```
**********************************************************
  143409 Thu Jan 10 17:07:56 2019
new/usr/src/uts/i86pc/os/cpuid.c
10208 Add x86 features for L1TF
**********************************************************
```

```
   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 (c) 2011, 2016 by Delphix. All rights reserved.
  24  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
  25  * Copyright 2014 Josef "Jeff" Sipek <jeffpc@josefsipek.net>
  26  */
  27 /*
  28  * Copyright (c) 2010, Intel Corporation.
  29  * All rights reserved.
  30  */
  31 /*
  32  * Portions Copyright 2009 Advanced Micro Devices, Inc.
  33  */
  34 /*
  35  * Copyright (c) 2019, Joyent, Inc.
  35  * Copyright 2018 Joyent, Inc.
  36  */
  37 /*
  38  * Various routines to handle identification
  39  * and classification of x86 processors.
  40  */

  42 #include <sys/types.h>
  43 #include <sys/archsystm.h>
  44 #include <sys/x86_archext.h>
  45 #include <sys/kmem.h>
  46 #include <sys/systm.h>
  47 #include <sys/cmn_err.h>
  48 #include <sys/sunddi.h>
  49 #include <sys/sunndi.h>
  50 #include <sys/cpuvar.h>
  51 #include <sys/processor.h>
  52 #include <sys/sysmacros.h>
  53 #include <sys/pg.h>
  54 #include <sys/fp.h>
  55 #include <sys/controlregs.h>
  56 #include <sys/bitmap.h>
  57 #include <sys/auxv_386.h>
  58 #include <sys/memnode.h>
  59 #include <sys/pci_cfgspace.h>
  60 #include <sys/comm_page.h>
```



```
  61 #include <sys/mach_mmu.h>
  62 #include <sys/ucode.h>
  63 #include <sys/tsc.h>

  65 #ifdef __xpv
  66 #include <sys/hypervisor.h>
  67 #else
  68 #include <sys/ontrap.h>
  69 #endif

  71 /*
  72  * Pass 0 of cpuid feature analysis happens in locore. It contains special code
  73  * to recognize Cyrix processors that are not cpuid-compliant, and to deal with
  74  * them accordingly. For most modern processors, feature detection occurs here
  75  * in pass 1.
  76  *
  77  * Pass 1 of cpuid feature analysis happens just at the beginning of mlsetup()
  78  * for the boot CPU and does the basic analysis that the early kernel needs.
  79  * x86_featureset is set based on the return value of cpuid_pass1() of the boot
  80  * CPU.
  81  *
  82  * Pass 1 includes:
  83  *
  84  *      o Determining vendor/model/family/stepping and setting x86_type and
  85  *        x86_vendor accordingly.
  86  *      o Processing the feature flags returned by the cpuid instruction while
  87  *        applying any workarounds or tricks for the specific processor.
  88  *      o Mapping the feature flags into illumos feature bits (X86_*).
  89  *      o Processing extended feature flags if supported by the processor,
  90  *        again while applying specific processor knowledge.
  91  *      o Determining the CMT characteristics of the system.
  92  *
  93  * Pass 1 is done on non-boot CPUs during their initialization and the results
  94  * are used only as a meager attempt at ensuring that all processors within the
  95  * system support the same features.
  96  *
  97  * Pass 2 of cpuid feature analysis happens just at the beginning
  98  * of startup().  It just copies in and corrects the remainder
  99  * of the cpuid data we depend on: standard cpuid functions that we didn't
 100  * need for pass1 feature analysis, and extended cpuid functions beyond the
 101  * simple feature processing done in pass1.
 102  *
 103  * Pass 3 of cpuid analysis is invoked after basic kernel services; in
 104  * particular kernel memory allocation has been made available. It creates a
 105  * readable brand string based on the data collected in the first two passes.
 106  *
 107  * Pass 4 of cpuid analysis is invoked after post_startup() when all
 108  * the support infrastructure for various hardware features has been
 109  * initialized. It determines which processor features will be reported
 110  * to userland via the aux vector.
 111  *
 112  * All passes are executed on all CPUs, but only the boot CPU determines what
 113  * features the kernel will use.
 114  *
 115  * Much of the worst junk in this file is for the support of processors
 116  * that didn't really implement the cpuid instruction properly.
 117  *
 118  * NOTE: The accessor functions (cpuid_get*) are aware of, and ASSERT upon,
 119  * the pass numbers.  Accordingly, changes to the pass code may require changes
 120  * to the accessor code.
