AMD

AMD Processor Recognition

Application Note

Publication # **20734** Rev: **J** Amendment/**0**
Issue Date: **November 1998**

Contents

List of Figures

List of Tables

Revision History

Date	Rev	Description
Sept 1997	F	Moved SYSCALL/SYSRET instruction feature bit (in extended feature function 8000_0001h) from bit 10 to bit 11. See Table 6 on page 15 and Table 10 on page 18.
Sept 1997	F	Added bit 31 to the extended feature function 8000_0001h for a new feature. See Table 4 on page 8 and Table 6 on page 15.
Sept 1997	F	Added support for AMD-K6® processor Models 7, 8, and 9 to Table 1 on page 4 and Table 2 on page 5.
Sept 1997	F	Added return values for AMD-K6 processor Model 7 to Table 10 on page 18.
Dec 1997	G	Changed part names for AMD-K6 processor Models 8 and 9 in Table 2 on page 5.
Dec 1997	G	Added 3DNow!™ instructions feature (bit 31) to Table 4 on page 8 and Table 6 on page 15.
Dec 1997	G	Added AMD-K6®-2 processor return values to Table 10 on page 18.
Jan 1998	H	Added revised bit 31 description and alternate test for AMD-K6-2 to "Identifying Supported Features" on page 6.
May 1998	I	Revised "Functions 8000_0002h, 8000_0003h, and 8000_0004h — Processor Name String" on page 16.
May 1998	I	Added return values for AMD-K6 processor Model 9 to Table 10 on page 18. Divided Appendix B table into two separate tables.
Nov 1998	J	In "Standard Functions" on page 12, clarified AMD's vendor identification string stored in registers EBX, EDX, and ECX.
Nov 1998	J	In Table 10, "Values Returned By AMD-K6® Processors," on page 18, changed function 8000_0002h, EDX value for the AMD-K6 processor Model 7 and deleted note 2.

Application Note

AMD Processor Recognition

Introduction

Due to the increasing number of choices available in the x86 processor marketplace, the need for a simple way for hardware and software to identify the type of processor and its feature set has become critical. The CPUID instruction was added to the x86 instruction set for this purpose.

The CPUID instruction provides complete information about the processor (vendor, type, name, etc.) and its capabilities (features). After detecting the processor and its capabilities, software can be accurately tuned to the system for maximum performance and benefit to users. For example, game software can test the performance level available from a particular processor by detecting the type or speed of the processor. If the performance level is high enough, the software can enable additional capabilities or more advanced algorithms. Another example involves testing for the presence of MMX™ instructions on the processor. If the software finds this feature present when it checks the feature bits, it can utilize these more powerful extensions for dramatically better performance on new multimedia software.

Using the CPUID Instruction

Overview

Software operating at any privilege level can execute the CPUID instruction to identify the processor and its feature set. In addition, the CPUID instruction implements multiple functions, each providing different information about the processor, including the vendor, model number, revision (stepping), features, cache organization, and processor name. The multiple-function approach allows the CPUID instruction to return a complete picture about the type of processor and its capabilities—more detailed information than could be returned by a single function. In addition to gathering all the information by calling multiple functions, the CPUID instruction provides the flexibility of making only one call to obtain the specific data requested once the processor vendor has been identified.

The functions are divided into two types: standard functions and extended functions. Standard functions provide a simple method for software to access information common to all x86 processors. Extended functions provide information on extensions specific to a vendor's processor (for example, AMD's processors).

The flexibility of the CPUID instruction allows for the addition of new CPUID functions in future generations of processors. Appendix A on page 11 contains a detailed description of the CPUID instruction.

Testing for the CPUID Instruction

Beginning with the Am486®DX4 processor, all AMD processors implement the CPUID instruction. In order to avoid an invalid opcode exception on those processors that do not support the CPUID instruction, software must first test to determine if the CPUID instruction is present on the processor. The presence of the CPUID instruction is indicated by the ID bit (21) in the EFLAGS register. If this bit is writeable, the CPUID instruction is implemented on the processor.

