AMD Processor Recognition

Application Note

Publication # **20734**	Revision: **3.12**
Issue Date: **August 2004**	

Trademarks

Contents

List of Tables

List of Figures

Revision History

Date	Revision	Description
August 2004	3.12	Extensive edit, rewrite, and reorganization of previous version; included information for AMD Sempron™ processors Models 6, 8 and 10. This public document includes all information in all previous versions.

Chapter 1 Introduction

The rapid evolution of x86 architecture led to the implementation of new computational features that greatly complicated the microprocessor software development environment. The differences between x86 processors of different vendors, or between one generation of processor and another from the same vendor, were radical enough that software was not immediately portable from platform to platform. It was difficult to develop operating systems, device drivers, and application software without determining the type of processor in a given system. To develop software to run on a variety of platforms, the detection of subtle differences between instruction set implementations and other features from one processor to the next became critical.

It was possible to detect the differences between early types of processors by running subroutines that tested the results of certain instruction sequences. If a particular instruction or feature was not implemented on the processor, alternative paths were selected by which to achieve the desired computational result. Such methods soon became cumbersome. The success of the x86 family of processors rapidly led to a complex landscape of architectural variation and technological innovation. To sift through the myriad possibilities, software required a simple, standardized, extensible method of processor identification. The CPUID instruction was added to the x86 instruction set for this purpose. Beginning with the Am486® DX4 processor, all AMD processors support the CPUID instruction.

Over the years, the implementation of certain details of the CPUID instruction has changed, as have the amount, type and detail of the information the instruction returns. This document covers implementation details on the CPUID instruction in all AMD microprocessors. Where there are differences between processor generations, the presentation is, for the most part, chronologically ordered and includes information on the following AMD microprocessors:

- AM486 Processor, $Am5_x86$® Processor
- AMD-K5 Processors
- AMD-K6® Processors Models 7, 8, and 9
- AMD-K6-2 Processor
- AMD-K6-III Processor
- AMD Athlon™ Processor, AMD Duron™ Processor
- AMD Sempron™ Processor
- AMD Athlon 64 Processor, AMD Opteron™ Processor

1.1 The CPUID Instruction

The CPUID instruction operates by putting a function or extended function number into the EAX register and calling the CPUID instruction. The results of this call are returned in various general purpose registers (EAX, EBX, ECX, EDX, etc.), depending upon the function or extended function specified in the EAX register at the time the CPUID instruction is invoked. A typical sequence is as follows:

```
...
push EAX          ; Save value in EAX.
push EBX          ; Save value in EBX.
push EDX          ; Save value in EBX.
mov EAX,1h       ; Call will be to CPUID standard function 1.
CPUID             ; Instruction call.
...               ; Test results in EAX, EBX, and EDX.
...               ; Continue computation.
```

In this example, this CPUID operation affects the bit settings of the EAX, EBX, and EDX registers. The EAX register bits are set to indicate the (extended) family, (extended) model, and stepping of the processor being tested; the EBX register provides the initial APIC ID, the size of the cache line flushed with the CLFLUSH instruction, and the brand ID; the EDX register settings indicate processor standard features supported. Details will be discussed in the sections that follows.

1.1.1 Static versus Idiosyncratic CPUID Information

Most of the values returned by the various CPUID functions and extended functions have straightforward interpretations. For instance, if standard function 1 returns a value of 1 in EDX register bit 8, the processor supports the CMPXCH8B instruction; extended function 8000_0008h always returns a value that represents the actual maximum virtual- and physical-address sizes in the EAX register. The interpretation of such static values does not depend on any further information and their meanings (although not necessarily their values) are invariant from one processor to the next.

The current document is principally concerned with idiosyncratic CPUID information. This includes information on:

- newly introduced features
- values whose interpretation depends on a correlation between values returned in different fields of one or more registers
- or values whose interpretation depends of correlation with a value found in a table in the processor's data sheet or other documentation.

The latter include information necessary for the BIOS to program the appropriate processor name string, which depends on the proper correlation between brand ID, processor frontside bus speed, and product marketing information. These topics are discussed in detail in the following chapters.

1.1.2 CPUID Functions

The CPUID instruction implements multiple functions, each providing different information about the processor, including the vendor, processor name, model number, revision (stepping), features, cache organization, and memory addressing. A multiple-function approach allows the CPUID instruction to return a complete picture of the processor and its capabilities—more detailed information than could be returned by a single function. The flexibility of the CPUID instruction allows for the addition of new CPUID functions in future generations of processors.

The functions are divided into two types: *standard functions* and *extended functions*. Standard functions are of the form 0000_*xxxx*h, *x* represents a hexadecimal value; extended functions are of the form 8000_*xxxx*h.

Standard Functions

Standard functions provide a simple method for software to access information common to all x86 ISA processors. Although the particular information returned may vary, all AMD processors provide the following CPUID standard functions; details are discussed in detail in the following chapters)

Table 1. CPUID Standard Functions

Standard Function	Register	Information Returned
0h	EAX	Largest CPUID standard function number implemented on this processor
	EBX	Processor vendor
	ECX	
	EDX	
1h	EAX	Processor signature
	EBX	Initial APIC ID, CLFLUSH size and 8-bit brand ID
	EDX	Standard feature support

Extended Functions

Extended functions provide information on vendor-specific processor features (such as those by AMD). The list of available extended functions has grown with each new generation of AMD processors. Currently, AMD processors support the extended functions specified in Table 2 on page 14.

Table 2. CPUID Extended Functions

Extended Function	Register(s) Affected	Information Returned
8000_0000h	EAX	Largest CPUID extended function number implemented on this processor
	EBX	Processor vendor
	ECX	
	EDX	
8000_0001h	EAX	Processor signature
	EBX	12-bit brand ID
	ECX	AMD extended feature support (bits 32–63)
	EDX	AMD extended feature support (bits 0–31)
8000_0002h	EAX	Processor name string
	EBX	
	ECX	
	EDX	
8000_0003h	EAX	Processor name string
	EBX	
	ECX	
	EDX	
8000_0004h	EAX	Processor name string
	EBX	
	ECX	
	EDX	
8000_0005h	EAX	TLB information for 2-Mbyte and 4-Mbyte pages
	EBX	TLB information for 4-Kbyte pages
	ECX	L1 data cache information
	EDX	L1 instruction cache information

Table 2. CPUID Extended Functions

Extended Function	Register(s) Affected	Information Returned
8000_0006h	EAX	L2 TLB information for 2-Mbyte and 4-Mbyte pages
	EBX	L2 TLB information for 4-Kbyte pages
	ECX	L2 cache information
8000_0007	EDX	Advanced power management features
8000_0008h	EAX	Long-mode address sizes
8000_0009h–8000_0018h	N/A	Reserved for future use

1.2 CPUID Privilege Level

Software operating at any privilege level can execute the CPUID instruction.

1.3 Additional Information

The details of the CPUID instruction and the CPUID standard and extended functions are provided in the "CPUID" section in the *AMD64 Architecture Programmer's Manual Volume 3: General Purpose and System Instructions*, order# 24594.

Chapter 2 Summary of CPUID Functions and Values

This chapter presents a brief summary of currently supported CPUID functions and extended functions and the registers they affect. For more complete discussion of these topics, see "CPUID" in the *AMD64 Architecture Programmer's Manual Volume 3: General Purpose and System Instructions*, order# 24594.

2.1 CPUID Standard Function 0 and Extended Function 8000_0000h

CPUID standard function 0 and extended function 8000_0000h return similar types of information.

Both functions return a string in the EBX, EDX, and ECX registers that represents the processor vendor name. For AMD processors, this string is always "AuthenticAMD". For a full discussion of the vendor name format, see "CPUID" in the *AMD64 Architecture Programmer's Manual Volume 3: General Purpose and System Instructions*, order# 24594.

All reserved bits read as zero (RAZ). If a field is not supported on a particular processor, all of the bits in that field will be returned as zeros. The CPUID instruction attempts to find all information associated with a given function number, even if the function does not exist. An attempt to call CPUID with a function number higher than that returned by standard function 0 or extended function 1 will result in filling the EAX, EBX, ECX, and EDX registers with zeros. The CPUID instruction never generates an exception.

2.1.1 Highest Standard Function Number Available

Standard function 0 returns the highest *standard* function number supported by the processor. For AMD processors, the highest standard function number is always 1 (standard functions 0 and 1 are the only standard functions supported by all AMD processors).

2.1.2 Highest Extended Function Number Available

Extended function 8000_0000h returns the highest *extended* function number supported by the processor. If it is not possible to assume that a given application is to run on a particular processor, extended function 8000_0000h should first test for the highest extended function number available on the processor. (The AMD-K5 processor model 0, Am486®DX4 microprocessor, and $Am5_x86$® processors did not support any CPUID extended functions; while the AMD-K5 processor models 1, 2, and 3 and the AMD-K6® processor models 6, 7, and the AMD-K6-2 processor model 8 supported CPUID extended functions 8000_0000h through 8000_0005. Subsequent AMD processor offerings have introduced additional CPUID extended functions.)

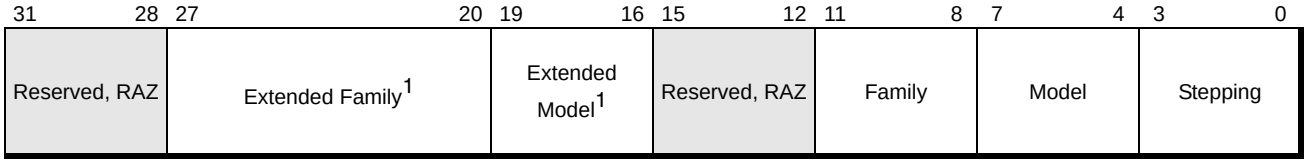
2.2 Standard Function 1 and Extended Function 8000_0001h

CPUID standard function 1 and extended function 8000_0001h return similar types of information. Both function identify supported processor features, as well as processor family, model, and version information. They also return the brand ID, which is used to determine the processor name string. Additionally, standard function 1 returns the initial APIC ID and CLFLUSH size.

