_flags & HAT_VLP) {
1965 #if defined(__amd64)
1966                 x86pte_t *vlpptep = CPU->cpu_hat_info->hci_vlp_l2ptes;

1968                 VLP_COPY(hat->hat_vlp_ptes, vlpptep);
1969 #elif defined(__i386)
1970                 reload_pae32(hat, CPU);
1971 #endif
```



```
1972         }
1973         reload_cr3();
1974         return (0);
1975 }
______unchanged_portion_omitted_
2042 #endif /* !__xpv */

2044 /*
2045  * Internal routine to do cross calls to invalidate a range of pages on
2046  * all CPUs using a given hat.
2047  */
2048 void
2049 hat_tlb_inval_range(hat_t *hat, range_info_t *range)
2037 hat_tlb_inval_range(hat_t *hat, uintptr_t va, size_t len)
2050 {
2051         extern int      flushes_require_xcalls; /* from mp_startup.c */
2052         cpuset_t        justme;
2053         cpuset_t        cpus_to_shootdown;
2054         uintptr_t       va = range->rng_va;
2055         size_t          len = range->rng_cnt << LEVEL_SHIFT(range->rng_level);
2056 #ifndef __xpv
2057         cpuset_t        check_cpus;
2058         cpu_t           *cpup;
2059         int             c;
2060 #endif

2062         /*
2063          * If the hat is being destroyed, there are no more users, so
2064          * demap need not do anything.
2065          */
2066         if (hat->hat_flags & HAT_FREEING)
2067                 return;

2069         /*
2070          * If demapping from a shared pagetable, we best demap the
2071          * entire set of user TLBs, since we don't know what addresses
2072          * these were shared at.
2073          */
2074         if (hat->hat_flags & HAT_SHARED) {
2075                 hat = kas.a_hat;
2076                 va = DEMAP_ALL_ADDR;
2077         }

2079         /*
2080          * if not running with multiple CPUs, don't use cross calls
2081          */
2082         if (panicstr || !flushes_require_xcalls) {
2083 #ifdef __xpv
2084                 if (va == DEMAP_ALL_ADDR) {
2085                         xen_flush_tlb();
2086                 } else {
2087                         for (size_t i = 0; i < len; i += MMU_PAGESIZE)
2088                                 xen_flush_va((caddr_t)(va + i));
2089                 }
2090 #else
2091                 (void) hati_demap_func((xc_arg_t)hat,
2092                     (xc_arg_t)range, (xc_arg_t)len);
2078                     (xc_arg_t)va, (xc_arg_t)len);
2093 #endif
2094                 return;
2095         }


2098         /*
2099          * Determine CPUs to shootdown. Kernel changes always do all CPUs.
2100          * Otherwise it's just CPUs currently executing in this hat.
```

```
2101          */
2102         kpreempt_disable();
2103         CPUSET_ONLY(justme, CPU->cpu_id);
2104         if (hat == kas.a_hat)
2105                 cpus_to_shootdown = khat_cpuset;
2106         else
2107                 cpus_to_shootdown = hat->hat_cpus;

2109 #ifndef __xpv
2110         /*
2111          * If any CPUs in the set are idle, just request a delayed flush
2112          * and avoid waking them up.
2113          */
2114         check_cpus = cpus_to_shootdown;
2115         for (c = 0; c < NCPU && !CPUSET_ISNULL(check_cpus); ++c) {
2116                 ulong_t tlb_info;

2118                 if (!CPU_IN_SET(check_cpus, c))
2119                         continue;
2120                 CPUSET_DEL(check_cpus, c);
2121                 cpup = cpu[c];
2122                 if (cpup == NULL)
2123                         continue;

2125                 tlb_info = cpup->cpu_m.mcpu_tlb_info;
2126                 while (tlb_info == TLB_CPU_HALTED) {
2127                         (void) CAS_TLB_INFO(cpup, TLB_CPU_HALTED,
2128                             TLB_CPU_HALTED | TLB_INVAL_ALL);
2129                         SMT_PAUSE();
2130                         tlb_info = cpup->cpu_m.mcpu_tlb_info;
2131                 }
2132                 if (tlb_info == (TLB_CPU_HALTED | TLB_INVAL_ALL)) {
2133                         HATSTAT_INC(hs_tlb_inval_delayed);
2134                         CPUSET_DEL(cpus_to_shootdown, c);
2135                 }
2136         }
2137 #endif

2139         if (CPUSET_ISNULL(cpus_to_shootdown) ||
2140             CPUSET_ISEQUAL(cpus_to_shootdown, justme)) {

2142 #ifdef __xpv
2143                 if (va == DEMAP_ALL_ADDR) {
2144                         xen_flush_tlb();
2145                 } else {
2146                         for (size_t i = 0; i < len; i += MMU_PAGESIZE)
2147                                 xen_flush_va((caddr_t)(va + i));
2148                 }
2149 #else
2150                 (void) hati_demap_func((xc_arg_t)hat,
2151                     (xc_arg_t)range, (xc_arg_t)len);
2137                     (xc_arg_t)va, (xc_arg_t)len);
2152 #endif

2154         } else {

2156                 CPUSET_ADD(cpus_to_shootdown, CPU->cpu_id);
2157 #ifdef __xpv
2158                 if (va == DEMAP_ALL_ADDR) {
2159                         xen_gflush_tlb(cpus_to_shootdown);
2160                 } else {
2161                         for (size_t i = 0; i < len; i += MMU_PAGESIZE) {
2162                                 xen_gflush_va((caddr_t)(va + i),
2163                                     cpus_to_shootdown);
2164                         }
2165                 }
```



```
2166 #else
2167                 xc_call((xc_arg_t)hat, (xc_arg_t)range, (xc_arg_t)len,
2153                 xc_call((xc_arg_t)hat, (xc_arg_t)va, (xc_arg_t)len,
2168                     CPUSET2BV(cpus_to_shootdown), hati_demap_func);
2169 #endif

2171         }
2172         kpreempt_enable();
2173 }

2175 void
2176 hat_tlb_inval(hat_t *hat, uintptr_t va)
2177 {
2178         /*
2179          * Create range for a single page.
2180          */
2181         range_info_t range;
2182         range.rng_va = va;
2183         range.rng_cnt = 1; /* one page */
2184         range.rng_level = MIN_PAGE_LEVEL; /* pages are MMU_PAGESIZE */

2186         hat_tlb_inval_range(hat, &range);
2164         hat_tlb_inval_range(hat, va, MMU_PAGESIZE);
2187 }
______unchanged_portion_omitted_

2352 /*
2331  * Do the callbacks for ranges being unloaded.
2332  */
2333 typedef struct range_info {
2334         uintptr_t       rng_va;
2335         ulong_t         rng_cnt;
2336         level_t         rng_level;
2337 } range_info_t;

2339 /*
2353  * Invalidate the TLB, and perform the callback to the upper level VM system,
2354  * for the specified ranges of contiguous pages.
2355  */
2356 static void
2357 handle_ranges(hat_t *hat, hat_callback_t *cb, uint_t cnt, range_info_t *range)
2358 {
2359         while (cnt > 0) {
2347                 size_t len;

2360                 --cnt;
2361                 hat_tlb_inval_range(hat, &range[cnt]);
2350                 len = range[cnt].rng_cnt << LEVEL_SHIFT(range[cnt].rng_level);
2351                 hat_tlb_inval_range(hat, (uintptr_t)range[cnt].rng_va, len);

2363                 if (cb != NULL) {
2364                         cb->hcb_start_addr = (caddr_t)range[cnt].rng_va;
2365                         cb->hcb_end_addr = cb->hcb_start_addr;
2366                         cb->hcb_end_addr += range[cnt].rng_cnt <<
2367                             LEVEL_SHIFT(range[cnt].rng_level);
2356                         cb->hcb_end_addr += len;
2368                         cb->hcb_function(cb);
2369                 }
2370         }
2371 }
______unchanged_portion_omitted_
```

```
**********************************************************
    9049 Wed Feb 28 17:34:23 2018
new/usr/src/uts/i86pc/vm/hat_pte.h
9208 hati_demap_func should take pagesize into account
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Tim Kordas <tim.kordas@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2017 Joyent, Inc.  All rights reserved.
 */

#ifndef _VM_HAT_PTE_H
#define _VM_HAT_PTE_H

#ifdef  __cplusplus
extern "C" {
#endif

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

/*
 * macros to get/set/clear the PTE fields
 */
#define PTE_SET(p, f)   ((p) |= (f))
#define PTE_CLR(p, f)   ((p) &= ~(x86pte_t)(f))
#define PTE_GET(p, f)   ((p) & (f))

/*
 * Handy macro to check if a pagetable entry or pointer is valid
 */
#define PTE_ISVALID(p)          PTE_GET(p, PT_VALID)

/*
 * Does a PTE map a large page.
 */
#define PTE_IS_LGPG(p, l)       ((l) > 0 && PTE_GET((p), PT_PAGESIZE))

/*
 * does this PTE represent a page (not a pointer to another page table)?
 */
#define PTE_ISPAGE(p, l)        \
        (PTE_ISVALID(p) && ((l) == 0 || PTE_GET(p, PT_PAGESIZE)))
```



```
/*
 * Handy macro to check if 2 PTE's are the same - ignores REF/MOD bits.
 * On the 64 bit hypervisor we also have to ignore the high order
 * software bits and the global/user bit which are set/cleared
 * capriciously (by the hypervisor!)
 */
#if defined(__amd64) && defined(__xpv)
#define PT_IGNORE       ((0x7ffUL << 52) | PT_GLOBAL | PT_USER)
#else
#define PT_IGNORE       (0)
#endif
#define PTE_EQUIV(a, b)  (((a) | (PT_IGNORE | PT_REF | PT_MOD)) == \
        ((b) | (PT_IGNORE | PT_REF | PT_MOD)))

/*
 * Shorthand for converting a PTE to it's pfn.
 */
#define PTE2MFN(p, l)   \
        mmu_btop(PTE_GET((p), PTE_IS_LGPG((p), (l)) ? PT_PADDR_LGPG : PT_PADDR))
#ifdef __xpv
#define PTE2PFN(p, l) pte2pfn(p, l)
#else
#define PTE2PFN(p, l) PTE2MFN(p, l)
#endif

#define PT_NX           (0x8000000000000000ull)
#define PT_PADDR        (0x000ffffffffff000ull)
#define PT_PADDR_LGPG   (0x000fffffffffe000ull) /* phys addr for large pages */

/*
 * Macros to create a PTP or PTE from the pfn and level
 */
#ifdef __xpv

/*
 * we use the highest order bit in physical address pfns to mark foreign mfns
 */
#ifdef _LP64
#define PFN_IS_FOREIGN_MFN (1ul << 51)
#else
#define PFN_IS_FOREIGN_MFN (1ul << 31)
#endif

#define MAKEPTP(pfn, l) \
        (pa_to_ma(pfn_to_pa(pfn)) | mmu.ptp_bits[(l) + 1])
#define MAKEPTE(pfn, l) \
        ((pfn & PFN_IS_FOREIGN_MFN) ? \
        ((pfn_to_pa(pfn & ~PFN_IS_FOREIGN_MFN) | mmu.pte_bits[l]) | \
        PT_FOREIGN | PT_REF | PT_MOD) : \
        (pa_to_ma(pfn_to_pa(pfn)) | mmu.pte_bits[l]))
#else
#define MAKEPTP(pfn, l) \
        (pfn_to_pa(pfn) | mmu.ptp_bits[(l) + 1])
#define MAKEPTE(pfn, l) \
        (pfn_to_pa(pfn) | mmu.pte_bits[l])
#endif

/*
 * The idea of "level" refers to the level where the page table is used in the
 * the hardware address translation steps. The level values correspond to the
 * following names of tables used in AMD/Intel architecture documents:
 *
 *      AMD/INTEL name          Level #
 *      ----------------------  -------
 *      Page Map Level 4           3
 *      Page Directory Pointer     2
```

```
 126  *      Page Directory             1
 127  *      Page Table                 0
 128  *
 129  * The numbering scheme is such that the values of 0 and 1 can correspond to
 130  * the pagesize codes used for MPSS support. For now the Maximum level at
 131  * which you can have a large page is a constant, that may change in
 132  * future processors.
 133  *
 134  * The type of "level_t" is signed so that it can be used like:
 135  *      level_t l;
 136  *      ...
 137  *      while (--l >= 0)
 138  *              ...
 139  */
 140 #define MAX_NUM_LEVEL           4
 141 #define MAX_PAGE_LEVEL          2
 142 #define MIN_PAGE_LEVEL          0
 143 typedef int8_t level_t;
 144 #define LEVEL_SHIFT(l)  (mmu.level_shift[l])
 145 #define LEVEL_SIZE(l)   (mmu.level_size[l])
 146 #define LEVEL_OFFSET(l) (mmu.level_offset[l])
 147 #define LEVEL_MASK(l)   (mmu.level_mask[l])

 149 /*
 150  * Macros to:
 151  * Check for a PFN above 4Gig and 64Gig for 32 bit PAE support
 152  */
 153 #define PFN_4G          (4ull * (1024 * 1024 * 1024 / MMU_PAGESIZE))
 154 #define PFN_64G         (64ull * (1024 * 1024 * 1024 / MMU_PAGESIZE))
 155 #define PFN_ABOVE4G(pfn) ((pfn) >= PFN_4G)
 156 #define PFN_ABOVE64G(pfn) ((pfn) >= PFN_64G)

 158 /*
 159  * The CR3 register holds the physical address of the top level page table.
 160  */
 161 #define MAKECR3(pfn)    mmu_ptob(pfn)

 163 /*
 164  * HAT/MMU parameters that depend on kernel mode and/or processor type
 165  */
 166 struct htable;
 167 struct hat_mmu_info {
 168         x86pte_t pt_nx;         /* either 0 or PT_NX */
 169         x86pte_t pt_global;     /* either 0 or PT_GLOBAL */

 171         pfn_t highest_pfn;

 173         uint_t num_level;       /* number of page table levels in use */
 174         uint_t max_level;       /* just num_level - 1 */
 175         uint_t max_page_level;  /* maximum level at which we can map a page */
 176         uint_t umax_page_level; /* max user page map level */
 177         uint_t ptes_per_table;  /* # of entries in lower level page tables */
 178         uint_t top_level_count; /* # of entries in top most level page table */

 180         uint_t  hash_cnt;       /* cnt of entries in htable_hash_cache */
 181         uint_t  vlp_hash_cnt;   /* cnt of entries in vlp htable_hash_cache */

 183         uint_t pae_hat;         /* either 0 or 1 */

 185         uintptr_t hole_start;   /* start of VA hole (or -1 if none) */
 186         uintptr_t hole_end;     /* end of VA hole (or 0 if none) */

 188         struct htable **kmap_htables; /* htables for segmap + 32 bit heap */
 189         x86pte_t *kmap_ptes;    /* mapping of pagetables that map kmap */
 190         uintptr_t kmap_addr;    /* start addr of kmap */
 191         uintptr_t kmap_eaddr;   /* end addr of kmap */
```



```
 193         uint_t pte_size;        /* either 4 or 8 */
 194         uint_t pte_size_shift;  /* either 2 or 3 */
 195         x86pte_t ptp_bits[MAX_NUM_LEVEL];       /* bits set for interior PTP */
 196         x86pte_t pte_bits[MAX_NUM_LEVEL];       /* bits set for leaf PTE */

 198         /*
 199          * A range of VA used to window pages in the i86pc/vm code.
 200          * See PWIN_XXX macros.
 201          */
 202         caddr_t pwin_base;
 203         caddr_t pwin_pte_va;
 204         paddr_t pwin_pte_pa;

 206         /*
 207          * The following tables are equivalent to PAGEXXXXX at different levels
 208          * in the page table hierarchy.
 209          */
 210         uint_t level_shift[MAX_NUM_LEVEL];      /* PAGESHIFT for given level */
 211         uintptr_t level_size[MAX_NUM_LEVEL];    /* PAGESIZE for given level */
 212         uintptr_t level_offset[MAX_NUM_LEVEL];  /* PAGEOFFSET for given level */
 213         uintptr_t level_mask[MAX_NUM_LEVEL];    /* PAGEMASK for given level */
 214 };
______unchanged_portion_omitted_
```

```
**********************************************************
    1978 Wed Feb 28 17:34:23 2018
new/usr/src/uts/intel/amd64/sys/kdi_regs.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#ifndef _AMD64_SYS_KDI_REGS_H
#define _AMD64_SYS_KDI_REGS_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#ifdef __cplusplus
extern "C" {
#endif