 121  */

 123 uint_t x86_vendor = X86_VENDOR_IntelClone;
 124 uint_t x86_type = X86_TYPE_OTHER;
 125 uint_t x86_clflush_size = 0;
```

```
 127 #if defined(__xpv)
 128 int x86_use_pcid = 0;
 129 int x86_use_invpcid = 0;
 130 #else
 131 int x86_use_pcid = -1;
 132 int x86_use_invpcid = -1;
 133 #endif

 135 uint_t pentiumpro_bug4046376;

 137 uchar_t x86_featureset[BT_SIZEOFMAP(NUM_X86_FEATURES)];

 139 static char *x86_feature_names[NUM_X86_FEATURES] = {
 140         "lgpg",
 141         "tsc",
 142         "msr",
 143         "mtrr",
 144         "pge",
 145         "de",
 146         "cmov",
 147         "mmx",
 148         "mca",
 149         "pae",
 150         "cv8",
 151         "pat",
 152         "sep",
 153         "sse",
 154         "sse2",
 155         "htt",
 156         "asysc",
 157         "nx",
 158         "sse3",
 159         "cx16",
 160         "cmp",
 161         "tscp",
 162         "mwait",
 163         "sse4a",
 164         "cpuid",
 165         "ssse3",
 166         "sse4_1",
 167         "sse4_2",
 168         "1gpg",
 169         "clfsh",
 170         "64",
 171         "aes",
 172         "pclmulqdq",
 173         "xsave",
 174         "avx",
 175         "vmx",
 176         "svm",
 177         "topoext",
 178         "f16c",
 179         "rdrand",
 180         "x2apic",
 181         "avx2",
 182         "bmi1",
 183         "bmi2",
 184         "fma",
 185         "smep",
 186         "smap",
 187         "adx",
 188         "rdseed",
 189         "mpx",
 190         "avx512f",
 191         "avx512dq",
 192         "avx512pf",
```



```
 193         "avx512er",
 194         "avx512cd",
 195         "avx512bw",
 196         "avx512vl",
 197         "avx512fma",
 198         "avx512vbmi",
 199         "avx512_vpopcntdq",
 200         "avx512_4vnniw",
 201         "avx512_4fmaps",
 202         "xsaveopt",
 203         "xsavec",
 204         "xsaves",
 205         "sha",
 206         "umip",
 207         "pku",
 208         "ospke",
 209         "pcid",
 210         "invpcid",
 211         "ibrs",
 212         "ibpb",
 213         "stibp",
 214         "ssbd",
 215         "ssbd_virt",
 216         "rdcl_no",
 217         "ibrs_all",
 218         "rsba",
 219         "ssb_no",
 220         "stibp_all",
 221         "flush_cmd",
 222         "l1d_vmentry_no"
 220         "stibp_all"
 223 };
______unchanged_portion_omitted_

 990 static void
 991 cpuid_scan_security(cpu_t *cpu, uchar_t *featureset)
 992 {
 993         struct cpuid_info *cpi = cpu->cpu_m.mcpu_cpi;

 995         if (cpi->cpi_vendor == X86_VENDOR_AMD &&
 996             cpi->cpi_xmaxeax >= 0x80000008) {
 997                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_IBPB)
 998                         add_x86_feature(featureset, X86FSET_IBPB);
 999                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_IBRS)
1000                         add_x86_feature(featureset, X86FSET_IBRS);
1001                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_STIBP)
1002                         add_x86_feature(featureset, X86FSET_STIBP);
1003                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_IBRS_ALL)
1004                         add_x86_feature(featureset, X86FSET_IBRS_ALL);
1005                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_STIBP_ALL)
1006                         add_x86_feature(featureset, X86FSET_STIBP_ALL);
1007                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_PREFER_IBRS)
1008                         add_x86_feature(featureset, X86FSET_RSBA);
1009                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_SSBD)
1010                         add_x86_feature(featureset, X86FSET_SSBD);
1011                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_VIRT_SSBD)
1012                         add_x86_feature(featureset, X86FSET_SSBD_VIRT);
1013                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_SSB_NO)
1014                         add_x86_feature(featureset, X86FSET_SSB_NO);
1015         } else if (cpi->cpi_vendor == X86_VENDOR_Intel &&
1016             cpi->cpi_maxeax >= 7) {
1017                 struct cpuid_regs *ecp;
1018                 ecp = &cpi->cpi_std[7];

1020                 if (ecp->cp_edx & CPUID_INTC_EDX_7_0_SPEC_CTRL) {
1021                         add_x86_feature(featureset, X86FSET_IBRS);
```

```
1022                        add_x86_feature(featureset, X86FSET_IBPB);
1023                }

1025                if (ecp->cp_edx & CPUID_INTC_EDX_7_0_STIBP) {
1026                        add_x86_feature(featureset, X86FSET_STIBP);
1027                }

1029                /*
1030                 * Don't read the arch caps MSR on xpv where we lack the
1031                 * on_trap().
1032                 */
1033 #ifndef __xpv
1034                if (ecp->cp_edx & CPUID_INTC_EDX_7_0_ARCH_CAPS) {
1035                        on_trap_data_t otd;

1037                        /*
1038                         * Be paranoid and assume we'll get a #GP.
