```
**********************************************************
   31898 Mon Dec 10 01:26:05 2012
new/usr/src/uts/i86pc/cpu/authenticamd/authamd_main.c
3408 detect socket type of newer AMD CPUs
Reviewed by: Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
**********************************************************
```

```
/*
 * 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) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 * "Generic AMD" model-specific support.  If no more-specific support can
 * be found, or such modules declines to initialize, then for AuthenticAMD
 * cpus this module can have a crack at providing some AMD model-specific
 * support that at least goes beyond common MCA architectural features
 * if not down to the nitty-gritty level for a particular model.  We
 * are layered on top of a cpu module, likely cpu.generic, so there is no
 * need for us to perform common architecturally-accessible functions.
 */

#include <sys/types.h>
#include <sys/cmn_err.h>
#include <sys/modctl.h>
#include <sys/cpu_module.h>
#include <sys/mca_x86.h>
#include <sys/pci_cfgspace.h>
#include <sys/x86_archext.h>
#include <sys/mc_amd.h>
#include <sys/fm/protocol.h>
#include <sys/fm/cpu/GENAMD.h>
#include <sys/fm/smb/fmsmb.h>
#include <sys/fm/util.h>
#include <sys/nvpair.h>
#include <sys/controlregs.h>
#include <sys/pghw.h>
#include <sys/sunddi.h>
#include <sys/sysmacros.h>
#include <sys/cpu_module_ms_impl.h>

#include "authamd.h"

extern int x86gentopo_legacy; /* x86 generic topo support */

int authamd_ms_support_disable = 0;
```



```
#define AUTHAMD_F_REVS_BCDE \
        (X86_CHIPREV_AMD_F_REV_B | X86_CHIPREV_AMD_F_REV_C0 | \
        X86_CHIPREV_AMD_F_REV_CG | X86_CHIPREV_AMD_F_REV_D | \
        X86_CHIPREV_AMD_F_REV_E)

#define AUTHAMD_F_REVS_FG \
        (X86_CHIPREV_AMD_F_REV_F | X86_CHIPREV_AMD_F_REV_G)

#define AUTHAMD_10_REVS_AB \
        (X86_CHIPREV_AMD_10_REV_A | X86_CHIPREV_AMD_10_REV_B)

/*
 * Bitmasks of support for various features.  Try to enable features
 * via inclusion in one of these bitmasks and check that at the
 * feature imlementation - that way new family support may often simply
 * simply need to update these bitmasks.
 */

/*
 * Models that include an on-chip NorthBridge.
 */
#define AUTHAMD_NBONCHIP(rev) \
        (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_F_REV_B) || \
        X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_A))

/*
 * Families/revisions for which we can recognise main memory ECC errors.
 */
#define AUTHAMD_MEMECC_RECOGNISED(rev) \
        (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_F_REV_B) || \
        X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_A))

/*
 * Families/revisions that have an Online Spare Control Register
 */
#define AUTHAMD_HAS_ONLINESPARECTL(rev) \
        (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_F_REV_F) || \
        X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_A))

/*
 * Families/revisions for which we will perform NB MCA Config changes
 */
#define AUTHAMD_DO_NBMCACFG(rev) \
        (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_F_REV_B) || \
        X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_A))

/*
 * Families/revisions that have chip cache scrubbers.
 */
#define AUTHAMD_HAS_CHIPSCRUB(rev) \
        (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_F_REV_B) || \
        X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_A))

/*
 * Families/revisions that have a NB misc register or registers -
 * evaluates to 0 if no support, otherwise the number of MC4_MISCj.
 */
#define AUTHAMD_NBMISC_NUM(rev) \
        (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_F_REV_F)? 1 : \
        (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_A) ? 3 : 0))

/*
 * Families/revision for which we wish not to machine check for GART
 * table walk errors - bit 10 of NB CTL.
 */
#define AUTHAMD_NOGARTTBLWLK_MC(rev) \
```

```
127         (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_F_REV_B) || \
128         X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_A))

130 /*
131  * Families/revisions that are potentially L3 capable
132  */
133 #define AUTHAMD_L3CAPABLE(rev) \
134         (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_A))

136 /*
137  * Families/revisions that support x8 ChipKill ECC
138  */
139 #define AUTHAMD_SUPPORTS_X8ECC(rev) \
140         (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_D0))
140         (X86_CHIPREV_ATLEAST(rev, X86_CHIPREV_AMD_10_REV_D))

142 /*
143  * We recognise main memory ECC errors for AUTHAMD_MEMECC_RECOGNISED
144  * revisions as:
145  *
146  *      - being reported by the NB
147  *      - being a compound bus/interconnect error (external to chip)
148  *      - having LL of LG
149  *      - having II of MEM (but could still be a master/target abort)
150  *      - having CECC or UECC set
151  *
152  * We do not check the extended error code (first nibble of the
153  * model-specific error code on AMD) since this has changed from
154  * family 0xf to family 0x10 (ext code 0 now reserved on family 0x10).
155  * Instead we use CECC/UECC to separate off the master/target
156  * abort cases.
157  *
158  * We insist that the detector be the NorthBridge bank;  although
159  * IC/DC can report some main memory errors, they do not capture
160  * an address at sufficient resolution to be useful and the NB will
161  * report most errors.
162  */
163 #define AUTHAMD_IS_MEMECCERR(bank, status) \
164         ((bank) == AMD_MCA_BANK_NB && \
165         MCAX86_ERRCODE_ISBUS_INTERCONNECT(MCAX86_ERRCODE(status)) && \
166         MCAX86_ERRCODE_LL(MCAX86_ERRCODE(status)) == MCAX86_ERRCODE_LL_LG && \
167         MCAX86_ERRCODE_II(MCAX86_ERRCODE(status)) == MCAX86_ERRCODE_II_MEM && \
168         ((status) & (AMD_BANK_STAT_CECC | AMD_BANK_STAT_UECC)))

170 static authamd_error_disp_t authamd_memce_disp = {
171         FM_EREPORT_CPU_GENAMD,
172         FM_EREPORT_CPU_GENAMD_MEM_CE,
173         FM_EREPORT_GENAMD_PAYLOAD_FLAGS_MEM_CE
174 };
```

*unchanged_portion_omitted_*

```
**********************************************************
   14134 Mon Dec 10 01:26:05 2012
new/usr/src/uts/i86pc/os/cpuid_subr.c
3408 detect socket type of newer AMD CPUs
Reviewed by: Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
**********************************************************
```

```
/*
 * 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.