#define KDIREG_NGREG    31

/*
 * A modified version of struct regs layout.
 */

#define KDIREG_SAVFP    0
#define KDIREG_SAVPC    1
#define KDIREG_RDI      2
#define KDIREG_RSI      3
#define KDIREG_RDX      4
#define KDIREG_RCX      5
#define KDIREG_R8       6
#define KDIREG_R9       7
#define KDIREG_RAX      8
#define KDIREG_RBX      9
#define KDIREG_RBP      10
#define KDIREG_R10      11
#define KDIREG_R11      12
#define KDIREG_R12      13
#define KDIREG_R13      14
```



```
#define KDIREG_R14      15
#define KDIREG_R15      16
#define KDIREG_FSBASE   17
#define KDIREG_GSBASE   18
#define KDIREG_KGSBASE  19
#define KDIREG_DS       20
#define KDIREG_ES       21
#define KDIREG_FS       22
#define KDIREG_GS       23
#define KDIREG_TRAPNO   24
#define KDIREG_ERR      25
#define KDIREG_RIP      26
#define KDIREG_CS       27
#define KDIREG_RFLAGS   28
#define KDIREG_RSP      29
#define KDIREG_SS       30

#define KDIREG_PC       KDIREG_RIP
#define KDIREG_SP       KDIREG_RSP
#define KDIREG_FP       KDIREG_RBP

#ifdef _ASM

/* Patch point for MSR clearing. */
#define KDI_MSR_PATCH \
        nop; nop; nop; nop; \
        nop; nop; nop; nop; \
        nop; nop; nop; nop; \
        nop; nop; nop; nop; \
        nop

#endif  /* _ASM */

#define KDI_MSR_PATCHOFF        8       /* bytes of code before patch point */
#define KDI_MSR_PATCHSZ         17      /* bytes in KDI_MSR_PATCH, above */

#ifdef __cplusplus
}
```

_____unchanged_portion_omitted_

```
**********************************************************
    1850 Wed Feb 28 17:34:23 2018
new/usr/src/uts/intel/ia32/sys/kdi_regs.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#ifndef _IA32_SYS_KDI_REGS_H
#define _IA32_SYS_KDI_REGS_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#ifdef __cplusplus
extern "C" {
#endif

#define KDIREG_NGREG    21

/*
 * %ss appears in a different place than a typical struct regs, since the
 * machine won't save %ss on a trap entry from the same privilege level.
 */

#define KDIREG_SAVFP    0
#define KDIREG_SAVPC    1
#define KDIREG_SS       2
#define KDIREG_GS       3
#define KDIREG_FS       4
#define KDIREG_ES       5
#define KDIREG_DS       6
#define KDIREG_EDI      7
#define KDIREG_ESI      8
#define KDIREG_EBP      9
#define KDIREG_ESP      10
#define KDIREG_EBX      11
#define KDIREG_EDX      12
#define KDIREG_ECX      13
```



```
#define KDIREG_EAX      14
#define KDIREG_TRAPNO   15
#define KDIREG_ERR      16
#define KDIREG_EIP      17
#define KDIREG_CS       18
#define KDIREG_EFLAGS   19
#define KDIREG_UESP     20

#define KDIREG_PC       KDIREG_EIP
#define KDIREG_SP       KDIREG_ESP
#define KDIREG_FP       KDIREG_EBP

#ifdef _ASM

/* Patch point for MSR clearing. */
#define KDI_MSR_PATCH \
        nop; nop; nop; nop; \
        nop; nop; nop; nop; \
        nop; nop; nop; nop; \
        nop

#endif /* _ASM */

#define KDI_MSR_PATCHOFF        8       /* bytes of code before patch point */
#define KDI_MSR_PATCHSZ         13      /* bytes in KDI_MSR_PATCH, above */

#ifdef __cplusplus
}
_____unchanged_portion_omitted_
```

```
**********************************************************
   15759 Wed Feb 28 17:34:24 2018
new/usr/src/uts/intel/kdi/amd64/kdi_asm.s
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Debugger entry for both master and slave CPUs
 */

#if defined(__lint)
#include <sys/types.h>
#endif

#include <sys/segments.h>
#include <sys/asm_linkage.h>
#include <sys/controlregs.h>
#include <sys/x86_archext.h>
#include <sys/privregs.h>
#include <sys/machprivregs.h>
#include <sys/kdi_regs.h>
#include <sys/psw.h>
#include <sys/uadmin.h>
#ifdef __xpv
#include <sys/hypervisor.h>
#endif

#ifdef _ASM

#include <kdi_assym.h>
#include <assym.h>

/* clobbers %rdx, %rcx, returns addr in %rax, CPU ID in %rbx */
#define GET_CPUSAVE_ADDR \
```



```
        movzbq  %gs:CPU_ID, %rbx;               \
        movq    %rbx, %rax;                     \
        movq    $KRS_SIZE, %rcx;                \
        mulq    %rcx;                           \
        movq    $kdi_cpusave, %rdx;             \
        /*CSTYLED*/                             \
        addq    (%rdx), %rax

/*
 * Save copies of the IDT and GDT descriptors.  Note that we only save the IDT
 * and GDT if the IDT isn't ours, as we may be legitimately re-entering the
 * debugger through the trap handler.  We don't want to clobber the saved IDT
 * in the process, as we'd end up resuming the world on our IDT.
 */
#define SAVE_IDTGDT                             \
        movq    %gs:CPU_IDT, %r11;              \
        leaq    kdi_idt(%rip), %rsi;            \
        cmpq    %rsi, %r11;                     \
        je      1f;                             \
        movq    %r11, KRS_IDT(%rax);            \
        movq    %gs:CPU_GDT, %r11;              \
        movq    %r11, KRS_GDT(%rax);            \
1:

#ifdef __xpv

#define SAVE_GSBASE(reg) /* nothing */
#define RESTORE_GSBASE(reg) /* nothing */

#else

#define SAVE_GSBASE(base)                               \
        movl    $MSR_AMD_GSBASE, %ecx;                  \
        rdmsr;                                          \
        shlq    $32, %rdx;                              \
        orq     %rax, %rdx;                             \
        movq    %rdx, REG_OFF(KDIREG_GSBASE)(base)

#define RESTORE_GSBASE(base)                            \
        movq    REG_OFF(KDIREG_GSBASE)(base), %rdx;     \
        movq    %rdx, %rax;                             \
        shrq    $32, %rdx;                              \
        movl    $MSR_AMD_GSBASE, %ecx;                  \
        wrmsr

#endif /* __xpv */

/*
 * %ss, %rsp, %rflags, %cs, %rip, %err, %trapno are already on the stack.  Note
 * that on the hypervisor, we skip the save/restore of GSBASE: it's slow, and
 * unnecessary.
 */
#define KDI_SAVE_REGS(base) \
        movq    %rdi, REG_OFF(KDIREG_RDI)(base);        \
        movq    %rsi, REG_OFF(KDIREG_RSI)(base);        \
        movq    %rdx, REG_OFF(KDIREG_RDX)(base);        \
        movq    %rcx, REG_OFF(KDIREG_RCX)(base);        \
        movq    %r8, REG_OFF(KDIREG_R8)(base);          \
        movq    %r9, REG_OFF(KDIREG_R9)(base);          \
        movq    %rax, REG_OFF(KDIREG_RAX)(base);        \
        movq    %rbx, REG_OFF(KDIREG_RBX)(base);        \
        movq    %rbp, REG_OFF(KDIREG_RBP)(base);        \
        movq    %r10, REG_OFF(KDIREG_R10)(base);        \
        movq    %r11, REG_OFF(KDIREG_R11)(base);        \
        movq    %r12, REG_OFF(KDIREG_R12)(base);        \
        movq    %r13, REG_OFF(KDIREG_R13)(base);        \
```

```
123         movq    %r14, REG_OFF(KDIREG_R14)(base);        \
124         movq    %r15, REG_OFF(KDIREG_R15)(base);        \
125         movq    %rbp, REG_OFF(KDIREG_SAVFP)(base);      \
126         movq    REG_OFF(KDIREG_RIP)(base), %rax;        \
127         movq    %rax, REG_OFF(KDIREG_SAVPC)(base);      \
128         clrq    %rax;                                   \
129         movw    %ds, %ax;                               \
130         movq    %rax, REG_OFF(KDIREG_DS)(base);         \
131         movw    %es, %ax;                               \
132         movq    %rax, REG_OFF(KDIREG_ES)(base);         \
133         movw    %fs, %ax;                               \
134         movq    %rax, REG_OFF(KDIREG_FS)(base);         \
135         movw    %gs, %ax;                               \
136         movq    %rax, REG_OFF(KDIREG_GS)(base);         \
137         SAVE_GSBASE(base)

139 #define KDI_RESTORE_REGS(base) \
140         movq    base, %rdi;                             \
141         RESTORE_GSBASE(%rdi);                           \
142         movq    REG_OFF(KDIREG_ES)(%rdi), %rax;         \
143         movw    %ax, %es;                               \
144         movq    REG_OFF(KDIREG_DS)(%rdi), %rax;         \
145         movw    %ax, %ds;                               \
146         movq    REG_OFF(KDIREG_R15)(%rdi), %r15;        \
147         movq    REG_OFF(KDIREG_R14)(%rdi), %r14;        \
148         movq    REG_OFF(KDIREG_R13)(%rdi), %r13;        \
149         movq    REG_OFF(KDIREG_R12)(%rdi), %r12;        \
150         movq    REG_OFF(KDIREG_R11)(%rdi), %r11;        \
151         movq    REG_OFF(KDIREG_R10)(%rdi), %r10;        \
152         movq    REG_OFF(KDIREG_RBP)(%rdi), %rbp;        \
153         movq    REG_OFF(KDIREG_RBX)(%rdi), %rbx;        \
154         movq    REG_OFF(KDIREG_RAX)(%rdi), %rax;        \
155         movq    REG_OFF(KDIREG_R9)(%rdi), %r9;          \
156         movq    REG_OFF(KDIREG_R8)(%rdi), %r8;          \
157         movq    REG_OFF(KDIREG_RCX)(%rdi), %rcx;        \
158         movq    REG_OFF(KDIREG_RDX)(%rdi), %rdx;        \
159         movq    REG_OFF(KDIREG_RSI)(%rdi), %rsi;        \
160         movq    REG_OFF(KDIREG_RDI)(%rdi), %rdi

162 /*
163  * Given the address of the current CPU's cpusave area in %rax, the following
164  * macro restores the debugging state to said CPU.  Restored state includes
165  * the debug registers from the global %dr variables.
165  * the debug registers from the global %dr variables, and debugging MSRs from
166  * the CPU save area.  This code would be in a separate routine, but for the
167  * fact that some of the MSRs are jump-sensitive.  As such, we need to minimize
168  * the number of jumps taken subsequent to the update of said MSRs.  We can
169  * remove one jump (the ret) by using a macro instead of a function for the
170  * debugging state restoration code.
166  *
167  * Takes the cpusave area in %rdi as a parameter.
172  * Takes the cpusave area in %rdi as a parameter, clobbers %rax-%rdx
168  */
169 #define KDI_RESTORE_DEBUGGING_STATE \
170         pushq   %rdi;                                           \
171         leaq    kdi_drreg(%rip), %r15;                          \
172         movl    $7, %edi;                                       \
173         movq    DR_CTL(%r15), %rsi;                             \
174         call    kdi_dreg_set;                                   \
175                                                                 \
176         movl    $6, %edi;                                       \
177         movq    $KDIREG_DRSTAT_RESERVED, %rsi;                  \
178         call    kdi_dreg_set;                                   \
179                                                                 \
180         movl    $0, %edi;                                       \
181         movq    DRADDR_OFF(0)(%r15), %rsi;                      \
```



```
182         call    kdi_dreg_set;                                   \
183         movl    $1, %edi;                                       \
184         movq    DRADDR_OFF(1)(%r15), %rsi;                      \
185         call    kdi_dreg_set;                                   \
186         movl    $2, %edi;                                       \
187         movq    DRADDR_OFF(2)(%r15), %rsi;                      \
188         call    kdi_dreg_set;                                   \
189         movl    $3, %edi;                                       \
190         movq    DRADDR_OFF(3)(%r15), %rsi;                      \
191         call    kdi_dreg_set;                                   \
192         popq    %rdi;
197         popq    %rdi;                                           \
198                                                                 \
199         /*                                                      \
200          * Write any requested MSRs.                            \
201          */                                                     \
202         movq    KRS_MSR(%rdi), %rbx;                            \
203         cmpq    $0, %rbx;                                       \
204         je      3f;                                             \
205 1:                                                              \
206         movl    MSR_NUM(%rbx), %ecx;                            \
207         cmpl    $0, %ecx;                                       \
208         je      3f;                                             \
209                                                                 \
210         movl    MSR_TYPE(%rbx), %edx;                           \
211         cmpl    $KDI_MSR_WRITE, %edx;                           \
212         jne     2f;                                             \
213                                                                 \
214         movq    MSR_VALP(%rbx), %rdx;                           \
215         movl    0(%rdx), %eax;                                  \
216         movl    4(%rdx), %edx;                                  \
217         wrmsr;                                                  \
218 2:                                                              \
219         addq    $MSR_SIZE, %rbx;                                \
220         jmp     1b;                                             \
221 3:                                                              \
222         /*                                                      \
223          * We must not branch after re-enabling LBR.  If       \
224          * kdi_wsr_wrexit_msr is set, it contains the number    \
225          * of the MSR that controls LBR.  kdi_wsr_wrexit_valp   \
226          * contains the value that is to be written to enable   \
227          * LBR.                                                 \
228          */                                                     \
229         leaq    kdi_msr_wrexit_msr(%rip), %rcx;                 \
230         movl    (%rcx), %ecx;                                   \
231         cmpl    $0, %ecx;                                       \
232         je      1f;                                             \
233                                                                 \
234         leaq    kdi_msr_wrexit_valp(%rip), %rdx;                \
235         movq    (%rdx), %rdx;                                   \
236         movl    0(%rdx), %eax;                                  \
237         movl    4(%rdx), %edx;                                  \
238                                                                 \
239         wrmsr;                                                  \
240 1:

194 /*
195  * Each cpusave buffer has an area set aside for a ring buffer of breadcrumbs.
196  * The following macros manage the buffer.
197  */

199 /* Advance the ring buffer */
200 #define ADVANCE_CRUMB_POINTER(cpusave, tmp1, tmp2) \
201         movq    KRS_CURCRUMBIDX(cpusave), tmp1; \
202         cmpq    $[KDI_NCRUMBS - 1], tmp1;       \
203         jge     1f;                             \
```

```
204         /* Advance the pointer and index */     \
205         addq    $1, tmp1;                       \
206         movq    tmp1, KRS_CURCRUMBIDX(cpusave); \
207         movq    KRS_CURCRUMB(cpusave), tmp1;    \
208         addq    $KRM_SIZE, tmp1;                \
209         jmp     2f;                             \
210 1:      /* Reset the pointer and index */       \
211         movq    $0, KRS_CURCRUMBIDX(cpusave);   \
212         leaq    KRS_CRUMBS(cpusave), tmp1;      \
213 2:      movq    tmp1, KRS_CURCRUMB(cpusave);    \
214         /* Clear the new crumb */               \
215         movq    $KDI_NCRUMBS, tmp2;             \
216 3:      movq    $0, -4(tmp1, tmp2, 4);          \
217         decq    tmp2;                           \
218         jnz     3b

220 /* Set a value in the current breadcrumb buffer */
221 #define ADD_CRUMB(cpusave, offset, value, tmp) \
222         movq    KRS_CURCRUMB(cpusave), tmp;     \
223         movq    value, offset(tmp)

225 #endif  /* _ASM */

227 #if defined(__lint)
228 void
229 kdi_cmnint(void)
230 {
231 }
_____unchanged_portion_omitted_

342 #endif  /* __lint */

344 /*
345  * The cross-call handler for slave CPUs.
346  *
347  * The debugger is single-threaded, so only one CPU, called the master, may be
348  * running it at any given time.  The other CPUs, known as slaves, spin in a
349  * busy loop until there's something for them to do.  This is the entry point
350  * for the slaves - they'll be sent here in response to a cross-call sent by the
351  * master.
352  */

354 #if defined(__lint)
403 char kdi_slave_entry_patch;

355 void
356 kdi_slave_entry(void)
357 {
358 }
359 #else /* __lint */
410         .globl  kdi_slave_entry_patch;

360         ENTRY_NP(kdi_slave_entry)

414         /* kdi_msr_add_clrentry knows where this is */
415 kdi_slave_entry_patch:
416         KDI_MSR_PATCH;

362         /*
363          * Cross calls are implemented as function calls, so our stack currently
364          * looks like one you'd get from a zero-argument function call.  That
365          * is, there's the return %rip at %rsp, and that's about it.  We need
366          * to make it look like an interrupt stack.  When we first save, we'll
367          * reverse the saved %ss and %rip, which we'll fix back up when we've
368          * freed up some general-purpose registers.  We'll also need to fix up
369          * the saved %rsp.
```



