```
**********************************************************
   28900 Wed Aug 12 10:41:01 2015
new/usr/src/uts/intel/sys/x86_archext.h
6116 remove unused FMT_CPUID_*
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2011 by Delphix. All rights reserved.
 * Copyright 2012 Nexenta Systems, Inc. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright (c) 2015, Joyent, Inc.
 * Copyright 2012 Jens Elkner <jel+illumos@cs.uni-magdeburg.de>
 * Copyright 2012 Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
 * Copyright 2014 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 */

#ifndef _SYS_X86_ARCHEXT_H
#define _SYS_X86_ARCHEXT_H

#if !defined(_ASM)
#include <sys/regset.h>
#include <sys/processor.h>
#include <vm/seg_enum.h>
#include <vm/page.h>
#endif  /* _ASM */

#ifdef __cplusplus
extern "C" {
#endif

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

#define CPUID_INTC_EDX_FPU      0x00000001      /* x87 fpu present */
#define CPUID_INTC_EDX_VME      0x00000002      /* virtual-8086 extension */
#define CPUID_INTC_EDX_DE       0x00000004      /* debugging extensions */
#define CPUID_INTC_EDX_PSE      0x00000008      /* page size extension */
#define CPUID_INTC_EDX_TSC      0x00000010      /* time stamp counter */
#define CPUID_INTC_EDX_MSR      0x00000020      /* rdmsr and wrmsr */
#define CPUID_INTC_EDX_PAE      0x00000040      /* physical addr extension */
```



```
#define CPUID_INTC_EDX_MCE      0x00000080      /* machine check exception */
#define CPUID_INTC_EDX_CX8      0x00000100      /* cmpxchg8b instruction */
#define CPUID_INTC_EDX_APIC     0x00000200      /* local APIC */
                                                /* 0x400 - reserved */
#define CPUID_INTC_EDX_SEP      0x00000800      /* sysenter and sysexit */
#define CPUID_INTC_EDX_MTRR     0x00001000      /* memory type range reg */
#define CPUID_INTC_EDX_PGE      0x00002000      /* page global enable */
#define CPUID_INTC_EDX_MCA      0x00004000      /* machine check arch */
#define CPUID_INTC_EDX_CMOV     0x00008000      /* conditional move insns */
#define CPUID_INTC_EDX_PAT      0x00010000      /* page attribute table */
#define CPUID_INTC_EDX_PSE36    0x00020000      /* 36-bit pagesize extension */
#define CPUID_INTC_EDX_PSN      0x00040000      /* processor serial number */
#define CPUID_INTC_EDX_CLFSH    0x00080000      /* clflush instruction */
                                                /* 0x100000 - reserved */
#define CPUID_INTC_EDX_DS       0x00200000      /* debug store exists */
#define CPUID_INTC_EDX_ACPI     0x00400000      /* monitoring + clock ctrl */
#define CPUID_INTC_EDX_MMX      0x00800000      /* MMX instructions */
#define CPUID_INTC_EDX_FXSR     0x01000000      /* fxsave and fxrstor */
#define CPUID_INTC_EDX_SSE      0x02000000      /* streaming SIMD extensions */
#define CPUID_INTC_EDX_SSE2     0x04000000      /* SSE extensions */
#define CPUID_INTC_EDX_SS       0x08000000      /* self-snoop */
#define CPUID_INTC_EDX_HTT      0x10000000      /* Hyper Thread Technology */
#define CPUID_INTC_EDX_TM       0x20000000      /* thermal monitoring */
#define CPUID_INTC_EDX_IA64     0x40000000      /* Itanium emulating IA32 */
#define CPUID_INTC_EDX_PBE      0x80000000      /* Pending Break Enable */

#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 */
#define CPUID_INTC_ECX_FMA      0x00001000      /* Fused Multiply Add */
#define CPUID_INTC_ECX_CX16     0x00002000      /* cmpxchg16 */
#define CPUID_INTC_ECX_ETPRD    0x00004000      /* extended task pri messages */
                                                /* 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_X2APIC   0x00200000      /* x2APIC */
#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 CPUID_INTC_ECX_F16C     0x20000000      /* F16C supported */
```

```
#define CPUID_INTC_ECX_RDRAND   0x40000000      /* RDRAND supported */
#define CPUID_INTC_ECX_HV       0x80000000      /* Hypervisor */

#define FMT_CPUID_INTC_ECX                                              \