2.2.1 Processor Signature

Each processor belongs to a certain family of processors (AMD Athlon™, AMD Duron™, AMD Sempron™, AMD Athlon 64, and AMD Opteron™ processors) and within a family, several models may exist (AMD Athlon 64 Model 4, AMD Opteron Model 8). Additionally, each processor has a stepping or revision number. The family number, model number, and stepping number form the *processor signature*. The processor signature is important for various purposes. For some AMD processors, the BIOS uses the processor signature to determine valid operating combinations of processor core voltage and frequency (P-states) used by AMD PowerNow!™ technology and Cool‘n’Quiet™ technology. The processor signature is also used to determine whether a microcode patch is applicable to a given processor.

After a call to CPUID with standard function 1 or extended function 8000_0001h, the value in the EAX registers represents the processor signature. Figure shows the format of the processor signature returned in EAX.



Bits	Mnemonic	Definition
31–28	Reserved	RAZ
27–20	Extended Family[1]	If (Family = 0Fh), provides additional family information
19–16	Extended Model[1]	If (Family = 0Fh), provides additional model information
15–12	Reserved	RAZ
11–8	Family	Processor/Instruction Family
7–4	Model	Processor Model
3–0	Stepping	Processor Stepping (revision)

Note: *1. These bits are only returned for the AMD Athlon™ 64 and AMD Opteron™ processors. For the AMD Athlon and AMD Duron™ processors these bits are reserved.*

Figure 1. Processor Signature Fields in EAX returned by Standard Function 0000_0001h and Extended Function 8000_0001h

Stepping

Each processor is assigned a stepping (or version) number. The latest stepping information is found in the revision guide for the particular processor.

Family and Model Number

Each processor family is assigned a 4-bit family number. For instance, processors employing the AMD64 architecture belongs to family 0Fh (1111b). Within a family, processors are further classified by a 4-bit model number. The AMD Athlon™ 64 Model 4 has model number 04h (0100b), while the AMD Opteron™ Model 5 has model number 05h (0101b).

Note: *On all AMD-K6® and AMD Athlon, AMD Sempron™ Model 8, and AMD Duron™ processors, the family number returned by extended function 8000_0001h is equal to the family number returned by standard function 1 + 1. For instance, for the AMD-K6 processor (Model 8), the family number returned by standard function 1 is 0101b (05h) and that returned by extended function 8000_0001h is 0110b (06h); for the AMD Athlon processor Model 6, the standard function returns family 0110b (06h) and the extended function returns 0111b (07h). While differences are small, they are important; for certain purposes, the use of the family number returned by the extended function may be required, rather than that returned by the standard function.*

On all AMD Athlon 64, AMD Sempron Model 10, and AMD Opteron processors, and all AMD processors produced prior to AMD-K6, the values returned for the processor signature are identical for both standard function 1 and extended function 8000_0001h.

Extended Family and Extended Model

The extended family and extended model fields, in conjunction with the family and model fields, expand the family and model numbering scheme. The *effective family* and *effective model* numbers are calculated from the values in the extended family, family, extended model, and model fields.

It is important to understand how the extended family and extended model fields combine with the family and model fields to produce the effective family and effective model numbers. Software must use the computed 8-bit effective family and effective model numbers when displaying or using the processor's family and model numbers. Failure to do so results in the display or use of incorrect values.

If the family field (EAX bits 11–8) contains the value 1111b (0Fh):

- the effective family is calculated by adding the 4-bit family to the 8-bit extended family.
- the effective model is calculated by shifting the 4-bit extended model field four bits to the left and adding the result to the 4-bit model number field.

If the family field is less than 1111b (0Fh):

- the effective family is equal to the value in the family field.
- the effective model is equal to the value in the model field.

The following code sample illustrates how to compute 8-bit effective family and effective model numbers:

```
;-----------------------------------------------------------------------------
; amdGetProcessorFamily:
;
; Returns:
;     AX = Processor EXTENDED FAMILY and FAMILY bits as one 12-bit value
;
;      Uses:  EAX
; Preserves:  EBX, ECX, EDX
;-----------------------------------------------------------------------------
amdGetProcessorFamily proc near
                               ;Family bits returned in:
   call    amdCpuidValue32       ; eax[27:20] and eax[11: 8]
   shr     eax, (20-16)          ; eax[23:16] and eax[ 7: 4]
   shl     ax, (15-7)            ; eax[23:16] and eax[15:12]
   shr     ax, 12                ; eax[11: 4] and eax[ 3: 0]
   ret
amdGetProcessorFamily endp
;-----------------------------------------------------------------------------

;-----------------------------------------------------------------------------
; amdGetProcessorModel:
;
; Returns:
;     AX = Processor EXTENDED MODEL and MODEL bits as one 12-bit value
;
;      Uses:  EAX
; Preserves:  EBX, ECX, EDX
;-----------------------------------------------------------------------------
amdGetProcessorModel proc near
                                ;Model bits returned in:
   call     amdCpuidValue32      ; eax[19:16] and eax[ 7: 4]
   shl      ax,(15-7)            ; eax[19:16] and eax[15:12]
   shr      ax,12                ; eax[11: 4] and eax[ 3: 0]
   ret
amdGetProcessorModel endp
;-----------------------------------------------------------------------------

;-----------------------------------------------------------------------------
; amdCpuidValue32:
;
; Returns:
;    EAX = 32-bit value returned CPUID Function 1 in EAX
;
;      Uses:  EAX
; Preserves:  EBX, ECX, EDX
;-----------------------------------------------------------------------------
amdCpuidValue32 proc near
   push     edx
```

```
   push     ecx
   push     ebx
   mov      eax,1                ;
   CPUID                        ;Return eax = 32-bit CPUID value
   pop      ebx
   pop      ecx
   pop      edx
   ret
amdCpuidValue32 endp
;------------------------------------------------------------------------------
```

Table 3 shows the processor signatures for AMD processors up to effective family 6:

Table 3. Processor Signatures for AMD Athlon™, AMD Duron™, and AMD Legacy Processors (CPUID Function 1)

Processor	Extended Family [27:20]	Extended Model [19:16]	Family [11:8]	Model [7:4]	Stepping ID[2] [3:0]
Am486® and Am5$_x$86® Processors	0	0	0100b (4h)	yyyy[1]	xxxx
AMD-K5 Model 0	0	0	0101b (5h)	0000b (0h)	xxxx
AMD-K5 Model 1	0	0	0101b (5h)	0001b (1h)	xxxx
AMD-K5 Model 2	0	0	0101b (5h)	0010b (2h)	xxxx
AMD-K5 Model 3	0	0	0101b (5h)	0011b (3h)	xxxx
AMD-K6® Model 6	0	0	0101b (5h)	0110b (6h)	xxxx
AMD-K6 Model 7	0	0	0101b (5h)	0111b (7h)	xxxx
AMD-K6®-2 Model 8	0	0	0101b (5h)	1000b (8h)	xxxx
AMD-K6®-III Model 9	0	0	0101b (5h)	1001b (9h)	xxxx
AMD Athlon™ Model 1	0	0	0110b (6h)	0001b (1h)	xxxx
AMD Athlon Model 2	0	0	0110b (6h)	0010b (2h)	xxxx
AMD Duron™ Model 3	0	0	0110b (6h)	0011b (3h)	xxxxb
mobile AMD Duron Model 3	0	0	0110b (6h)	0011b (3h)	xxxxb
AMD Athlon Model 4	0	0	0110b (6h)	0100b (4h)	xxxxb
AMD Athlon MP Model 6	0	0	0110b (6h)	0110b (6h)	xxxxb
AMD Athlon XP Model 6	0	0	0110b (6h)	0110b (6h)	xxxxb
Mobile AMD Athlon 4 Model 6	0	0	0110b (6h)	0110b (6h)	xxxxb
AMD Duron Model 6	0	0	0110b (6h)	0110b (6h)	xxxxb
Mobile AMD Duron Model 6	0	0	0110b (6h)	0110b (6h)	xxxxb
AMD Duron Model 7	0	0	0110b (6h)	0111b (7h)	xxxxb
Mobile AMD Duron Model 7	0	0	0110b (6h)	0111b (7h)	xxxxb

Notes:

1. *Contact your AMD representative for model identifier information.*
2. *Stepping ID may change. Consult the appropriate processor revision guide, or contact your AMD representative for the latest stepping information. AMD Athlon processors of the same model numbers share the same revision guide. AMD Duron processors of the same model number share the same revision guide.*

Table 3. Processor Signatures for AMD Athlon™, AMD Duron™, and AMD Legacy Processors (CPUID Function 1) (Continued)

Processor	Extended Family [27:20]	Extended Model [19:16]	Family [11:8]	Model [7:4]	Stepping ID[2] [3:0]
AMD Athlon XP Model 8	0	0	0110b (6h)	1000b (8h)	xxxxb
AMD Athlon MP Model 8	0	0	0110b (6h)	1000b (8h)	xxxxb
AMD Sempron™ Model 8	0	0	0110b (6h	1000b (8h)	xxxxb
Mobile AMD Athlon XP–M Model 8	0	0	0110b (6h)	1010b (Ah)	xxxxb
Mobile AMD Athlon XP–M (LV) Model 8	0	0	0110b (6h)	1010b (Ah)	xxxxb
AMD Athlon XP Model 10	0	0	0110b (6h)	1010b (Ah)	xxxxb
AMD Sempron Model 10	0	0	0110b (6h)	1010b (Ah)	xxxxb
AMD Athlon MP Model 10	0	0	0110b (6h)	1010b (Ah)	xxxxb
Mobile AMD Athlon XP–M Model 10	0	0	0110b (6h)	1010b (Ah)	xxxxb
Mobile AMD Athlon XP–M (LV) Model 10	0	0	0110b (6h)	1010b (Ah)	xxxxb

Notes:

1. *Contact your AMD representative for model identifier information.*
2. *Stepping ID may change. Consult the appropriate processor revision guide, or contact your AMD representative for the latest stepping information. AMD Athlon processors of the same model numbers share the same revision guide. AMD Duron processors of the same model number share the same revision guide.*

2.2.2 Feature Support—Standard Function 1 and Extended Function 8000_0001h

Standard function 1 sets EDX register bits representing each standard processor feature supported; extended function 8000_0001h sets EDX and ECX register bits representing each extended feature supported.