Software uses the PUSHFD and POPFD instructions to write to the ID bit in the EFLAGS register. After reading the ID bit, a comparison determines if this operation changed the value of the ID bit. If the value changed, the CPUID instruction is available for identifying the processor and its features. The following code sample demonstrates the way a program uses the PUSHFD and POPFD instructions to test the ID bit.

```
pushfd                  ; Save EFLAGS to stack
pop   eax               ; Store EFLAGS in EAX
mov   ebx, eax          ; Save in EBX for testing later
xor   eax, 00200000h    ; Switch bit 21
push  eax               ; Copy "changed" value to stack
popfd                   ; Save "changed" EAX to EFLAGS
pushfd                  ; Push EFLAGS to top of stack
pop   eax               ; Store EFLAGS in EAX
cmp   eax, ebx          ; See if bit 21 has changed
jz    NO_CPUID          ; If no change, no CPUID
```

Using CPUID Functions

When software uses the CPUID instruction to identify a processor, it is important that it uses the instruction appropriately. The instruction has been defined to make it easy to identify the type and features of x86 processors manufactured by many different vendors.

The standard functions (EAX=0 and EAX=1) are the same for all processors. Having standard functions simplifies software's task of testing for and implementing features common to x86 processors. Software can test for these features and, as new x86 processors are released, benefit from these capabilities immediately.

Extended functions are specific to a vendor's processor. These functions provide additional information about AMD processors that software can use to identify enhanced features and functions. To test for extended functions, software checks for "AuthenticAMD" in the vendor identification string returned by function 0 and for a non-zero value in the EAX register returned by function 8000_0000h.

Within AMD's family of processors, different members can execute a different number of functions. Table 1 summarizes the CPUID functions currently implemented on AMD processors.

Table 1. Summary of CPUID Functions in AMD Processors
(Appendix A contains detailed descriptions of the functions.)

Standard Function	Extended Function[1]	Description	Am486®DX4 and Am5$_X$86™ Processors	AMD-K5™ Processor (Model 0)	AMD-K5™ Processor (Models 1, 2, and 3)	AMD-K6® Processor (Models 6, 7, and 8)	AMD-K6® Processor (Model 9)[2]
0	–	Vendor String and Largest Standard Function Value	X	X	X	X	X
1	–	Processor Signature and Standard Feature Bits	X	X	X	X	X
–	8000_0000h	Largest Extended Function Value	–	–	X	X	X
–	8000_0001h	Extended Processor Signature and Extended Feature Bits	–	–	X	X	X
–	8000_0002h	Processor Name	–	–	X	X	X
–	8000_0003h	Processor Name	–	–	X	X	X
–	8000_0004h	Processor Name	–	–	X	X	X
–	8000_0005h	L1 Cache Information	–	–	X	X	X
–	TBD	TBD	–	–	–	–	X

Notes:

1. *Future versions of these processors may implement additional functions.*
2. *A future revision of this application note will fully describe the AMD-K6 processor Model 9 and its CPUID functions.*

Identifying the Processor's Vendor

Software must execute the standard function EAX=0. The CPUID instruction returns a 12-character string that identifies the processor's vendor. The instruction also returns the largest standard function input value defined for the CPUID instruction on the processor.

For AMD processors, function 0 returns a vendor string of "AuthenticAMD". This string informs the software to follow AMD's definition for subsequent CPUID functions and the registers returned for those functions.

Once the software identifies the processor's vendor, it knows the definition for all the functions supplied by the CPUID instruction. By using these functions, the software obtains the processor information needed to properly tune its functionality to the capabilities of the processor.

Determining the Processor Signature (Standard Function)

Standard function 1 (EAX=1) of the CPUID instruction returns the standard processor signature and feature bits. The standard processor signature is returned in the EAX register and provides information regarding the specific revision (stepping) and model of the processor and the instruction family level supported by the processor. The revision level is used to determine if the processor requires the implementation of software workarounds. Figure 1 shows the contents of the EAX register obtained by function 1. Table 2 summarizes the specific processor signature values returned for AMD processors.