 */

/*
 * Portions Copyright 2009 Advanced Micro Devices, Inc.
 */

/*
 * Copyright 2012 Jens Elkner <jel+illumos@cs.uni-magdeburg.de>
 * Copyright 2012 Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
 */

/*
 * Support functions that interpret CPUID and similar information.
 * These should not be used from anywhere other than cpuid.c and
 * cmi_hw.c - as such we will not list them in any header file
 * such as x86_archext.h.
 *
 * In cpuid.c we process CPUID information for each cpu_t instance
 * we're presented with, and stash this raw information and material
 * derived from it in per-cpu_t structures.
 *
 * If we are virtualized then the CPUID information derived from CPUID
 * instructions executed in the guest is based on whatever the hypervisor
 * wanted to make things look like, and the cpu_t are not necessarily in 1:1
 * or fixed correspondence with real processor execution resources.  In cmi_hw.c
 * we are interested in the native properties of a processor - for fault
 * management (and potentially other, such as power management) purposes;
 * it will tunnel through to real hardware information, and use the
 * functionality provided in this file to process it.
 */

#include <sys/types.h>
#include <sys/systm.h>
#include <sys/bitmap.h>
#include <sys/x86_archext.h>
#include <sys/pci_cfgspace.h>
```



```
#ifdef __xpv
#include <sys/hypervisor.h>
#endif

/*
 * AMD socket types.
 * First index :
 *              0 for family 0xf, revs B thru E
 *              1 for family 0xf, revs F and G
 *              2 for family 0x10
 *              3 for family 0x11
 *              4 for family 0x12
 *              5 for family 0x14
 *              6 for family 0x15, models 00 - 0f
 *              7 for family 0x15, models 10 - 1f
 * Second index by (model & 0x3) for family 0fh,
 * CPUID pkg bits (Fn8000_0001_EBX[31:28]) for later families.
 * Second index by (model & 0x3) for family 0fh
 * or CPUID bits for later families
 */
static uint32_t amd_skts[8][8] = {
static uint32_t amd_skts[4][8] = {
        /*
         * Family 0xf revisions B through E
         */
#define A_SKTS_0                        0
        {
                X86_SOCKET_754,         /* 0b000 */
                X86_SOCKET_940,         /* 0b001 */
                X86_SOCKET_754,         /* 0b010 */
                X86_SOCKET_939,         /* 0b011 */
                X86_SOCKET_UNKNOWN,     /* 0b100 */
                X86_SOCKET_UNKNOWN,     /* 0b101 */
                X86_SOCKET_UNKNOWN,     /* 0b110 */
                X86_SOCKET_UNKNOWN      /* 0b111 */
        },
        /*
         * Family 0xf revisions F and G
         */
#define A_SKTS_1                        1
        {
                X86_SOCKET_S1g1,        /* 0b000 */
                X86_SOCKET_F1207,       /* 0b001 */
                X86_SOCKET_UNKNOWN,     /* 0b010 */
                X86_SOCKET_AM2,         /* 0b011 */
                X86_SOCKET_UNKNOWN,     /* 0b100 */
                X86_SOCKET_UNKNOWN,     /* 0b101 */
                X86_SOCKET_UNKNOWN,     /* 0b110 */
                X86_SOCKET_UNKNOWN      /* 0b111 */
        },
        /*
         * Family 0x10
         */
#define A_SKTS_2                        2
        {
                X86_SOCKET_F1207,       /* 0b000 */
                X86_SOCKET_AM2R2,       /* 0b001 */
                X86_SOCKET_AM,          /* 0b001 */
                X86_SOCKET_S1g3,        /* 0b010 */
                X86_SOCKET_G34,         /* 0b011 */
                X86_SOCKET_ASB2,        /* 0b100 */
                X86_SOCKET_C32,         /* 0b101 */
                X86_SOCKET_UNKNOWN,     /* 0b110 */
                X86_SOCKET_UNKNOWN      /* 0b111 */
        },
```

```
123         /*
124          * Family 0x11
125          */
126 #define A_SKTS_3                        3
127         {
128                 X86_SOCKET_UNKNOWN,     /* 0b000 */
129                 X86_SOCKET_UNKNOWN,     /* 0b001 */
130                 X86_SOCKET_S1g2,        /* 0b010 */
131                 X86_SOCKET_UNKNOWN,     /* 0b011 */
132                 X86_SOCKET_UNKNOWN,     /* 0b100 */
133                 X86_SOCKET_UNKNOWN,     /* 0b101 */
134                 X86_SOCKET_UNKNOWN,     /* 0b110 */
135                 X86_SOCKET_UNKNOWN      /* 0b111 */
136         },

138         /*
139          * Family 0x12
140          */
141 #define A_SKTS_4                        4
142         {
143                 X86_SOCKET_UNKNOWN,     /* 0b000 */
144                 X86_SOCKET_FS1,         /* 0b001 */
145                 X86_SOCKET_FM1,         /* 0b010 */
146                 X86_SOCKET_UNKNOWN,     /* 0b011 */
147                 X86_SOCKET_UNKNOWN,     /* 0b100 */
148                 X86_SOCKET_UNKNOWN,     /* 0b101 */
149                 X86_SOCKET_UNKNOWN,     /* 0b110 */
150                 X86_SOCKET_UNKNOWN      /* 0b111 */
151         },

153         /*
154          * Family 0x14
155          */
156 #define A_SKTS_5                        5
157         {
158                 X86_SOCKET_FT1,         /* 0b000 */
159                 X86_SOCKET_UNKNOWN,     /* 0b001 */
160                 X86_SOCKET_UNKNOWN,     /* 0b010 */
161                 X86_SOCKET_UNKNOWN,     /* 0b011 */
162                 X86_SOCKET_UNKNOWN,     /* 0b100 */
163                 X86_SOCKET_UNKNOWN,     /* 0b101 */
164                 X86_SOCKET_UNKNOWN,     /* 0b110 */