        "\20"                                                           \
        "\37rdrand\36f16c\35avx\34osxsav\33xsave"                       \
        "\32aes"                                                        \
        "\30popcnt\27movbe\26x2apic\25sse4.2\24sse4.1\23dca"            \
        "\20\17etprd\16cx16\13cid\12ssse3\11tm2"                        \
        "\10est\7smx\6vmx\5dscpl\4mon\2pclmulqdq\1sse3"

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

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

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

#define CPUID_AMD_ECX_AHF64     0x00000001      /* LAHF and SAHF in long mode */
#define CPUID_AMD_ECX_CMP_LGCY  0x00000002      /* AMD: multicore chip */
#define CPUID_AMD_ECX_SVM       0x00000004      /* AMD: secure VM */
#define CPUID_AMD_ECX_EAS       0x00000008      /* extended apic space */
#define CPUID_AMD_ECX_CR8D      0x00000010      /* AMD: 32-bit mov %cr8 */
#define CPUID_AMD_ECX_LZCNT     0x00000020      /* AMD: LZCNT insn */
#define CPUID_AMD_ECX_SSE4A     0x00000040      /* AMD: SSE4A insns */
#define CPUID_AMD_ECX_MAS       0x00000080      /* AMD: MisAlignSse mnode */
#define CPUID_AMD_ECX_3DNP      0x00000100      /* AMD: 3DNowPrefectch */
#define CPUID_AMD_ECX_OSVW      0x00000200      /* AMD: OSVW */
```



```
#define CPUID_AMD_ECX_IBS       0x00000400      /* AMD: IBS */
#define CPUID_AMD_ECX_SSE5      0x00000800      /* AMD: SSE5 */
#define CPUID_AMD_ECX_SKINIT    0x00001000      /* AMD: SKINIT */
#define CPUID_AMD_ECX_WDT       0x00002000      /* AMD: WDT */
#define CPUID_AMD_ECX_TOPOEXT   0x00400000      /* AMD: Topology Extensions */

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

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

/*
 * Intel also uses cpuid leaf 7 to have additional instructions and features.
 * Like some other leaves, but unlike the current ones we care about, it
 * requires us to specify both a leaf in %eax and a sub-leaf in %ecx. To deal
 * with the potential use of additional sub-leaves in the future, we now
 * specifically label the EBX features with their leaf and sub-leaf.
 */
#define CPUID_INTC_EBX_7_0_BMI1         0x00000008      /* BMI1 instrs */
#define CPUID_INTC_EBX_7_0_AVX2         0x00000020      /* AVX2 supported */
#define CPUID_INTC_EBX_7_0_SMEP         0x00000080      /* SMEP in CR4 */
#define CPUID_INTC_EBX_7_0_BMI2         0x00000100      /* BMI2 Instrs */

#define P5_MCHADDR      0x0
#define P5_CESR         0x11
#define P5_CTR0         0x12
#define P5_CTR1         0x13

#define K5_MCHADDR      0x0
#define K5_MCHTYPE      0x01
#define K5_TSC          0x10
#define K5_TR12         0x12

#define REG_PAT         0x277

#define REG_MC0_CTL             0x400
#define REG_MC5_MISC            0x417
#define REG_PERFCTR0            0xc1
#define REG_PERFCTR1            0xc2

#define REG_PERFEVNT0           0x186
#define REG_PERFEVNT1           0x187

#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
```

```
232 #define MSR_DEBUGCTL            0x1d9

234 #define DEBUGCTL_LBR            0x01
235 #define DEBUGCTL_BTF            0x02

237 /* Intel P6, AMD */
238 #define MSR_LBR_FROM            0x1db
239 #define MSR_LBR_TO              0x1dc
240 #define MSR_LEX_FROM            0x1dd
241 #define MSR_LEX_TO              0x1de

243 /* Intel P4 (pre-Prescott, non P4 M) */
244 #define MSR_P4_LBSTK_TOS        0x1da
245 #define MSR_P4_LBSTK_0          0x1db
246 #define MSR_P4_LBSTK_1          0x1dc
247 #define MSR_P4_LBSTK_2          0x1dd
248 #define MSR_P4_LBSTK_3          0x1de

250 /* Intel Pentium M */
251 #define MSR_P6M_LBSTK_TOS       0x1c9
252 #define MSR_P6M_LBSTK_0         0x040
253 #define MSR_P6M_LBSTK_1         0x041
254 #define MSR_P6M_LBSTK_2         0x042
255 #define MSR_P6M_LBSTK_3         0x043
256 #define MSR_P6M_LBSTK_4         0x044
257 #define MSR_P6M_LBSTK_5         0x045
258 #define MSR_P6M_LBSTK_6         0x046
259 #define MSR_P6M_LBSTK_7         0x047