Standard Feature Support

For a full discussion of standard feature support see “CPUID” in the *AMD64 Architecture Programmer’s Manual Volume 3: General Purpose and System Instructions*, order# 24594. Currently, none of the standard feature bits require special rules of interpretation.

Extended Feature Support

Some EDX register bits are subject to idiosyncratic interpretation, depending on the processor family, model, and other factors. These are summarized in Table 4 on page 23:

Table 4. Extended Feature EDX Bit Interpretation

EDX Register Bit	Bit Name	FSB	Processor Family	Value	Meaning
19	ECC	≤ 266FSB	AMD Athlon™ MP Processor	1	Multiprocessor support
		≥ 333FSB	AMD Sempron™ Processor	1	AMD Sempron processor brand feature support

Enabling SSE Support

All AMD Athlon™ XP and AMD Sempron™ processors fully support the SSE instruction set extensions. This support is disabled by default; so, the BIOS must enable the feature. See the *AMD Athlon™ and AMD Duron™ Processor BIOS, Software, and Debug Developers Guide*, order# 21656. Motherboards will not pass AMD validation or be posted on the AMD recommended motherboard Web site, if SSE is not enabled by the BIOS for model 6 and above processors.

2.2.3 Brand ID—Standard Function 1 and Extended Function 8000_0001

AMD processor family number 0Fh and later processors support a brand ID value. CPUID standard function 1 and extended function 8000_0001h both return value in the EBX register. The brand ID identifies a processor having the set of features that are unique to a specific brand. This identifier maps to a name string value that identifies the processor in various program displays (such as that which appears during POST on bootup, or that on the General tab in the System dialog box under Microsoft® Windows®). Standard function 1 returns an 8-bit brand ID and extended function 8000_0001 returns a 12-bit brand ID.

The 8-bit and 12-bit brand ID fields were introduced with the AMD Athlon 64 and AMD Opteron processors. For all previous processors, these fields are reserved but read as zero (RAZ).

8-Bit Brand ID

On certain models, CPUID standard function 1 sets EBX bits 0–7 to represent the brand ID. If the 8-bit brand ID is non-zero, the 12-bit brand ID is zero. If the 8-bit brand ID is zero and the 12-bit brand ID is zero, the processor is an engineering sample.

12-Bit Brand ID

On certain models, CPUID extended function 0 sets EBX bits 0–11 to represent the brand ID. If the 12-bit brand ID is non-zero, the 8-bit brand ID is zero. If the 12-bit brand ID is zero and the 8-bit brand ID is zero, the processor is an engineering sample.

For AMD Athlon 64 and AMD Opteron processors and the AMD Sempron™ processor model 10,

the BIOS uses the 8-bit or 12-bit brand ID to program the ASCII processor name string (returned by CPUID functions 8000_0002h, 8000_0003h, and 8000_0004h).

The following code example shows how to read both the 8-bit and 12-bit Brand ID (whichever is applicable), returning a value in the 12-bit format ready for name string look-up.

```
;-----------------------------------------------------------------------------
; amdGetBrandIDValue:
;
; Returns:
;   ZF=0 if BRANDID != 0
;      1 if BRANDID  = 0
;   bx = BRANDID value (in 12-bit brand ID format)
;
;     ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___
;    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
; bx | - | - | - | - | - | - | - | - |b10| b9| b8| b4| b3| b2| b1| b0|  8-bit
;    |___|___|___|___|___|___|___|___|___|___|___|___|___|___|___|___| BRANDID
;    15            12 11            8 7             4 3             0
;
;     ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___   8-bit
;    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | BRANDID
; bx | 0 | 0 | 0 | 0 | 0 |b10| b9| b8| 0 | 0 | 0 | b4| b3| b2| b1| b0| in 12-bit
;    |___|___|___|___|___|___|___|___|___|___|___|___|___|___|___|___| format
;    15            12 11            8 7             4 3             0
;
;     ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___
;    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
; bx | - | - | - | - |b11|b10| b9| b8| b7| b6| b5| b4| b3| b2| b1| b0| 12-bit
;    |___|___|___|___|___|___|___|___|___|___|___|___|___|___|___|___| BRANDID
;    15            12 11            8 7             4 3             0
;
;-----------------------------------------------------------------------------
amdGetBrandIDValue proc near
        push    eax                     ;
        push    ebx                     ;Save registers overwritten by
        push    ecx                     ;  CPUID instruction
        push    edx                     ;
        _CPUID  1                       ;Fn 1 Rtns 8-bit BRANDID in bl;
                                        ; if 0, get 12-bit value
        xor     bh, bh                  ;
        shl     bx, 3                   ;
        shr     bl, 3                   ;bx = 12-bit compatible BrandID
        or      bx, bx                  ;If 0, check for 12-bit BRANDID
        jnz     @F                      ;Jump if 8-bit Brand ID (OLD CPUs)
;
; 8-bit BRANDID is 0, check for 12-bit BRANDID
;
        _CPUID  80000001h               ;BX = 12-bit BRANDID
@@:
;
        pop     edx
```

```
        pop     ecx
        mov     ax, bx                  ;save return value
        pop     ebx
        mov     bx, ax                  ;return values in bx
        pop     eax
        and     bx, 0FFFh               ;Set/clear ZF for return
        ret
amdGetBrandIDValue endp
;------------------------------------------------------------------------
;------------------------------------------------------------------------
;                Macro Definition:  _CPUID  <function>
;------------------------------------------------------------------------
_CPUID macro    cpuidindex
    IFNB <cpuidindex>
        mov     eax, cpuidindex
    ENDIF
        db      0Fh, 0A2h       ; {eax,ebx,ecx,edx} <-- CPUID Fn[eax]
        endm
```

For details on how the brand ID is used to construction the processor name string, Section 3.4.1, "Brand ID", on page 43.

2.2.4 Initial APIC ID, CLFLUSH size—Standard Function 1

On AMD processor family 0Fh, and later, processors introduced an 8-bit initial APIC ID field and an 8-bit CLFLUSH size field, in addition to the 8-bit brand ID.

Initial APIC ID

The initial APIC ID is an 8-bit value in EBX bits 31-24. This is the initial value of the processor's local APIC physical ID register. Bits 26–24 of this field represent the Northbridge NodeID and bits 31–27 represent the CPU number within the node.

CLFLUSH Size

The CLFLUSH size is an 8-bit value in EBX bits 15-8. This field specifies the size in quadwords of the cache line that is flushed by the CLFLUSH instruction.

For detailed information on the initial APIC ID and CLFLUSH size see "CLFLUSH" and "CPUID" in the *AMD64 Architecture Programmer's Manual Volume 3: General Purpose and System Instructions*, order# 24594.

2.3 Processor Name String—Extended Functions 8000_0002h–8000_0004h

CPUID extended 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 EAX, EBX, ECX, and EDX registers to return an ASCII string of up to

48 characters in little-endian byte order. 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. Functions 8000_0003h and 8000_0004h return the second and third group of 16 characters in a similar fashion. The ASCII NUL character (00h) indicates the end of the processor name string.

These functions first became available on the AMD-K5 Model 1. Initially, the processor name string was hardwired onto the processor; all AMD-K5 processors Models 1 through Model 3 returned the string `"AMD-K5(tm) Processor"`.

The AMD Athlon™ processor model 2 added the name string model specific registers (MSRs) C001_0030h–C001_0035h. These model specific registers were initialized to a *default* name string on reset. (The default name string for AMD Athlon processors is `"AMD Athlon(tm) Processor"`.) This innovation allowed the BIOS to change the value of the name string in the MSRs through the WRMSR (Write to Model Specific Register) instruction. (See “WRMSR” in the *AMD64 Programmer’s Manual Volume 3: General Purpose and System Programming*, order #24594.) This facilitated retrieval of the name string for display and other purposes, since the CPUID extended functions 8000_0002h–8000_0004h directly access the value in the name string MSRs. Table 6 in Chapter 3, "Programming the Processor Name String," provides a complete list of default name strings for all AMD processors.

Beginning with the AMD Athlon 64 and AMD Opteron™ processors, MSRs C001_0030h–8000_0035h are initialized to zeros (ASCII NULs) on reset. There is no default name string and the correct name string must be determined by the BIOS. The selection of the appropriate name string is a more or less complicated process and can depend on a variety of information, such as processor signature, front-side bus speed, and brand ID.

Software must be running at privilege level 0 to write to MSRs C0001_0030h–C001_0035h.

See Chapter 3, "Programming the Processor Name String," for more detailed information about programming and using the processor name string.

2.4 L1 Cache and TLB Information—Extended Function 8000_0005h

Extended function 8000_0005h returns information about the processor L1 TLBs and caches in the EAX, EBX, ECX, and EDX registers.

TLB Information for 2-Mbyte and 4-Mbyte Pages

Figure 2 shows the format of TLB information for 2-Mbyte and 4-Mbyte pages returned in the EAX register following execution of CPUID extended function 8000_0005h.

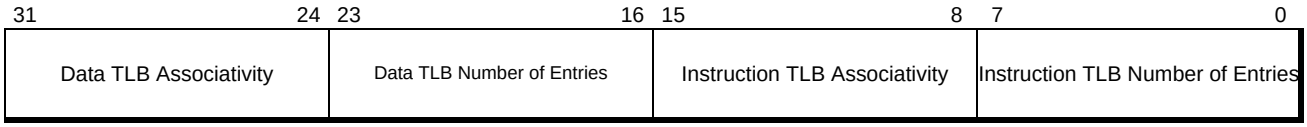


Figure 2. TLB Information for 2-Mbyte and 4-Mbyte Pages returned in EAX by Extended Function 8000_0005h

TLB Information for 4-Kbyte Pages

Figure 3 on page 27 shows the format TLB information for 4-Kbyte pages returned in the EBX register following execution of CPUID extended function 8000_0005h.