165                 X86_SOCKET_UNKNOWN      /* 0b111 */
166         },

168         /*
169          * Family 0x15 models 00 - 0f
170          */
171 #define A_SKTS_6                        6
172         {
173                 X86_SOCKET_UNKNOWN,     /* 0b000 */
174                 X86_SOCKET_AM3R2,       /* 0b001 */
175                 X86_SOCKET_UNKNOWN,     /* 0b010 */
176                 X86_SOCKET_G34,         /* 0b011 */
177                 X86_SOCKET_UNKNOWN,     /* 0b100 */
178                 X86_SOCKET_C32,         /* 0b101 */
179                 X86_SOCKET_UNKNOWN,     /* 0b110 */
180                 X86_SOCKET_UNKNOWN      /* 0b111 */
181         },

183         /*
184          * Family 0x15 models 10 - 1f
185          */
186 #define A_SKTS_7                        7
187         {
188                 X86_SOCKET_FP2,         /* 0b000 */
```



```
189                 X86_SOCKET_FS1R2,       /* 0b001 */
190                 X86_SOCKET_FM2,         /* 0b010 */
191                 X86_SOCKET_UNKNOWN,     /* 0b011 */
192                 X86_SOCKET_UNKNOWN,     /* 0b100 */
193                 X86_SOCKET_UNKNOWN,     /* 0b101 */
194                 X86_SOCKET_UNKNOWN,     /* 0b110 */
195                 X86_SOCKET_UNKNOWN      /* 0b111 */
196         },

127         }
198 };
_____unchanged_portion_omitted_
204 static struct amd_sktmap_s amd_sktmap[23] = {
134 static struct amd_sktmap_s amd_sktmap[15] = {
205         { X86_SOCKET_754,       "754" },
206         { X86_SOCKET_939,       "939" },
207         { X86_SOCKET_940,       "940" },
208         { X86_SOCKET_S1g1,      "S1g1" },
209         { X86_SOCKET_AM2,       "AM2" },
210         { X86_SOCKET_F1207,     "F(1207)" },
211         { X86_SOCKET_S1g2,      "S1g2" },
212         { X86_SOCKET_S1g3,      "S1g3" },
213         { X86_SOCKET_AM,        "AM" },
214         { X86_SOCKET_AM2R2,     "AM2r2" },
215         { X86_SOCKET_AM3,       "AM3" },
216         { X86_SOCKET_G34,       "G34" },
217         { X86_SOCKET_ASB2,      "ASB2" },
218         { X86_SOCKET_C32,       "C32" },
219         { X86_SOCKET_FT1,       "FT1" },
220         { X86_SOCKET_FM1,       "FM1" },
221         { X86_SOCKET_FS1,       "FS1" },
222         { X86_SOCKET_AM3R2,     "AM3r2" },
223         { X86_SOCKET_FP2,       "FP2" },
224         { X86_SOCKET_FS1R2,     "FS1r2" },
225         { X86_SOCKET_FM2,       "FM2" },
226         { X86_SOCKET_UNKNOWN,   "Unknown" }
227 };
_____unchanged_portion_omitted_
246         /*
247          * =============== AuthenticAMD Family 0xf ===============
248          */

250         /*
251          * Rev B includes model 0x4 stepping 0 and model 0x5 stepping 0 and 1.
252          */
253         { 0xf, 0x04, 0x04, 0x0, 0x0, X86_CHIPREV_AMD_F_REV_B, "B", A_SKTS_0 },
254         { 0xf, 0x05, 0x05, 0x0, 0x1, X86_CHIPREV_AMD_F_REV_B, "B", A_SKTS_0 },
255         /*
256          * Rev C0 includes model 0x4 stepping 8 and model 0x5 stepping 8
257          */
258         { 0xf, 0x04, 0x05, 0x8, 0x8, X86_CHIPREV_AMD_F_REV_C0, "C0", A_SKTS_0 },
259         /*
260          * Rev CG is the rest of extended model 0x0 - i.e., everything
261          * but the rev B and C0 combinations covered above.
262          */
263         { 0xf, 0x00, 0x0f, 0x0, 0xf, X86_CHIPREV_AMD_F_REV_CG, "CG", A_SKTS_0 },
264         /*
265          * Rev D has extended model 0x1.
266          */
267         { 0xf, 0x10, 0x1f, 0x0, 0xf, X86_CHIPREV_AMD_F_REV_D, "D", A_SKTS_0 },
268         /*
269          * Rev E has extended model 0x2.
270          * Extended model 0x3 is unused but available to grow into.
271          */
272         { 0xf, 0x20, 0x3f, 0x0, 0xf, X86_CHIPREV_AMD_F_REV_E, "E", A_SKTS_0 },
273         /*
```

```
274          * Rev F has extended models 0x4 and 0x5.
275          */
276         { 0xf, 0x40, 0x5f, 0x0, 0xf, X86_CHIPREV_AMD_F_REV_F, "F", A_SKTS_1 },
277         /*
278          * Rev G has extended model 0x6.
279          */
280         { 0xf, 0x60, 0x6f, 0x0, 0xf, X86_CHIPREV_AMD_F_REV_G, "G", A_SKTS_1 },

282         /*
283          * ================ AuthenticAMD Family 0x10 ================
284          */

286         /*
287          * Rev A has model 0 and stepping 0/1/2 for DR-{A0,A1,A2}.
288          * Give all of model 0 stepping range to rev A.
289          */
290         { 0x10, 0x00, 0x00, 0x0, 0x2, X86_CHIPREV_AMD_10_REV_A, "A", A_SKTS_2 },

292         /*
293          * Rev B has model 2 and steppings 0/1/0xa/2 for DR-{B0,B1,BA,B2}.
294          * Give all of model 2 stepping range to rev B.
295          */
296         { 0x10, 0x02, 0x02, 0x0, 0xf, X86_CHIPREV_AMD_10_REV_B, "B", A_SKTS_2 },

298         /*
299          * Rev C has models 4-6 (depending on L3 cache configuration)
300          * Give all of models 4-6 stepping range 0-2 to rev C2.
223          * Give all of models 4-6 stepping range to rev C.
301          */
302         { 0x10, 0x4, 0x6, 0x0, 0x2, X86_CHIPREV_AMD_10_REV_C2, "C2", A_SKTS_2 },
225         { 0x10, 0x04, 0x06, 0x0, 0xf, X86_CHIPREV_AMD_10_REV_C, "C", A_SKTS_2 },

304         /*
305          * Rev C has models 4-6 (depending on L3 cache configuration)
306          * Give all of models 4-6 stepping range >= 3 to rev C3.
307          */
308         { 0x10, 0x4, 0x6, 0x3, 0xf, X86_CHIPREV_AMD_10_REV_C3, "C3", A_SKTS_2 },

310         /*
311          * Rev D has models 8 and 9
312          * Give all of model 8 and 9 stepping 0 to rev D0.