1039                         */
1040                        if (!on_trap(&otd, OT_DATA_ACCESS)) {
1041                                uint64_t reg;

1043                                reg = rdmsr(MSR_IA32_ARCH_CAPABILITIES);
1044                                if (reg & IA32_ARCH_CAP_RDCL_NO) {
1045                                        add_x86_feature(featureset,
1046                                            X86FSET_RDCL_NO);
1047                                }
1048                                if (reg & IA32_ARCH_CAP_IBRS_ALL) {
1049                                        add_x86_feature(featureset,
1050                                            X86FSET_IBRS_ALL);
1051                                }
1052                                if (reg & IA32_ARCH_CAP_RSBA) {
1053                                        add_x86_feature(featureset,
1054                                            X86FSET_RSBA);
1055                                }
1056                                if (reg & IA32_ARCH_CAP_SKIP_L1DFL_VMENTRY) {
1057                                        add_x86_feature(featureset,
1058                                            X86FSET_L1D_VM_NO);
1059                                }
1060                                if (reg & IA32_ARCH_CAP_SSB_NO) {
1061                                        add_x86_feature(featureset,
1062                                            X86FSET_SSB_NO);
1063                                }
1064                        }
1065                        no_trap();
1066                }
1067 #endif  /* !__xpv */

1069                if (ecp->cp_edx & CPUID_INTC_EDX_7_0_SSBD)
1070                        add_x86_feature(featureset, X86FSET_SSBD);

1072                if (ecp->cp_edx & CPUID_INTC_EDX_7_0_FLUSH_CMD)
1073                        add_x86_feature(featureset, X86FSET_FLUSH_CMD);
1074        }
1075 }
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   35207 Thu Jan 10 17:07:56 2019
new/usr/src/uts/intel/sys/x86_archext.h
10208 Add x86 features for L1TF
**********************************************************
```

```
   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) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
  23  * Copyright (c) 2011 by Delphix. All rights reserved.
  24  */
  25 /*
  26  * Copyright (c) 2010, Intel Corporation.
  27  * All rights reserved.
  28  */
  29 /*
  30  * Copyright (c) 2019, Joyent, Inc.
  30  * Copyright 2018 Joyent, Inc.
  31  * Copyright 2012 Jens Elkner <jel+illumos@cs.uni-magdeburg.de>
  32  * Copyright 2012 Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
  33  * Copyright 2014 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
  34  * Copyright 2018 Nexenta Systems, Inc.
  35  */

  37 #ifndef _SYS_X86_ARCHEXT_H
  38 #define _SYS_X86_ARCHEXT_H

  40 #if !defined(_ASM)
  41 #include <sys/regset.h>
  42 #include <sys/processor.h>
  43 #include <vm/seg_enum.h>
  44 #include <vm/page.h>
  45 #endif  /* _ASM */

  47 #ifdef  __cplusplus
  48 extern "C" {
  49 #endif

  51 /*
  52  * cpuid instruction feature flags in %edx (standard function 1)
  53  */

  55 #define CPUID_INTC_EDX_FPU      0x00000001      /* x87 fpu present */
  56 #define CPUID_INTC_EDX_VME      0x00000002      /* virtual-8086 extension */
  57 #define CPUID_INTC_EDX_DE       0x00000004      /* debugging extensions */
  58 #define CPUID_INTC_EDX_PSE      0x00000008      /* page size extension */
  59 #define CPUID_INTC_EDX_TSC      0x00000010      /* time stamp counter */
  60 #define CPUID_INTC_EDX_MSR      0x00000020      /* rdmsr and wrmsr */
```



```
  61 #define CPUID_INTC_EDX_PAE      0x00000040      /* physical addr extension */
  62 #define CPUID_INTC_EDX_MCE      0x00000080      /* machine check exception */
  63 #define CPUID_INTC_EDX_CX8      0x00000100      /* cmpxchg8b instruction */
  64 #define CPUID_INTC_EDX_APIC     0x00000200      /* local APIC */
  65                                                 /* 0x400 - reserved */
  66 #define CPUID_INTC_EDX_SEP      0x00000800      /* sysenter and sysexit */
  67 #define CPUID_INTC_EDX_MTRR     0x00001000      /* memory type range reg */
  68 #define CPUID_INTC_EDX_PGE      0x00002000      /* page global enable */
  69 #define CPUID_INTC_EDX_MCA      0x00004000      /* machine check arch */
  70 #define CPUID_INTC_EDX_CMOV     0x00008000      /* conditional move insns */
  71 #define CPUID_INTC_EDX_PAT      0x00010000      /* page attribute table */
  72 #define CPUID_INTC_EDX_PSE36    0x00020000      /* 36-bit pagesize extension */
  73 #define CPUID_INTC_EDX_PSN      0x00040000      /* processor serial number */
  74 #define CPUID_INTC_EDX_CLFSH    0x00080000      /* clflush instruction */