261 /* Intel P4 (Prescott) */
262 #define MSR_PRP4_LBSTK_TOS      0x1da
263 #define MSR_PRP4_LBSTK_FROM_0   0x680
264 #define MSR_PRP4_LBSTK_FROM_1   0x681
265 #define MSR_PRP4_LBSTK_FROM_2   0x682
266 #define MSR_PRP4_LBSTK_FROM_3   0x683
267 #define MSR_PRP4_LBSTK_FROM_4   0x684
268 #define MSR_PRP4_LBSTK_FROM_5   0x685
269 #define MSR_PRP4_LBSTK_FROM_6   0x686
270 #define MSR_PRP4_LBSTK_FROM_7   0x687
271 #define MSR_PRP4_LBSTK_FROM_8   0x688
272 #define MSR_PRP4_LBSTK_FROM_9   0x689
273 #define MSR_PRP4_LBSTK_FROM_10  0x68a
274 #define MSR_PRP4_LBSTK_FROM_11  0x68b
275 #define MSR_PRP4_LBSTK_FROM_12  0x68c
276 #define MSR_PRP4_LBSTK_FROM_13  0x68d
277 #define MSR_PRP4_LBSTK_FROM_14  0x68e
278 #define MSR_PRP4_LBSTK_FROM_15  0x68f
279 #define MSR_PRP4_LBSTK_TO_0     0x6c0
280 #define MSR_PRP4_LBSTK_TO_1     0x6c1
281 #define MSR_PRP4_LBSTK_TO_2     0x6c2
282 #define MSR_PRP4_LBSTK_TO_3     0x6c3
283 #define MSR_PRP4_LBSTK_TO_4     0x6c4
284 #define MSR_PRP4_LBSTK_TO_5     0x6c5
285 #define MSR_PRP4_LBSTK_TO_6     0x6c6
286 #define MSR_PRP4_LBSTK_TO_7     0x6c7
287 #define MSR_PRP4_LBSTK_TO_8     0x6c8
288 #define MSR_PRP4_LBSTK_TO_9     0x6c9
289 #define MSR_PRP4_LBSTK_TO_10    0x6ca
290 #define MSR_PRP4_LBSTK_TO_11    0x6cb
291 #define MSR_PRP4_LBSTK_TO_12    0x6cc
292 #define MSR_PRP4_LBSTK_TO_13    0x6cd
293 #define MSR_PRP4_LBSTK_TO_14    0x6ce
294 #define MSR_PRP4_LBSTK_TO_15    0x6cf

296 #define MCI_CTL_VALUE           0xffffffff
```



```
298 #define MTRR_TYPE_UC            0
299 #define MTRR_TYPE_WC            1
300 #define MTRR_TYPE_WT            4
301 #define MTRR_TYPE_WP            5
302 #define MTRR_TYPE_WB            6
303 #define MTRR_TYPE_UC_           7

305 /*
306  * For Solaris we set up the page attritubute table in the following way:
307  * PAT0 Write-Back
308  * PAT1 Write-Through
309  * PAT2 Unchacheable-
310  * PAT3 Uncacheable
311  * PAT4 Write-Back
312  * PAT5 Write-Through
313  * PAT6 Write-Combine
314  * PAT7 Uncacheable
315  * The only difference from h/w default is entry 6.
316  */
317 #define PAT_DEFAULT_ATTRIBUTE                   \
318         ((uint64_t)MTRR_TYPE_WB |               \
319         ((uint64_t)MTRR_TYPE_WT << 8) |         \
320         ((uint64_t)MTRR_TYPE_UC_ << 16) |       \
321         ((uint64_t)MTRR_TYPE_UC << 24) |        \
322         ((uint64_t)MTRR_TYPE_WB << 32) |        \
323         ((uint64_t)MTRR_TYPE_WT << 40) |        \
324         ((uint64_t)MTRR_TYPE_WC << 48) |        \
325         ((uint64_t)MTRR_TYPE_UC << 56))

327 #define X86FSET_LARGEPAGE       0
328 #define X86FSET_TSC             1
329 #define X86FSET_MSR             2
330 #define X86FSET_MTRR            3
331 #define X86FSET_PGE             4
332 #define X86FSET_DE              5
333 #define X86FSET_CMOV            6
334 #define X86FSET_MMX             7
335 #define X86FSET_MCA             8
336 #define X86FSET_PAE             9
337 #define X86FSET_CX8             10
338 #define X86FSET_PAT             11
339 #define X86FSET_SEP             12
340 #define X86FSET_SSE             13
341 #define X86FSET_SSE2            14
342 #define X86FSET_HTT             15
343 #define X86FSET_ASYSC           16
344 #define X86FSET_NX              17
345 #define X86FSET_SSE3            18
346 #define X86FSET_CX16            19
347 #define X86FSET_CMP             20
348 #define X86FSET_TSCP            21
349 #define X86FSET_MWAIT           22
350 #define X86FSET_SSE4A           23
351 #define X86FSET_CPUID           24
352 #define X86FSET_SSSE3           25
353 #define X86FSET_SSE4_1          26
354 #define X86FSET_SSE4_2          27
355 #define X86FSET_1GPG            28
356 #define X86FSET_CLFSH           29
357 #define X86FSET_64              30
358 #define X86FSET_AES             31
359 #define X86FSET_PCLMULQDQ       32
360 #define X86FSET_XSAVE           33
361 #define X86FSET_AVX             34
362 #define X86FSET_VMX             35
363 #define X86FSET_SVM             36
```

```
364 #define X86FSET_TOPOEXT         37