229          * Give all of model 8 and 9 stepping range to rev D.
313          */
314         { 0x10, 0x8, 0x9, 0x0, 0x0, X86_CHIPREV_AMD_10_REV_D0, "D0", A_SKTS_2 },
231         { 0x10, 0x08, 0x09, 0x0, 0xf, X86_CHIPREV_AMD_10_REV_D, "D", A_SKTS_2 },

316         /*
317          * Rev D has models 8 and 9
318          * Give all of model 8 and 9 stepping range >= 1 to rev D1.
319          */
320         { 0x10, 0x8, 0x9, 0x1, 0xf, X86_CHIPREV_AMD_10_REV_D1, "D1", A_SKTS_2 },

322         /*
323          * Rev E has models A and stepping 0
324          * Give all of model A stepping range to rev E.
325          */
326         { 0x10, 0xA, 0xA, 0x0, 0xf, X86_CHIPREV_AMD_10_REV_E, "E", A_SKTS_2 },

328         /*
329          * ================ AuthenticAMD Family 0x11 ================
330          */
331         { 0x11, 0x03, 0x03, 0x0, 0xf, X86_CHIPREV_AMD_11_REV_B, "B", A_SKTS_3 },

333         /*
334          * ================ AuthenticAMD Family 0x12 ================
335          */
```



```
336         { 0x12, 0x01, 0x01, 0x0, 0xf, X86_CHIPREV_AMD_12_REV_B, "B", A_SKTS_4 },

338         /*
339          * ================ AuthenticAMD Family 0x14 ================
340          */
341         { 0x14, 0x01, 0x01, 0x0, 0xf, X86_CHIPREV_AMD_14_REV_B, "B", A_SKTS_5 },
342         { 0x14, 0x02, 0x02, 0x0, 0xf, X86_CHIPREV_AMD_14_REV_C, "C", A_SKTS_5 },

344         /*
345          * ================ AuthenticAMD Family 0x15 ================
346          */
347         { 0x15, 0x01, 0x01, 0x2, 0x2, X86_CHIPREV_AMD_15OR_REV_B2, "B2",
348             A_SKTS_6 },
349         { 0x15, 0x10, 0x10, 0x1, 0x1, X86_CHIPREV_AMD_15TN_REV_A1, "A1",
350             A_SKTS_7 },
236         { 0x11, 0x03, 0x3, 0x0, 0xf, X86_CHIPREV_AMD_11, "B", A_SKTS_3 },
351 };

353 static void
354 synth_amd_info(uint_t family, uint_t model, uint_t step,
355     uint32_t *skt_p, uint32_t *chiprev_p, const char **chiprevstr_p)
356 {
357         const struct amd_rev_mapent *rmp;
358         int found = 0;
359         int i;

361         if (family < 0xf)
362                 return;

364         for (i = 0, rmp = amd_revmap; i < sizeof (amd_revmap) / sizeof (*rmp);
365             i++, rmp++) {
366                 if (family == rmp->rm_family &&
367                     model >= rmp->rm_modello && model <= rmp->rm_modelhi &&
368                     step >= rmp->rm_steplo && step <= rmp->rm_stephi) {
369                         found = 1;
370                         break;
371                 }
372         }

374         if (!found)
375                 return;

377         if (chiprev_p != NULL)
378                 *chiprev_p = rmp->rm_chiprev;
379         if (chiprevstr_p != NULL)
380                 *chiprevstr_p = rmp->rm_chiprevstr;

382         if (skt_p != NULL) {
383                 int platform;

385 #ifdef __xpv
386                 /* PV guest */
387                 if (!is_controldom()) {
388                         *skt_p = X86_SOCKET_UNKNOWN;
389                         return;
390                 }
391 #endif
392                 platform = get_hwenv();

394                 if ((platform == HW_XEN_HVM) || (platform == HW_VMWARE)) {
395                         *skt_p = X86_SOCKET_UNKNOWN;
396                 } else if (family == 0xf) {
397                         *skt_p = amd_skts[rmp->rm_sktidx][model & 0x3];
398                 } else {
399                         /*
400                          * Starting with family 10h, socket type is stored in
```

```
401                          * CPUID Fn8000_0001_EBX
402                          */
403                         struct cpuid_regs cp;
404                         int idx;
405
406                         cp.cp_eax = 0x80000001;
407                         (void) __cpuid_insn(&cp);
408
409                         /* PkgType bits */
410                         idx = BITX(cp.cp_ebx, 31, 28);
411
412                         if (idx > 7) {
413                                 /* Reserved bits */
414                                 *skt_p = X86_SOCKET_UNKNOWN;
415                         } else {
416                                 *skt_p = amd_skts[rmp->rm_sktidx][idx];
417                         }
418                         if (family == 0x10) {
301                         } else if (family == 0x10 &&
302                             amd_skts[rmp->rm_sktidx][idx] ==
303                             X86_SOCKET_AM) {
419                                 /*
420                                  * Look at Ddr3Mode bit of DRAM Configuration
421                                  * High Register to decide whether this is
422                                  * actually AM3 or S1g4.
307                                  * AM2r2 (aka AM2+) or AM3.
423                                  */
424                                 uint32_t val;
425
426                                 val = pci_getl_func(0, 24, 2, 0x94);
427                                 if (BITX(val, 8, 8)) {
428                                         if (*skt_p == X86_SOCKET_AM2R2)
312                                 if (BITX(val, 8, 8))
429                                                 *skt_p = X86_SOCKET_AM3;
430                                         else if (*skt_p == X86_SOCKET_S1g3)
431                                                 *skt_p = X86_SOCKET_S1g4;
314                                 else
315                                         *skt_p = X86_SOCKET_AM2R2;
316                         } else {
317                                 *skt_p = amd_skts[rmp->rm_sktidx][idx];
432                                 }
433                         }
434                 }
435         }
436 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   28098 Mon Dec 10 01:26:06 2012
new/usr/src/uts/intel/sys/x86_archext.h
3408 detect socket type of newer AMD CPUs
Reviewed by: Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
**********************************************************
```

```
/*
 * 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 (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright (c) 2011, Joyent, Inc. All rights reserved.