  75                                                 /* 0x100000 - reserved */
  76 #define CPUID_INTC_EDX_DS       0x00200000      /* debug store exists */
  77 #define CPUID_INTC_EDX_ACPI     0x00400000      /* monitoring + clock ctrl */
  78 #define CPUID_INTC_EDX_MMX      0x00800000      /* MMX instructions */
  79 #define CPUID_INTC_EDX_FXSR     0x01000000      /* fxsave and fxrstor */
  80 #define CPUID_INTC_EDX_SSE      0x02000000      /* streaming SIMD extensions */
  81 #define CPUID_INTC_EDX_SSE2     0x04000000      /* SSE extensions */
  82 #define CPUID_INTC_EDX_SS       0x08000000      /* self-snoop */
  83 #define CPUID_INTC_EDX_HTT      0x10000000      /* Hyper Thread Technology */
  84 #define CPUID_INTC_EDX_TM       0x20000000      /* thermal monitoring */
  85 #define CPUID_INTC_EDX_IA64     0x40000000      /* Itanium emulating IA32 */
  86 #define CPUID_INTC_EDX_PBE      0x80000000      /* Pending Break Enable */

  88 /*
  89  * cpuid instruction feature flags in %ecx (standard function 1)
  90  */

  92 #define CPUID_INTC_ECX_SSE3     0x00000001      /* Yet more SSE extensions */
  93 #define CPUID_INTC_ECX_PCLMULQDQ 0x00000002     /* PCLMULQDQ insn */
  94 #define CPUID_INTC_ECX_DTES64   0x00000004      /* 64-bit DS area */
  95 #define CPUID_INTC_ECX_MON      0x00000008      /* MONITOR/MWAIT */
  96 #define CPUID_INTC_ECX_DSCPL    0x00000010      /* CPL-qualified debug store */
  97 #define CPUID_INTC_ECX_VMX      0x00000020      /* Hardware VM extensions */
  98 #define CPUID_INTC_ECX_SMX      0x00000040      /* Secure mode extensions */
  99 #define CPUID_INTC_ECX_EST      0x00000080      /* enhanced SpeedStep */
 100 #define CPUID_INTC_ECX_TM2      0x00000100      /* thermal monitoring */
 101 #define CPUID_INTC_ECX_SSSE3    0x00000200      /* Supplemental SSE3 insns */
 102 #define CPUID_INTC_ECX_CID      0x00000400      /* L1 context ID */
 103                                                 /* 0x00000800 - reserved */
 104 #define CPUID_INTC_ECX_FMA      0x00001000      /* Fused Multiply Add */
 105 #define CPUID_INTC_ECX_CX16     0x00002000      /* cmpxchg16 */
 106 #define CPUID_INTC_ECX_ETPRD    0x00004000      /* extended task pri messages */
 107 #define CPUID_INTC_ECX_PDCM     0x00008000      /* Perf/Debug Capability MSR */
 108                                                 /* 0x00010000 - reserved */
 109 #define CPUID_INTC_ECX_PCID     0x00020000      /* process-context ids */
 110 #define CPUID_INTC_ECX_DCA      0x00040000      /* direct cache access */
 111 #define CPUID_INTC_ECX_SSE4_1   0x00080000      /* SSE4.1 insns */
 112 #define CPUID_INTC_ECX_SSE4_2   0x00100000      /* SSE4.2 insns */
 113 #define CPUID_INTC_ECX_X2APIC   0x00200000      /* x2APIC */
 114 #define CPUID_INTC_ECX_MOVBE    0x00400000      /* MOVBE insn */
 115 #define CPUID_INTC_ECX_POPCNT   0x00800000      /* POPCNT insn */
 116 #define CPUID_INTC_ECX_TSCDL    0x01000000      /* Deadline TSC */
 117 #define CPUID_INTC_ECX_AES      0x02000000      /* AES insns */
 118 #define CPUID_INTC_ECX_XSAVE    0x04000000      /* XSAVE/XRESTOR insns */
 119 #define CPUID_INTC_ECX_OSXSAVE  0x08000000      /* OS supports XSAVE insns */
 120 #define CPUID_INTC_ECX_AVX      0x10000000      /* AVX supported */
 121 #define CPUID_INTC_ECX_F16C     0x20000000      /* F16C supported */
 122 #define CPUID_INTC_ECX_RDRAND   0x40000000      /* RDRAND supported */
 123 #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 */
 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      0x000000004 /* AMD: FP Err. Ptr. Zero */
 191 #define CPUID_AMD_EBX_IBPB              0x000001000 /* AMD: IBPB */
 192 #define CPUID_AMD_EBX_IBRS              0x000004000 /* AMD: IBRS */
```



```
 193 #define CPUID_AMD_EBX_STIBP             0x000008000 /* AMD: STIBP */
 194 #define CPUID_AMD_EBX_IBRS_ALL          0x000010000 /* AMD: Enhanced IBRS */
 195 #define CPUID_AMD_EBX_STIBP_ALL         0x000020000 /* AMD: STIBP ALL */
 196 #define CPUID_AMD_EBX_PREFER_IBRS       0x000040000 /* AMD: Don't retpoline */
 197 #define CPUID_AMD_EBX_SSBD              0x001000000 /* AMD: SSBD */
 198 #define CPUID_AMD_EBX_VIRT_SSBD         0x002000000 /* AMD: VIRT SSBD */
 199 #define CPUID_AMD_EBX_SSB_NO            0x004000000 /* AMD: SSB Fixed */

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

 210 /*
 211  * Intel also uses cpuid leaf 7 to have additional instructions and features.
 212  * Like some other leaves, but unlike the current ones we care about, it
 213  * requires us to specify both a leaf in %eax and a sub-leaf in %ecx. To deal
 214  * with the potential use of additional sub-leaves in the future, we now
 215  * specifically label the EBX features with their leaf and sub-leaf.