```
370          */

372         pushq   %rsp            /* pushed value off by 8 */
373         pushfq
374         CLI(%rax)
375         pushq   $KCS_SEL
376         clrq    %rax
377         movw    %ss, %ax
378         pushq   %rax            /* rip should be here */
379         pushq   $-1             /* phony trap error code */
380         pushq   $-1             /* phony trap number */

382         subq    $REG_OFF(KDIREG_TRAPNO), %rsp
383         KDI_SAVE_REGS(%rsp)

385         movq    REG_OFF(KDIREG_SS)(%rsp), %rax
386         xchgq   REG_OFF(KDIREG_RIP)(%rsp), %rax
387         movq    %rax, REG_OFF(KDIREG_SS)(%rsp)

389         movq    REG_OFF(KDIREG_RSP)(%rsp), %rax
390         addq    $8, %rax
391         movq    %rax, REG_OFF(KDIREG_RSP)(%rsp)

393         /*
394          * We've saved all of the general-purpose registers, and have a stack
395          * that is irettable (after we strip down to the error code)
396          */

398         GET_CPUSAVE_ADDR        /* %rax = cpusave, %rbx = CPU ID */

400         ADVANCE_CRUMB_POINTER(%rax, %rcx, %rdx)

402         ADD_CRUMB(%rax, KRM_CPU_STATE, $KDI_CPU_STATE_SLAVE, %rdx)

404         movq    REG_OFF(KDIREG_RIP)(%rsp), %rcx
405         ADD_CRUMB(%rax, KRM_PC, %rcx, %rdx)

407         pushq   %rax
408         jmp     kdi_save_common_state

410         SET_SIZE(kdi_slave_entry)

412 #endif  /* __lint */

414 /*
415  * The state of the world:
416  *
417  * The stack has a complete set of saved registers and segment
418  * selectors, arranged in the kdi_regs.h order.  It also has a pointer
419  * to our cpusave area.
420  *
421  * We need to save, into the cpusave area, a pointer to these saved
422  * registers.  First we check whether we should jump straight back to
423  * the kernel.  If not, we save a few more registers, ready the
424  * machine for debugger entry, and enter the debugger.
425  */

427 #if !defined(__lint)

429         ENTRY_NP(kdi_save_common_state)

431         popq    %rdi                    /* the cpusave area */
432         movq    %rsp, KRS_GREGS(%rdi)   /* save ptr to current saved regs */

434         pushq   %rdi
435         call    kdi_trap_pass
```

```
436         cmpq    $1, %rax
437         je      kdi_pass_to_kernel
438         popq    %rax /* cpusave in %rax */

440         SAVE_IDTGDT

442 #if !defined(__xpv)
443         /* Save off %cr0, and clear write protect */
444         movq    %cr0, %rcx
445         movq    %rcx, KRS_CR0(%rax)
446         andq    $_BITNOT(CR0_WP), %rcx
447         movq    %rcx, %cr0
448 #endif

450         /* Save the debug registers and disable any active watchpoints */

452         movq    %rax, %r15              /* save cpusave area ptr */
453         movl    $7, %edi
454         call    kdi_dreg_get
455         movq    %rax, KRS_DRCTL(%r15)

457         andq    $_BITNOT(KDIREG_DRCTL_WPALLEN_MASK), %rax
458         movq    %rax, %rsi
459         movl    $7, %edi
460         call    kdi_dreg_set

462         movl    $6, %edi
463         call    kdi_dreg_get
464         movq    %rax, KRS_DRSTAT(%r15)

466         movl    $0, %edi
467         call    kdi_dreg_get
468         movq    %rax, KRS_DROFF(0)(%r15)

470         movl    $1, %edi
471         call    kdi_dreg_get
472         movq    %rax, KRS_DROFF(1)(%r15)

474         movl    $2, %edi
475         call    kdi_dreg_get
476         movq    %rax, KRS_DROFF(2)(%r15)

478         movl    $3, %edi
479         call    kdi_dreg_get
480         movq    %rax, KRS_DROFF(3)(%r15)

482         movq    %r15, %rax      /* restore cpu save area to rax */

540         /*
541          * Save any requested MSRs.
542          */
543         movq    KRS_MSR(%rax), %rcx
544         cmpq    $0, %rcx
545         je      no_msr

547         pushq   %rax            /* rdmsr clobbers %eax */
548         movq    %rcx, %rbx

550 1:
551         movl    MSR_NUM(%rbx), %ecx
552         cmpl    $0, %ecx
553         je      msr_done

555         movl    MSR_TYPE(%rbx), %edx
556         cmpl    $KDI_MSR_READ, %edx
557         jne     msr_next
```



```
559         rdmsr                   /* addr in %ecx, value into %edx:%eax */
560         movl    %eax, MSR_VAL(%rbx)
561         movl    %edx, _CONST(MSR_VAL + 4)(%rbx)

563 msr_next:
564         addq    $MSR_SIZE, %rbx
565         jmp     1b

567 msr_done:
568         popq    %rax

570 no_msr:
484         clrq    %rbp            /* stack traces should end here */

486         pushq   %rax
487         movq    %rax, %rdi      /* cpusave */

489         call    kdi_debugger_entry

491         /* Pass cpusave to kdi_resume */
492         popq    %rdi

494         jmp     kdi_resume

496         SET_SIZE(kdi_save_common_state)
```

_____unchanged_portion_omitted_

```
**********************************************************
   15151 Wed Feb 28 17:34:24 2018
new/usr/src/uts/intel/kdi/ia32/kdi_asm.s
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Debugger entry for both master and slave CPUs
 */

#if defined(__lint)
#include <sys/types.h>
#endif

#include <sys/segments.h>
#include <sys/asm_linkage.h>
#include <sys/controlregs.h>
#include <sys/x86_archext.h>
#include <sys/privregs.h>
#include <sys/machprivregs.h>
#include <sys/kdi_regs.h>
#include <sys/uadmin.h>
#include <sys/psw.h>

#ifdef _ASM

#include <kdi_assym.h>
#include <assym.h>

/* clobbers %edx, %ecx, returns addr in %eax, cpu id in %ebx */
#define GET_CPUSAVE_ADDR \
        movl    %gs:CPU_ID, %ebx;               \
        movl    %ebx, %eax;                     \
        movl    $KRS_SIZE, %ecx;                \
        mull    %ecx;                           \
        movl    $kdi_cpusave, %edx;             \
        /*CSTYLED*/                             \
        addl    (%edx), %eax

/*
 * Save copies of the IDT and GDT descriptors.  Note that we only save the IDT
 * and GDT if the IDT isn't ours, as we may be legitimately re-entering the
 * debugger through the trap handler.  We don't want to clobber the saved IDT
 * in the process, as we'd end up resuming the world on our IDT.
 */
#define SAVE_IDTGDT                             \
        movl    %gs:CPU_IDT, %edx;              \
        cmpl    $kdi_idt, %edx;                 \
        je      1f;                             \
        movl    %edx, KRS_IDT(%eax);            \
        movl    %gs:CPU_GDT, %edx;              \
        movl    %edx, KRS_GDT(%eax);            \
1:

/*
 * Given the address of the current CPU's cpusave area in %edi, the following
 * macro restores the debugging state to said CPU.  Restored state includes
 * the debug registers from the global %dr variables.
 * the debug registers from the global %dr variables, and debugging MSRs from
 * the CPU save area.  This code would be in a separate routine, but for the
 * fact that some of the MSRs are jump-sensitive.  As such, we need to minimize
 * the number of jumps taken subsequent to the update of said MSRs.  We can
 * remove one jump (the ret) by using a macro instead of a function for the
 * debugging state restoration code.
 *
 * Takes the cpusave area in %edi as a parameter, clobbers %eax-%edx
 */
#define KDI_RESTORE_DEBUGGING_STATE \
        leal    kdi_drreg, %ebx;                                \
                                                                \
        pushl   DR_CTL(%ebx);                                   \
        pushl   $7;                                             \
        call    kdi_dreg_set;                                   \
        addl    $8, %esp;                                       \
                                                                \
        pushl   $KDIREG_DRSTAT_RESERVED;                                \
        pushl   $6;                                             \
        call    kdi_dreg_set;                                   \
        addl    $8, %esp;                                       \
                                                                \
        pushl   DRADDR_OFF(0)(%ebx);                            \
        pushl   $0;                                             \
        call    kdi_dreg_set;                                   \
        addl    $8, %esp;                                       \
                                                                \
        pushl   DRADDR_OFF(1)(%ebx);                            \
        pushl   $1;                                             \
        call    kdi_dreg_set;                                   \
        addl    $8, %esp;                                       \
                                                                \
        pushl   DRADDR_OFF(2)(%ebx);                            \
        pushl   $2;                                             \
        call    kdi_dreg_set;                                   \
        addl    $8, %esp;                                       \
                                                                \
        pushl   DRADDR_OFF(3)(%ebx);                            \
        pushl   $3;                                             \
        call    kdi_dreg_set;                                   \
        addl    $8, %esp;                                       \
```

```
121                                                                     \
122         /*                                                          \
123          * Write any requested MSRs.                                \
124          */                                                         \
125         movl    KRS_MSR(%edi), %ebx;                                \
126         cmpl    $0, %ebx;                                           \
127         je      3f;                                                 \
128 1:                                                                  \
129         movl    MSR_NUM(%ebx), %ecx;                                \
130         cmpl    $0, %ecx;                                           \
131         je      3f;                                                 \
132                                                                     \
133         movl    MSR_TYPE(%ebx), %edx;                               \
134         cmpl    $KDI_MSR_WRITE, %edx;                               \
135         jne     2f;                                                 \
136                                                                     \
137         movl    MSR_VALP(%ebx), %edx;                               \
138         movl    0(%edx), %eax;                                      \
139         movl    4(%edx), %edx;                                      \
140         wrmsr;                                                      \
141 2:                                                                  \
142         addl    $MSR_SIZE, %ebx;                                    \
143         jmp     1b;                                                 \
144 3:                                                                  \
145         /*                                                          \
146          * We must not branch after re-enabling LBR.  If           \
147          * kdi_wsr_wrexit_msr is set, it contains the number       \
148          * of the MSR that controls LBR.  kdi_wsr_wrexit_valp      \
149          * contains the value that is to be written to enable      \
150          * LBR.                                                    \
151          */                                                         \
152         movl    kdi_msr_wrexit_msr, %ecx;                           \
153         cmpl    $0, %ecx;                                           \
154         je      1f;                                                 \
155                                                                     \
156         movl    kdi_msr_wrexit_valp, %edx;                          \
157         movl    0(%edx), %eax;                                      \
158         movl    4(%edx), %edx;                                      \
159                                                                     \
160         wrmsr;                                                      \
161 1:
```

```
115 #define KDI_RESTORE_REGS() \
116         /* Discard savfp and savpc */ \
117         addl    $8, %esp; \
118         popl    %ss; \
119         popl    %gs; \
120         popl    %fs; \
121         popl    %es; \
122         popl    %ds; \
123         popal; \
124         /* Discard trapno and err */ \
125         addl    $8, %esp

127 /*
128  * Each cpusave buffer has an area set aside for a ring buffer of breadcrumbs.
129  * The following macros manage the buffer.
130  */

132 /* Advance the ring buffer */
133 #define ADVANCE_CRUMB_POINTER(cpusave, tmp1, tmp2) \
134         movl    KRS_CURCRUMBIDX(cpusave), tmp1; \
135         cmpl    $[KDI_NCRUMBS - 1], tmp1;       \
136         jge     1f;                             \
137         /* Advance the pointer and index */     \
138         addl    $1, tmp1;                       \
```



```
139         movl    tmp1, KRS_CURCRUMBIDX(cpusave); \
140         movl    KRS_CURCRUMB(cpusave), tmp1;    \
141         addl    $KRM_SIZE, tmp1;                \
142         jmp     2f;                             \
143 1:      /* Reset the pointer and index */       \
144         movw    $0, KRS_CURCRUMBIDX(cpusave);   \
145         leal    KRS_CRUMBS(cpusave), tmp1;      \
146 2:      movl    tmp1, KRS_CURCRUMB(cpusave);    \
147         /* Clear the new crumb */               \
148         movl    $KDI_NCRUMBS, tmp2;             \
149 3:      movl    $0, -4(tmp1, tmp2, 4);          \
150         decl    tmp2;                           \
151         jnz     3b

153 /* Set a value in the current breadcrumb buffer */
154 #define ADD_CRUMB(cpusave, offset, value, tmp) \
155         movl    KRS_CURCRUMB(cpusave), tmp;     \
156         movl    value, offset(tmp)

158 #endif  /* _ASM */

160 /*
161  * The main entry point for master CPUs.  It also serves as the trap handler
162  * for all traps and interrupts taken during single-step.
163  */
164 #if defined(__lint)
165 void
166 kdi_cmnint(void)
167 {
168 }
_____*unchanged_portion_omitted_*

316 #endif  /* __lint */

318 /*
319  * The cross-call handler for slave CPUs.
320  *
321  * The debugger is single-threaded, so only one CPU, called the master, may be
322  * running it at any given time.  The other CPUs, known as slaves, spin in a
323  * busy loop until there's something for them to do.  This is the entry point
324  * for the slaves - they'll be sent here in response to a cross-call sent by the
325  * master.
326  */

328 #if defined(__lint)
377 char kdi_slave_entry_patch;

329 void
330 kdi_slave_entry(void)
331 {
332 }
333 #else /* __lint */
384         .globl  kdi_slave_entry_patch;

334         ENTRY_NP(kdi_slave_entry)

388         /* kdi_msr_add_clrentry knows where this is */
389 kdi_slave_entry_patch:
390         KDI_MSR_PATCH;

336         /*
337          * Cross calls are implemented as function calls, so our stack
338          * currently looks like one you'd get from a zero-argument function
339          * call. There's an %eip at %esp, and that's about it.  We want to
340          * make it look like the master CPU's stack.  By doing this, we can
341          * use the same resume code for both master and slave.  We need to
```

```
342          * make our stack look like a 'struct regs' before we jump into the
343          * common save routine.
344          */

346         pushl   %cs
347         pushfl
348         pushl   $-1             /* A phony trap error code */
349         pushl   $-1             /* A phony trap number */
350         pushal
351         pushl   %ds
352         pushl   %es
353         pushl   %fs
354         pushl   %gs
355         pushl   %ss

357         subl    $8, %esp
358         movl    %ebp, REG_OFF(KDIREG_SAVFP)(%esp)
359         movl    REG_OFF(KDIREG_EIP)(%esp), %eax
360         movl    %eax, REG_OFF(KDIREG_SAVPC)(%esp)

362         /*
363          * Swap our saved EFLAGS and %eip.  Each is where the other
364          * should be.
365          */
366         movl    REG_OFF(KDIREG_EFLAGS)(%esp), %eax
367         xchgl   REG_OFF(KDIREG_EIP)(%esp), %eax
368         movl    %eax, REG_OFF(KDIREG_EFLAGS)(%esp)

370         /*
371          * Our stack now matches struct regs, and is irettable.  We don't need
372          * to do anything special for the hypervisor w.r.t. PS_IE since we
373          * iret twice anyway; the second iret back to the hypervisor
374          * will re-enable interrupts.
375          */
376         CLI(%eax)

378         /* Load sanitized segment selectors */
379         movw    kdi_ds, %ds
380         movw    kdi_ds, %es
381         movw    kdi_fs, %fs
382         movw    kdi_gs, %gs
383         movw    kdi_ds, %ss

385         GET_CPUSAVE_ADDR        /* %eax = cpusave, %ebx = CPU ID */

387         ADVANCE_CRUMB_POINTER(%eax, %ecx, %edx)

389         ADD_CRUMB(%eax, KRM_CPU_STATE, $KDI_CPU_STATE_SLAVE, %edx)

391         movl    REG_OFF(KDIREG_EIP)(%esp), %ecx
392         ADD_CRUMB(%eax, KRM_PC, %ecx, %edx)

394         pushl   %eax
395         jmp     kdi_save_common_state

397         SET_SIZE(kdi_slave_entry)

399 #endif  /* __lint */

401 /*
402  * The state of the world:
403  *
404  * The stack has a complete set of saved registers and segment
405  * selectors, arranged in 'struct regs' order (or vice-versa), up to
406  * and including EFLAGS.  It also has a pointer to our cpusave area.
407  *
```



```
408  * We need to save a pointer to these saved registers.  We also want
409  * to adjust the saved %esp - it should point just beyond the saved
410  * registers to the last frame of the thread we interrupted.  Finally,
411  * we want to clear out bits 16-31 of the saved selectors, as the
412  * selector pushls don't automatically clear them.
413  */
414 #if !defined(__lint)