 * Copyright 2012 Jens Elkner <jel+illumos@cs.uni-magdeburg.de>
 * Copyright 2012 Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
 */

#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 */

#define FMT_CPUID_INTC_EDX                                              \
        "\20"                                                           \
        "\40pbe\37ia64\36tm\35htt\34ss\33sse2\32sse\31fxsr"     \
        "\30mmx\27acpi\26ds\24clfsh\23psn\22pse36\21pat"        \
        "\20cmov\17mca\16pge\15mtrr\14sep\12apic\11cx8"         \
        "\10mce\7pae\6msr\5tsc\4pse\3de\2vme\1fpu"

/*
 * 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 */
                                                /* 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 */
                                                /* 0x00001000 - reserved */
#define CPUID_INTC_ECX_CX16     0x00002000      /* cmpxchg16 */
#define CPUID_INTC_ECX_ETPRD    0x00004000      /* extended task pri messages */
                                                /* 0x00008000 - reserved */
                                                /* 0x00010000 - reserved */
                                                /* 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_MOVBE    0x00400000      /* MOVBE insn */
#define CPUID_INTC_ECX_POPCNT   0x00800000      /* POPCNT insn */
#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 FMT_CPUID_INTC_ECX                                              \
        "\20"                                                           \
```

```
127         "\35avx\34osxsav\33xsave"                                   \
128         "\32aes"                                                    \
129         "\30popcnt\27movbe\25sse4.2\24sse4.1\23dca"                 \
130         "\20\17etprd\16cx16\13cid\12ssse3\11tm2"                    \
131         "\10est\7smx\6vmx\5dscpl\4mon\2pclmulqdq\1sse3"

133 /*
134  * cpuid instruction feature flags in %edx (extended function 0x80000001)
135  */

137 #define CPUID_AMD_EDX_FPU       0x00000001      /* x87 fpu present */
138 #define CPUID_AMD_EDX_VME       0x00000002      /* virtual-8086 extension */
139 #define CPUID_AMD_EDX_DE        0x00000004      /* debugging extensions */
140 #define CPUID_AMD_EDX_PSE       0x00000008      /* page size extensions */
141 #define CPUID_AMD_EDX_TSC       0x00000010      /* time stamp counter */
142 #define CPUID_AMD_EDX_MSR       0x00000020      /* rdmsr and wrmsr */
143 #define CPUID_AMD_EDX_PAE       0x00000040      /* physical addr extension */
144 #define CPUID_AMD_EDX_MCE       0x00000080      /* machine check exception */
145 #define CPUID_AMD_EDX_CX8       0x00000100      /* cmpxchg8b instruction */
146 #define CPUID_AMD_EDX_APIC      0x00000200      /* local APIC */
147                                                 /* 0x00000400 - sysc on K6m6 */
148 #define CPUID_AMD_EDX_SYSC      0x00000800      /* AMD: syscall and sysret */
149 #define CPUID_AMD_EDX_MTRR      0x00001000      /* memory type and range reg */
150 #define CPUID_AMD_EDX_PGE       0x00002000      /* page global enable */
151 #define CPUID_AMD_EDX_MCA       0x00004000      /* machine check arch */
152 #define CPUID_AMD_EDX_CMOV      0x00008000      /* conditional move insns */
153 #define CPUID_AMD_EDX_PAT       0x00010000      /* K7: page attribute table */
154 #define CPUID_AMD_EDX_FCMOV     0x00010000      /* FCMOVcc etc. */
155 #define CPUID_AMD_EDX_PSE36     0x00020000      /* 36-bit pagesize extension */
156                                 /* 0x00040000 - reserved */
157                                 /* 0x00080000 - reserved */
158 #define CPUID_AMD_EDX_NX        0x00100000      /* AMD: no-execute page prot */
159                                 /* 0x00200000 - reserved */
160 #define CPUID_AMD_EDX_MMXamd    0x00400000      /* AMD: MMX extensions */
161 #define CPUID_AMD_EDX_MMX       0x00800000      /* MMX instructions */
162 #define CPUID_AMD_EDX_FXSR      0x01000000      /* fxsave and fxrstor */
163 #define CPUID_AMD_EDX_FFXSR     0x02000000      /* fast fxsave/fxrstor */
164 #define CPUID_AMD_EDX_1GPG      0x04000000      /* 1GB page */
165 #define CPUID_AMD_EDX_TSCP      0x08000000      /* rdtscp instruction */
166                                 /* 0x10000000 - reserved */
167 #define CPUID_AMD_EDX_LM        0x20000000      /* AMD: long mode */
168 #define CPUID_AMD_EDX_3DNowx    0x40000000      /* AMD: extensions to 3DNow! */
169 #define CPUID_AMD_EDX_3DNow     0x80000000      /* AMD: 3DNow! instructions */

171 #define FMT_CPUID_AMD_EDX                                           \
172         "\20"                                                       \
173         "\40a3d\37a3d+\36lm\34tscp\32ffxsr\31fxsr"                  \
174         "\30mmx\27mmxext\25nx\22pse\21pat"                          \
175         "\20cmov\17mca\16pge\15mtrr\14syscall\12apic\11cx8"         \
176         "\10mce\7pae\6msr\5tsc\4pse\3de\2vme\1fpu"

178 #define CPUID_AMD_ECX_AHF64     0x00000001      /* LAHF and SAHF in long mode */
179 #define CPUID_AMD_ECX_CMP_LGCY  0x00000002      /* AMD: multicore chip */
180 #define CPUID_AMD_ECX_SVM       0x00000004      /* AMD: secure VM */
181 #define CPUID_AMD_ECX_EAS       0x00000008      /* extended apic space */
182 #define CPUID_AMD_ECX_CR8D      0x00000010      /* AMD: 32-bit mov %cr8 */
183 #define CPUID_AMD_ECX_LZCNT     0x00000020      /* AMD: LZCNT insn */
184 #define CPUID_AMD_ECX_SSE4A     0x00000040      /* AMD: SSE4A insns */
185 #define CPUID_AMD_ECX_MAS       0x00000080      /* AMD: MisAlignSse mnode */
186 #define CPUID_AMD_ECX_3DNP      0x00000100      /* AMD: 3DNowPrefectch */
187 #define CPUID_AMD_ECX_OSVW      0x00000200      /* AMD: OSVW */
188 #define CPUID_AMD_ECX_IBS       0x00000400      /* AMD: IBS */
189 #define CPUID_AMD_ECX_SSE5      0x00000800      /* AMD: SSE5 */
190 #define CPUID_AMD_ECX_SKINIT    0x00001000      /* AMD: SKINIT */
191 #define CPUID_AMD_ECX_WDT       0x00002000      /* AMD: WDT */
192 #define CPUID_AMD_ECX_TOPOEXT   0x00400000      /* AMD: Topology Extensions */
```



```
194 #define FMT_CPUID_AMD_ECX                                           \
195         "\20"                                                       \
196         "\22topoext"                                                \
197         "\14wdt\13skinit\12sse5\11ibs\10osvw\93dnp\8mas"            \
198         "\7sse4a\6lzcnt\5cr8d\3svm\2lcmplgcy\1ahf64"

200 /*
201  * Intel now seems to have claimed part of the "extended" function
202  * space that we previously for non-Intel implementors to use.