 216  */
 217 #define CPUID_INTC_EBX_7_0_BMI1         0x00000008      /* BMI1 instrs */
 218 #define CPUID_INTC_EBX_7_0_HLE          0x00000010      /* HLE */
 219 #define CPUID_INTC_EBX_7_0_AVX2         0x00000020      /* AVX2 supported */
 220 #define CPUID_INTC_EBX_7_0_SMEP         0x00000080      /* SMEP in CR4 */
 221 #define CPUID_INTC_EBX_7_0_BMI2         0x00000100      /* BMI2 instrs */
 222 #define CPUID_INTC_EBX_7_0_INVPCID      0x00000400      /* invpcid instr */
 223 #define CPUID_INTC_EBX_7_0_MPX          0x00004000      /* Mem. Prot. Ext. */
 224 #define CPUID_INTC_EBX_7_0_AVX512F      0x00010000      /* AVX512 foundation */
 225 #define CPUID_INTC_EBX_7_0_AVX512DQ     0x00020000      /* AVX512DQ */
 226 #define CPUID_INTC_EBX_7_0_RDSEED       0x00040000      /* RDSEED instr */
 227 #define CPUID_INTC_EBX_7_0_ADX          0x00080000      /* ADX instrs */
 228 #define CPUID_INTC_EBX_7_0_SMAP         0x00100000      /* SMAP in CR 4 */
 229 #define CPUID_INTC_EBX_7_0_AVX512IFMA   0x00200000      /* AVX512IFMA */
 230 #define CPUID_INTC_EBX_7_0_CLWB         0x01000000      /* CLWB */
 231 #define CPUID_INTC_EBX_7_0_AVX512PF     0x04000000      /* AVX512PF */
 232 #define CPUID_INTC_EBX_7_0_AVX512ER     0x08000000      /* AVX512ER */
 233 #define CPUID_INTC_EBX_7_0_AVX512CD     0x10000000      /* AVX512CD */
 234 #define CPUID_INTC_EBX_7_0_SHA          0x20000000      /* SHA extensions */
 235 #define CPUID_INTC_EBX_7_0_AVX512BW     0x40000000      /* AVX512BW */
 236 #define CPUID_INTC_EBX_7_0_AVX512VL     0x80000000      /* AVX512VL */

 238 #define CPUID_INTC_EBX_7_0_ALL_AVX512 \
 239         (CPUID_INTC_EBX_7_0_AVX512F | CPUID_INTC_EBX_7_0_AVX512DQ | \
 240         CPUID_INTC_EBX_7_0_AVX512IFMA | CPUID_INTC_EBX_7_0_AVX512PF | \
 241         CPUID_INTC_EBX_7_0_AVX512ER | CPUID_INTC_EBX_7_0_AVX512CD | \
 242         CPUID_INTC_EBX_7_0_AVX512BW | CPUID_INTC_EBX_7_0_AVX512VL)

 244 #define CPUID_INTC_ECX_7_0_AVX512VBMI   0x00000002      /* AVX512VBMI */
 245 #define CPUID_INTC_ECX_7_0_UMIP         0x00000004      /* UMIP */
 246 #define CPUID_INTC_ECX_7_0_PKU          0x00000008      /* umode prot. keys */
 247 #define CPUID_INTC_ECX_7_0_OSPKE        0x00000010      /* OSPKE */
 248 #define CPUID_INTC_ECX_7_0_AVX512VPOPCDQ 0x00004000     /* AVX512 VPOPCNTDQ */

 250 #define CPUID_INTC_ECX_7_0_ALL_AVX512 \
 251         (CPUID_INTC_ECX_7_0_AVX512VBMI | CPUID_INTC_ECX_7_0_AVX512VPOPCDQ)

 253 #define CPUID_INTC_EDX_7_0_AVX5124NNIW  0x00000004      /* AVX512 4NNIW */
 254 #define CPUID_INTC_EDX_7_0_AVX5124FMAPS 0x00000008      /* AVX512 4FMAPS */
 255 #define CPUID_INTC_EDX_7_0_SPEC_CTRL    0x04000000      /* Spec, IBPB, IBRS */
 256 #define CPUID_INTC_EDX_7_0_STIBP        0x08000000      /* STIBP */
 257 #define CPUID_INTC_EDX_7_0_FLUSH_CMD    0x10000000      /* IA32_FLUSH_CMD */
 258 #define CPUID_INTC_EDX_7_0_ARCH_CAPS    0x20000000      /* IA32_ARCH_CAPS */
```

```
#define CPUID_INTC_EDX_7_0_SSBD         0x80000000      /* SSBD */

#define CPUID_INTC_EDX_7_0_ALL_AVX512 \
        (CPUID_INTC_EDX_7_0_AVX5124NNIW | CPUID_INTC_EDX_7_0_AVX5124FMAPS)

/*
 * Intel also uses cpuid leaf 0xd to report additional instructions and features
 * when the sub-leaf in %ecx == 1. We label these using the same convention as
 * with leaf 7.