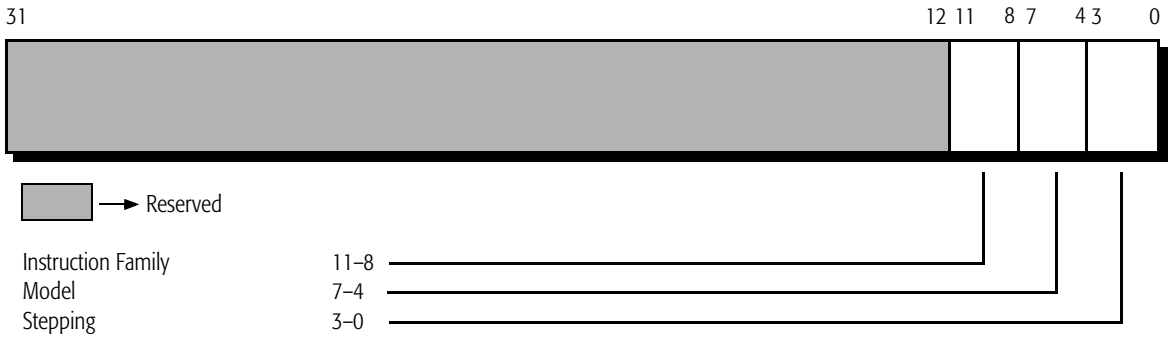

Figure 1. Contents of EAX Register Returned by Function 1

Table 2. Summary of Processor Signatures for AMD Processors
(Appendix A contains details on bit locations and values.)

Processor	Instruction Family	Model	Stepping ID
Am486® and Am5$_X$86™ Processors	0100b (4h)	yyyy[2]	xxxx[1]
AMD-K5™ Processor (Model 0)	0101b (5h)	0000b (0h)	xxxx[1]
AMD-K5 Processor (Model 1)	0101b (5h)	0001b (1h)	xxxx[1]
AMD-K5 Processor (Model 2)	0101b (5h)	0010b (2h)	xxxx[1]
AMD-K5 Processor (Model 3)	0101b (5h)	0011b (3h)	xxxx[1]
AMD-K6® Processor (Model 6)	0101b (5h)	0110b (6h)	xxxx[1]
AMD-K6 Processor (Model 7)	0101b (5h)	0111b (7h)	xxxx[1]
AMD-K6®-2 Processor (Model 8)	0101b (5h)	1000b (8h)	xxxx[1]
AMD-K6® Processor (Model 9)	0101b (5h)	1001b (9h)	xxxx[1]

Notes:
1. *Contact your AMD representative for the latest stepping information.*
2. *Model identifier information is provided in the AMD BIOS Development Guide, order# 19720.*

Identifying Supported Features

The feature bits are returned in the EDX register for two CPUID functions: standard function 1 and extended function 8000_0001h. Each bit corresponds to a specific feature and indicates if that feature is present on the processor. Table 3 summarizes the standard feature bits, and Table 4 summarizes the extended feature bits.

Before using any of the enhanced features added to the latest generation of processors, software should test each feature bit returned by functions 1 and 8000_0001h to identify the capabilities available on the processor. For example, software must test bit 23 to determine if the processor executes MMX instructions. Attempting to execute an unavailable feature can cause errors and exceptions.

Bit 31, as returned by extended function 8000_0001h, designates the presence of 3DNow!™ technology. Other processor vendors have adopted this technology so now bit 31 is considered an open standard. An alternate way to test for the presence of 3DNow! technology (as opposed to testing for AuthenticAMD) is for software to implement the following algorithm:

1. Test for the CPUID instruction. (See "Testing for the CPUID Instruction" on page 2.)
2. Execute the CPUID extended function 8000_0000h.
3. Test if the value returned in the EAX register is greater than or equal to 8000_0000h.
4. Execute the CPUID extended function 8000_0001h.
5. Test bit 31 in the EDX register for 3DNow! technology.

Table 3. Summary of Standard Feature Bits for AMD Processors
(Appendix A contains details on bit locations and values.)

Feature	Description
Floating-Point Unit	A floating-point unit is available.
Virtual Mode Extensions	Virtual mode extensions are available.
Debugging Extensions	I/O breakpoint debug extensions are supported.
Page Size Extensions	4-Mbyte pages are supported.
Time Stamp Counter (with RDTSC and CR4 disable bit)	A time stamp counter is available in the processor, and the RDTSC instruction is supported.
K86™ Family of Processors' Model-Specific Registers (with RDMSR and WRMSR)	The K86 model-specific registers are available in the processor, and the RDMSR and WRMSR instructions are supported.
Machine Check Exception	The machine check exception is supported.
CMPXCHG8B Instruction	The CMPXCHG8B instruction is supported.
APIC	A local APIC unit is available.
Global Paging Extension	Global paging extensions are available.
Conditional Move Instructions	The conditional move instructions CMOV, FCMOV, and FCOMI are supported.
MMX™ Instructions	The MMX instruction set is supported.