203  * More excitingly still, they've claimed bit 20 to mean LAHF/SAHF
204  * is available in long mode i.e. what AMD indicate using bit 0.
205  * On the other hand, everything else is labelled as reserved.
206  */
207 #define CPUID_INTC_ECX_AHF64    0x00100000      /* LAHF and SAHF in long mode */


210 #define P5_MCHADDR      0x0
211 #define P5_CESR         0x11
212 #define P5_CTR0         0x12
213 #define P5_CTR1         0x13

215 #define K5_MCHADDR      0x0
216 #define K5_MCHTYPE      0x01
217 #define K5_TSC          0x10
218 #define K5_TR12         0x12

220 #define REG_PAT         0x277

222 #define REG_MC0_CTL             0x400
223 #define REG_MC5_MISC            0x417
224 #define REG_PERFCTR0            0xc1
225 #define REG_PERFCTR1            0xc2

227 #define REG_PERFEVNT0           0x186
228 #define REG_PERFEVNT1           0x187

230 #define REG_TSC                 0x10    /* timestamp counter */
231 #define REG_APIC_BASE_MSR       0x1b
232 #define REG_X2APIC_BASE_MSR     0x800   /* The MSR address offset of x2APIC */

234 #if !defined(__xpv)
235 /*
236  * AMD C1E
237  */
238 #define MSR_AMD_INT_PENDING_CMP_HALT    0xC0010055
239 #define AMD_ACTONCMPHALT_SHIFT  27
240 #define AMD_ACTONCMPHALT_MASK   3
241 #endif

243 #define MSR_DEBUGCTL            0x1d9

245 #define DEBUGCTL_LBR            0x01
246 #define DEBUGCTL_BTF            0x02

248 /* Intel P6, AMD */
249 #define MSR_LBR_FROM            0x1db
250 #define MSR_LBR_TO              0x1dc
251 #define MSR_LEX_FROM            0x1dd
252 #define MSR_LEX_TO              0x1de

254 /* Intel P4 (pre-Prescott, non P4 M) */
255 #define MSR_P4_LBSTK_TOS        0x1da
256 #define MSR_P4_LBSTK_0          0x1db
257 #define MSR_P4_LBSTK_1          0x1dc
258 #define MSR_P4_LBSTK_2          0x1dd
```

```
#define MSR_P4_LBSTK_3          0x1de

/* Intel Pentium M */
#define MSR_P6M_LBSTK_TOS       0x1c9
#define MSR_P6M_LBSTK_0         0x040
#define MSR_P6M_LBSTK_1         0x041
#define MSR_P6M_LBSTK_2         0x042
#define MSR_P6M_LBSTK_3         0x043
#define MSR_P6M_LBSTK_4         0x044
#define MSR_P6M_LBSTK_5         0x045
#define MSR_P6M_LBSTK_6         0x046
#define MSR_P6M_LBSTK_7         0x047

/* Intel P4 (Prescott) */
#define MSR_PRP4_LBSTK_TOS      0x1da
#define MSR_PRP4_LBSTK_FROM_0   0x680
#define MSR_PRP4_LBSTK_FROM_1   0x681
#define MSR_PRP4_LBSTK_FROM_2   0x682
#define MSR_PRP4_LBSTK_FROM_3   0x683
#define MSR_PRP4_LBSTK_FROM_4   0x684
#define MSR_PRP4_LBSTK_FROM_5   0x685
#define MSR_PRP4_LBSTK_FROM_6   0x686
#define MSR_PRP4_LBSTK_FROM_7   0x687
#define MSR_PRP4_LBSTK_FROM_8   0x688
#define MSR_PRP4_LBSTK_FROM_9   0x689
#define MSR_PRP4_LBSTK_FROM_10  0x68a
#define MSR_PRP4_LBSTK_FROM_11  0x68b
#define MSR_PRP4_LBSTK_FROM_12  0x68c
#define MSR_PRP4_LBSTK_FROM_13  0x68d
#define MSR_PRP4_LBSTK_FROM_14  0x68e
#define MSR_PRP4_LBSTK_FROM_15  0x68f
#define MSR_PRP4_LBSTK_TO_0     0x6c0
#define MSR_PRP4_LBSTK_TO_1     0x6c1
#define MSR_PRP4_LBSTK_TO_2     0x6c2
#define MSR_PRP4_LBSTK_TO_3     0x6c3
#define MSR_PRP4_LBSTK_TO_4     0x6c4
#define MSR_PRP4_LBSTK_TO_5     0x6c5
#define MSR_PRP4_LBSTK_TO_6     0x6c6
#define MSR_PRP4_LBSTK_TO_7     0x6c7
#define MSR_PRP4_LBSTK_TO_8     0x6c8
#define MSR_PRP4_LBSTK_TO_9     0x6c9
#define MSR_PRP4_LBSTK_TO_10    0x6ca
#define MSR_PRP4_LBSTK_TO_11    0x6cb
#define MSR_PRP4_LBSTK_TO_12    0x6cc
#define MSR_PRP4_LBSTK_TO_13    0x6cd
#define MSR_PRP4_LBSTK_TO_14    0x6ce
#define MSR_PRP4_LBSTK_TO_15    0x6cf

#define MCI_CTL_VALUE           0xffffffff

#define MTRR_TYPE_UC            0
#define MTRR_TYPE_WC            1
#define MTRR_TYPE_WT            4
#define MTRR_TYPE_WP            5
#define MTRR_TYPE_WB            6
#define MTRR_TYPE_UC_           7

/*
 * For Solaris we set up the page attritubute table in the following way:
 * PAT0 Write-Back
 * PAT1 Write-Through
 * PAT2 Unchacheable-
 * PAT3 Uncacheable
 * PAT4 Write-Back
 * PAT5 Write-Through
 * PAT6 Write-Combine
```



```
 * PAT7 Uncacheable
 * The only difference from h/w default is entry 6.