416         ENTRY_NP(kdi_save_common_state)

418         popl    %eax                    /* the cpusave area */

420         movl    %esp, KRS_GREGS(%eax)   /* save ptr to current saved regs */

422         addl    $REG_OFF(KDIREG_EFLAGS - KDIREG_EAX), KDIREG_OFF(KDIREG_ESP)(%es

424         andl    $0xffff, KDIREG_OFF(KDIREG_SS)(%esp)
425         andl    $0xffff, KDIREG_OFF(KDIREG_GS)(%esp)
426         andl    $0xffff, KDIREG_OFF(KDIREG_FS)(%esp)
427         andl    $0xffff, KDIREG_OFF(KDIREG_ES)(%esp)
428         andl    $0xffff, KDIREG_OFF(KDIREG_DS)(%esp)

430         pushl   %eax
431         call    kdi_trap_pass
432         cmpl    $1, %eax
433         je      kdi_pass_to_kernel
434         popl    %eax

436         SAVE_IDTGDT

438 #if !defined(__xpv)
439         /* Save off %cr0, and clear write protect */
440         movl    %cr0, %ecx
441         movl    %ecx, KRS_CR0(%eax)
442         andl    $_BITNOT(CR0_WP), %ecx
443         movl    %ecx, %cr0
444 #endif
445         pushl   %edi
446         movl    %eax, %edi

448         /* Save the debug registers and disable any active watchpoints */
449         pushl   $7
450         call    kdi_dreg_get
451         addl    $4, %esp

453         movl    %eax, KRS_DRCTL(%edi)
454         andl    $_BITNOT(KDIREG_DRCTL_WPALLEN_MASK), %eax

456         pushl   %eax
457         pushl   $7
458         call    kdi_dreg_set
459         addl    $8, %esp

461         pushl   $6
462         call    kdi_dreg_get
463         addl    $4, %esp
464         movl    %eax, KRS_DRSTAT(%edi)

466         pushl   $0
467         call    kdi_dreg_get
468         addl    $4, %esp
469         movl    %eax, KRS_DROFF(0)(%edi)

471         pushl   $1
472         call    kdi_dreg_get
473         addl    $4, %esp
```

```
 474         movl    %eax, KRS_DROFF(1)(%edi)

 476         pushl   $2
 477         call    kdi_dreg_get
 478         addl    $4, %esp
 479         movl    %eax, KRS_DROFF(2)(%edi)

 481         pushl   $3
 482         call    kdi_dreg_get
 483         addl    $4, %esp
 484         movl    %eax, KRS_DROFF(3)(%edi)

 486         movl    %edi, %eax
 487         popl    %edi

 545         /*
 546          * Save any requested MSRs.
 547          */
 548         movl    KRS_MSR(%eax), %ecx
 549         cmpl    $0, %ecx
 550         je      no_msr

 552         pushl   %eax            /* rdmsr clobbers %eax */
 553         movl    %ecx, %ebx
 554 1:
 555         movl    MSR_NUM(%ebx), %ecx
 556         cmpl    $0, %ecx
 557         je      msr_done

 559         movl    MSR_TYPE(%ebx), %edx
 560         cmpl    $KDI_MSR_READ, %edx
 561         jne     msr_next

 563         rdmsr                   /* addr in %ecx, value into %edx:%eax */
 564         movl    %eax, MSR_VAL(%ebx)
 565         movl    %edx, _CONST(MSR_VAL + 4)(%ebx)

 567 msr_next:
 568         addl    $MSR_SIZE, %ebx
 569         jmp     1b

 571 msr_done:
 572         popl    %eax

 574 no_msr:
 489         clr     %ebp            /* stack traces should end here */

 491         pushl   %eax
 492         call    kdi_debugger_entry
 493         popl    %eax

 495         jmp     kdi_resume

 497         SET_SIZE(kdi_save_common_state)
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   11742 Wed Feb 28 17:34:24 2018
new/usr/src/uts/intel/kdi/kdi_idt.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
9207 kdi_idt: Cast GATESEG_GETOFFSET through uintptr_t
**********************************************************
```

```
/*
 * 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 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

/*
 * Management of KMDB's IDT, which is installed upon KMDB activation.
 *
 * Debugger activation has two flavors, which cover the cases where KMDB is
 * loaded at boot, and when it is loaded after boot.  In brief, in both cases,
 * the KDI needs to interpose upon several handlers in the IDT.  When
 * mod-loaded KMDB is deactivated, we undo the IDT interposition, restoring the
 * handlers to what they were before we started.
 *
 * We also take over the entirety of IDT (except the double-fault handler) on
 * the active CPU when we're in kmdb so we can handle things like page faults
 * sensibly.
 *
 * Boot-loaded KMDB
 *
 * When we're first activated, we're running on boot's IDT.  We need to be able
 * to function in this world, so we'll install our handlers into boot's IDT.
 * This is a little complicated: we're using the fake cpu_t set up by
 * boot_kdi_tmpinit(), so we can't access cpu_idt directly.  Instead,
 * kdi_idt_write() notices that cpu_idt is NULL, and works around this problem.
 *
 * Later, when we're about to switch to the kernel's IDT, it'll call us via
 * kdi_idt_sync(), allowing us to add our handlers to the new IDT.  While
 * boot-loaded KMDB can't be unloaded, we still need to save the descriptors we
 * replace so we can pass traps back to the kernel as necessary.
 *
 * The last phase of boot-loaded KMDB activation occurs at non-boot CPU
 * startup.  We will be called on each non-boot CPU, thus allowing us to set up
 * any watchpoints that may have been configured on the boot CPU and interpose
 * on the given CPU's IDT.  We don't save the interposed descriptors in this
```



```
 * case -- see kdi_cpu_init() for details.
 *
 * Mod-loaded KMDB
 *
 * This style of activation is much simpler, as the CPUs are already running,
 * and are using their own copy of the kernel's IDT.  We simply interpose upon
 * each CPU's IDT.  We save the handlers we replace, both for deactivation and
 * for passing traps back to the kernel.  Note that for the hypervisors'
 * benefit, we need to xcall to the other CPUs to do this, since we need to
 * actively set the trap entries in its virtual IDT from that vcpu's context
 * rather than just modifying the IDT table from the CPU running kdi_activate().
 */

#include <sys/types.h>
#include <sys/segments.h>
#include <sys/trap.h>
#include <sys/cpuvar.h>
#include <sys/reboot.h>
#include <sys/sunddi.h>
#include <sys/archsystm.h>
#include <sys/kdi_impl.h>
#include <sys/x_call.h>
#include <ia32/sys/psw.h>

#define KDI_GATE_NVECS  3

#define KDI_IDT_NOSAVE  0
#define KDI_IDT_SAVE    1

#define KDI_IDT_DTYPE_KERNEL    0
#define KDI_IDT_DTYPE_BOOT      1

kdi_cpusave_t *kdi_cpusave;
int kdi_ncpusave;

static kdi_main_t kdi_kmdb_main;

kdi_drreg_t kdi_drreg;

#ifndef __amd64
/* Used to track the current set of valid kernel selectors. */
uint32_t        kdi_cs;
uint32_t        kdi_ds;
uint32_t        kdi_fs;
uint32_t        kdi_gs;
#endif

uint_t          kdi_msr_wrexit_msr;
uint64_t        *kdi_msr_wrexit_valp;

uintptr_t       kdi_kernel_handler;

int             kdi_trap_switch;

#define KDI_MEMRANGES_MAX       2

kdi_memrange_t  kdi_memranges[KDI_MEMRANGES_MAX];
int             kdi_nmemranges;

typedef void idt_hdlr_f(void);

extern idt_hdlr_f kdi_trap0, kdi_trap1, kdi_int2, kdi_trap3, kdi_trap4;
extern idt_hdlr_f kdi_trap5, kdi_trap6, kdi_trap7, kdi_trap9;
extern idt_hdlr_f kdi_traperr10, kdi_traperr11, kdi_traperr12;
extern idt_hdlr_f kdi_traperr13, kdi_traperr14, kdi_trap16, kdi_trap17;
extern idt_hdlr_f kdi_trap18, kdi_trap19, kdi_trap20, kdi_ivct32;
```

```
121 extern idt_hdlr_f kdi_invaltrap;
122 extern size_t kdi_ivct_size;
124 extern char kdi_slave_entry_patch;

124 typedef struct kdi_gate_spec {
125         uint_t kgs_vec;
126         uint_t kgs_dpl;
127 } kdi_gate_spec_t;
```

_unchanged_portion_omitted_

```
197 /*
198  * Patch caller-provided code into the debugger's IDT handlers.  This code is
199  * used to save MSRs that must be saved before the first branch.  All handlers
200  * are essentially the same, and end with a branch to kdi_cmnint.  To save the
201  * MSR, we need to patch in before the branch.  The handlers have the following
202  * structure: KDI_MSR_PATCHOFF bytes of code, KDI_MSR_PATCHSZ bytes of
203  * patchable space, followed by more code.
204  */
205 void
206 kdi_idt_patch(caddr_t code, size_t sz)
207 {
208         int i;

210         ASSERT(sz <= KDI_MSR_PATCHSZ);

212         for (i = 0; i < sizeof (kdi_idt) / sizeof (struct gate_desc); i++) {
213                 gate_desc_t *gd;
214                 uchar_t *patch;

216                 if (i == T_DBLFLT)
217                         continue;       /* uses kernel's handler */

219                 gd = &kdi_idt[i];
220                 patch = (uchar_t *)GATESEG_GETOFFSET(gd) + KDI_MSR_PATCHOFF;

222                 /*
223                  * We can't ASSERT that there's a nop here, because this may be
224                  * a debugger restart.  In that case, we're copying the new
225                  * patch point over the old one.
226                  */
227                 /* FIXME: dtrace fbt ... */
228                 bcopy(code, patch, sz);

230                 /* Fill the rest with nops to be sure */
231                 while (sz < KDI_MSR_PATCHSZ)
232                         patch[sz++] = 0x90; /* nop */
233         }
234 }

195 static void
196 kdi_idt_gates_install(selector_t sel, int saveold)
197 {
198         gate_desc_t gates[KDI_GATE_NVECS];
199         int i;

201         bzero(gates, sizeof (*gates));

203         for (i = 0; i < KDI_GATE_NVECS; i++) {
204                 const kdi_gate_spec_t *gs = &kdi_gate_specs[i];
205                 uintptr_t func = GATESEG_GETOFFSET(&kdi_idt[gs->kgs_vec]);
206                 set_gatesegd(&gates[i], (void (*)())func, sel, SDT_SYSIGT,
207                     gs->kgs_dpl, gs->kgs_vec);
208         }

210         for (i = 0; i < KDI_GATE_NVECS; i++) {
211                 uint_t vec = kdi_gate_specs[i].kgs_vec;
```



```
213                 if (saveold)
214                         kdi_kgates[i] = CPU->cpu_m.mcpu_idt[vec];

216                 kdi_idt_write(&gates[i], vec);
217         }
218 }
```

_unchanged_portion_omitted_

```
281 /*
282  * On some processors, we'll need to clear a certain MSR before proceeding into
283  * the debugger.  Complicating matters, this MSR must be cleared before we take
284  * any branches.  We have patch points in every trap handler, which will cover
285  * all entry paths for master CPUs.  We also have a patch point in the slave
286  * entry code.
287  */
288 static void
289 kdi_msr_add_clrentry(uint_t msr)
290 {
291 #ifdef __amd64
292         uchar_t code[] = {
293                 0x51, 0x50, 0x52,               /* pushq %rcx, %rax, %rdx */
294                 0xb9, 0x00, 0x00, 0x00, 0x00,   /* movl $MSRNUM, %ecx */
295                 0x31, 0xc0,                     /* clr %eax */
296                 0x31, 0xd2,                     /* clr %edx */
297                 0x0f, 0x30,                     /* wrmsr */
298                 0x5a, 0x58, 0x59                /* popq %rdx, %rax, %rcx */
299         };
300         uchar_t *patch = &code[4];
301 #else
302         uchar_t code[] = {
303                 0x60,                           /* pushal */
304                 0xb9, 0x00, 0x00, 0x00, 0x00,   /* movl $MSRNUM, %ecx */
305                 0x31, 0xc0,                     /* clr %eax */
306                 0x31, 0xd2,                     /* clr %edx */
307                 0x0f, 0x30,                     /* wrmsr */
308                 0x61                            /* popal */
309         };
310         uchar_t *patch = &code[2];
311 #endif

313         bcopy(&msr, patch, sizeof (uint32_t));

315         kdi_idt_patch((caddr_t)code, sizeof (code));

317         bcopy(code, &kdi_slave_entry_patch, sizeof (code));
318 }

320 static void
321 kdi_msr_add_wrexit(uint_t msr, uint64_t *valp)
322 {
323         kdi_msr_wrexit_msr = msr;
324         kdi_msr_wrexit_valp = valp;
325 }

240 void
328 kdi_set_debug_msrs(kdi_msr_t *msrs)
329 {
330         int nmsrs, i;

332         ASSERT(kdi_cpusave[0].krs_msr == NULL);

334         /* Look in CPU0's MSRs for any special MSRs. */
335         for (nmsrs = 0; msrs[nmsrs].msr_num != 0; nmsrs++) {
336                 switch (msrs[nmsrs].msr_type) {
337                 case KDI_MSR_CLEARENTRY:
```

```
                        kdi_msr_add_clrentry(msrs[nmsrs].msr_num);
                        break;

                case KDI_MSR_WRITEDELAY:
                        kdi_msr_add_wrexit(msrs[nmsrs].msr_num,
                            msrs[nmsrs].kdi_msr_valp);
                        break;
                }
        }

        nmsrs++;

        for (i = 0; i < kdi_ncpusave; i++)
                kdi_cpusave[i].krs_msr = &msrs[nmsrs * i];
}

void
kdi_update_drreg(kdi_drreg_t *drreg)
{
        kdi_drreg = *drreg;
}
```

*unchanged_portion_omitted_*

```
/*
 * Activation for CPUs other than the boot CPU, called from that CPU's
 * mp_startup().  We saved the kernel's descriptors when we initialized the
 * boot CPU, so we don't want to do it again.  Saving the handlers from this
 * CPU's IDT would actually be dangerous with the CPU initialization method in
 * use at the time of this writing.  With that method, the startup code creates
 * the IDTs for slave CPUs by copying the one used by the boot CPU, which has
 * already been interposed upon by KMDB.  Were we to interpose again, we'd
 * replace the kernel's descriptors with our own in the save area.  By not
 * saving, but still overwriting, we'll work in the current world, and in any
 * future world where the IDT is generated from scratch.
 */
void
kdi_cpu_init(void)
{
        kdi_idt_gates_install(KCS_SEL, KDI_IDT_NOSAVE);
        /* Load the debug registers. */
        /* Load the debug registers and MSRs */
        kdi_cpu_debug_init(&kdi_cpusave[CPU->cpu_id]);
}
```

*unchanged_portion_omitted_*

```
void
kdi_activate(kdi_main_t main, kdi_cpusave_t *cpusave, uint_t ncpusave)
{
        int i;
        cpuset_t cpuset;

        CPUSET_ALL(cpuset);

        kdi_cpusave = cpusave;
        kdi_ncpusave = ncpusave;

        kdi_kmdb_main = main;

        for (i = 0; i < kdi_ncpusave; i++) {
                kdi_cpusave[i].krs_cpu_id = i;

                kdi_cpusave[i].krs_curcrumb =
                    &kdi_cpusave[i].krs_crumbs[KDI_NCRUMBS - 1];
                kdi_cpusave[i].krs_curcrumbidx = KDI_NCRUMBS - 1;
        }
```



```
        if (boothowto & RB_KMDB)
                kdi_idt_init(KMDBCODE_SEL);
        else
                kdi_idt_init(KCS_SEL);

        /* The initial selector set.  Updated by the debugger-entry code */
#ifndef __amd64
        kdi_cs = B32CODE_SEL;
        kdi_ds = kdi_fs = kdi_gs = B32DATA_SEL;
#endif

        kdi_memranges[0].mr_base = kdi_segdebugbase;
        kdi_memranges[0].mr_lim = kdi_segdebugbase + kdi_segdebugsize - 1;
        kdi_nmemranges = 1;

        kdi_drreg.dr_ctl = KDIREG_DRCTL_RESERVED;
        kdi_drreg.dr_stat = KDIREG_DRSTAT_RESERVED;

        kdi_msr_wrexit_msr = 0;
        kdi_msr_wrexit_valp = NULL;

        if (boothowto & RB_KMDB) {
                kdi_idt_gates_install(KMDBCODE_SEL, KDI_IDT_NOSAVE);
        } else {
                xc_call(0, 0, 0, CPUSET2BV(cpuset),
                    (xc_func_t)kdi_cpu_activate);
        }
}
```

*unchanged_portion_omitted_*