365 #define X86FSET_F16C            38
366 #define X86FSET_RDRAND          39
367 #define X86FSET_X2APIC          40
368 #define X86FSET_AVX2            41
369 #define X86FSET_BMI1            42
370 #define X86FSET_BMI2            43
371 #define X86FSET_FMA             44
372 #define X86FSET_SMEP            45

374 /*
375  * flags to patch tsc_read routine.
376  */
377 #define X86_NO_TSC              0x0
378 #define X86_HAVE_TSCP           0x1
379 #define X86_TSC_MFENCE          0x2
380 #define X86_TSC_LFENCE          0x4

382 /*
383  * Intel Deep C-State invariant TSC in leaf 0x80000007.
384  */
385 #define CPUID_TSC_CSTATE_INVARIANCE     (0x100)

387 /*
388  * Intel Deep C-state always-running local APIC timer
389  */
390 #define CPUID_CSTATE_ARAT       (0x4)

392 /*
393  * Intel ENERGY_PERF_BIAS MSR indicated by feature bit CPUID.6.ECX[3].
394  */
395 #define CPUID_EPB_SUPPORT       (1 << 3)

397 /*
398  * Intel TSC deadline timer
399  */
400 #define CPUID_DEADLINE_TSC      (1 << 24)

402 /*
403  * x86_type is a legacy concept; this is supplanted
404  * for most purposes by x86_featureset; modern CPUs
405  * should be X86_TYPE_OTHER
406  */
407 #define X86_TYPE_OTHER          0
408 #define X86_TYPE_486            1
409 #define X86_TYPE_P5             2
410 #define X86_TYPE_P6             3
411 #define X86_TYPE_CYRIX_486      4
412 #define X86_TYPE_CYRIX_6x86L    5
413 #define X86_TYPE_CYRIX_6x86     6
414 #define X86_TYPE_CYRIX_GXm      7
415 #define X86_TYPE_CYRIX_6x86MX   8
416 #define X86_TYPE_CYRIX_MediaGX  9
417 #define X86_TYPE_CYRIX_MII      10
418 #define X86_TYPE_VIA_CYRIX_III  11
419 #define X86_TYPE_P4             12

421 /*
422  * x86_vendor allows us to select between
423  * implementation features and helps guide
424  * the interpretation of the cpuid instruction.
425  */
426 #define X86_VENDOR_Intel        0
427 #define X86_VENDORSTR_Intel     "GenuineIntel"

429 #define X86_VENDOR_IntelClone   1
```



```
431 #define X86_VENDOR_AMD          2
432 #define X86_VENDORSTR_AMD       "AuthenticAMD"

434 #define X86_VENDOR_Cyrix        3
435 #define X86_VENDORSTR_CYRIX     "CyrixInstead"

437 #define X86_VENDOR_UMC          4
438 #define X86_VENDORSTR_UMC       "UMC UMC UMC "

440 #define X86_VENDOR_NexGen       5
441 #define X86_VENDORSTR_NexGen    "NexGenDriven"

443 #define X86_VENDOR_Centaur      6
444 #define X86_VENDORSTR_Centaur   "CentaurHauls"

446 #define X86_VENDOR_Rise         7
447 #define X86_VENDORSTR_Rise      "RiseRiseRise"

449 #define X86_VENDOR_SiS          8
450 #define X86_VENDORSTR_SiS       "SiS SiS SiS "

452 #define X86_VENDOR_TM           9
453 #define X86_VENDORSTR_TM        "GenuineTMx86"

455 #define X86_VENDOR_NSC          10
456 #define X86_VENDORSTR_NSC       "Geode by NSC"

458 /*
459  * Vendor string max len + \0
460  */
461 #define X86_VENDOR_STRLEN       13

463 /*
464  * Some vendor/family/model/stepping ranges are commonly grouped under
465  * a single identifying banner by the vendor.  The following encode
466  * that "revision" in a uint32_t with the 8 most significant bits
467  * identifying the vendor with X86_VENDOR_*, the next 8 identifying the
468  * family, and the remaining 16 typically forming a bitmask of revisions
469  * within that family with more significant bits indicating "later" revisions.
470  */

472 #define _X86_CHIPREV_VENDOR_MASK        0xff000000u
473 #define _X86_CHIPREV_VENDOR_SHIFT       24
474 #define _X86_CHIPREV_FAMILY_MASK        0x00ff0000u
475 #define _X86_CHIPREV_FAMILY_SHIFT       16
476 #define _X86_CHIPREV_REV_MASK           0x0000ffffu

478 #define _X86_CHIPREV_VENDOR(x) \
479         (((x) & _X86_CHIPREV_VENDOR_MASK) >> _X86_CHIPREV_VENDOR_SHIFT)
480 #define _X86_CHIPREV_FAMILY(x) \
481         (((x) & _X86_CHIPREV_FAMILY_MASK) >> _X86_CHIPREV_FAMILY_SHIFT)
482 #define _X86_CHIPREV_REV(x) \
483         ((x) & _X86_CHIPREV_REV_MASK)

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

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