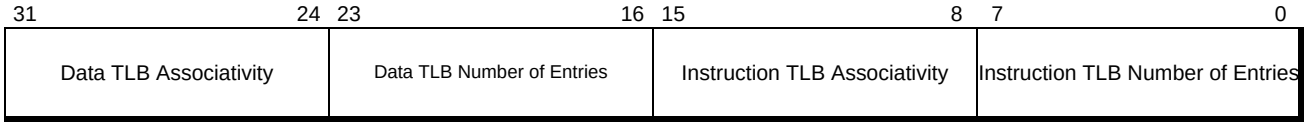


Figure 3. TLB Information for 4-Kbyte Pages returned by Extended Function 8000_0005h in EBX

L1 Data Cache Information

Figure 4 shows the format of the L1 data cache information for 4-Kbyte pages returned in the ECX register following execution of CPUID extended function 8000_0005h.

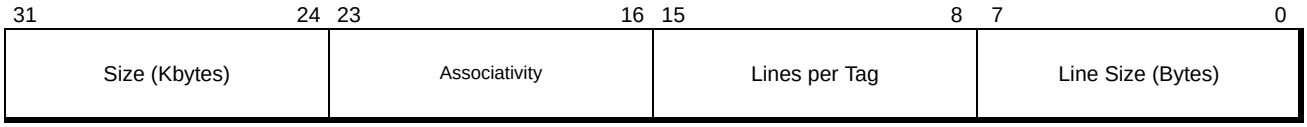


Figure 4. L1 Data Cache Information returned in ECX by Extended Function 8000_0005h

L1 Instruction Cache Information

Figure 5 shows the format of the L1 Instruction cache information returned in the EDX register following execution of CPUID extended function 8000_0005h.

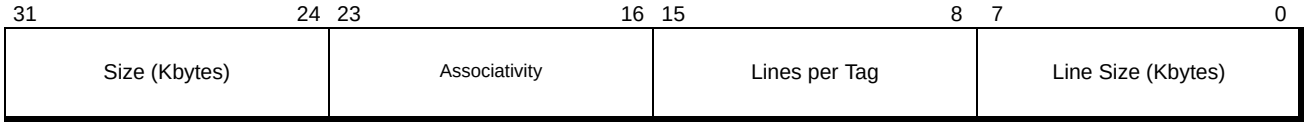


Figure 5. L1 Instruction Cache Information Returned in EDX by Extended Function 8000_0005h

Size for L1 Caches and TLBs. The size fields for the L1 data cache, L1 instruction cache, L1 data TLB, and L1 instruction TLB are all eight bits wide. The value in this field represents the size of the cache in Kbytes.

Associativity for L1 Caches and L1 TLBs. The associativity fields for the L1 data cache, L1 instruction cache, L1 data TLB, and L1 instruction TLB are all eight bits wide. Except for 00h (reserved) and FFh (Full), the number returned in the associativity field represents the actual number of ways. with a range of 01h through FEh. For example, a returned value of 02h indicates two-way associativity and a returned value of 04h indicates four-way associativity.

Lines per Tag for L1 Caches and TLBs. The lines per tag fields for the L1 data cache, L1 instruction cache, L1 data TLB, and L1 instruction TLB are all eight bits wide. The value in this field represents the actual number of cache lines per tag.

Line Size for L1 Caches and TLBs. The line size fields for the L1 data cache, L1 instruction cache, L1 data TLB, and L1 instruction TLB are all eight bits wide. The value in this field represents the size of a cache line in Kbytes.

2.5 L2 Cache Information—Extended Function 8000_0006h

Extended function 8000_0006h returns information about the integrated L2 cache.

2.5.1 L2 TLB Information for 2-Mbyte and 4-Mbyte Pages

Figure 6 shows the format of the L2 TLB information for 2-Mbyte and 4-Mbyte pages returned in the EAX register following execution of CPUID extended function 8000_0006h.

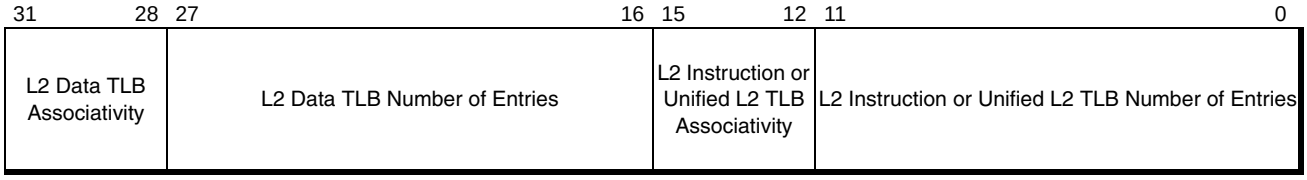


Figure 6. L2 TLB Information for 2-Mbyte and 4-Mbyte Pages Returned in EAX by Extended Function 8000_0006h

2.5.2 L2 TLB Information for 4-Kbyte Pages

Figure 7 shows the format of the L2 TLB information for 4-Kbyte pages returned in the EBX register following execution of CPUID extended function 8000_0006h.

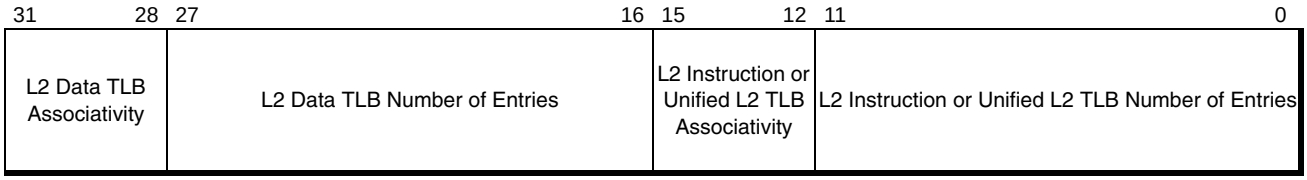


Figure 7. L2 TLB Information for 4-Kbyte Pages Returned in EBX by Extended Function 8000_0006h

2.5.3 L2 Cache Information

Figure 8 shows the format of the L2 TLB information for 4-Kbyte pages returned in the EBX register following execution of CPUID extended function 8000_0006h.

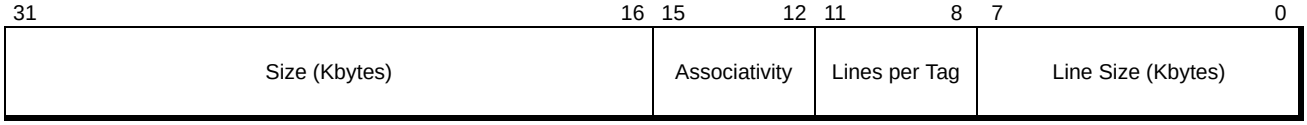


Figure 8. L2 TLB Information returned in ECX by Extended Function 8000_0006h

Associativity for L2 Caches and L2 TLBs. The associativity fields for the L2 cache, L2 data TLB, and L2 instruction TLB are four bits wide as specified in.Table 5.

Table 5. Associativity Values for L2 Caches and TLBs

Bits 15–12	Associativity
0000b	L2 off
0001b	Direct mapped
0010b	2-way
0011b	*Reserved*
0100b	4-way
0101b	*Reserved*
0110b	8-way
0111b	*Reserved*
1000b	16-way
1001b	*Reserved*
1010b	*Reserved*
1011b	*Reserved*
1100b	*Reserved*
1101b	*Reserved*
1110b	*Reserved*
1111b	Full

2.6 Advanced Power Management Features

Extended function 8000_0007h returns information about the advanced power management features supported by the processor. Figure 9 shows the format of the EDX register returned by CPUID extended function 8000_0007h.

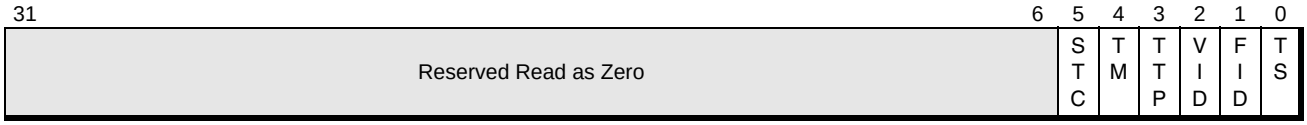


Figure 9. Advanced Power Management Features Returned in EDX by Extended Function 8000_0007h

Thermal Sensor (TS). EDX bit 0 indicates the availability of a thermal sensor.

Frequency ID control (FID). EDX bit 1 indicates the availability of FID control.

Voltage ID Control (VID). EDX bit 2 indicates the availability of VID control.

Thermal Trip (TTP). EDX bit 3 indicates the availability of thermal trip control.

Thermal Monitor (TM). EDX bit 4 indicates the presence of an external thermal monitor.

Software Thermal Control (STC). EDX bit 5 indicates whether software (such as OS or other) can control the thermal state of the processor.

For more details, see the *BIOS Requirements for AMD PowerNow!™ Technology Application Note,* order# 25264, and the *BIOS Requirements for AMD PowerNow!™ Technology Low-Power Desktop Processors Application Note,* order# 25541.

2.7 Address Sizes—Extended Function 8000_0008h

Extended function 8000_0008h returns maximum supported virtual-address and physical-address sizes in the EAX register.

Figure 10 shows the format of the address size information returned in the EAX register following execution of CPUID extended function 8000_0005h.