```
**********************************************************
    6076 Wed Feb 28 17:34:24 2018
new/usr/src/uts/intel/kdi/kdi_idthdl.s
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Companion to kdi_idt.c - the implementation of the trap and interrupt
 * handlers.  For the most part, these handlers do the same thing - they
 * push a trap number onto the stack, followed by a jump to kdi_cmnint.
 * Each trap and interrupt has its own handler because each one pushes a
 * different number.
 */

#include <sys/asm_linkage.h>
#include <sys/kdi_regs.h>

/* Nothing in this file is of interest to lint. */
#if !defined(__lint)

/*
 * The default ASM_ENTRY_ALIGN (16) wastes far too much space.  Pay no
 * attention to the fleet of nop's we're adding to each handler.
 */
#undef  ASM_ENTRY_ALIGN
#define ASM_ENTRY_ALIGN 8

/*
 * We need the .align in ENTRY_NP (defined to be ASM_ENTRY_ALIGN) to match our
 * manual .align (KDI_MSR_PATCHOFF) in order to ensure that the space reserved
 * at the beginning of the handler for code is exactly KDI_MSR_PATCHOFF bytes
 * long.  Note that the #error below isn't supported by the preprocessor invoked
 * by as(1), and won't stop the build, but it'll emit a noticeable error message
 * which won't escape the filters.
 */
```



```
#if ASM_ENTRY_ALIGN != KDI_MSR_PATCHOFF
#error "ASM_ENTRY_ALIGN != KDI_MSR_PATCHOFF"
this won't assemble
#endif

/*
 * kdi_idt_patch will, on certain processors, replace the patch points below
 * with MSR-clearing code.  kdi_id_patch has intimate knowledge of the size of
 * the nop hole, as well as the structure of the handlers.  Do not change
 * anything here without also changing kdi_idt_patch.
 */

/*
 * Generic trap and interrupt handlers.
 */

#if defined(__xpv) && defined(__amd64)

/*
 * The hypervisor places r11 and rcx on the stack.
 */

#define TRAP_NOERR(trapno) \
        popq    %rcx;           \
        popq    %r11;           \
        pushq   $trapno

#define TRAP_ERR(trapno)        \
        popq    %rcx;           \
        popq    %r11;           \
        pushq   $0;             \
        pushq   $trapno

#else

#define TRAP_NOERR(trapno)      \
        push    $trapno

#define TRAP_ERR(trapno)        \
        push    $0;             \
        push    $trapno

#endif  /* __xpv && __amd64 */


#define MKIVCT(n) \
        ENTRY_NP(kdi_ivct/**/n/**/);    \
        TRAP_ERR(n);                    \
        .align  KDI_MSR_PATCHOFF;       \
        KDI_MSR_PATCH;                  \
        jmp     kdi_cmnint;             \
        SET_SIZE(kdi_ivct/**/n/**/)

#define MKTRAPHDLR(n) \
        ENTRY_NP(kdi_trap/**/n);        \
        TRAP_ERR(n);                    \
        .align  KDI_MSR_PATCHOFF;       \
        KDI_MSR_PATCH;                  \
        jmp     kdi_cmnint;             \
        SET_SIZE(kdi_trap/**/n/**/)

#define MKTRAPERRHDLR(n) \
        ENTRY_NP(kdi_traperr/**/n);     \
        TRAP_NOERR(n);                  \
        .align  KDI_MSR_PATCHOFF;       \
        KDI_MSR_PATCH;                  \
```

```
  97         jmp     kdi_cmnint;             \
  98         SET_SIZE(kdi_traperr/**/n)

 100 #define MKNMIHDLR \
 101         ENTRY_NP(kdi_int2);             \
 102         TRAP_NOERR(2);                  \
 129         .align  KDI_MSR_PATCHOFF;       \
 130         KDI_MSR_PATCH;                  \
 103         jmp     kdi_nmiint;             \
 104         SET_SIZE(kdi_int2)

 106 #define MKINVALHDLR \
 107         ENTRY_NP(kdi_invaltrap);        \
 108         TRAP_NOERR(255);                \
 137         .align  KDI_MSR_PATCHOFF;       \
 138         KDI_MSR_PATCH;                  \
 109         jmp     kdi_cmnint;             \
 110         SET_SIZE(kdi_invaltrap)
```

*unchanged_portion_omitted_*

```
**********************************************************
    1844 Wed Feb 28 17:34:24 2018
new/usr/src/uts/intel/kdi/kdi_offsets.in
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
\
\ Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
\ Use is subject to license terms.
\
\ Copyright 2018 Joyent, Inc.
\
\ 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
\
\ ident "%Z%%M% %I%     %E% SMI"
\
\ CPU-save structure offsets for use in assembly code.
\
\ Keep in sync with kdi_state.h
\

#include <sys/cpuvar.h>
#include <sys/kdi_impl.h>

kdi_memrange_t  MR_SIZE
                mr_base
                mr_lim

kdi_crumb_t     KRM_SIZE
                krm_cpu_state
                krm_pc
                krm_sp
                krm_trapno
                krm_flag

kdi_drreg_t
                dr_ctl
                dr_stat
                dr_addr

kdi_msr_t       MSR_SIZE
                msr_num
                msr_type
                _u._msr_valp                    MSR_VALP
                _u._msr_val                     MSR_VAL

kdi_cpusave_t   KRS_SIZE
```



```
                krs_gregs
                krs_dr
                krs_dr.dr_ctl                   KRS_DRCTL
                krs_dr.dr_stat                  KRS_DRSTAT
                krs_gdt
                krs_idt
                krs_cr0
                krs_msr
                krs_cpu_state
                krs_curcrumbidx
                krs_curcrumb
                krs_crumbs

cpu
                cpu_id

greg_t          KREG_SIZE

#if defined(__amd64)
\#define        REG_SHIFT               3
#else
\#define        REG_SHIFT               2
#endif

\#define        DRADDR_IDX(num)         _CONST(_MUL(num, DR_ADDR_INCR))
\#define        DRADDR_OFF(num)         _CONST(DRADDR_IDX(num) + DR_ADDR)
\#define        KRS_DROFF(num)          _CONST(DRADDR_OFF(num) + KRS_DR)
\#define        REG_OFF(reg)            _CONST(_CONST(reg) << REG_SHIFT)
\#define        KDIREG_OFF(reg)         _CONST(_MUL(KREG_SIZE, reg) + KRS_GREGS)
```

```
**********************************************************
    4230 Wed Feb 28 17:34:24 2018
new/usr/src/uts/intel/os/arch_kdi.c
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

/*
 * Kernel/Debugger Interface (KDI) routines.  Called during debugger under
 * various system states (boot, while running, while the debugger has control).
 * Functions intended for use while the debugger has control may not grab any
 * locks or perform any functions that assume the availability of other system
 * services.
 */

#include <sys/systm.h>
#include <sys/x86_archext.h>
#include <sys/kdi_impl.h>
#include <sys/smp_impldefs.h>
#include <sys/psm_types.h>
#include <sys/segments.h>
#include <sys/archsystm.h>
#include <sys/controlregs.h>
#include <sys/trap.h>
#include <sys/kobj.h>
#include <sys/kobj_impl.h>
#include <sys/clock_impl.h>

static void
kdi_system_claim(void)
{
        lbolt_debug_entry();

        psm_notifyf(PSM_DEBUG_ENTER);
}
```

*unchanged_portion_omitted_*

```
/*
```



```
 * On Intel, most of these are shared between i86*, so this is really an
 * arch_kdi_init().
 */
void
mach_kdi_init(kdi_t *kdi)
{
        kdi->kdi_plat_call = kdi_plat_call;
        kdi->kdi_kmdb_enter = kmdb_enter;
        kdi->mkdi_activate = kdi_activate;
        kdi->mkdi_deactivate = kdi_deactivate;
        kdi->mkdi_idt_switch = kdi_idt_switch;
        kdi->mkdi_update_drreg = kdi_update_drreg;
        kdi->mkdi_set_debug_msrs = kdi_set_debug_msrs;
        kdi->mkdi_get_userlimit = kdi_get_userlimit;
        kdi->mkdi_get_cpuinfo = kdi_get_cpuinfo;
        kdi->mkdi_stop_slaves = kdi_stop_slaves;
        kdi->mkdi_start_slaves = kdi_start_slaves;
        kdi->mkdi_slave_wait = kdi_slave_wait;
        kdi->mkdi_memrange_add = kdi_memrange_add;
        kdi->mkdi_reboot = kdi_reboot;
}
```

*unchanged_portion_omitted_*

```
**********************************************************
    7788 Wed Feb 28 17:34:25 2018
new/usr/src/uts/intel/sys/controlregs.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright (c) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
  23  * Copyright 2015, Joyent, Inc.
  24  *
  25  * Copyright 2018 Joyent, Inc.
  26  */

  28 #ifndef _SYS_CONTROLREGS_H
  29 #define _SYS_CONTROLREGS_H

  31 #ifndef _ASM
  32 #include <sys/types.h>
  33 #endif

  35 #ifdef __cplusplus
  36 extern "C" {
  37 #endif

  39 /*
  40  * This file describes the x86 architecture control registers which
  41  * are part of the privileged architecture.
  42  *
  43  * Many of these definitions are shared between IA-32-style and
  44  * AMD64-style processors.
  45  */

  47 /* CR0 Register */

  49 #define CR0_PG  0x80000000              /* paging enabled       */
  50 #define CR0_CD  0x40000000              /* cache disable        */
  51 #define CR0_NW  0x20000000              /* not writethrough     */
  52 #define CR0_AM  0x00040000              /* alignment mask       */
  53 #define CR0_WP  0x00010000              /* write protect        */
  54 #define CR0_NE  0x00000020              /* numeric error        */
  55 #define CR0_ET  0x00000010              /* extension type       */
  56 #define CR0_TS  0x00000008              /* task switch          */
  57 #define CR0_EM  0x00000004              /* emulation            */
  58 #define CR0_MP  0x00000002              /* monitor coprocessor  */
```



```
  59 #define CR0_PE  0x00000001              /* protection enabled   */

  61 /* XX64 eliminate these compatibility defines */

  63 #define CR0_CE  CR0_CD
  64 #define CR0_WT  CR0_NW

  66 #define FMT_CR0 \
  67         "\20\40pg\37cd\36nw\35am\21wp\6ne\5et\4ts\3em\2mp\1pe"

  69 /*
  70  * Set the FPU-related control bits to explain to the processor that
  71  * we're managing FPU state:
  72  * - set monitor coprocessor (allow TS bit to control FPU)
  73  * - set numeric exception (disable IGNNE# mechanism)
  74  * - set task switch (#nm on first fp instruction)
  75  * - clear emulate math bit (cause we're not emulating!)
  76  */
  77 #define CR0_ENABLE_FPU_FLAGS(cr)        \
  78         (((cr) | CR0_MP | CR0_NE | CR0_TS) & (uint32_t)~CR0_EM)

  80 /*
  81  * Set the FPU-related control bits to explain to the processor that
  82  * we're -not- managing FPU state:
  83  * - set emulate (all fp instructions cause #nm)
  84  * - clear monitor coprocessor (so fwait/wait doesn't #nm)
  85  */
  86 #define CR0_DISABLE_FPU_FLAGS(cr)       \
  87         (((cr) | CR0_EM) & (uint32_t)~CR0_MP)

  89 /* CR3 Register */

  91 #define CR3_PCD 0x00000010              /* cache disable                */
  92 #define CR3_PWT 0x00000008              /* write through                */

  94 #define FMT_CR3 "\20\5pcd\4pwt"

  96 /* CR4 Register */

  98 #define CR4_VME         0x0001          /* virtual-8086 mode extensions */
  99 #define CR4_PVI         0x0002          /* protected-mode virtual interrupts */
 100 #define CR4_TSD         0x0004          /* time stamp disable           */
 101 #define CR4_DE          0x0008          /* debugging extensions         */
 102 #define CR4_PSE         0x0010          /* page size extensions         */
 103 #define CR4_PAE         0x0020          /* physical address extension   */
 104 #define CR4_MCE         0x0040          /* machine check enable         */
 105 #define CR4_PGE         0x0080          /* page global enable           */
 106 #define CR4_PCE         0x0100          /* perf-monitoring counter enable */
 107 #define CR4_OSFXSR      0x0200          /* OS fxsave/fxrstor support    */
 108 #define CR4_OSXMMEXCPT  0x0400          /* OS unmasked exception support */
 109                                         /* 0x0800 reserved */
 110                                         /* 0x1000 reserved */
 111 #define CR4_VMXE        0x2000
 112 #define CR4_SMXE        0x4000
 113 #define CR4_PCIDE       0x20000         /* PCID enable */
 114 #define CR4_OSXSAVE     0x40000         /* OS xsave/xrestore support    */
 115 #define CR4_SMEP        0x100000        /* NX for user pages in kernel */
 116 #define CR4_SMAP        0x200000        /* kernel can't access user pages */

 118 #define FMT_CR4                                             \
 119         "\20\26smap\25smep\23osxsav\22pcide"                \
 116         "\20\26smap\25smep\23osxsav"                        \
 120         "\17smxe\16vmxe\13xmme\12fxsr\11pce\10pge"      \
 121         "\7mce\6pae\5pse\4de\3tsd\2pvi\1vme"

 123 /*
```

```
 124  * Enable the SSE-related control bits to explain to the processor that
 125  * we're managing XMM state and exceptions
 126  */
 127 #define CR4_ENABLE_SSE_FLAGS(cr)        \
 128         ((cr) | CR4_OSFXSR | CR4_OSXMMEXCPT)

 130 /*
 131  * Disable the SSE-related control bits to explain to the processor
 132  * that we're NOT managing XMM state
 133  */
 134 #define CR4_DISABLE_SSE_FLAGS(cr)       \
 135         ((cr) & ~(uint32_t)(CR4_OSFXSR | CR4_OSXMMEXCPT))

 137 /* Intel's SYSENTER configuration registers */

 139 #define MSR_INTC_SEP_CS 0x174           /* kernel code selector MSR */
 140 #define MSR_INTC_SEP_ESP 0x175          /* kernel esp MSR */
 141 #define MSR_INTC_SEP_EIP 0x176          /* kernel eip MSR */

 143 /* Intel's microcode registers */
 144 #define MSR_INTC_UCODE_WRITE            0x79    /* microcode write */
 145 #define MSR_INTC_UCODE_REV              0x8b    /* microcode revision */
 146 #define INTC_UCODE_REV_SHIFT            32      /* Bits 63:32 */

 148 /* Intel's platform identification */
 149 #define MSR_INTC_PLATFORM_ID            0x17
 150 #define INTC_PLATFORM_ID_SHIFT          50      /* Bit 52:50 */
 151 #define INTC_PLATFORM_ID_MASK           0x7

 153 /* AMD's EFER register */

 155 #define MSR_AMD_EFER    0xc0000080      /* extended feature enable MSR */

 157 #define AMD_EFER_FFXSR  0x4000          /* fast fxsave/fxrstor          */
 158 #define AMD_EFER_SVME   0x1000          /* svm enable                   */
 159 #define AMD_EFER_NXE    0x0800          /* no-execute enable            */
 160 #define AMD_EFER_LMA    0x0400          /* long mode active (read-only) */
 161 #define AMD_EFER_LME    0x0100          /* long mode enable             */
 162 #define AMD_EFER_SCE    0x0001          /* system call extensions       */

 164 #define FMT_AMD_EFER \
 165         "\20\17ffxsr\15svme\14nxe\13lma\11lme\1sce"

 167 /* AMD's SYSCFG register */

 169 #define MSR_AMD_SYSCFG  0xc0000010      /* system configuration MSR */

 171 #define AMD_SYSCFG_TOM2 0x200000        /* MtrrTom2En */
 172 #define AMD_SYSCFG_MVDM 0x100000        /* MtrrVarDramEn */
 173 #define AMD_SYSCFG_MFDM 0x080000        /* MtrrFixDramModEn */
 174 #define AMD_SYSCFG_MFDE 0x040000        /* MtrrFixDramEn */

 176 #define FMT_AMD_SYSCFG \
 177         "\20\26tom2\25mvdm\24mfdm\23mfde"

 179 /* AMD's syscall/sysret MSRs */

 181 #define MSR_AMD_STAR    0xc0000081      /* %cs:%ss:%cs:%ss:%eip for syscall */
 182 #define MSR_AMD_LSTAR   0xc0000082      /* target %rip of 64-bit syscall */
 183 #define MSR_AMD_CSTAR   0xc0000083      /* target %rip of 32-bit syscall */
 184 #define MSR_AMD_SFMASK  0xc0000084      /* syscall flag mask */