```
497 #define _X86_CHIPREV_MKREV(vendor, family, rev) \
498         ((uint32_t)(vendor) << _X86_CHIPREV_VENDOR_SHIFT | \
499         (family) << _X86_CHIPREV_FAMILY_SHIFT | (rev))

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

506 /* Revision default */
507 #define X86_CHIPREV_UNKNOWN     0x0

509 /*
510  * Definitions for AMD Family 0xf. Minor revisions C0 and CG are
511  * sufficiently different that we will distinguish them; in all other
512  * case we will identify the major revision.
513  */
514 #define X86_CHIPREV_AMD_F_REV_B _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0001)
515 #define X86_CHIPREV_AMD_F_REV_C0 _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0002)
516 #define X86_CHIPREV_AMD_F_REV_CG _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0004)
517 #define X86_CHIPREV_AMD_F_REV_D _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0008)
518 #define X86_CHIPREV_AMD_F_REV_E _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0010)
519 #define X86_CHIPREV_AMD_F_REV_F _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0020)
520 #define X86_CHIPREV_AMD_F_REV_G _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0040)

522 /*
523  * Definitions for AMD Family 0x10.  Rev A was Engineering Samples only.
524  */
525 #define X86_CHIPREV_AMD_10_REV_A \
526         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0001)
527 #define X86_CHIPREV_AMD_10_REV_B \
528         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0002)
529 #define X86_CHIPREV_AMD_10_REV_C2 \
530         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0004)
531 #define X86_CHIPREV_AMD_10_REV_C3 \
532         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0008)
533 #define X86_CHIPREV_AMD_10_REV_D0 \
534         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0010)
535 #define X86_CHIPREV_AMD_10_REV_D1 \
536         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0020)
537 #define X86_CHIPREV_AMD_10_REV_E \
538         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0040)

540 /*
541  * Definitions for AMD Family 0x11.
542  */
543 #define X86_CHIPREV_AMD_11_REV_B \
544         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x11, 0x0002)

546 /*
547  * Definitions for AMD Family 0x12.
548  */
549 #define X86_CHIPREV_AMD_12_REV_B \
550         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x12, 0x0002)

552 /*
553  * Definitions for AMD Family 0x14.
554  */
555 #define X86_CHIPREV_AMD_14_REV_B \
556         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x14, 0x0002)
557 #define X86_CHIPREV_AMD_14_REV_C \
558         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x14, 0x0004)