 */
#define CPUID_INTC_EAX_D_1_XSAVEOPT     0x00000001      /* xsaveopt inst. */
#define CPUID_INTC_EAX_D_1_XSAVEC       0x00000002      /* xsavec inst. */
#define CPUID_INTC_EAX_D_1_XSAVES       0x00000008      /* xsaves inst. */

#define REG_PAT                 0x277
#define REG_TSC                 0x10    /* timestamp counter */
#define REG_APIC_BASE_MSR       0x1b
#define REG_X2APIC_BASE_MSR     0x800   /* The MSR address offset of x2APIC */

#if !defined(__xpv)
/*
 * AMD C1E
 */
#define MSR_AMD_INT_PENDING_CMP_HALT    0xC0010055
#define AMD_ACTONCMPHALT_SHIFT  27
#define AMD_ACTONCMPHALT_MASK   3
#endif

#define MSR_DEBUGCTL            0x1d9

#define DEBUGCTL_LBR            0x01
#define DEBUGCTL_BTF            0x02

/* Intel P6, AMD */
#define MSR_LBR_FROM            0x1db
#define MSR_LBR_TO              0x1dc
#define MSR_LEX_FROM            0x1dd
#define MSR_LEX_TO              0x1de

/* Intel P4 (pre-Prescott, non P4 M) */
#define MSR_P4_LBSTK_TOS        0x1da
#define MSR_P4_LBSTK_0          0x1db
#define MSR_P4_LBSTK_1          0x1dc
#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

/*
 * Intel IA32_ARCH_CAPABILITIES MSR.
 */
#define MSR_IA32_ARCH_CAPABILITIES              0x10a
#define IA32_ARCH_CAP_RDCL_NO                   0x0001
#define IA32_ARCH_CAP_IBRS_ALL                  0x0002
#define IA32_ARCH_CAP_RSBA                      0x0004
#define IA32_ARCH_CAP_SKIP_L1DFL_VMENTRY        0x0008
#define IA32_ARCH_CAP_SSB_NO                    0x0010

/*
 * Intel Speculation related MSRs
 */
#define MSR_IA32_SPEC_CTRL      0x48
#define IA32_SPEC_CTRL_IBRS     0x01
#define IA32_SPEC_CTRL_STIBP    0x02
#define IA32_SPEC_CTRL_SSBD     0x04

#define MSR_IA32_PRED_CMD       0x49
#define IA32_PRED_CMD_IBPB      0x01

#define MSR_IA32_FLUSH_CMD      0x10b
#define IA32_FLUSH_CMD_L1D      0x01

#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
```

```
391  * PAT5 Write-Through
392  * PAT6 Write-Combine
393  * PAT7 Uncacheable
394  * The only difference from h/w default is entry 6.
395  */
396 #define PAT_DEFAULT_ATTRIBUTE                   \
397         ((uint64_t)MTRR_TYPE_WB |               \
398         ((uint64_t)MTRR_TYPE_WT << 8) |         \
399         ((uint64_t)MTRR_TYPE_UC_ << 16) |       \
400         ((uint64_t)MTRR_TYPE_UC << 24) |        \
401         ((uint64_t)MTRR_TYPE_WB << 32) |        \
402         ((uint64_t)MTRR_TYPE_WT << 40) |        \
403         ((uint64_t)MTRR_TYPE_WC << 48) |        \
404         ((uint64_t)MTRR_TYPE_UC << 56))

406 #define X86FSET_LARGEPAGE       0
407 #define X86FSET_TSC             1
408 #define X86FSET_MSR             2
409 #define X86FSET_MTRR            3
410 #define X86FSET_PGE             4
411 #define X86FSET_DE              5
412 #define X86FSET_CMOV            6
413 #define X86FSET_MMX             7
414 #define X86FSET_MCA             8
415 #define X86FSET_PAE             9
416 #define X86FSET_CX8             10
417 #define X86FSET_PAT             11
418 #define X86FSET_SEP             12
419 #define X86FSET_SSE             13
420 #define X86FSET_SSE2            14
421 #define X86FSET_HTT             15
422 #define X86FSET_ASYSC           16
423 #define X86FSET_NX              17
424 #define X86FSET_SSE3            18
425 #define X86FSET_CX16            19
426 #define X86FSET_CMP             20
427 #define X86FSET_TSCP            21
428 #define X86FSET_MWAIT           22
429 #define X86FSET_SSE4A           23
430 #define X86FSET_CPUID           24
431 #define X86FSET_SSSE3           25
432 #define X86FSET_SSE4_1          26
433 #define X86FSET_SSE4_2          27
434 #define X86FSET_1GPG            28
435 #define X86FSET_CLFSH           29
436 #define X86FSET_64              30
437 #define X86FSET_AES             31
438 #define X86FSET_PCLMULQDQ       32
439 #define X86FSET_XSAVE           33
440 #define X86FSET_AVX             34
441 #define X86FSET_VMX             35
442 #define X86FSET_SVM             36
443 #define X86FSET_TOPOEXT         37
444 #define X86FSET_F16C            38
445 #define X86FSET_RDRAND          39
446 #define X86FSET_X2APIC          40
447 #define X86FSET_AVX2            41
448 #define X86FSET_BMI1            42
449 #define X86FSET_BMI2            43
450 #define X86FSET_FMA             44