 186 /* AMD's FS.base and GS.base MSRs */

 188 #define MSR_AMD_FSBASE  0xc0000100      /* 64-bit base address for %fs */
 189 #define MSR_AMD_GSBASE  0xc0000101      /* 64-bit base address for %gs */
```



```
 190 #define MSR_AMD_KGSBASE 0xc0000102      /* swapgs swaps this with gsbase */
 191 #define MSR_AMD_TSCAUX  0xc0000103      /* %ecx value on rdtscp insn */

 193 /* AMD's configuration MSRs, weakly documented in the revision guide */

 195 #define MSR_AMD_DC_CFG  0xc0011022

 197 #define AMD_DC_CFG_DIS_CNV_WC_SSO       (UINT64_C(1) << 3)
 198 #define AMD_DC_CFG_DIS_SMC_CHK_BUF      (UINT64_C(1) << 10)

 200 /* AMD's HWCR MSR */

 202 #define MSR_AMD_HWCR    0xc0010015

 204 #define AMD_HWCR_TLBCACHEDIS            (UINT64_C(1) << 3)
 205 #define AMD_HWCR_FFDIS                  0x00040 /* disable TLB Flush Filter */
 206 #define AMD_HWCR_MCI_STATUS_WREN        0x40000 /* enable write of MCi_STATUS */

 208 /* AMD's NorthBridge Config MSR, SHOULD ONLY BE WRITTEN TO BY BIOS */

 210 #define MSR_AMD_NB_CFG  0xc001001f

 212 #define AMD_NB_CFG_SRQ_HEARTBEAT        (UINT64_C(1) << 20)
 213 #define AMD_NB_CFG_SRQ_SPR              (UINT64_C(1) << 32)

 215 #define MSR_AMD_BU_CFG  0xc0011023

 217 #define AMD_BU_CFG_E298                 (UINT64_C(1) << 1)

 219 #define MSR_AMD_DE_CFG  0xc0011029

 221 #define AMD_DE_CFG_E721                 (UINT64_C(1))

 223 /* AMD's osvw MSRs */
 224 #define MSR_AMD_OSVW_ID_LEN             0xc0010140
 225 #define MSR_AMD_OSVW_STATUS             0xc0010141


 228 #define OSVW_ID_LEN_MASK                0xffffULL
 229 #define OSVW_ID_CNT_PER_MSR             64

 231 /*
 232  * Enable PCI Extended Configuration Space (ECS) on Greyhound
 233  */
 234 #define AMD_GH_NB_CFG_EN_ECS            (UINT64_C(1) << 46)

 236 /* AMD microcode patch loader */
 237 #define MSR_AMD_PATCHLEVEL      0x8b
 238 #define MSR_AMD_PATCHLOADER     0xc0010020

 240 #ifdef __cplusplus
 241 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    3809 Wed Feb 28 17:34:25 2018
new/usr/src/uts/intel/sys/kdi_machimpl.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  *
  25  * Copyright 2018 Joyent, Inc.
  26  */

  28 #ifndef _SYS_KDI_MACHIMPL_H
  29 #define _SYS_KDI_MACHIMPL_H

  29 #pragma ident   "%Z%%M% %I%     %E% SMI"

  31 /*
  32  * The Kernel/Debugger interface.  The operations provided by the kdi_t,
  33  * defined below, comprise the Debugger -> Kernel portion of the interface,
  34  * and are to be used only when the system has been stopped.
  35  */

  37 #include <sys/modctl.h>
  38 #include <sys/types.h>
  39 #include <sys/cpuvar.h>
  40 #include <sys/kdi_regs.h>

  42 #ifdef __cplusplus
  43 extern "C" {
  44 #endif

  46 typedef void (*kdi_main_t)(kdi_cpusave_t *);

  48 typedef struct kdi_memrange {
  49         caddr_t mr_base;
  50         caddr_t mr_lim;
  51 } kdi_memrange_t;

  53 #define KDI_MEMRANGES_MAX       2

  55 typedef struct kdi_mach {
  56         void (*mkdi_activate)(kdi_main_t, kdi_cpusave_t *, uint_t);
```



```
  57         void (*mkdi_deactivate)(void);

  59         void (*mkdi_idt_switch)(kdi_cpusave_t *);

  61         void (*mkdi_update_drreg)(kdi_drreg_t *);
  62         void (*mkdi_set_debug_msrs)(kdi_msr_t *);

  63         uintptr_t (*mkdi_get_userlimit)(void);

  65         int (*mkdi_get_cpuinfo)(uint_t *, uint_t *, uint_t *);

  67         void (*mkdi_stop_slaves)(int, int);

  69         void (*mkdi_start_slaves)(void);

  71         void (*mkdi_slave_wait)(void);

  73         void (*mkdi_memrange_add)(caddr_t, size_t);

  75         void (*mkdi_reboot)(void);
  76 } kdi_mach_t;

  78 #define mkdi_activate                   kdi_mach.mkdi_activate
  79 #define mkdi_deactivate                 kdi_mach.mkdi_deactivate
  80 #define mkdi_idt_switch                 kdi_mach.mkdi_idt_switch
  81 #define mkdi_update_drreg               kdi_mach.mkdi_update_drreg
  83 #define mkdi_set_debug_msrs             kdi_mach.mkdi_set_debug_msrs
  82 #define mkdi_get_userlimit              kdi_mach.mkdi_get_userlimit
  83 #define mkdi_get_cpuinfo                kdi_mach.mkdi_get_cpuinfo
  84 #define mkdi_stop_slaves                kdi_mach.mkdi_stop_slaves
  85 #define mkdi_start_slaves               kdi_mach.mkdi_start_slaves
  86 #define mkdi_slave_wait                 kdi_mach.mkdi_slave_wait
  87 #define mkdi_memrange_add               kdi_mach.mkdi_memrange_add
  88 #define mkdi_reboot                     kdi_mach.mkdi_reboot

  90 extern void hat_kdi_init(void);

  92 extern ulong_t kdi_getdr0(void), kdi_getdr1(void), kdi_getdr2(void);
  93 extern ulong_t kdi_getdr3(void), kdi_getdr6(void), kdi_getdr7(void);
  94 extern void kdi_setdr0(ulong_t), kdi_setdr1(ulong_t), kdi_setdr2(ulong_t);
  95 extern void kdi_setdr3(ulong_t), kdi_setdr6(ulong_t), kdi_setdr7(ulong_t);
  96 extern ulong_t kdi_dreg_get(int);
  97 extern void kdi_dreg_set(int, ulong_t);
  98 extern void kdi_update_drreg(kdi_drreg_t *);
 101 extern void kdi_set_debug_msrs(kdi_msr_t *);
  99 extern void kdi_cpu_debug_init(kdi_cpusave_t *);

 101 extern void kdi_cpu_init(void);
 102 extern void kdi_xc_others(int, void (*)(void));
 103 extern void kdi_start_slaves(void);
 104 extern void kdi_slave_wait(void);

 106 extern void kdi_idtr_set(gate_desc_t *, size_t);
 107 extern void kdi_idt_write(struct gate_desc *, uint_t);
 108 extern void kdi_idt_sync(void);
 109 extern void kdi_idt_switch(kdi_cpusave_t *);
 110 #ifdef __xpv
 111 extern void kdi_idtr_write(desctbr_t *);
 112 #else
 113 #define kdi_idtr_write(idtr) wr_idtr(idtr)
 114 #endif

 116 extern void kdi_activate(kdi_main_t, kdi_cpusave_t *, uint_t);
 117 extern void kdi_deactivate(void);
 118 extern void kdi_stop_slaves(int, int);
 119 extern void kdi_memrange_add(caddr_t, size_t);
```

```
 120 extern void kdi_reboot(void);

 122 #ifdef __cplusplus
 123 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    3021 Wed Feb 28 17:34:25 2018
new/usr/src/uts/intel/sys/kdi_regs.h
9210 remove KMDB branch debugging support
9211 ::crregs could do with cr2/cr3 support
9209 ::ttrace should be able to filter by thread
Reviewed by: Patrick Mooney <patrick.mooney@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2018 Joyent, Inc.
 */

#ifndef _SYS_KDI_REGS_H
#define _SYS_KDI_REGS_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#ifndef _ASM
#include <sys/types.h>
#include <sys/segments.h>
#include <sys/regset.h>
#include <sys/privregs.h>
#endif

#if defined(__amd64)
#include <amd64/sys/kdi_regs.h>
#elif defined(__i386)
#include <ia32/sys/kdi_regs.h>
#endif

#ifdef __cplusplus
extern "C" {
#endif

#define KDI_NCRUMBS     5

#define KDI_CPU_STATE_NONE              0
#define KDI_CPU_STATE_MASTER            1
#define KDI_CPU_STATE_SLAVE             2

#define KDIREG_DRCTL_WPALLEN_MASK       0x000000ff
#define KDIREG_DRSTAT_RESERVED          0xffff0ff0
#define KDIREG_DRCTL_RESERVED           0x00000700
```



```
#define KDI_MSR_READ            0x1     /* read during entry (unlimited) */
#define KDI_MSR_WRITE           0x2     /* write during exit (unlimited) */
#define KDI_MSR_WRITEDELAY      0x4     /* write after last branch (<= 1) */
#define KDI_MSR_CLEARENTRY      0x3     /* clear before 1st branch (<= 1) */

#ifndef _ASM

/*
 * We maintain a ring buffer of bread crumbs for debugging purposes.  The
 * current buffer pointer is advanced along the ring with each intercepted
 * trap (debugger entry, invalid memory access, fault during step, etc).
 */
typedef struct kdi_crumb {
        greg_t krm_cpu_state;   /* This CPU's state at last entry */
        greg_t krm_pc;          /* Instruction pointer at trap */
        greg_t krm_sp;          /* Stack pointer at trap */
        greg_t krm_trapno;      /* The last trap number */
        greg_t krm_flag;        /* KAIF_CRUMB_F_* */
} kdi_crumb_t;
______unchanged_portion_omitted_

typedef struct kdi_msr {
        uint_t          msr_num;
        uint_t          msr_type;
        union {
                uint64_t *_msr_valp;
                uint64_t _msr_val;
        } _u;
} kdi_msr_t;

#define kdi_msr_val     _u._msr_val
#define kdi_msr_valp    _u._msr_valp

/*
 * Data structure used to hold all of the state for a given CPU.
 */
typedef struct kdi_cpusave {
        greg_t                  *krs_gregs;     /* saved registers */

        kdi_drreg_t             krs_dr;         /* saved debug registers */

        user_desc_t             *krs_gdt;       /* GDT address */
        gate_desc_t             *krs_idt;       /* IDT address */

        greg_t                  krs_cr0;        /* saved %cr0 */

        kdi_msr_t               *krs_msr;       /* ptr to MSR save area */

        uint_t                  krs_cpu_state;  /* KDI_CPU_STATE_* mstr/slv */
        uint_t                  krs_cpu_flushed; /* Have caches been flushed? */
        uint_t                  krs_cpu_id;     /* this CPU's ID */

        /* Bread crumb ring buffer */
        ulong_t                 krs_curcrumbidx; /* Current krs_crumbs idx */
        kdi_crumb_t             *krs_curcrumb;  /* Pointer to current crumb */
        kdi_crumb_t             krs_crumbs[KDI_NCRUMBS]; /* Crumbs */
} kdi_cpusave_t;
______unchanged_portion_omitted_
```

```
**********************************************************
   32725 Wed Feb 28 17:34:25 2018
new/usr/src/uts/intel/sys/x86_archext.h
9215 update CPUID defines
**********************************************************
```

```
/*
 * 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) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2011 by Delphix. All rights reserved.
 * Copyright 2012 Nexenta Systems, Inc. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright 2017 Joyent, Inc.
 * Copyright 2012 Jens Elkner <jel+illumos@cs.uni-magdeburg.de>
 * Copyright 2012 Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
 * Copyright 2014 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 */

#ifndef _SYS_X86_ARCHEXT_H
#define _SYS_X86_ARCHEXT_H

#if !defined(_ASM)
#include <sys/regset.h>
#include <sys/processor.h>
#include <vm/seg_enum.h>
#include <vm/page.h>
#endif  /* _ASM */

#ifdef __cplusplus
extern "C" {
#endif

/*
 * cpuid instruction feature flags in %edx (standard function 1)
 */

#define CPUID_INTC_EDX_FPU      0x00000001      /* x87 fpu present */
#define CPUID_INTC_EDX_VME      0x00000002      /* virtual-8086 extension */
#define CPUID_INTC_EDX_DE       0x00000004      /* debugging extensions */
#define CPUID_INTC_EDX_PSE      0x00000008      /* page size extension */
#define CPUID_INTC_EDX_TSC      0x00000010      /* time stamp counter */
#define CPUID_INTC_EDX_MSR      0x00000020      /* rdmsr and wrmsr */
#define CPUID_INTC_EDX_PAE      0x00000040      /* physical addr extension */
```



```
#define CPUID_INTC_EDX_MCE      0x00000080      /* machine check exception */
#define CPUID_INTC_EDX_CX8      0x00000100      /* cmpxchg8b instruction */
#define CPUID_INTC_EDX_APIC     0x00000200      /* local APIC */
                                                /* 0x400 - reserved */
#define CPUID_INTC_EDX_SEP      0x00000800      /* sysenter and sysexit */
#define CPUID_INTC_EDX_MTRR     0x00001000      /* memory type range reg */
#define CPUID_INTC_EDX_PGE      0x00002000      /* page global enable */
#define CPUID_INTC_EDX_MCA      0x00004000      /* machine check arch */
#define CPUID_INTC_EDX_CMOV     0x00008000      /* conditional move insns */
#define CPUID_INTC_EDX_PAT      0x00010000      /* page attribute table */
#define CPUID_INTC_EDX_PSE36    0x00020000      /* 36-bit pagesize extension */
#define CPUID_INTC_EDX_PSN      0x00040000      /* processor serial number */
#define CPUID_INTC_EDX_CLFSH    0x00080000      /* clflush instruction */
                                                /* 0x100000 - reserved */
#define CPUID_INTC_EDX_DS       0x00200000      /* debug store exists */
#define CPUID_INTC_EDX_ACPI     0x00400000      /* monitoring + clock ctrl */
#define CPUID_INTC_EDX_MMX      0x00800000      /* MMX instructions */
#define CPUID_INTC_EDX_FXSR     0x01000000      /* fxsave and fxrstor */
#define CPUID_INTC_EDX_SSE      0x02000000      /* streaming SIMD extensions */
#define CPUID_INTC_EDX_SSE2     0x04000000      /* SSE extensions */
#define CPUID_INTC_EDX_SS       0x08000000      /* self-snoop */
#define CPUID_INTC_EDX_HTT      0x10000000      /* Hyper Thread Technology */
#define CPUID_INTC_EDX_TM       0x20000000      /* thermal monitoring */
#define CPUID_INTC_EDX_IA64     0x40000000      /* Itanium emulating IA32 */
#define CPUID_INTC_EDX_PBE      0x80000000      /* Pending Break Enable */

/*
 * cpuid instruction feature flags in %ecx (standard function 1)
 */

#define CPUID_INTC_ECX_SSE3     0x00000001      /* Yet more SSE extensions */
#define CPUID_INTC_ECX_PCLMULQDQ 0x00000002     /* PCLMULQDQ insn */
#define CPUID_INTC_ECX_DTES64   0x00000004      /* 64-bit DS area */
                                                /* 0x00000004 - reserved */
#define CPUID_INTC_ECX_MON      0x00000008      /* MONITOR/MWAIT */
#define CPUID_INTC_ECX_DSCPL    0x00000010      /* CPL-qualified debug store */
#define CPUID_INTC_ECX_VMX      0x00000020      /* Hardware VM extensions */
#define CPUID_INTC_ECX_SMX      0x00000040      /* Secure mode extensions */
#define CPUID_INTC_ECX_EST      0x00000080      /* enhanced SpeedStep */
#define CPUID_INTC_ECX_TM2      0x00000100      /* thermal monitoring */
#define CPUID_INTC_ECX_SSSE3    0x00000200      /* Supplemental SSE3 insns */
#define CPUID_INTC_ECX_CID      0x00000400      /* L1 context ID */
                                                /* 0x00000800 - reserved */
#define CPUID_INTC_ECX_FMA      0x00001000      /* Fused Multiply Add */
#define CPUID_INTC_ECX_CX16     0x00002000      /* cmpxchg16 */
#define CPUID_INTC_ECX_ETPRD    0x00004000      /* extended task pri messages */
#define CPUID_INTC_ECX_PDCM     0x00008000      /* Perf/Debug Capability MSR */
                                                /* 0x00008000 - reserved */
                                                /* 0x00010000 - reserved */
#define CPUID_INTC_ECX_PCID     0x00020000      /* process-context ids */
                                                /* 0x00020000 - reserved */
#define CPUID_INTC_ECX_DCA      0x00040000      /* direct cache access */
#define CPUID_INTC_ECX_SSE4_1   0x00080000      /* SSE4.1 insns */
#define CPUID_INTC_ECX_SSE4_2   0x00100000      /* SSE4.2 insns */
#define CPUID_INTC_ECX_X2APIC   0x00200000      /* x2APIC */
#define CPUID_INTC_ECX_MOVBE    0x00400000      /* MOVBE insn */
#define CPUID_INTC_ECX_POPCNT   0x00800000      /* POPCNT insn */
#define CPUID_INTC_ECX_TSCDL    0x01000000      /* Deadline TSC */
#define CPUID_INTC_ECX_AES      0x02000000      /* AES insns */
#define CPUID_INTC_ECX_XSAVE    0x04000000      /* XSAVE/XRESTOR insns */
#define CPUID_INTC_ECX_OSXSAVE  0x08000000      /* OS supports XSAVE insns */
#define CPUID_INTC_ECX_AVX      0x10000000      /* AVX supported */
#define CPUID_INTC_ECX_F16C     0x20000000      /* F16C supported */
#define CPUID_INTC_ECX_RDRAND   0x40000000      /* RDRAND supported */
#define CPUID_INTC_ECX_HV       0x80000000      /* Hypervisor */
```