Table 4. Summary of Extended Feature Bits for AMD Processors
(Appendix A contains details on bit locations and values.)

Feature	Description
Floating-Point Unit	A floating-point unit is available.
Virtual Mode Extensions	Virtual mode extensions are available.
Debugging Extensions	I/O breakpoint debug extensions are supported.
Page Size Extensions	4-Mbyte pages are supported.
Time Stamp Counter (with RDTSC and CR4 disable bit)	A time stamp counter is available in the processor, and the RDTSC instruction is supported.
K86™ Family of Processors' Model-Specific Registers (with RDMSR and WRMSR)	The K86 model-specific registers are available in the processor, and the RDMSR and WRMSR instructions are supported.
Machine Check Exception	The machine check exception is supported.
CMPXCHG8B Instruction	The CMPXCHG8B instruction is supported.
Global Paging Extension	Global paging extensions are available.
SYSCALL and SYSRET Instructions	The SYSCALL and SYSRET instructions and associated extensions are supported.
Integer Conditional Move Instruction	The integer conditional move instruction CMOV is supported.
Floating-Point Conditional Move Instructions	The floating-point conditional move instructions FCMOV and FCOMI are supported.
MMX™ Instructions	The MMX instruction set is supported.
3DNow!™ Instructions	The 3DNow! instruction set is supported.

Testing For Extended Functions

Once software has identified the processor's vendor as AMD, it must test for extended functions by executing function 8000_0000h. The EAX register returns the largest extended function input value defined for the CPUID instruction on the processor. If this value is non-zero, extended functions are supported.

To simplify identifying processors and their features, the AMD extended functions include all the information provided in the standard functions as well as the additional AMD-specific feature enhancements. This duplication can minimize the number of function calls required by software.

Determining the Processor Signature (Extended Function)

Extended function 8000_0001h returns the AMD processor signature. The signature is returned in the EAX register and provides generation, model, and stepping information for AMD processors. Figure 2 shows the contents returned in the EAX register.

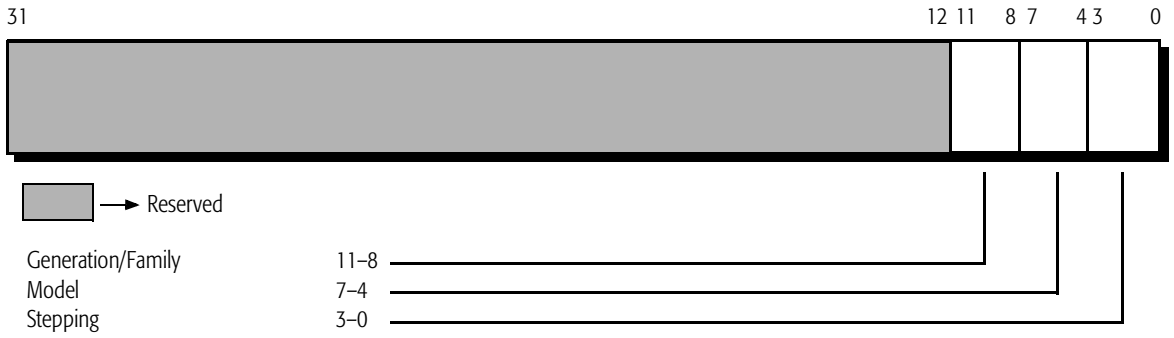

Figure 2. Contents of EAX Register Returned by Extended Function 8000_0001h

Displaying the Processor's Name

Functions 8000_0002h, 8000_0003h, and 8000_0004h return an ASCII string containing the name of the processor. These functions eliminate the need for software to search for the processor name in a lookup table, a process requiring a large block of memory and frequent updates. Instead, software can

simply call these three functions to obtain the name string (48 ASCII characters in little endian format) and display it on the screen. Although the name string can be up to 48 characters in length, shorter names have the remaining byte locations filled with the ASCII NULL character (00h). To simplify the display routines and avoid using screen space, software only needs to display characters until a NULL character is detected.