451 #define X86FSET_SMEP            45
452 #define X86FSET_SMAP            46
453 #define X86FSET_ADX             47
454 #define X86FSET_RDSEED          48
455 #define X86FSET_MPX             49
456 #define X86FSET_AVX512F         50
```



```
457 #define X86FSET_AVX512DQ        51
458 #define X86FSET_AVX512PF        52
459 #define X86FSET_AVX512ER        53
460 #define X86FSET_AVX512CD        54
461 #define X86FSET_AVX512BW        55
462 #define X86FSET_AVX512VL        56
463 #define X86FSET_AVX512FMA       57
464 #define X86FSET_AVX512VBMI      58
465 #define X86FSET_AVX512VPOPCDQ   59
466 #define X86FSET_AVX512NNIW      60
467 #define X86FSET_AVX512FMAPS     61
468 #define X86FSET_XSAVEOPT        62
469 #define X86FSET_XSAVEC          63
470 #define X86FSET_XSAVES          64
471 #define X86FSET_SHA             65
472 #define X86FSET_UMIP            66
473 #define X86FSET_PKU             67
474 #define X86FSET_OSPKE           68
475 #define X86FSET_PCID            69
476 #define X86FSET_INVPCID         70
477 #define X86FSET_IBRS            71
478 #define X86FSET_IBPB            72
479 #define X86FSET_STIBP           73
480 #define X86FSET_SSBD            74
481 #define X86FSET_SSBD_VIRT       75
482 #define X86FSET_RDCL_NO         76
483 #define X86FSET_IBRS_ALL        77
484 #define X86FSET_RSBA            78
485 #define X86FSET_SSB_NO          79
486 #define X86FSET_STIBP_ALL       80
487 #define X86FSET_FLUSH_CMD       81
488 #define X86FSET_L1D_VM_NO       82

490 /*
491  * Intel Deep C-State invariant TSC in leaf 0x80000007.
492  */
493 #define CPUID_TSC_CSTATE_INVARIANCE     (0x100)

495 /*
496  * Intel Deep C-state always-running local APIC timer
497  */
498 #define CPUID_CSTATE_ARAT       (0x4)

500 /*
501  * Intel ENERGY_PERF_BIAS MSR indicated by feature bit CPUID.6.ECX[3].
502  */
503 #define CPUID_EPB_SUPPORT       (1 << 3)

505 /*
506  * Intel TSC deadline timer
507  */
508 #define CPUID_DEADLINE_TSC      (1 << 24)

510 /*
511  * x86_type is a legacy concept; this is supplanted
512  * for most purposes by x86_featureset; modern CPUs
513  * should be X86_TYPE_OTHER
514  */
515 #define X86_TYPE_OTHER          0
516 #define X86_TYPE_486            1
517 #define X86_TYPE_P5             2
518 #define X86_TYPE_P6             3
519 #define X86_TYPE_CYRIX_486      4
520 #define X86_TYPE_CYRIX_6x86L    5
521 #define X86_TYPE_CYRIX_6x86     6
522 #define X86_TYPE_CYRIX_GXm      7
```

```
#define X86_TYPE_CYRIX_6x86MX   8
#define X86_TYPE_CYRIX_MediaGX  9
#define X86_TYPE_CYRIX_MII      10
#define X86_TYPE_VIA_CYRIX_III  11
#define X86_TYPE_P4             12

/*
 * x86_vendor allows us to select between
 * implementation features and helps guide
 * the interpretation of the cpuid instruction.
 */
#define X86_VENDOR_Intel        0
#define X86_VENDORSTR_Intel     "GenuineIntel"

#define X86_VENDOR_IntelClone   1

#define X86_VENDOR_AMD          2
#define X86_VENDORSTR_AMD       "AuthenticAMD"

#define X86_VENDOR_Cyrix        3
#define X86_VENDORSTR_CYRIX     "CyrixInstead"

#define X86_VENDOR_UMC          4
#define X86_VENDORSTR_UMC       "UMC UMC UMC "

#define X86_VENDOR_NexGen       5
#define X86_VENDORSTR_NexGen    "NexGenDriven"

#define X86_VENDOR_Centaur      6
#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)

/*
```

```
 655  * Definitions for AMD Family 0x12.
 656  */
 657 #define X86_CHIPREV_AMD_12_REV_B \
 658         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x12, 0x0002)

 660 /*
 661  * Definitions for AMD Family 0x14.
 662  */
 663 #define X86_CHIPREV_AMD_14_REV_B \
 664         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x14, 0x0002)
 665 #define X86_CHIPREV_AMD_14_REV_C \
 666         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x14, 0x0004)

 668 /*
 669  * Definitions for AMD Family 0x15
 670  */
 671 #define X86_CHIPREV_AMD_15OR_REV_B2 \
 672         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0001)

 674 #define X86_CHIPREV_AMD_15TN_REV_A1 \
 675         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0002)

 677 /*
 678  * Various socket/package types, extended as the need to distinguish
 679  * a new type arises.  The top 8 byte identfies the vendor and the
 680  * remaining 24 bits describe 24 socket types.