```
125 /*
126  * cpuid instruction feature flags in %edx (extended function 0x80000001)
127  */

129 #define CPUID_AMD_EDX_FPU       0x00000001      /* x87 fpu present */
130 #define CPUID_AMD_EDX_VME       0x00000002      /* virtual-8086 extension */
131 #define CPUID_AMD_EDX_DE        0x00000004      /* debugging extensions */
132 #define CPUID_AMD_EDX_PSE       0x00000008      /* page size extensions */
133 #define CPUID_AMD_EDX_TSC       0x00000010      /* time stamp counter */
134 #define CPUID_AMD_EDX_MSR       0x00000020      /* rdmsr and wrmsr */
135 #define CPUID_AMD_EDX_PAE       0x00000040      /* physical addr extension */
136 #define CPUID_AMD_EDX_MCE       0x00000080      /* machine check exception */
137 #define CPUID_AMD_EDX_CX8       0x00000100      /* cmpxchg8b instruction */
138 #define CPUID_AMD_EDX_APIC      0x00000200      /* local APIC */
139                                                 /* 0x00000400 - sysc on K6m6 */
140 #define CPUID_AMD_EDX_SYSC      0x00000800      /* AMD: syscall and sysret */
141 #define CPUID_AMD_EDX_MTRR      0x00001000      /* memory type and range reg */
142 #define CPUID_AMD_EDX_PGE       0x00002000      /* page global enable */
143 #define CPUID_AMD_EDX_MCA       0x00004000      /* machine check arch */
144 #define CPUID_AMD_EDX_CMOV      0x00008000      /* conditional move insns */
145 #define CPUID_AMD_EDX_PAT       0x00010000      /* K7: page attribute table */
146 #define CPUID_AMD_EDX_FCMOV     0x00010000      /* FCMOVcc etc. */
147 #define CPUID_AMD_EDX_PSE36     0x00020000      /* 36-bit pagesize extension */
148                                 /* 0x00040000 - reserved */
149                                 /* 0x00080000 - reserved */
150 #define CPUID_AMD_EDX_NX        0x00100000      /* AMD: no-execute page prot */
151                                 /* 0x00200000 - reserved */
152 #define CPUID_AMD_EDX_MMXamd    0x00400000      /* AMD: MMX extensions */
153 #define CPUID_AMD_EDX_MMX       0x00800000      /* MMX instructions */
154 #define CPUID_AMD_EDX_FXSR      0x01000000      /* fxsave and fxrstor */
155 #define CPUID_AMD_EDX_FFXSR     0x02000000      /* fast fxsave/fxrstor */
156 #define CPUID_AMD_EDX_1GPG      0x04000000      /* 1GB page */
157 #define CPUID_AMD_EDX_TSCP      0x08000000      /* rdtscp instruction */
158                                 /* 0x10000000 - reserved */
159 #define CPUID_AMD_EDX_LM        0x20000000      /* AMD: long mode */
160 #define CPUID_AMD_EDX_3DNowx    0x40000000      /* AMD: extensions to 3DNow! */
161 #define CPUID_AMD_EDX_3DNow     0x80000000      /* AMD: 3DNow! instructions */

163 #define CPUID_AMD_ECX_AHF64     0x00000001      /* LAHF and SAHF in long mode */
164 #define CPUID_AMD_ECX_CMP_LGCY  0x00000002      /* AMD: multicore chip */
165 #define CPUID_AMD_ECX_SVM       0x00000004      /* AMD: secure VM */
166 #define CPUID_AMD_ECX_EAS       0x00000008      /* extended apic space */
167 #define CPUID_AMD_ECX_CR8D      0x00000010      /* AMD: 32-bit mov %cr8 */
168 #define CPUID_AMD_ECX_LZCNT     0x00000020      /* AMD: LZCNT insn */
169 #define CPUID_AMD_ECX_SSE4A     0x00000040      /* AMD: SSE4A insns */
170 #define CPUID_AMD_ECX_MAS       0x00000080      /* AMD: MisAlignSse mnode */
171 #define CPUID_AMD_ECX_3DNP      0x00000100      /* AMD: 3DNowPrefectch */
172 #define CPUID_AMD_ECX_OSVW      0x00000200      /* AMD: OSVW */
173 #define CPUID_AMD_ECX_IBS       0x00000400      /* AMD: IBS */
174 #define CPUID_AMD_ECX_SSE5      0x00000800      /* AMD: Extended AVX */
173 #define CPUID_AMD_ECX_SSE5      0x00000800      /* AMD: SSE5 */
175 #define CPUID_AMD_ECX_SKINIT    0x00001000      /* AMD: SKINIT */
176 #define CPUID_AMD_ECX_WDT       0x00002000      /* AMD: WDT */
177                                 /* 0x00004000 - reserved */
178 #define CPUID_AMD_ECX_LWP       0x00008000      /* AMD: Lightweight profiling */
179 #define CPUID_AMD_ECX_FMA4      0x00010000      /* AMD: 4-operand FMA support */
180                                 /* 0x00020000 - reserved */
181                                 /* 0x00040000 - reserved */
182 #define CPUID_AMD_ECX_NIDMSR    0x00080000      /* AMD: Node ID MSR */
183                                 /* 0x00100000 - reserved */
184 #define CPUID_AMD_ECX_TBM       0x00200000      /* AMD: trailing bit manips. */
185 #define CPUID_AMD_ECX_TOPOEXT   0x00400000      /* AMD: Topology Extensions */

187 /*
188  * AMD uses %ebx for some of their features (extended function 0x80000008).
189  */
```



```
190 #define CPUID_AMD_EBX_ERR_PTR_ZERO      0x00000004 /* AMD: FP Err. Ptr. Zero */

192 /*
193  * Intel now seems to have claimed part of the "extended" function
194  * space that we previously for non-Intel implementors to use.
195  * More excitingly still, they've claimed bit 20 to mean LAHF/SAHF
196  * is available in long mode i.e. what AMD indicate using bit 0.
197  * On the other hand, everything else is labelled as reserved.
198  */
199 #define CPUID_INTC_ECX_AHF64    0x00100000      /* LAHF and SAHF in long mode */

201 /*
202  * Intel also uses cpuid leaf 7 to have additional instructions and features.
203  * Like some other leaves, but unlike the current ones we care about, it
204  * requires us to specify both a leaf in %eax and a sub-leaf in %ecx. To deal
205  * with the potential use of additional sub-leaves in the future, we now
206  * specifically label the EBX features with their leaf and sub-leaf.
207  */
208 #define CPUID_INTC_EBX_7_0_BMI1         0x00000008      /* BMI1 instrs */
209 #define CPUID_INTC_EBX_7_0_HLE          0x00000010      /* HLE */
210 #define CPUID_INTC_EBX_7_0_AVX2         0x00000020      /* AVX2 supported */
211 #define CPUID_INTC_EBX_7_0_SMEP         0x00000080      /* SMEP in CR4 */
212 #define CPUID_INTC_EBX_7_0_BMI2         0x00000100      /* BMI2 instrs */
213 #define CPUID_INTC_EBX_7_0_MPX          0x00004000      /* Mem. Prot. Ext. */
214 #define CPUID_INTC_EBX_7_0_AVX512F      0x00010000      /* AVX512 foundation */
215 #define CPUID_INTC_EBX_7_0_AVX512DQ     0x00020000      /* AVX512DQ */
216 #define CPUID_INTC_EBX_7_0_RDSEED       0x00040000      /* RDSEED instr */
217 #define CPUID_INTC_EBX_7_0_ADX          0x00080000      /* ADX instrs */
218 #define CPUID_INTC_EBX_7_0_SMAP         0x00100000      /* SMAP in CR 4 */
219 #define CPUID_INTC_EBX_7_0_AVX512IFMA   0x00200000      /* AVX512IFMA */
220 #define CPUID_INTC_EBX_7_0_CLWB         0x01000000      /* CLWB */
221 #define CPUID_INTC_EBX_7_0_AVX512PF     0x04000000      /* AVX512PF */
222 #define CPUID_INTC_EBX_7_0_AVX512ER     0x08000000      /* AVX512ER */
223 #define CPUID_INTC_EBX_7_0_AVX512CD     0x10000000      /* AVX512CD */
224 #define CPUID_INTC_EBX_7_0_SHA          0x20000000      /* SHA extensions */
225 #define CPUID_INTC_EBX_7_0_AVX512BW     0x40000000      /* AVX512BW */
226 #define CPUID_INTC_EBX_7_0_AVX512VL     0x80000000      /* AVX512VL */

228 #define CPUID_INTC_EBX_7_0_ALL_AVX512 \
229         (CPUID_INTC_EBX_7_0_AVX512F | CPUID_INTC_EBX_7_0_AVX512DQ | \
230         CPUID_INTC_EBX_7_0_AVX512IFMA | CPUID_INTC_EBX_7_0_AVX512PF | \
231         CPUID_INTC_EBX_7_0_AVX512ER | CPUID_INTC_EBX_7_0_AVX512CD | \
232         CPUID_INTC_EBX_7_0_AVX512BW | CPUID_INTC_EBX_7_0_AVX512VL)

234 #define CPUID_INTC_ECX_7_0_AVX512VBMI   0x00000002      /* AVX512VBMI */
235 #define CPUID_INTC_ECX_7_0_UMIP         0x00000004      /* UMIP */
236 #define CPUID_INTC_ECX_7_0_PKU          0x00000008      /* umode prot. keys */
237 #define CPUID_INTC_ECX_7_0_OSPKE        0x00000010      /* OSPKE */
238 #define CPUID_INTC_ECX_7_0_AVX512VPOPCDQ 0x00004000     /* AVX512 VPOPCNTDQ */

240 #define CPUID_INTC_ECX_7_0_ALL_AVX512 \
241         (CPUID_INTC_ECX_7_0_AVX512VBMI | CPUID_INTC_ECX_7_0_AVX512VPOPCDQ)

243 #define CPUID_INTC_EDX_7_0_AVX5124NNIW  0x00000004      /* AVX512 4NNIW */
244 #define CPUID_INTC_EDX_7_0_AVX5124FMAPS 0x00000008      /* AVX512 4FMAPS */

246 #define CPUID_INTC_EDX_7_0_ALL_AVX512 \
247         (CPUID_INTC_EDX_7_0_AVX5124NNIW | CPUID_INTC_EDX_7_0_AVX5124FMAPS)

249 /*
250  * Intel also uses cpuid leaf 0xd to report additional instructions and features
251  * when the sub-leaf in %ecx == 1. We label these using the same convention as
252  * with leaf 7.
253  */
254 #define CPUID_INTC_EAX_D_1_XSAVEOPT     0x00000001      /* xsaveopt inst. */
255 #define CPUID_INTC_EAX_D_1_XSAVEC       0x00000002      /* xsavec inst. */
```

```
256 #define CPUID_INTC_EAX_D_1_XSAVES       0x00000008      /* xsaves inst. */

258 #define P5_MCHADDR      0x0
259 #define P5_CESR         0x11
260 #define P5_CTR0         0x12
261 #define P5_CTR1         0x13

263 #define K5_MCHADDR      0x0
264 #define K5_MCHTYPE      0x01
265 #define K5_TSC          0x10
266 #define K5_TR12         0x12

268 #define REG_PAT         0x277

270 #define REG_MC0_CTL             0x400
271 #define REG_MC5_MISC            0x417
272 #define REG_PERFCTR0            0xc1
273 #define REG_PERFCTR1            0xc2

275 #define REG_PERFEVNT0           0x186
276 #define REG_PERFEVNT1           0x187

278 #define REG_TSC                 0x10    /* timestamp counter */
279 #define REG_APIC_BASE_MSR       0x1b
280 #define REG_X2APIC_BASE_MSR     0x800   /* The MSR address offset of x2APIC */

282 #if !defined(__xpv)
283 /*
284  * AMD C1E
285  */
286 #define MSR_AMD_INT_PENDING_CMP_HALT    0xC0010055
287 #define AMD_ACTONCMPHALT_SHIFT  27
288 #define AMD_ACTONCMPHALT_MASK   3
289 #endif

291 #define MSR_DEBUGCTL            0x1d9

293 #define DEBUGCTL_LBR            0x01
294 #define DEBUGCTL_BTF            0x02

296 /* Intel P6, AMD */
297 #define MSR_LBR_FROM            0x1db
298 #define MSR_LBR_TO              0x1dc
299 #define MSR_LEX_FROM            0x1dd
300 #define MSR_LEX_TO              0x1de

302 /* Intel P4 (pre-Prescott, non P4 M) */
303 #define MSR_P4_LBSTK_TOS        0x1da
304 #define MSR_P4_LBSTK_0          0x1db
305 #define MSR_P4_LBSTK_1          0x1dc
306 #define MSR_P4_LBSTK_2          0x1dd
307 #define MSR_P4_LBSTK_3          0x1de

309 /* Intel Pentium M */
310 #define MSR_P6M_LBSTK_TOS       0x1c9
311 #define MSR_P6M_LBSTK_0         0x040
312 #define MSR_P6M_LBSTK_1         0x041
313 #define MSR_P6M_LBSTK_2         0x042
314 #define MSR_P6M_LBSTK_3         0x043
315 #define MSR_P6M_LBSTK_4         0x044
316 #define MSR_P6M_LBSTK_5         0x045
317 #define MSR_P6M_LBSTK_6         0x046
318 #define MSR_P6M_LBSTK_7         0x047

320 /* Intel P4 (Prescott) */
321 #define MSR_PRP4_LBSTK_TOS      0x1da
```



```
322 #define MSR_PRP4_LBSTK_FROM_0   0x680
323 #define MSR_PRP4_LBSTK_FROM_1   0x681
324 #define MSR_PRP4_LBSTK_FROM_2   0x682
325 #define MSR_PRP4_LBSTK_FROM_3   0x683
326 #define MSR_PRP4_LBSTK_FROM_4   0x684
327 #define MSR_PRP4_LBSTK_FROM_5   0x685
328 #define MSR_PRP4_LBSTK_FROM_6   0x686
329 #define MSR_PRP4_LBSTK_FROM_7   0x687
330 #define MSR_PRP4_LBSTK_FROM_8   0x688
331 #define MSR_PRP4_LBSTK_FROM_9   0x689
332 #define MSR_PRP4_LBSTK_FROM_10  0x68a
333 #define MSR_PRP4_LBSTK_FROM_11  0x68b
334 #define MSR_PRP4_LBSTK_FROM_12  0x68c
335 #define MSR_PRP4_LBSTK_FROM_13  0x68d
336 #define MSR_PRP4_LBSTK_FROM_14  0x68e
337 #define MSR_PRP4_LBSTK_FROM_15  0x68f
338 #define MSR_PRP4_LBSTK_TO_0     0x6c0
339 #define MSR_PRP4_LBSTK_TO_1     0x6c1
340 #define MSR_PRP4_LBSTK_TO_2     0x6c2
341 #define MSR_PRP4_LBSTK_TO_3     0x6c3
342 #define MSR_PRP4_LBSTK_TO_4     0x6c4
343 #define MSR_PRP4_LBSTK_TO_5     0x6c5
344 #define MSR_PRP4_LBSTK_TO_6     0x6c6
345 #define MSR_PRP4_LBSTK_TO_7     0x6c7
346 #define MSR_PRP4_LBSTK_TO_8     0x6c8
347 #define MSR_PRP4_LBSTK_TO_9     0x6c9
348 #define MSR_PRP4_LBSTK_TO_10    0x6ca
349 #define MSR_PRP4_LBSTK_TO_11    0x6cb
350 #define MSR_PRP4_LBSTK_TO_12    0x6cc
351 #define MSR_PRP4_LBSTK_TO_13    0x6cd
352 #define MSR_PRP4_LBSTK_TO_14    0x6ce
353 #define MSR_PRP4_LBSTK_TO_15    0x6cf