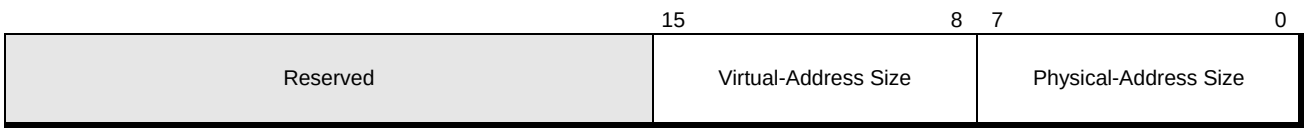


Figure 10. Long-Mode Address Sizes

Physical-Address Size. EAX bits 0–7 represent the actual numeric value of the maximum physical-address that the process can address.

Virtual-Address Size. EAX bits 8–15 represent the actual numeric value of the maximum virtual-address that the process can address.

2.7.1 Vendor-Specific Feature Flags—Extended Functions 8000_0009h–8000_0018h

Functions 8000_0009h through 8000_0018h are reserved for future expansion of vendor-specific feature flags. Each function is reserved for a specific vendor.

Features that are compatible with extensions made by other vendors will be reported in the feature function reserved for that vendor. So, to check for a certain feature across multiple vendors, software need only check the vendor-specific feature flags for the vendor that first implemented that feature.

Currently, Intel features are reported by function 0000_0001h and AMD features are reported by function 8000_0001h. Functions 8000_0009h and 8000_000Ah are reserved for expansion by Intel and AMD respectively.

- 8000_0009h reserved for Intel feature flag expansion
- 8000_000Ah reserved for AMD feature flag expansion
- 8000_000B–8000_0018h reserved for vendors to be determined

Chapter 3 Programming the Processor Name String

When an x86 processor-based computer boots, the BIOS displays the processor name string and other information about the system. This character string appears in a variety of contexts that provide operating system and application software configuration information. This section, for BIOS programmers, describes how to program the *processor name string* that is returned by the CPUID instruction with extended functions 8000_0002h–8000_0004h.

3.1 The Name String on AMD Legacy Processors

The earliest AMD processors—the AM486® processor, AM5$_x$86® processor, and AMD-K5 processor model 0—do not support the processor name string. Mechanisms completely external to the processor (BIOS, OS and other) are responsible for displaying the correct AMD specified name string. At the time these processors appeared on the market, the processor landscape was simple enough that the name string could be programmed into the BIOS and operating system software by reference to the processor signature returned by CPUID standard function 1.

The AMD-K5 processor models 1, 2, and 3 added CPUID extended functions 8000_0002h–8000_0004h, but a name string is hardwired on the processor and cannot be altered. This practice prevailed through the advent of the AMD Athlon™ processor model 1.

Beginning with the AMD Athlon processor model 2, six 64-bit name string model specific registers (MSRs) C001_0030h through C001_0035h were incorporated onto the processor. Throughout the life of the AMD Athlon and AMD Duron™ processors, these registers have been initialized on reset to a default name string value, either `"AMD Athlon(tm) Processor"` or `"AMD Duron(tm) Processor"`, as appropriate. These name string MSRs are programmable; software operating at privilege level 0 (such as the BIOS or OS) may modify or replace the default name string in MSRs C001_0030h–C001_0035h, as required. CPUID extended function 8000_0002h–8000_0004h read the contents of the six name string MSRs, providing efficient high-level access to the processor name string by software operating at any privilege level.

Table 6 shows the default name strings for all AMD processors through the AMD Athlon and AMD Sempron™ Processors.

Table 6. Default Name Strings for AMD486® and AM5$_x$86® Processors through AMD Athlon™ and AMD Sempron™ Processors

Processor	Default Name String	Modifiable?
AM486® and AM5$_x$86®	Undefined	N/A
AMD-K5 Processor Model 0	Undefined	N/A

Table 6. Default Name Strings for AMD486® and AM5$_x$86® Processors through AMD Athlon™ and AMD Sempron™ Processors

AMD-K5 Processor Model 1	"AMD-K5(tm) Processor"	No
AMD-K5 Processor Model 2	"AMD-K5(tm) Processor"	No
AMD-K5 Processor Model 3	"AMD-K5(tm) Processor"	No
AMD-K6® Processor Model 6	"AMD-K6(tm) w/ multimedia extensions"	No
AMD-K6 Processor Model 7	"AMD-K6(tm) w/ multimedia extensions"	No
AMD-K6-2 Processor Model 8	"AMD-K6(tm) 3D Processor"	No
AMD-K6-III Processor Model 9	"AMD-K6(tm) 3D+ Processor"	No
AMD Athlon™ Processor Model 1	"AMD-K7(tm) Processor"	No
AMD Athlon Processor Model 2	"AMD Athlon(tm) Processor"	Yes
AMD Duron™ Processor Model 3	"AMD Duron(tm) Processor"	Yes
Mobile AMD Duron Processor Model 3	"AMD Duron(tm) Processor"	Yes
AMD Athlon Processor Model 4	"AMD Athlon(tm) Processor"	Yes
AMD Athlon MP Model 6	"AMD Athlon(tm) Processor"	Yes
AMD Athlon XP Model 6	"AMD Athlon(tm) Processor"	Yes
Mobile AMD Athlon 4 Processor Model 6	"AMD Athlon(tm) Processor"	Yes
AMD Duron Processor Model 6	"AMD Duron(tm) Processor"	Yes
Mobile AMD Duron Processor Model 6	"AMD Duron(tm) Processor"	Yes
AMD Duron Processor Model 7	"AMD Duron(tm) Processor"	Yes
Mobile AMD Duron Processor Model 7	"AMD Duron(tm) Processor"	Yes
AMD Duron Processor Model 8	"AMD Duron(tm) Processor'	Yes
AMD Athlon XP Model 8	"AMD Athlon(tm) Processor"	Yes
AMD Athlon MP Model 8	"AMD Athlon(tm) Processor"	Yes
Mobile AMD Athlon XP-M Model 8	"AMD Athlon(tm) Processor"	Yes
Mobile AMD Athlon XP-M(LV) Model 8	"AMD Athlon(tm) Processor"	Yes
AMD Athlon XP Model 10	"AMD Athlon(tm) Processor"	Yes
AMD Athlon MP Model 10	"AMD Athlon(tm) Processor"	Yes
Mobile AMD Athlon XP-M Model 10	"AMD Athlon(tm) Processor"	Yes
Mobile AMD Athlon XP-M(LV) Model 10	"AMD Athlon(tm) Processor"	Yes
AMD Sempron™ Model 8	"AMD Athlon(tm) Processor"	Yes
AMD Sempron Model 10	"AMD Athlon(tm) Processor"	Yes

3.2 Default Name String on AMD Athlon™ 64 and AMD Opteron™ Processors

Like the AMD Athlon and AMD Duron™ processors, AMD Athlon™ 64 and AMD Opteron™ processors support MSRs C001_0030h through C001_0035h. Unlike the AMD Athlon and AMD Duron processors, the AMD Athlon 64 and AMD Opteron processors do *not* initialize the name string MSRs on reset with a default name string; rather, the name string MSRs are initialized with 48 ASCII NUL characters (all zeros). The BIOS must determine the appropriate name string for the particular processor and program it into the MSRs. (The WRMSR instruction can only be used by software operating at a privilege level of 0.) CPUID extended functions 8000_0002h, 8000_0003h, and 8000_0004h can then read the name string. (The CPUID instruction does not return an exception if MSRs C001_0030h–C001_0035h contain no name string. The CPUID name string functions simply return a string of ASCII NULs in the EAX, EBX, ECX, and EDX registers.) See section 2.3 “Processor Name String—Extended Functions 8000_0002h–8000_0004h” on page 25 for details concerning the name string functions.

3.3 Processor Name String Code Requirements

All AMD Athlon, AMD Duron, AMD Sempron™, AMD Athlon 64, and AMD Opteron processors are assigned an AMD approved name string that the BIOS must use to initialize MSRs C001_0030h–C001_0035h. The BIOS and operating system software display this processor name string and model number whenever they display processor information during boot up.

Note: *To pass AMD validation for inclusion on the AMD Recommended Motherboard Website, motherboards must incorporate a BIOS that programs the proper processor name string and model number.*

Determining the correct name string depends on several pieces of information, depending on the processor family. The assigning process is different for AMD Athlon, AMD Duron, and some AMD Sempron processors than for AMD Athlon 64 and AMD Opteron processors.

3.3.1 Name String Format

A processor name string, such as `"mobile AMD Athlon(tm) XP-M 1800+", ` consists of two principal components:

- Processor brand (Example: `"mobile AMD Athlon(tm) XP-M "`)
- Model number (Example: `"1800+"`)

The way the name string constituents are determined varies depending on several factors, including processor type (mobile, workstation, server), frontside bus speed of the processor, processor frequency, and L2 cache size.

Some processor models do not have a model number.

3.3.2 Name String Determination for AMD Athlon™, AMD Duron™, AMD Sempron™ Processors

The AMD Athlon processor model 2, AMD Duron processor model 3 and AMD Athlon processor model 4 were the first AMD processors to implement the name string MSRs (C001_0030h–C001_0035h). These MSRs were initialized with a default name string on reset, but the BIOS could write over the default reset value. At that time, AMD began requiring all AMD approved BIOS and motherboard vendors to display an AMD-approved name string. The method of determining the correct AMD preferred name string has become fairly complex.

For the most part, the descriptions that follow break processors into subgroups by platform type within processor family.

3.3.3 AMD Recommended Name String for AMD Athlon™ Processor Models 2 and 4 and AMD Duron™ Processor Model 3

The name strings for the AMD Athlon processor model 2, AMD Duron processor model 3 and AMD Athlon processor model 4 are determined by referring to Table 7. The distinction between the AMD Duron processor model 3 and the mobile AMD Duron processor model 3 is arbitrary; the processor signatures are identical. Custom software designed specifically for the mobile platform must program the name string MSRs with the correct name string for this mobile processor.