Displaying Cache Information

Function 8000_0005h provides cache information for the processor. Some diagnostic software displays information about the system and the processor's configuration. It is common for this type of software to provide cache size and organization of information. Function 8000_0005h provides a simple way for software to obtain information about the on-chip L1 caches and Translation Lookaside Buffer (TLB) structures. The size and organization information is returned in the registers as described in Appendix A on page 11. Software can simply display these values, eliminating the need for large pieces of code to test the memory structures.

Sample Code

A code sample that uses the CPUID instruction to identify the processor and its features is available from AMD's website at http://www.amd.com/K6/k6docs/.

Appendix A

CPUID

mnemonic	*opcode*	*description*
CPUID	0F A2h	Identify the processor and its feature set
Privilege:	none	
Registers Affected:	EAX, EBX, ECX, EDX	
Flags Affected:	none	
Exceptions Generated:	none	

The CPUID instruction is an application-level instruction that software executes to identify the processor and its feature set. This instruction offers multiple functions, each providing a different set of information about the processor. The CPUID instruction can be executed from any privilege level. Software can use the information returned by this instruction to tune its functionality for the specific processor and its features.

Not all processors implement the CPUID instruction. Therefore, software must test to determine if the instruction is present on the processor. If the ID bit (21) in the EFLAGS register is writeable, the CPUID instruction is implemented.

The CPUID instruction supports multiple functions. The information associated with each function is obtained by executing the CPUID instruction with the function number in the EAX register. Functions are divided into two types: standard functions and extended functions. Standard functions are found in the low function space, 0000_0000h–7FFF_FFFFh. In general, all x86 processors have the same standard function definitions.

Extended functions are defined specifically for processors supplied by the vendor listed in the vendor identification string. Extended functions are found in the high function space, 8000_0000h–8FFF_FFFFh. Because not all vendors have defined extended functions, software must test for their presence on the processor.

AMD processors have extended functions under the following conditions:

- The processor returns the “AuthenticAMD” vendor identification string.
- The 8000_0000h function returns a non-zero value in the EAX register.

Standard Functions

Function 0 — Largest Standard Function Input Value and Vendor Identification String

Input: EAX = 0

Output: EAX = Largest function input value recognized by the CPUID instruction
EBX, EDX, ECX = Vendor identification string

This is a standard function found in all processors implementing the CPUID instruction. It returns two values. The first value is returned in the EAX register and indicates the largest standard function value recognized by the processor. The second value is the vendor identification string. This 12-character ASCII string is returned in the EBX, EDX, and ECX registers in little endian format. AMD processors return a vendor identification string of "AuthenticAMD" as follows:

EBX				EDX				ECX				← Registers
h	t	u	A	i	t	n	e	D	M	A	c	← Alpha Characters
68	74	75	41	69	74	6E	65	44	4D	41	63	← ASCII Codes

Software uses the vendor identification string as follows:

- To identify the processor as an AMD processor
- To apply AMD's definition of the CPUID instruction for all additional function calls

Function 1 — Processor Signature and Standard Feature Flags

Input: EAX = 1

Output: EAX = Processor Signature
EBX = Reserved
ECX = Reserved
EDX = Standard Feature Flags

Function 1 returns two values—the Processor Signature and the Standard Feature Flags. The processor signature is returned in the EAX register and identifies the specific processor by providing information on its type—instruction family, model, and revision (stepping). The information is formatted as follows:

- EAX[3–0] Stepping ID
- EAX[7–4] Model
- EAX[11–8] Instruction Family
- EAX[31–12] Reserved

The standard feature flags are returned in the EDX register and indicate the presence of specific features. In most cases, a "1" indicates the feature is present, and a "0" indicates the feature is not present. Table 5 contains a list of the currently defined standard feature flags. Reserved bits will be used for new features as they are added.