 */
#define PAT_DEFAULT_ATTRIBUTE                   \
        ((uint64_t)MTRR_TYPE_WB |               \
        ((uint64_t)MTRR_TYPE_WT << 8) |         \
        ((uint64_t)MTRR_TYPE_UC_ << 16) |       \
        ((uint64_t)MTRR_TYPE_UC << 24) |        \
        ((uint64_t)MTRR_TYPE_WB << 32) |        \
        ((uint64_t)MTRR_TYPE_WT << 40) |        \
        ((uint64_t)MTRR_TYPE_WC << 48) |        \
        ((uint64_t)MTRR_TYPE_UC << 56))

#define X86FSET_LARGEPAGE       0
#define X86FSET_TSC             1
#define X86FSET_MSR             2
#define X86FSET_MTRR            3
#define X86FSET_PGE             4
#define X86FSET_DE              5
#define X86FSET_CMOV            6
#define X86FSET_MMX             7
#define X86FSET_MCA             8
#define X86FSET_PAE             9
#define X86FSET_CX8             10
#define X86FSET_PAT             11
#define X86FSET_SEP             12
#define X86FSET_SSE             13
#define X86FSET_SSE2            14
#define X86FSET_HTT             15
#define X86FSET_ASYSC           16
#define X86FSET_NX              17
#define X86FSET_SSE3            18
#define X86FSET_CX16            19
#define X86FSET_CMP             20
#define X86FSET_TSCP            21
#define X86FSET_MWAIT           22
#define X86FSET_SSE4A           23
#define X86FSET_CPUID           24
#define X86FSET_SSSE3           25
#define X86FSET_SSE4_1          26
#define X86FSET_SSE4_2          27
#define X86FSET_1GPG            28
#define X86FSET_CLFSH           29
#define X86FSET_64              30
#define X86FSET_AES             31
#define X86FSET_PCLMULQDQ       32
#define X86FSET_XSAVE           33
#define X86FSET_AVX             34
#define X86FSET_VMX             35
#define X86FSET_SVM             36
#define X86FSET_TOPOEXT         37

/*
 * flags to patch tsc_read routine.
 */
#define X86_NO_TSC              0x0
#define X86_HAVE_TSCP           0x1
#define X86_TSC_MFENCE          0x2
#define X86_TSC_LFENCE          0x4

/*
 * Intel Deep C-State invariant TSC in leaf 0x80000007.
 */
#define CPUID_TSC_CSTATE_INVARIANCE     (0x100)

/*
```

```
 391  * Intel Deep C-state always-running local APIC timer
 392  */
 393 #define CPUID_CSTATE_ARAT       (0x4)

 395 /*
 396  * Intel ENERGY_PERF_BIAS MSR indicated by feature bit CPUID.6.ECX[3].
 397  */
 398 #define CPUID_EPB_SUPPORT       (1 << 3)

 400 /*
 401  * Intel TSC deadline timer
 402  */
 403 #define CPUID_DEADLINE_TSC      (1 << 24)

 405 /*
 406  * x86_type is a legacy concept; this is supplanted
 407  * for most purposes by x86_featureset; modern CPUs
 408  * should be X86_TYPE_OTHER
 409  */
 410 #define X86_TYPE_OTHER          0
 411 #define X86_TYPE_486            1
 412 #define X86_TYPE_P5             2
 413 #define X86_TYPE_P6             3
 414 #define X86_TYPE_CYRIX_486      4
 415 #define X86_TYPE_CYRIX_6x86L    5
 416 #define X86_TYPE_CYRIX_6x86     6
 417 #define X86_TYPE_CYRIX_GXm      7
 418 #define X86_TYPE_CYRIX_6x86MX   8
 419 #define X86_TYPE_CYRIX_MediaGX  9
 420 #define X86_TYPE_CYRIX_MII      10
 421 #define X86_TYPE_VIA_CYRIX_III  11
 422 #define X86_TYPE_P4             12

 424 /*
 425  * x86_vendor allows us to select between
 426  * implementation features and helps guide
 427  * the interpretation of the cpuid instruction.
 428  */
 429 #define X86_VENDOR_Intel        0
 430 #define X86_VENDORSTR_Intel     "GenuineIntel"

 432 #define X86_VENDOR_IntelClone   1

 434 #define X86_VENDOR_AMD          2
 435 #define X86_VENDORSTR_AMD       "AuthenticAMD"

 437 #define X86_VENDOR_Cyrix        3
 438 #define X86_VENDORSTR_CYRIX     "CyrixInstead"

 440 #define X86_VENDOR_UMC          4
 441 #define X86_VENDORSTR_UMC       "UMC UMC UMC "

 443 #define X86_VENDOR_NexGen       5
 444 #define X86_VENDORSTR_NexGen    "NexGenDriven"

 446 #define X86_VENDOR_Centaur      6
 447 #define X86_VENDORSTR_Centaur   "CentaurHauls"

 449 #define X86_VENDOR_Rise         7
 450 #define X86_VENDORSTR_Rise      "RiseRiseRise"

 452 #define X86_VENDOR_SiS          8
 453 #define X86_VENDORSTR_SiS       "SiS SiS SiS "

 455 #define X86_VENDOR_TM           9
 456 #define X86_VENDORSTR_TM        "GenuineTMx86"
```



```
 458 #define X86_VENDOR_NSC          10
 459 #define X86_VENDORSTR_NSC       "Geode by NSC"

 461 /*
 462  * Vendor string max len + \0
 463  */
 464 #define X86_VENDOR_STRLEN       13

 466 /*
 467  * Some vendor/family/model/stepping ranges are commonly grouped under
 468  * a single identifying banner by the vendor.  The following encode
 469  * that "revision" in a uint32_t with the 8 most significant bits
 470  * identifying the vendor with X86_VENDOR_*, the next 8 identifying the
 471  * family, and the remaining 16 typically forming a bitmask of revisions
 472  * within that family with more significant bits indicating "later" revisions.