Table 7. AMD-Recommended Name String for AMD Athlon™ Processor Models 2 and 4 and AMD Duron™ Processor Model 3

Processor	Processor Signature[1]	Recommended Name String
AMD Athlon™ Model 2	62*x*h	"AMD Athlon(tm) Processor"
AMD Duron™ Model 3	63*x*h	"AMD Duron(tm) Processor"
mobile AMD Duron Model 3	63*x*h	"mobile AMD Duron(tm) Processor"
AMD Athlon Model 4	64*x*h	"AMD Athlon(tm) Processor"

Note: *1. Processor stepping (x) is irrelevant.*

3.3.4 AMD Recommended Name String for Mobile AMD Athlon™ 4 Processor Model 6 and Mobile AMD Duron™ Processor Model 3

The Mobile AMD Athlon 4 processor Model 6 and Mobile AMD Duron processor Model 6 have identical family and model numbers (66h), but are distinguished by the value of the processor's L2 cache size as reported by ECX register bits[31:16] by CPUID extended function 8000_0006h.

- If the L2 cache size value reported by extended function 8000_0006h ECX bits[31:16] is 256 or greater, then the processor name string should be the “mobile AMD Athlon(tm) 4 processor”.
- If the L2 cache size reported is less than 256, then the processor name string should be the “mobile AMD Duron(tm) processor”.

Table 8 summarizes the name string information for these processors.

Table 8. AMD Recommended Name String for Mobile AMD Athlon™ 4 Processor Model 6 and Mobile AMD Duron™ Processor Model 6

Processor	Processor Signature	L2 Cache Size	Recommended Name String
Mobile AMD Athlon™ 4 Model 6	660h	≥256	“mobile AMD Athlon(tm) 4 Processor”
Mobile AMD Duron™ Model 6	660h	<256	“mobile AMD Duron(tm) Processor”

3.3.5 Identifying the Platform Context—AMD Athlon™ Processor Model 6 and Later

The selection of the appropriate name string for AMD Athlon™ processor Model 6 and above involves identifying whether the processor is employed in a desktop, multiprocessor, or mobile system. This is accomplished by examination of the processor configuration and the core logic of the platform.

3.3.5.1 Identifying Mobile Platforms

If CPUID Extended Function 8000_0007h returns EDX[2:1] = 11b (i.e, the processor supports FID and VID control) *and* if the Maximum FID field of the FidVidStatus MSR (C001_0042h) is *not* equal to the Startup FID, then the processor is a *mobile* processor and is operating in a mobile platform. If the processor supports FID and VID control, but the Maximum FID and Startup FID values are equal, then the processor is a *low-power desktop* processor.

The Startup FID and Maximum FID can only be determined by reading them from the FidVidStatus MSR C001_0042h, if AMD PowerNow! technology is enabled. For more information refer to the *BIOS Requirements for AMD PowerNow!™ Technology Application Note*, order# 25264, and the *BIOS Requirements for Cool‘n’Quiet Technology Application Note*, order# 25541.

3.3.5.2 Identifying Multiprocessing Platforms

If the Northbridge of the platform’s core logic is an AMD-762™ controller (IGD4-2P), the processor is operating in a *multiprocessing* (workstation/server) platform.

3.3.5.3 Identifying Desktop Platforms

If the platform is neither a mobile system, nor a multiprocessor system, the processor is operating in a desktop (or low-power desktop) platform.

3.3.6 AMD-Recommended Name String for Mobile AMD Athlon™ and Mobile AMD Duron™ Processors

Rules for displaying the name string depend (for most mobile models) on front side bus speed and processor frequency.

The name string for Mobile AMD Athlon processors Model 6 includes a model number suffix for frequencies greater than 1300 MHz. Mobile AMD Athlon processors models 8 and 10 always include a processor model number. This model number is derived from the processor frequency as shown in Table 14 on page 54. For instance, the model number for an AMD Athlon processor Model 8 OPGA with an operating frequency of 1400 MHz has a model number of 1600+ with a 256 Kbyte L2 cache, or a model number of 1800+ with a 512 Kbyte L2 cache. The complete name string is either "mobile AMD Athlon(tm) XP-M 1600+" or "mobile AMD Athlon(tm) XP-M 1800+

Table 9 specifies the name strings for all mobile AMD Athlon models 6, model 8 and model 10, and mobile AMD Duron processors model 7 and model 8.

Table 9. Recommended Name Strings for Mobile AMD Athlon™ Model 6 and Later and Mobile AMD Duron™ Processors

Processor	Processor Signature	Recommended Name String
AMD Athlon™ Model 6	661	"mobile AMD Athlon(tm) 4 "
AMD Athlon Model 6	662	"mobile AMD Athlon(tm) 4 [xxxxx]"[1]
AMD Duron™ Model 6	N/A[2]	"mobile AMD Duron(tm) "
AMD Duron Model 7	670	"mobile AMD Duron(tm) "
AMD Duron Model 7	671	
AMD Duron Model 8	N/A[2]	"mobile AMD Athlon(tm) XP-M [xxxxx]"
AMD Athlon Model 8 OPGA Mobile	N/A[2]	
AMD Athlon Model 10 OPGA Mobile	N/A[2]	
AMD Athlon Model 8 µPGA Mobile	N/A[2]	"mobile AMD Athlon(tm) XP-M (LV) [xxxxx]"
AMD Athlon Model 10 µPGA Mobile	N/A[2]	

Notes:

1. The name string for mobile AMD Athlon model 6 is "mobile AMD Athlon(tm) 4" at frequencies less than 1300 MHz. No model number is appended to the processor brand..

2. Processor signature is irrelevant.

3.3.7 Name String for Desktop Processors

Processor name strings for AMD desktop processors are given in Table 10 on page 39 and Table 11 on page 41. These processors include all AMD Athlon, AMD Duron, and AMD Sempron processors for desktop applications. The desktop platform segment includes both desktop and low-power desktop platforms.

The size of the L2 cache distinguishes an AMD Athlon processor or AMD Sempron processor from an AMD Duron processor. If the L2 cache size value reported by extended function 8000_0006h ECX bits[31:16] is 256 or greater, then the processor is an AMD Athlon or AMD Sempron family processor. If the L2 cache size reported is less than 256, then the processor is an AMD Duron family processor. See Table 14, "Model Number Mappings for Desktop and Mobile AMD Athlon™ XP and AMD Athlon™ MP Processors," on page 54 for details.

It is important to distinguish between a desktop and a multiprocessing system when selecting the appropriate name string. In most cases this determination can be made either as described in "Identifying Multiprocessing Platforms" on page 37 or by looking at the value of the ECC flag (EBX bit 19, returned by CPUID extended function 8000_0001h). For multiprocessing systems this bit will be set to 1; for desktop systems the ECC bit will be cleared. AMD Sempron™ processors are exceptions to this rule; The ECC bit is always set to 1 on these processors.

3.3.7.1 Desktop Model Numbers

As is the case for mobile processors, the name string for most desktop processors includes a model number suffix. The model number is determined by the L2 cache size, front side bus speed, and processor frequency as shown in Table 14 on page 54.

Table 10 specifies the name strings for all desktop AMD Athlon models 6, model 8 and model 10. A five character placeholder [*xxxxx*] indicates the model number. Where present, the model number always consists of four digits followed by '+' (1400+, 1800+, etc.).

Table 10. Name Strings for AMD Athlon™ and AMD Duron™ Desktop Processors

Processor	Processor Signature	Bit 19 of Extended Feature Flags	Recommended Name String
AMD Athlon™ Model 6	660h or 661h	Reserved	`AMD Athlon(tm)`
AMD Athlon Model 6	662h	0	`AMD Athlon(tm) XP [xxxxx]`
AMD Duron™ Model 6	N/A[1]	0	`AMD Duron(tm)`
AMD Duron Model 6	N/A[1]	0	`AMD Duron(tm)`
AMD Athlon Model 6	662h	0	`AMD Athlon(tm) XP processor [xxxxx]`
AMD Duron Model 7	670h	Reserved	`AMD Duron(tm)`
AMD Duron Model 7	671h	0	`AMD Duron(tm)`

Note: *1. Processor signature is irrelevant.*

Table 10. Name Strings for AMD Athlon™ and AMD Duron™ Desktop Processors (Continued)

Processor	Processor Signature	Bit 19 of Extended Feature Flags	Recommended Name String
AMD Duron Model 7	671h	N/A	`AMD Duron(tm)`
AMD Athlon Model 8	N/A[1]	0	`AMD Athlon(tm) XP [xxxxx]`
AMD Athlon Model 8	N/A[1]	0	`AMD Athlon(tm) XP [xxxxx]`
AMD Athlon™ XP Model 10	N/A[1]	0	`AMD Athlon(tm) XP [`*`xxxx`*`]`
AMD Athlon Model 2	N/A[1]	N/A	`AMD Athlon(tm) Processor`
AMD Duron™ Model 3	N/A[1]	N/A	`AMD Duron(tm)`
AMD Duron Model 8	N/A[1]	0	`AMD Duron(tm)`
AMD Duron Model 8	N/A[1]	0	`AMD Duron(tm)`
AMD Athlon Model 10	N/A[1]	0	`AMD Athlon(tm) XP [xxxxx]`

Note: *1. Processor signature is irrelevant.*

3.3.8 Name Strings for AMD Sempron™ Processors

The rules for determining the name strings for AMD Sempron processors depends on the effective family and and effective model of the processor. If the processor family is 6h (models 8 and 10 (Ah)), the name strings are determined as described in this section; otherwise, the name strings for AMD Sempron processors in effective family 0Fh are determined as described in "Constructing the Name String on AMD Athlon™ 64, AMD Opteron™, and AMD Sempron™ Processors" on page 43.

The name string on AMD Sempron processors Models 8 and 10 are determined by a number of factors. First, these processors always return a value of 1 for the ECC flag (EDX bit 19 as returned by CPUID extended function 8000_0001h), even though the processor is not a multiprocessing part. Second, these processors have a 333 MHz front side bus.