Table 5. Standard Feature Flag Descriptions

Bit	Feature	Description
0	Floating-Point Unit	0 = No FPU 1 = FPU Present
1	Virtual Mode Extensions	0 = No Support 1= Support
2	Debugging Extensions	0 = No Support 1= Support
3	Page Size Extensions	0 = No Support 1= Support 4-Mbyte Pages
4	Time Stamp Counter (with RDTSC and CR4 disable bit)	0 = No Support 1= Support
5	K86™ Family of Processors' Model-Specific Registers (with RDMSR and WRMSR)	0 = No Support 1= Support
6	Reserved	–
7	Machine Check Exception	0 = No Support 1= Support
8	CMPXCHG8B Instruction	0 = No Support 1= Support
9	APIC*	0 = No Support 1= Support
10–11	Reserved	–
12	Memory Type Range Registers	0 = No Support 1= Support
13	Global Paging Extension*	0 = No Support 1= Support
14	Reserved	–
15	Conditional Move Instruction	0 = No Support 1= Support
16–22	Reserved	–
23	MMX™ Instructions	0 = No Support 1= Support
24–31	Reserved	–

Note:

* *The AMD-K5™ processor (model 0) reserves bit 13 and implements feature bit 9 to indicate support for Global Paging Extensions instead of support for APIC.*

Extended Functions

Function 8000_0000h – Largest Extended Function Input Value

Input: EAX = 8000_0000h

Output: EAX = Largest function input value recognized by the CPUID instruction
EBX = Reserved
ECX = Reserved
EDX = Reserved

Function 8000_0000h returns a value in the EAX register that indicates the largest extended function value recognized by the processor.

Function 8000_0001h – AMD Processor Signature and Extended Feature Flags

Input: EAX = 8000_0001h

Output: EAX = AMD Processor Signature
EBX = Reserved
ECX = Reserved
EDX = Extended Feature Flags

Function 8000_0001h returns two values—the AMD Processor Signature and the Extended Feature Flags. The AMD processor signature is returned in the EAX register and identifies the specific processor by providing information regarding its type—generation/family, model, and revision (stepping). The information is formatted as follows:

- EAX[3–0] Stepping ID
- EAX[7–4] Model
- EAX[11–8] Generation/Family
- EAX[31–12] Reserved

The extended feature flags are returned in the EDX register and indicate the presence of specific features found in AMD processors. In most cases, a “1” indicates the feature is present, and a “0” indicates the feature is not present. Table 6 contains a list of the currently defined extended feature flags. Reserved bits will be used for new features as they are added.

Table 6. Extended Feature Flag Descriptions

Bit	Feature	Description
0	Floating-Point Unit	0 = No FPU 1 = FPU Present
1	Virtual Mode Extensions	0 = No Support 1= Support
2	Debugging Extensions	0 = No Support 1= Support
3	Page Size Extensions	0 = No Support 1= Support 4-Mbyte Pages
4	Time Stamp Counter (with RDTSC and CR4 disable bit)	0 = No Support 1= Support
5	K86™ Family of Processors' Model-Specific Registers (with RDMSR and WRMSR)	0 = No Support 1= Support
6	Reserved	—
7	Machine Check Exception	0 = No Support 1= Support
8	CMPXCHG8B Instruction	0 = No Support 1= Support
9–10	Reserved	—
11	SYSCALL and SYSRET Instructions	0 = No Support 1= Support
12	Reserved	—
13	Global Paging Extension	0 = No Support 1= Support
14	Reserved	—
15	Integer Conditional Move Instruction	0 = No Support 1= Support
16	Floating-Point Conditional Move Instructions	0 = No Support 1= Support
17–22	Reserved	—
23	MMX™ Instructions	0 = No Support 1= Support
24–30	Reserved	—
31	3DNow!™ Instructions	0 = No Support 1= Support

Functions 8000_0002h, 8000_0003h, and 8000_0004h — Processor Name String

Input: EAX = 8000_0002h, 8000_0003h, or 8000_0004h

Output: EAX = Processor Name String
EBX = Processor Name String
ECX = Processor Name String
EDX = Processor Name String

Functions 8000_0002h, 8000_0003h, and 8000_0004h each return part of the processor name string in the EAX, EBX, ECX, and EDX registers. These three functions use the four registers to return an ASCII string of up to 48 characters in little endian format. For example, function 8000_0002h returns the first 16 characters of the processor name. The first character resides in the least significant byte of EAX, and the last character (of this group of 16) resides in the most significant byte of EDX. The NULL character (ASCII 00h) is used to indicate the end of the processor name string. This feature is useful for processor names that require fewer than 48 characters.