560 /*
561  * Definitions for AMD Family 0x15
```



```
562  */
563 #define X86_CHIPREV_AMD_15OR_REV_B2 \
564         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0001)

566 #define X86_CHIPREV_AMD_15TN_REV_A1 \
567         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0002)

569 /*
570  * Various socket/package types, extended as the need to distinguish
571  * a new type arises.  The top 8 byte identfies the vendor and the
572  * remaining 24 bits describe 24 socket types.
573  */

575 #define _X86_SOCKET_VENDOR_SHIFT        24
576 #define _X86_SOCKET_VENDOR(x)   ((x) >> _X86_SOCKET_VENDOR_SHIFT)
577 #define _X86_SOCKET_TYPE_MASK   0x00ffffff
578 #define _X86_SOCKET_TYPE(x)             ((x) & _X86_SOCKET_TYPE_MASK)

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

583 #define X86_SOCKET_MATCH(s, mask) \
584         (_X86_SOCKET_VENDOR(s) == _X86_SOCKET_VENDOR(mask) && \
585         (_X86_SOCKET_TYPE(s) & _X86_SOCKET_TYPE(mask)) != 0)

587 #define X86_SOCKET_UNKNOWN 0x0
588         /*
589          * AMD socket types
590          */
591 #define X86_SOCKET_754          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000001)
592 #define X86_SOCKET_939          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000002)
593 #define X86_SOCKET_940          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000004)
594 #define X86_SOCKET_S1g1         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000008)
595 #define X86_SOCKET_AM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000010)
596 #define X86_SOCKET_F1207        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000020)
597 #define X86_SOCKET_S1g2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000040)
598 #define X86_SOCKET_S1g3         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000080)
599 #define X86_SOCKET_AM           _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000100)
600 #define X86_SOCKET_AM2R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000200)
601 #define X86_SOCKET_AM3          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000400)
602 #define X86_SOCKET_G34          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x000800)
603 #define X86_SOCKET_ASB2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x001000)
604 #define X86_SOCKET_C32          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x002000)
605 #define X86_SOCKET_S1g4         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x004000)
606 #define X86_SOCKET_FT1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x008000)
607 #define X86_SOCKET_FM1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x010000)
608 #define X86_SOCKET_FS1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x020000)
609 #define X86_SOCKET_AM3R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x040000)
610 #define X86_SOCKET_FP2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x080000)
611 #define X86_SOCKET_FS1R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x100000)
612 #define X86_SOCKET_FM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x200000)

614 /*
615  * xgetbv/xsetbv support
616  */

618 #define XFEATURE_ENABLED_MASK   0x0
619 /*
620  * XFEATURE_ENABLED_MASK values (eax)
621  */
622 #define XFEATURE_LEGACY_FP      0x1
623 #define XFEATURE_SSE            0x2
624 #define XFEATURE_AVX            0x4
625 #define XFEATURE_MAX            XFEATURE_AVX
626 #define XFEATURE_FP_ALL \
627         (XFEATURE_LEGACY_FP|XFEATURE_SSE|XFEATURE_AVX)
```

```
 629 #if !defined(_ASM)

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

 633 #define NUM_X86_FEATURES        46
 634 extern uchar_t x86_featureset[];

 636 extern void free_x86_featureset(void *featureset);
 637 extern boolean_t is_x86_feature(void *featureset, uint_t feature);
 638 extern void add_x86_feature(void *featureset, uint_t feature);
 639 extern void remove_x86_feature(void *featureset, uint_t feature);
 640 extern boolean_t compare_x86_featureset(void *setA, void *setB);
 641 extern void print_x86_featureset(void *featureset);


 644 extern uint_t x86_type;
 645 extern uint_t x86_vendor;
 646 extern uint_t x86_clflush_size;

 648 extern uint_t pentiumpro_bug4046376;

 650 extern const char CyrixInstead[];

 652 #endif

 654 #if defined(_KERNEL)

 656 /*
 657  * This structure is used to pass arguments and get return values back
 658  * from the CPUID instruction in __cpuid_insn() routine.
 659  */
 660 struct cpuid_regs {
 661         uint32_t        cp_eax;
 662         uint32_t        cp_ebx;
 663         uint32_t        cp_ecx;
 664         uint32_t        cp_edx;
 665 };
```

*unchanged_portion_omitted_*