L2 cache size distinguishes an AMD Sempron processor from an AMD Duron™ processor. If the L2 cache size value reported by extended function 8000_0006h ECX bits[31:16] is 256 or greater, then the processor is an AMD Athlon™ or AMD Sempron family processor.

Table 11 provides the name strings for AMD Sempron processors. Complete information necessary for mapping the correct model numbers is given in Table 15, "AMD AMD Sempron™ Processors Model 8 and Model 10," on page 56.

Table 11. Recommended Name Strings for AMD Sempron™ Processors Models 8 and 10

Processor	Processor Signature	Bit 19 of Extended Feature Flags	Platform	Recommended Name String
AMD Sempron™ Model 8	N/A[1]	1	Desktop	`AMD Sempron(tm) [xxxx]`
AMD Sempron Model 10	N/A[1]	1	Desktop	`AMD Sempron(tm) [xxxx]`

Note: *1. Processor signature is irrelevant.*

3.3.9 AMD Name String for AMD Athlon™ and AMD Duron™ MP Server/Workstation Processors

Several ways to determine whether or not a particular processor can be used in a multiprocessing server or workstation have been described. A direct method is to test the status of the ECC flag (EDX bit 19 returned by CPUID extended function 8000_0001h). This methods is described in "Name String for Desktop Processors" on page 39. Another method of identifying an MP processor is described in "Identifying Multiprocessing Platforms" on page 37. Table 12 shows the AMD-recommended name strings for AMD Athlon™ and AMD Duron™ MP systems. More complete information on model number mapping is provided in Table 14 on page 54.

Table 12. Recommended Name Strings for AMD Athlon™ and AMD Duron™ MP Server/Workstations

Processor	Processor Signature	Bit 19 of Extended Feature Flags	Platform	Recommended Name String
AMD Athlon™ Model 6	660h, 661h	Reserved	Multiprocessing	`AMD Athlon(tm) MP`
AMD Athlon Model 6	662h	1	Multiprocessing	`AMD Athlon(tm) MP [xxxxx]`
AMD Duron™ Model 6	N/A[1]	1	Multiprocessing	`AMD Duron(tm) MP`
AMD Athlon Model 6	662h	1	Multiprocessing	`AMD Athlon(tm) MP processor [xxxxx]`
AMD Duron Model 7	670h	Reserved	Multiprocessing	`AMD Duron(tm) MP`
AMD Duron Model 7	671h	1	Multiprocessing	`AMD Duron(tm) MP`
AMD Athlon Model 8	N/A[1]	1	Multiprocessing	`AMD Athlon(tm) MP [xxxxx]`
AMD Duron Model 8	N/A[1]	1	Multiprocessing	`AMD Duron(tm) MP`
AMD Duron Model 8	N/A[1]	1	Desktop	`AMD Duron(tm) MP`
AMD Athlon Model 10	N/A[1]	1	Multiprocessing	`AMD Athlon(tm) MP [xxxxx]`

Note: *1. Processor signature is irrelevant.*

3.4 Constructing the Name String on AMD Athlon™ 64, AMD Opteron™, and AMD Sempron™ Processors

On AMD Athlon 64, AMD Opteron, and AMD Sempron processors, all in effective family 0Fh, the name string is correlated to the processor's brand ID.

3.4.1 Brand ID

On the AMD Athlon 64 and AMD Opteron processors, and the AMD Sempron processor Model 10, the BIOS code uses the brand ID value to construct the proper name string from tables of ASCII name string information. The brand ID is either an 8-bit field returned in EBX by CPUID standard function 1 or a 12-bit field returned in EBX by CPUID extended function 8000_0001h (see "Brand ID—Standard Function 1 and Extended Function 8000_0001" on page 23).

If the 8-bit brand ID and the 12-bit brand ID are both zero, the processor is an engineering sample. All AMD Athlon 64 and AMD Opteron processors support the brand ID and thus report a non-zero value.

3.4.2 Constructing the Name String

The BIOS must construct the processor name string from information in a table that is a BIOS data structure. The bulk of the name string comes from the name string table, which contains the required processor name string for each AMD Athlon 64 and AMD Opteron processor with placeholders for the specific model number. After all of the component parts of the name string are determined, they must be concatenated into a NUL terminated string that is no longer than 48 bytes in length, including the terminating NUL.

3.4.2.1 Selecting the Name String

If the processor has a nonzero 8-bit brand ID, the three most-significant bits (EBX[7-5]) are used to choose a name string from those listed in Table 13 on page 44.

Otherwise, if the processor has a 12-bit brand ID that is non-zero, the six most-significant bits (EBX[11–7]) are used to index into the name strings listed in Table 13 on page 44.

3.4.2.2 Calculating the Model Number

If the processor has a non-zero 8-bit brand ID, the decimal equivalent of the *five* least-significant bits (*NN* in Table 13) are used to calculate two digits of the model number to be inserted into the appropriate name string.

If the processor has a 12-bit brand ID that is non-zero, the decimal equivalent of the *six* least-significant bits (*NN* in Table 13) are used to calculate two digits of the model number to be inserted into the appropriate name string.

The factor *NN* is used in several ways to construct the model number, depending on the processor. See the explanatory notes in Table 13 for details.

Table 13. Processor Name String Values for AMD Athlon™ 64, AMD Sempron™, and AMD Opteron™ Processors

Brand ID MSBs[6]	Name String	Processor
000000b	`AMD Engineering Sample` *NN*[1]	Reserved
000100b	`AMD Athlon(tm) 64 Processor` *XX*`00+`[2]	UP Client
001000b	`Mobile AMD Athlon(tm) 64 Processor` *XX*`00+`[2]	Mobile Client
001001b	`Mobile AMD Athlon(tm) 64 Processor` *XX*`00+`	Mobile Client Low-Power Mobiles VIDs
001100b	`AMD Opteron(tm) Processor 1`*YY*[3]	UP Server
001110b	`AMD Opteron(tm) Processor 1`*YY*[3] `HE`	UP Server Low-Power 55W
001111b	`AMD Opteron(tm) Processor 1`*YY*[3] `EE`	UP Server Low-Power 30W
010000b	`AMD Opteron(tm) Processor 2`*YY*[3]	2P Server
010010b	`AMD Opteron(tm) Processor 2`*YY*[3] `HE`	2P Server Low-Power 55W
010011b	`AMD Opteron(tm) Processor 2`*YY*[3] `EE`	2P Server Low-Power 30W
010100b	`AMD Opteron(tm) Processor 8`*YY*[3]	MP Server
010110b	`AMD Opteron(tm) Processor 8`*YY*[3] `HE`	MP Server Low-Power 55W
010111b	`AMD Opteron(tm) Processor 8`*YY*[3] `EE`	MP Server Low-Power 30W
011101b	`Mobile AMD Athlon(tm) XP-M Processor` *XX*`00+`	Mobile Client, 32-Bit
011110b	`Mobile AMD Athlon(tm) XP-M Processor` *XX*`00+`	Mobile Client, 32-Bit Low-Power Mobile VIDs

Notes: *For more information, see "8-Bit Brand ID" on page 56 and "EBX—12-bit Brand ID" on page 62.*

1. *NN = two digit decimal equivalent of bits 4–0 of the 8-bit brand ID or bits 5-0 of the 12-bit brand ID.*
2. *XX = 22 + NN. For example, 000001b stands for "23", 011111b stands for "53", and 111111b stands for "85".*
3. *YY = 38 + 2 * NN. For example, 000001b stands for "40" and 011110b stands for "98". Encodings 011111b through 111111b are reserved.*
4. *ZZ = 24 + NN. For example, 011011b stands for "51" and 100011b stands for "59".*
5. *TT = 24 + NN. For example, 000001b stands for "25", 011111b stands for "55", and 111111b stands for "87".*
6. *Bits 11–6 of the 12-bit Brand ID are given. Brand IDs not listed in this table are reserved.*

Table 13. Processor Name String Values for AMD Athlon™ 64, AMD Sempron™, and AMD Opteron™ Processors

Brand ID MSBs[6]	Name String	Processor
100000b	AMD Athlon(tm) XP Processor *XX*00+	Desktop/DTR Client 32-Bit
100001b	Mobile AMD Sempron(tm) Processor *TT*00+	Mobile Client, 32-Bit
100010b	AMD Sempron(tm) Processor *TT*00+	Desktop/DTR Client, 32-Bit
100011b	Mobile AMD Sempron(tm) Processor *TT*00+	Mobile Client, 32-Bit Low-Power Mobile VIDs
100100b	AMD Athlon(tm) 64 FX-*ZZ* Processor[4]	Desktop Client

Notes: *For more information, see "8-Bit Brand ID" on page 56 and "EBX—12-bit Brand ID" on page 62.*

1. *NN = two digit decimal equivalent of bits 4–0 of the 8-bit brand ID or bits 5-0 of the 12-bit brand ID.*
2. *XX = 22 + NN. For example, 000001b stands for "23", 011111b stands for "53", and 111111b stands for "85".*
3. *YY = 38 + 2 * NN. For example, 000001b stands for "40" and 011110b stands for "98". Encodings 011111b through 111111b are reserved.*
4. *ZZ = 24 + NN. For example, 011011b stands for "51" and 100011b stands for "59".*
5. *TT = 24 + NN. For example, 000001b stands for "25", 011111b stands for "55", and 111111b stands for "87".*
6. *Bits 11–6 of the 12-bit Brand ID are given. Brand IDs not listed in this table are reserved.*

3.4.3 Exceptional Cases

BIOS software must be prepared to handle the following anomalous cases:

- The brand ID value does not match an entry in Table 13 on page 44. This case can occur if the user upgrades the processor, but not the BIOS. For this case, it is recommended that the BIOS program the name string as

  `"AMD Athlon(tm) or Opteron(tm) CPU-model unknown"`

- The 8-bit brand ID and the 12-bit brand ID are both zero, indicating that the processor is an engineering sample.

3.5 Programming the Processor Name String

After the BIOS has found the proper string in the name table and constructed the correct model number, the BIOS programs the processor name string by writing to six MSRs with the WRMSR instruction.