Function 8000_0005h — L1 Cache Information

Input: EAX = 8000_0005h

Output: EAX = Reserved
EBX = TLB Information
ECX = L1 Data Cache Information
EDX = L1 Instruction Cache Information

Function 8000_0005h returns information about the processor's on-chip L1 caches and associated TLBs. Tables 7, 8, and 9 provide the format for the information returned by the 8000_0005h function.

Table 7. EBX Format Returned by Function 8000_0005h

	Data TLB		Instruction TLB	
	Associativity*	**# Entries**	**Associativity***	**# Entries**
EBX	Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0

Note:
* *Full associativity is indicated by a value of 0FFh.*

Table 8. ECX Format Returned by Function 8000_0005h

	L1 Data Cache			
	Size (Kbytes)	**Associativity***	**Lines per Tag**	**Line Size (bytes)**
ECX	Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0

Note:
* *Full associativity is indicated by a value of 0FFh.*

Table 9. EDX Format Returned by Function 8000_0005h

	L1 Instruction Cache			
	Size (Kbytes)	**Associativity***	**Lines per Tag**	**Line Size (bytes)**
EDX	Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0

Note:
* *Full associativity is indicated by a value of 0FFh.*

Appendix B

Tables 10 and 11 contain all the values returned for AMD processors by the CPUID instruction.

Table 10. Values Returned By AMD-K6® Processors

Function / Register	AMD-K6® Processor (Model 6)	AMD-K6® Processor (Model 7)	AMD-K6®-2 Processor (Model 8)	AMD-K6® Processor (Model 9)*
Function: 0				
EAX	0000_0001h	0000_0001h	0000_0001h	0000_0001h
EBX	6874_7541h	6874_7541h	6874_7541h	6874_7541h
ECX	444D_4163h	444D_4163h	444D_4163h	444D_4163h
EDX	6974_6E65h	6974_6E65h	6974_6E65h	6974_6E65h
Function: 1				
EAX	0000_056Xh	0000_057Xh	0000_058Xh	0000_059Xh
EBX	Reserved	Reserved	Reserved	Reserved
ECX	Reserved	Reserved	Reserved	Reserved
EDX	0080_01BFh	0080_01BFh	0080_01BFh	TBD
Function: 8000_0000h				
EAX	8000_0005h	8000_0005h	8000_0005h	TBD
EBX	Reserved	Reserved	Reserved	Reserved
ECX	Reserved	Reserved	Reserved	Reserved
EDX	Reserved	Reserved	Reserved	Reserved
Function: 8000_0001h				
EAX	0000_066Xh	0000_067Xh	0000_068Xh	0000_069Xh
EBX	Reserved	Reserved	Reserved	Reserved
ECX	Reserved	Reserved	Reserved	Reserved
EDX	0080_01BFh	0080_01BFh	8080_09BFh	TBD
Function: 8000_0002h				
EAX	2D44_4D41h	2D44_4D41h	2D44_4D41h	TBD
EBX	6D74_364Bh	6D74_364Bh	7428_364Bh	
ECX	202F_7720h	202F_7720h	3320_296Dh	
EDX	746C_756Dh	746C_756Dh	7270_2044h	

Note:
** A future revision of this application note will fully describe the AMD-K6 processor Model 9 and its CPUID functions.*

Table 10. Values Returned By AMD-K6® Processors (continued)

Function / Register	AMD-K6® Processor (Model 6)	AMD-K6® Processor (Model 7)	AMD-K6®-2 Processor (Model 8)	AMD-K6® Processor (Model 9)*
Function: 8000_0003h				
EAX	6465_6D69h	6465_6D69h	7365_636Fh	TBD
EBX	6520_6169h	6520_6169h	0072_6F73h	
ECX	6E65_7478h	6E65_7478h	0000_0000h	
EDX	6E6F_6973h	6E6F_6973h	0000_0000h	
Function: 8000_0004h				
EAX	0000_0073h	0000_0073h	0000_0000h	TBD
EBX	0000_0000h	0000_0000h	0000_0000h	
ECX	0000_0000h	0000_0000h	0000_0000h	
EDX	0000_0000h	0000_0000h	0000_0000h	
Function: 8000_0005h				
EAX	Reserved	Reserved	Reserved	Reserved
EBX	0280_0140h	0280_0140h	0280_0140h	TBD
ECX	2002_0220h	2002_0220h	2002_0220h	TBD
EDX	2002_0220h	2002_0220h	2002_0220h	TBD