355 #define MCI_CTL_VALUE           0xffffffff

357 #define MTRR_TYPE_UC            0
358 #define MTRR_TYPE_WC            1
359 #define MTRR_TYPE_WT            4
360 #define MTRR_TYPE_WP            5
361 #define MTRR_TYPE_WB            6
362 #define MTRR_TYPE_UC_           7

364 /*
365  * For Solaris we set up the page attritubute table in the following way:
366  * PAT0 Write-Back
367  * PAT1 Write-Through
368  * PAT2 Unchacheable-
369  * PAT3 Uncacheable
370  * PAT4 Write-Back
371  * PAT5 Write-Through
372  * PAT6 Write-Combine
373  * PAT7 Uncacheable
374  * The only difference from h/w default is entry 6.
375  */
376 #define PAT_DEFAULT_ATTRIBUTE                   \
377         ((uint64_t)MTRR_TYPE_WB |               \
378         ((uint64_t)MTRR_TYPE_WT << 8) |         \
379         ((uint64_t)MTRR_TYPE_UC_ << 16) |       \
380         ((uint64_t)MTRR_TYPE_UC << 24) |        \
381         ((uint64_t)MTRR_TYPE_WB << 32) |        \
382         ((uint64_t)MTRR_TYPE_WT << 40) |        \
383         ((uint64_t)MTRR_TYPE_WC << 48) |        \
384         ((uint64_t)MTRR_TYPE_UC << 56))

386 #define X86FSET_LARGEPAGE       0
387 #define X86FSET_TSC             1
```

```
388 #define X86FSET_MSR             2
389 #define X86FSET_MTRR            3
390 #define X86FSET_PGE             4
391 #define X86FSET_DE              5
392 #define X86FSET_CMOV            6
393 #define X86FSET_MMX             7
394 #define X86FSET_MCA             8
395 #define X86FSET_PAE             9
396 #define X86FSET_CX8             10
397 #define X86FSET_PAT             11
398 #define X86FSET_SEP             12
399 #define X86FSET_SSE             13
400 #define X86FSET_SSE2            14
401 #define X86FSET_HTT             15
402 #define X86FSET_ASYSC           16
403 #define X86FSET_NX              17
404 #define X86FSET_SSE3            18
405 #define X86FSET_CX16            19
406 #define X86FSET_CMP             20
407 #define X86FSET_TSCP            21
408 #define X86FSET_MWAIT           22
409 #define X86FSET_SSE4A           23
410 #define X86FSET_CPUID           24
411 #define X86FSET_SSSE3           25
412 #define X86FSET_SSE4_1          26
413 #define X86FSET_SSE4_2          27
414 #define X86FSET_1GPG            28
415 #define X86FSET_CLFSH           29
416 #define X86FSET_64              30
417 #define X86FSET_AES             31
418 #define X86FSET_PCLMULQDQ       32
419 #define X86FSET_XSAVE           33
420 #define X86FSET_AVX             34
421 #define X86FSET_VMX             35
422 #define X86FSET_SVM             36
423 #define X86FSET_TOPOEXT         37
424 #define X86FSET_F16C            38
425 #define X86FSET_RDRAND          39
426 #define X86FSET_X2APIC          40
427 #define X86FSET_AVX2            41
428 #define X86FSET_BMI1            42
429 #define X86FSET_BMI2            43
430 #define X86FSET_FMA             44
431 #define X86FSET_SMEP            45
432 #define X86FSET_SMAP            46
433 #define X86FSET_ADX             47
434 #define X86FSET_RDSEED          48
435 #define X86FSET_MPX             49
436 #define X86FSET_AVX512F         50
437 #define X86FSET_AVX512DQ        51
438 #define X86FSET_AVX512PF        52
439 #define X86FSET_AVX512ER        53
440 #define X86FSET_AVX512CD        54
441 #define X86FSET_AVX512BW        55
442 #define X86FSET_AVX512VL        56
443 #define X86FSET_AVX512FMA       57
444 #define X86FSET_AVX512VBMI      58
445 #define X86FSET_AVX512VPOPCDQ   59
446 #define X86FSET_AVX512NNIW      60
447 #define X86FSET_AVX512FMAPS     61
448 #define X86FSET_XSAVEOPT        62
449 #define X86FSET_XSAVEC          63
450 #define X86FSET_XSAVES          64
451 #define X86FSET_SHA             65
452 #define X86FSET_UMIP            66
453 #define X86FSET_PKU             67
```



```
454 #define X86FSET_OSPKE           68

456 /*
457  * Intel Deep C-State invariant TSC in leaf 0x80000007.
458  */
459 #define CPUID_TSC_CSTATE_INVARIANCE     (0x100)

461 /*
462  * Intel Deep C-state always-running local APIC timer
463  */
464 #define CPUID_CSTATE_ARAT       (0x4)

466 /*
467  * Intel ENERGY_PERF_BIAS MSR indicated by feature bit CPUID.6.ECX[3].
468  */
469 #define CPUID_EPB_SUPPORT       (1 << 3)

471 /*
472  * Intel TSC deadline timer
473  */
474 #define CPUID_DEADLINE_TSC      (1 << 24)

476 /*
477  * x86_type is a legacy concept; this is supplanted
478  * for most purposes by x86_featureset; modern CPUs
479  * should be X86_TYPE_OTHER
480  */
481 #define X86_TYPE_OTHER          0
482 #define X86_TYPE_486            1
483 #define X86_TYPE_P5             2
484 #define X86_TYPE_P6             3
485 #define X86_TYPE_CYRIX_486      4
486 #define X86_TYPE_CYRIX_6x86L    5
487 #define X86_TYPE_CYRIX_6x86     6
488 #define X86_TYPE_CYRIX_GXm      7
489 #define X86_TYPE_CYRIX_6x86MX   8
490 #define X86_TYPE_CYRIX_MediaGX  9
491 #define X86_TYPE_CYRIX_MII      10
492 #define X86_TYPE_VIA_CYRIX_III  11
493 #define X86_TYPE_P4             12

495 /*
496  * x86_vendor allows us to select between
497  * implementation features and helps guide
498  * the interpretation of the cpuid instruction.
499  */
500 #define X86_VENDOR_Intel        0
501 #define X86_VENDORSTR_Intel     "GenuineIntel"

503 #define X86_VENDOR_IntelClone   1

505 #define X86_VENDOR_AMD          2
506 #define X86_VENDORSTR_AMD       "AuthenticAMD"

508 #define X86_VENDOR_Cyrix        3
509 #define X86_VENDORSTR_CYRIX     "CyrixInstead"

511 #define X86_VENDOR_UMC          4
512 #define X86_VENDORSTR_UMC       "UMC UMC UMC "

514 #define X86_VENDOR_NexGen       5
515 #define X86_VENDORSTR_NexGen    "NexGenDriven"

517 #define X86_VENDOR_Centaur      6
518 #define X86_VENDORSTR_Centaur   "CentaurHauls"
```

```
#define X86_VENDOR_Rise         7
#define X86_VENDORSTR_Rise      "RiseRiseRise"

#define X86_VENDOR_SiS          8
#define X86_VENDORSTR_SiS       "SiS SiS SiS "

#define X86_VENDOR_TM           9
#define X86_VENDORSTR_TM        "GenuineTMx86"

#define X86_VENDOR_NSC          10
#define X86_VENDORSTR_NSC       "Geode by NSC"

/*
 * Vendor string max len + \0
 */
#define X86_VENDOR_STRLEN       13

/*
 * Some vendor/family/model/stepping ranges are commonly grouped under
 * a single identifying banner by the vendor.  The following encode
 * that "revision" in a uint32_t with the 8 most significant bits
 * identifying the vendor with X86_VENDOR_*, the next 8 identifying the
 * family, and the remaining 16 typically forming a bitmask of revisions
 * within that family with more significant bits indicating "later" revisions.
 */

#define _X86_CHIPREV_VENDOR_MASK        0xff000000u
#define _X86_CHIPREV_VENDOR_SHIFT       24
#define _X86_CHIPREV_FAMILY_MASK        0x00ff0000u
#define _X86_CHIPREV_FAMILY_SHIFT       16
#define _X86_CHIPREV_REV_MASK           0x0000ffffu

#define _X86_CHIPREV_VENDOR(x) \
        (((x) & _X86_CHIPREV_VENDOR_MASK) >> _X86_CHIPREV_VENDOR_SHIFT)
#define _X86_CHIPREV_FAMILY(x) \
        (((x) & _X86_CHIPREV_FAMILY_MASK) >> _X86_CHIPREV_FAMILY_SHIFT)
#define _X86_CHIPREV_REV(x) \
        ((x) & _X86_CHIPREV_REV_MASK)

/* True if x matches in vendor and family and if x matches the given rev mask */
#define X86_CHIPREV_MATCH(x, mask) \
        (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(mask) && \
        _X86_CHIPREV_FAMILY(x) == _X86_CHIPREV_FAMILY(mask) && \
        ((_X86_CHIPREV_REV(x) & _X86_CHIPREV_REV(mask)) != 0))

/* True if x matches in vendor and family, and rev is at least minx */
#define X86_CHIPREV_ATLEAST(x, minx) \
        (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(minx) && \
        _X86_CHIPREV_FAMILY(x) == _X86_CHIPREV_FAMILY(minx) && \
        _X86_CHIPREV_REV(x) >= _X86_CHIPREV_REV(minx))

#define _X86_CHIPREV_MKREV(vendor, family, rev) \
        ((uint32_t)(vendor) << _X86_CHIPREV_VENDOR_SHIFT | \
        (family) << _X86_CHIPREV_FAMILY_SHIFT | (rev))

/* True if x matches in vendor, and family is at least minx */
#define X86_CHIPFAM_ATLEAST(x, minx) \
        (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(minx) && \
        _X86_CHIPREV_FAMILY(x) >= _X86_CHIPREV_FAMILY(minx))

/* Revision default */
#define X86_CHIPREV_UNKNOWN     0x0

/*
 * Definitions for AMD Family 0xf. Minor revisions C0 and CG are
 * sufficiently different that we will distinguish them; in all other
```



```
 * case we will identify the major revision.
 */
#define X86_CHIPREV_AMD_F_REV_B _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0001)
#define X86_CHIPREV_AMD_F_REV_C0 _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0002)
#define X86_CHIPREV_AMD_F_REV_CG _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0004)
#define X86_CHIPREV_AMD_F_REV_D _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0008)
#define X86_CHIPREV_AMD_F_REV_E _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0010)
#define X86_CHIPREV_AMD_F_REV_F _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0020)
#define X86_CHIPREV_AMD_F_REV_G _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0040)

/*
 * Definitions for AMD Family 0x10.  Rev A was Engineering Samples only.
 */
#define X86_CHIPREV_AMD_10_REV_A \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0001)
#define X86_CHIPREV_AMD_10_REV_B \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0002)
#define X86_CHIPREV_AMD_10_REV_C2 \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0004)
#define X86_CHIPREV_AMD_10_REV_C3 \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0008)
#define X86_CHIPREV_AMD_10_REV_D0 \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0010)
#define X86_CHIPREV_AMD_10_REV_D1 \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0020)
#define X86_CHIPREV_AMD_10_REV_E \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0040)

/*
 * Definitions for AMD Family 0x11.
 */
#define X86_CHIPREV_AMD_11_REV_B \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x11, 0x0002)

/*
 * Definitions for AMD Family 0x12.
 */
#define X86_CHIPREV_AMD_12_REV_B \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x12, 0x0002)

/*
 * Definitions for AMD Family 0x14.
 */
#define X86_CHIPREV_AMD_14_REV_B \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x14, 0x0002)
#define X86_CHIPREV_AMD_14_REV_C \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x14, 0x0004)

/*
 * Definitions for AMD Family 0x15
 */
#define X86_CHIPREV_AMD_15OR_REV_B2 \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0001)

#define X86_CHIPREV_AMD_15TN_REV_A1 \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0002)

/*
 * Various socket/package types, extended as the need to distinguish
 * a new type arises.  The top 8 byte identfies the vendor and the
 * remaining 24 bits describe 24 socket types.
 */

#define _X86_SOCKET_VENDOR_SHIFT        24
#define _X86_SOCKET_VENDOR(x)   ((x) >> _X86_SOCKET_VENDOR_SHIFT)
#define _X86_SOCKET_TYPE_MASK   0x00ffffff
```

```
652 #define _X86_SOCKET_TYPE(x)             ((x) & _X86_SOCKET_TYPE_MASK)

654 #define _X86_SOCKET_MKVAL(vendor, bitval) \
655         ((uint32_t)(vendor) << _X86_SOCKET_VENDOR_SHIFT | (bitval))

657 #define X86_SOCKET_MATCH(s, mask) \
658         (_X86_SOCKET_VENDOR(s) == _X86_SOCKET_VENDOR(mask) && \
659         (_X86_SOCKET_TYPE(s) & _X86_SOCKET_TYPE(mask)) != 0)

661 #define X86_SOCKET_UNKNOWN 0x0
662         /*
663          * AMD socket types
664          */
665 #define X86_SOCKET_754          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000001)
666 #define X86_SOCKET_939          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000002)
667 #define X86_SOCKET_940          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000004)
668 #define X86_SOCKET_S1g1         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000008)
669 #define X86_SOCKET_AM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000010)
670 #define X86_SOCKET_F1207        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000020)
671 #define X86_SOCKET_S1g2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000040)
672 #define X86_SOCKET_S1g3         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000080)
673 #define X86_SOCKET_AM           _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000100)
674 #define X86_SOCKET_AM2R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000200)
675 #define X86_SOCKET_AM3          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000400)
676 #define X86_SOCKET_G34          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000800)
677 #define X86_SOCKET_ASB2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x001000)
678 #define X86_SOCKET_C32          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x002000)
679 #define X86_SOCKET_S1g4         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x004000)
680 #define X86_SOCKET_FT1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x008000)
681 #define X86_SOCKET_FM1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x010000)
682 #define X86_SOCKET_FS1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x020000)
683 #define X86_SOCKET_AM3R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x040000)
684 #define X86_SOCKET_FP2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x080000)
685 #define X86_SOCKET_FS1R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x100000)
686 #define X86_SOCKET_FM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x200000)

688 /*
689  * xgetbv/xsetbv support
690  * See section 13.3 in vol. 1 of the Intel devlopers manual.
691  */

693 #define XFEATURE_ENABLED_MASK   0x0
694 /*
695  * XFEATURE_ENABLED_MASK values (eax)
696  * See setup_xfem().
697  */
698 #define XFEATURE_LEGACY_FP      0x1
699 #define XFEATURE_SSE            0x2
700 #define XFEATURE_AVX            0x4
701 #define XFEATURE_MPX            0x18    /* 2 bits, both 0 or 1 */
702 #define XFEATURE_AVX512         0xe0    /* 3 bits, all 0 or 1 */
703         /* bit 8 unused */
704 #define XFEATURE_PKRU           0x200
705 #define XFEATURE_FP_ALL \
706         (XFEATURE_LEGACY_FP | XFEATURE_SSE | XFEATURE_AVX | XFEATURE_MPX | \
707         XFEATURE_AVX512 | XFEATURE_PKRU)

709 #if !defined(_ASM)

711 #if defined(_KERNEL) || defined(_KMEMUSER)

713 #define NUM_X86_FEATURES        69
714 extern uchar_t x86_featureset[];

716 extern void free_x86_featureset(void *featureset);
717 extern boolean_t is_x86_feature(void *featureset, uint_t feature);
```



```
718 extern void add_x86_feature(void *featureset, uint_t feature);
719 extern void remove_x86_feature(void *featureset, uint_t feature);
720 extern boolean_t compare_x86_featureset(void *setA, void *setB);
721 extern void print_x86_featureset(void *featureset);


724 extern uint_t x86_type;
725 extern uint_t x86_vendor;
726 extern uint_t x86_clflush_size;

728 extern uint_t pentiumpro_bug4046376;

730 extern const char CyrixInstead[];

732 #endif

734 #if defined(_KERNEL)

736 /*
737  * This structure is used to pass arguments and get return values back
738  * from the CPUID instruction in __cpuid_insn() routine.
739  */
740 struct cpuid_regs {
741         uint32_t        cp_eax;
742         uint32_t        cp_ebx;
743         uint32_t        cp_ecx;
744         uint32_t        cp_edx;
745 };
```

_____***unchanged_portion_omitted_***