 681  */

 683 #define _X86_SOCKET_VENDOR_SHIFT        24
 684 #define _X86_SOCKET_VENDOR(x)   ((x) >> _X86_SOCKET_VENDOR_SHIFT)
 685 #define _X86_SOCKET_TYPE_MASK   0x00ffffff
 686 #define _X86_SOCKET_TYPE(x)             ((x) & _X86_SOCKET_TYPE_MASK)

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

 691 #define X86_SOCKET_MATCH(s, mask) \
 692         (_X86_SOCKET_VENDOR(s) == _X86_SOCKET_VENDOR(mask) && \
 693         (_X86_SOCKET_TYPE(s) & _X86_SOCKET_TYPE(mask)) != 0)

 695 #define X86_SOCKET_UNKNOWN 0x0
 696         /*
 697          * AMD socket types
 698          */
 699 #define X86_SOCKET_754          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000001)
 700 #define X86_SOCKET_939          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000002)
 701 #define X86_SOCKET_940          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000004)
 702 #define X86_SOCKET_S1g1         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000008)
 703 #define X86_SOCKET_AM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000010)
 704 #define X86_SOCKET_F1207        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000020)
 705 #define X86_SOCKET_S1g2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000040)
 706 #define X86_SOCKET_S1g3         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000080)
 707 #define X86_SOCKET_AM           _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000100)
 708 #define X86_SOCKET_AM2R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000200)
 709 #define X86_SOCKET_AM3          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000400)
 710 #define X86_SOCKET_G34          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000800)
 711 #define X86_SOCKET_ASB2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x001000)
 712 #define X86_SOCKET_C32          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x002000)
 713 #define X86_SOCKET_S1g4         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x004000)
 714 #define X86_SOCKET_FT1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x008000)
 715 #define X86_SOCKET_FM1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x010000)
 716 #define X86_SOCKET_FS1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x020000)
 717 #define X86_SOCKET_AM3R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x040000)
 718 #define X86_SOCKET_FP2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x080000)
 719 #define X86_SOCKET_FS1R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x100000)
 720 #define X86_SOCKET_FM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x200000)
```



```
 722 /*
 723  * xgetbv/xsetbv support
 724  * See section 13.3 in vol. 1 of the Intel devlopers manual.
 725  */

 727 #define XFEATURE_ENABLED_MASK   0x0
 728 /*
 729  * XFEATURE_ENABLED_MASK values (eax)
 730  * See setup_xfem().
 731  */
 732 #define XFEATURE_LEGACY_FP      0x1
 733 #define XFEATURE_SSE            0x2
 734 #define XFEATURE_AVX            0x4
 735 #define XFEATURE_MPX            0x18    /* 2 bits, both 0 or 1 */
 736 #define XFEATURE_AVX512         0xe0    /* 3 bits, all 0 or 1 */
 737         /* bit 8 unused */
 738 #define XFEATURE_PKRU           0x200
 739 #define XFEATURE_FP_ALL \
 740         (XFEATURE_LEGACY_FP | XFEATURE_SSE | XFEATURE_AVX | XFEATURE_MPX | \
 741         XFEATURE_AVX512 | XFEATURE_PKRU)

 743 /*
 744  * Define the set of xfeature flags that should be considered valid in the xsave
 745  * state vector when we initialize an lwp. This is distinct from the full set so
 746  * that all of the processor's normal logic and tracking of the xsave state is
 747  * usable. This should correspond to the state that's been initialized by the
 748  * ABI to hold meaningful values. Adding additional bits here can have serious
 749  * performance implications and cause performance degradations when using the
 750  * FPU vector (xmm) registers.
 751  */
 752 #define XFEATURE_FP_INITIAL     (XFEATURE_LEGACY_FP | XFEATURE_SSE)

 754 #if !defined(_ASM)

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

 758 #define NUM_X86_FEATURES        83
 751 #define NUM_X86_FEATURES        81
 759 extern uchar_t x86_featureset[];

 761 extern void free_x86_featureset(void *featureset);
 762 extern boolean_t is_x86_feature(void *featureset, uint_t feature);
 763 extern void add_x86_feature(void *featureset, uint_t feature);
 764 extern void remove_x86_feature(void *featureset, uint_t feature);
 765 extern boolean_t compare_x86_featureset(void *setA, void *setB);
 766 extern void print_x86_featureset(void *featureset);


 769 extern uint_t x86_type;
 770 extern uint_t x86_vendor;
 771 extern uint_t x86_clflush_size;

 773 extern uint_t pentiumpro_bug4046376;

 775 extern const char CyrixInstead[];

 777 #endif

 779 #if defined(_KERNEL)

 781 /*
 782  * This structure is used to pass arguments and get return values back
 783  * from the CPUID instruction in __cpuid_insn() routine.
 784  */
 785 struct cpuid_regs {
```

```
 786         uint32_t        cp_eax;
 787         uint32_t        cp_ebx;
 788         uint32_t        cp_ecx;
 789         uint32_t        cp_edx;
 790 };
______unchanged_portion_omitted_
```