Note:

** A future revision of this application note will fully describe the AMD-K6 processor Model 9 and its CPUID functions.*

Table 11. Values Returned By Am486®, Am5$_X$86™, and AMD-K5™ Processors

Function Register	Am486® and Am5$_X$86™ Processors	AMD-K5™ Processor (Model 0)	AMD-K5™ Processor (Model 1)	AMD-K5™ Processor (Model 2)	AMD-K5™ Processor (Model 3)
Function: 0					
EAX	0000_0001h	0000_0001h	0000_0001h	0000_0001h	0000_0001h
EBX	6874_7541h	6874_7541h	6874_7541h	6874_7541h	6874_7541h
ECX	444D_4163h	444D_4163h	444D_4163h	444D_4163h	444D_4163h
EDX	6974_6E65h	6974_6E65h	6974_6E65h	6974_6E65h	6974_6E65h
Function: 1					
EAX	0000_04XXh	0000_050Xh	0000_051Xh	0000_052Xh	0000_053Xh
EBX	Reserved	Reserved	Reserved	Reserved	Reserved
ECX	Reserved	Reserved	Reserved	Reserved	Reserved
EDX	0000_0001h	0000_03BFh	0000_21BFh	0000_21BFh	0000_21BFh
Function: 8000_0000h					
EAX	0000_0000h	0000_0000h	8000_0005h	8000_0005h	8000_0005h
EBX	Undefined	Undefined	Reserved	Reserved	Reserved
ECX	Undefined	Undefined	Reserved	Reserved	Reserved
EDX	Undefined	Undefined	Reserved	Reserved	Reserved
Function: 8000_0001h					
EAX	Undefined	Undefined	0000_051Xh	0000_052Xh	0000_053Xh
EBX	Undefined	Undefined	Reserved	Reserved	Reserved
ECX	Undefined	Undefined	Reserved	Reserved	Reserved
EDX	Undefined	Undefined	0000_21BFh	0000_21BFh	0000_21BFh
Function: 8000_0002h					
EAX	Undefined	Undefined	2D44_4D41h	2D44_4D41h	2D44_4D41h
EBX	Undefined	Undefined	7428_354Bh	7428_354Bh	7428_354Bh
ECX	Undefined	Undefined	5020_296Dh	5020_296Dh	5020_296Dh
EDX	Undefined	Undefined	6563_6F72h	6563_6F72h	6563_6F72h
Function: 8000_0003h					
EAX	Undefined	Undefined	726F_7373h	726F_7373h	726F_7373h
EBX	Undefined	Undefined	0000_0000h	0000_0000h	0000_0000h
ECX	Undefined	Undefined	0000_0000h	0000_0000h	0000_0000h
EDX	Undefined	Undefined	0000_0000h	0000_0000h	0000_0000h

Table 11. Values Returned By Am486®, Am5$_X$86™, and AMD-K5™ Processors (continued)

Function Register	Am486® and Am5$_X$86™ Processors	AMD-K5™ Processor (Model 0)	AMD-K5™ Processor (Model 1)	AMD-K5™ Processor (Model 2)	AMD-K5™ Processor (Model 3)
Function: 8000_0004h					
EAX	Undefined	Undefined	0000_0000h	0000_0000h	0000_0000h
EBX	Undefined	Undefined	0000_0000h	0000_0000h	0000_0000h
ECX	Undefined	Undefined	0000_0000h	0000_0000h	0000_0000h
EDX	Undefined	Undefined	0000_0000h	0000_0000h	0000_0000h
Function: 8000_0005h					
EAX	Undefined	Undefined	Reserved	Reserved	Reserved
EBX	Undefined	Undefined	0480_0000h	0480_0000h	0480_0000h
ECX	Undefined	Undefined	0804_0120h	0804_0120h	0804_0120h
EDX	Undefined	Undefined	1004_0120h	1004_0120h	1004_0120h