 473  */

 475 #define _X86_CHIPREV_VENDOR_MASK        0xff000000u
 476 #define _X86_CHIPREV_VENDOR_SHIFT       24
 477 #define _X86_CHIPREV_FAMILY_MASK        0x00ff0000u
 478 #define _X86_CHIPREV_FAMILY_SHIFT       16
 479 #define _X86_CHIPREV_REV_MASK           0x0000ffffu

 481 #define _X86_CHIPREV_VENDOR(x) \
 482         (((x) & _X86_CHIPREV_VENDOR_MASK) >> _X86_CHIPREV_VENDOR_SHIFT)
 483 #define _X86_CHIPREV_FAMILY(x) \
 484         (((x) & _X86_CHIPREV_FAMILY_MASK) >> _X86_CHIPREV_FAMILY_SHIFT)
 485 #define _X86_CHIPREV_REV(x) \
 486         ((x) & _X86_CHIPREV_REV_MASK)

 488 /* True if x matches in vendor and family and if x matches the given rev mask */
 489 #define X86_CHIPREV_MATCH(x, mask) \
 490         (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(mask) && \
 491         _X86_CHIPREV_FAMILY(x) == _X86_CHIPREV_FAMILY(mask) && \
 492         ((_X86_CHIPREV_REV(x) & _X86_CHIPREV_REV(mask)) != 0))

 494 /* True if x matches in vendor and family, and rev is at least minx */
 495 #define X86_CHIPREV_ATLEAST(x, minx) \
 496         (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(minx) && \
 497         _X86_CHIPREV_FAMILY(x) == _X86_CHIPREV_FAMILY(minx) && \
 498         _X86_CHIPREV_REV(x) >= _X86_CHIPREV_REV(minx))

 500 #define _X86_CHIPREV_MKREV(vendor, family, rev) \
 501         ((uint32_t)(vendor) << _X86_CHIPREV_VENDOR_SHIFT | \
 502         (family) << _X86_CHIPREV_FAMILY_SHIFT | (rev))

 504 /* True if x matches in vendor, and family is at least minx */
 505 #define X86_CHIPFAM_ATLEAST(x, minx) \
 506         (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(minx) && \
 507         _X86_CHIPREV_FAMILY(x) >= _X86_CHIPREV_FAMILY(minx))

 509 /* Revision default */
 510 #define X86_CHIPREV_UNKNOWN     0x0

 512 /*
 513  * Definitions for AMD Family 0xf. Minor revisions C0 and CG are
 514  * sufficiently different that we will distinguish them; in all other
 515  * case we will identify the major revision.
 516  */
 517 #define X86_CHIPREV_AMD_F_REV_B _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0001)
 518 #define X86_CHIPREV_AMD_F_REV_C0 _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0002)
 519 #define X86_CHIPREV_AMD_F_REV_CG _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0004)
 520 #define X86_CHIPREV_AMD_F_REV_D _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0008)
 521 #define X86_CHIPREV_AMD_F_REV_E _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0010)
 522 #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 \
#define X86_CHIPREV_AMD_10_REV_C \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0004)
#define X86_CHIPREV_AMD_10_REV_C3 \
#define X86_CHIPREV_AMD_10_REV_D \
        _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)
#define X86_CHIPREV_AMD_11 \
        _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x11, 0x0001)

/*
 * 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
#define _X86_SOCKET_TYPE(x)             ((x) & _X86_SOCKET_TYPE_MASK)

#define _X86_SOCKET_MKVAL(vendor, bitval) \
        ((uint32_t)(vendor) << _X86_SOCKET_VENDOR_SHIFT | (bitval))
```



```
#define X86_SOCKET_MATCH(s, mask) \
        (_X86_SOCKET_VENDOR(s) == _X86_SOCKET_VENDOR(mask) && \
        (_X86_SOCKET_TYPE(s) & _X86_SOCKET_TYPE(mask)) != 0)

#define X86_SOCKET_UNKNOWN 0x0
        /*
         * AMD socket types
         */
#define X86_SOCKET_754          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000001)
#define X86_SOCKET_939          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000002)
#define X86_SOCKET_940          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000004)
#define X86_SOCKET_S1g1         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000008)
#define X86_SOCKET_AM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000010)
#define X86_SOCKET_F1207        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000020)
#define X86_SOCKET_S1g2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000040)
#define X86_SOCKET_S1g3         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000080)
#define X86_SOCKET_AM           _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000100)
#define X86_SOCKET_AM2R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000200)
#define X86_SOCKET_AM3          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000400)
#define X86_SOCKET_G34          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000800)
#define X86_SOCKET_ASB2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x001000)
#define X86_SOCKET_C32          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x002000)
#define X86_SOCKET_S1g4         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x004000)
#define X86_SOCKET_FT1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x008000)
#define X86_SOCKET_FM1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x010000)
#define X86_SOCKET_FS1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x020000)
#define X86_SOCKET_AM3R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x040000)
#define X86_SOCKET_FP2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x080000)
#define X86_SOCKET_FS1R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x100000)
#define X86_SOCKET_FM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x200000)

/*
 * xgetbv/xsetbv support
 */

#define XFEATURE_ENABLED_MASK   0x0
/*
 * XFEATURE_ENABLED_MASK values (eax)
 */
#define XFEATURE_LEGACY_FP      0x1
#define XFEATURE_SSE            0x2
#define XFEATURE_AVX            0x4
#define XFEATURE_MAX            XFEATURE_AVX
#define XFEATURE_FP_ALL         (XFEATURE_LEGACY_FP|XFEATURE_SSE|XFEATURE_AVX)

#if !defined(_ASM)

#if defined(_KERNEL) || defined(_KMEMUSER)

#define NUM_X86_FEATURES        38
extern uchar_t x86_featureset[];

extern void free_x86_featureset(void *featureset);
extern boolean_t is_x86_feature(void *featureset, uint_t feature);
extern void add_x86_feature(void *featureset, uint_t feature);
extern void remove_x86_feature(void *featureset, uint_t feature);
extern boolean_t compare_x86_featureset(void *setA, void *setB);
extern void print_x86_featureset(void *featureset);


extern uint_t x86_type;
extern uint_t x86_vendor;
extern uint_t x86_clflush_size;

extern uint_t pentiumpro_bug4046376;
```

```
 651 extern uint_t pentiumpro_bug4064495;

 653 extern uint_t enable486;

 655 extern const char CyrixInstead[];

 657 #endif

 659 #if defined(_KERNEL)

 661 /*
 662  * This structure is used to pass arguments and get return values back
 663  * from the CPUID instruction in __cpuid_insn() routine.
 664  */
 665 struct cpuid_regs {
 666         uint32_t        cp_eax;
 667         uint32_t        cp_ebx;
 668         uint32_t        cp_ecx;
 669         uint32_t        cp_edx;
 670 };
```

_____***unchanged_portion_omitted_***