MSRs C001_0030h–C001_0035h are used to set the 48 character processor name string returned by CPUID functions 8000_0002h, 8000_0003h, and 8000_0004h. Each MSR sets eight of the 48 characters in little-endian byte order. The first character resides in the least significant byte of MSR

C001_0030h and the last character (always a NUL) resides in the most significant byte of MSR C001_0035h. (The WRMSR instruction must be executed at a privilege level of 0; otherwise, a general protection (#GP(0)) exception will be triggered.) For more information on using the WRMSR instruction, see "WRMSR" in the *AMD64 Architecture Programmer's Manual, Volume 3: General Purpose and System Instructions*, order# 24594.

3.6 Name String Code Example

The following code illustrates how to program the name string into the name string MSRs (C001_0030h–C001_C0035h). This example assumes that DS:SI is pointing to a name string table entry. Each entry is a 48-character string in little-endian format padded with NUL characters.

```
;-----------------------------------------------;
; amdLoadNameStringMsrs:
;
;   Input:
;       DS:SI = ptr to new name string
;  Output:
;       none

;

;---------------------------------------;
amdLoadNameStringMsrs proc near               ;
                PUSH    EAX                   ;
                PUSH    EBX                   ;
                PUSH    ECX                   ;
                PUSH    EDX                   ;
;
; Zero out the name string MSRs
;
                MOV     ECX, 0C0010030h       ;Point ecx at 1st Name Str MSR
                XOR     EAX, EAX              ;edx:eax = 8 chars of string
                XOR     EDX, EDX              ;
;-----------------------------------------------;fill MSRs with 00's
@@:             WRMSR                         ;
                INC     CL                    ;Next MSR
                CMP     CL, 35h               ;Done?
                JBE     @b                    ; NO---
;-----------------------------------------------;
;
; Now load string at DS:SI into CPU's Name String MSRs ...
;
;-----------------------------------------------;
                mOV     CL, 30h               ;Point ecx at 1st Name Str MSR
NextNsMsr:                                    ;
                XOR     EDX, EDX              ;
                CALL    amdLoadEaxStr         ;Get next 4 chars -- end of str?
                JC      @f                    ; YES
                XCHG    EAX, EDX              ;
```

```
                CALL    amdLoadEaxStr           ;Get next 4 chars -- end of str?
                XCHG    EAX, EDX                ;(cf=1 if end of string)
@@:                                             ;
                WRMSR                           ;Load into CPU. End of String?
                JC      @f                      ; YES--break out of loop
                INC     CL                      ;Next MSR
                CMP     CL, 35h                 ;Reached 48 char limit?
                JBE     NextNsMsr               ; NO---process next MSR
@@:                                             ;done - name string patched
;-----------------------------------------------;
amdLoadNameStringMsrsExit:                      ;
                POP     EDX                     ;
                POP     ECX                     ;
                POP     EBX                     ;
                POP     EAX                     ;
                RET                             ;
amdLoadNameStringMsrs endp                      ;
;-----------------------------------------------;
;-----------------------------------------------;


;-----------------------------------------------;
; amdLoadEaxStr proc near                       ;
;
;   Input:
;       DS:SI = ptr to string
;  Output:
;       EAX = next 4 chars of string (0-extended if end of string reached)
;       SI = ptr advanced 4 characters
;       CF = 0 if end of string not reached
;            1 if end of string reached
;-----------------------------------------------;
amdLoadEaxStr proc near                         ;
        XOR     EAX, EAX                        ;
;-----------------------------------------------;
@@:     OR      AL, AL                          ;All 4 bytes loaded?
        CLC                                     ;(assume yes)
        JNZ     DoneNoAdjust                    ; YES--rtn w/CF=0
        MOV     AL, DS:[SI]                     ;get next char in string
        OR      AL, AL                          ;End of String?
        JZ      DoneCheckNull                   ; YES--reposition chars & rtn
        INC     SI                              ;POINT TO NEXT CHAR
        ROR     EAX, 8                          ;MOVE LAST CHAR OUT OF WAY
        JMP     @b                              ;
;-----------------------------------------------;
DoneCheckNull:                                  ;
        OR      EAX, EAX                        ;NULL STRING?
        JZ      @f                              ; YES--return with CF=1
                                                ;
DoneAdjust:                                     ;
        ROR     EAX, 8                          ;
```

```
        OR      AL, AL                          ;1ST CHAR IN POSITION?
        JZ      DoneAdjust                      ; NO---rotate again
@@:     STC                                     ;Flag end of string was reached
                                                ;
DoneNoAdjust:                                   ;
        RET                                     ;
amdLoadEaxStr endp                              ;
;-----------------------------------------------;
```

3.7 Processor Recognition Code Sample

The following example shows the recommended algorithm for proper determination of the processor name string for AMD Athlon, AMD Sempron, and AMD Duron family processors models 6 and greater.

```
int modelnum;                  // Model number for processor's specific frequency
                                 // already taken from table 14 lookup.

int modelnum512k;              // AMD Athlon(tm) XP processor model number for
                               // specific frequency of processor with 512KB L2
                                 // cache already taken from table 14 look-up.

int modelnum256k;              // AMD Athlon(tm) XP processor model number for
                               // specific frequency of processor with 256KB L2
                                 // cache already taken from table 14 look-up.


int modelnum_sempron;          // AMD Sempron(tm) processor model number for
                               // specific frequency already taken from table 15
                                 // lookup.

int modelnum_sempron512k;        // Model number for processor's specific frequency
                                // already taken from table 15 lookup.

string cpuid_name_string;       // Final name string to be written to name string
                                  // MSRs C001_0030-C001_0035h.

if ((PowerNow == enabled) && (StartupFID != MaxFID)) {
// mobile processor present
      if (L2size >= 512KB)    // L2size equals size of on-die processor L2 cache
           {
             if (cpu_package == uPGA)     // Display different name string based on
                                          // processor package. Note: the BIOS has
                                          // no way to detect the processor package
                                          // type. By mobile default, the mobile
                                          // OPGA package name should be displayed.
                                          // The name string should then be hand-
                                          // modified in mobile uPGA
                                          // implementations.

             { cpuid_name_string = “mobile AMD Athlon(tm) XP-M (LV) ” +
               modelnum512k;}
             else
             { cpuid_name_string = “mobile AMD Athlon(tm) XP-M ” + modelnum512k;}
           }
      else if (L2size == 256KB) { // L2size equals size of on-die processor L2
                                    cache.
             if (cpu == Model 8 or 10)
               {
```

```
            if (cpu_package == uPGA)     // Display different name string based on
                                         // processor package. Note: the BIOS has
                                         // no way to detect teh processor package
                                         // type. By mobile default, the mobile
                                         // OPGA package name should be displayed.
                                         // The name string should then be
                                         // hand-modified in mobile uPGA
                                         // implementations.

            { cpuid_name_string = “mobile AMD Athlon(tm) XP-M (LV) ” + modelnum;}
            else
            { cpuid_name_string = “mobile AMD Athlon(tm) XP-M ” + modelnum;}
              }
            else if ((cpu == Model 6) && (frequency >= 1300MHz)) {
                  cpuid_name_string = "mobile AMD Athlon(tm) 4 " + modelnum; }
            else { cpuid_name_string = "mobile AMD Athlon(tm) 4"; }
      }
      else { cpuid_name_string = "mobile AMD Duron(tm)"; }
}
else if (NB == AMD-762) {      // MP chipset for Workstation / Server
                                 // platform detection
// The following assumes only one processor is present.  If two processors are
// present, both processors should undergo the following check with uniprocessor
// mode resulting if either cpu is not MP capable --- Refer to BIOS Identification
// of Multiprocessing Capabilities Application Note, order#25477, for more
// information on the determination of multiprocessor capability.

      if ((cpu == Model 6, 8 or 10) && (L2size == 256KB)) {
            if (cpuid <= 661) {cpuid_name_string = "AMD Athlon(tm) MP"; }
            else if (EFF bit 19 == 1 && FSB_speed <= 266) { cpuid_name_string =
            "AMD Athlon(tm) MP " + modelnum; }
            else {cpuid_name_string = "AMD Athlon(tm) XP " + modelnum;
                 // Desktop processor with 256-Kbyte L2 cache--- bios has detected
                  // a non-MP capable processor and should operate in uniprocessor
                   // modesee MP bios application note
                    }
      }
      else if ((cpu == Model 10) && (L2size >= 512KB)) {
            if (EFF bit 19 == 1 && FSB_speed <= 266) {cpuid_name_string = "AMD
            Athlon(tm) MP " + modelnum512k; }
            else {cpuid_name_string = "AMD Athlon(tm) XP " + modelnum512k;
                 // Desktop processor with 512-Kbyte L2 cache--- bios has detected
                 //  a non-MP capable processor and should operate in uniprocessor
                  //  modesee MP bios application note
                    }
      }

      else if ((cpu == Model 6, 7, or 8) && (L2size < 256KB)) {
            if (cpuid == 670) { cpuid_name_string = "AMD Duron(tm) MP"; }
            else if (EFF bit 19 == 1  && FSB_speed <= 266) {
                        cpuid_name_string = "AMD Duron(tm) MP"; }
```

```
            else { cpuid_name_string = "AMD Duron(tm)";
                    // Desktop AMD Duron --- BIOS has detected a non-MP capable
                    // processor and should operate in uniprocessor mode see
                    //MP BIOS application note.
                  }
      }
      else { // Covers AMD Athlon Model 4 and AMD Duron Model 3 --- BIOS has
             // detected a non-MP capable processor and should operate in
             // uniprocessor mode see MP BIOS application note
            }
}
else {      // Low-power desktop or desktop platform
      if (cpu == Model 6) {
            if (L2size >= 256KB) {
                  if (frequency >= 1300MHz) {
                        cpuid_name_string = "AMD Athlon(tm) XP " + modelnum; }
                  else { cpuid_name_string = "AMD Athlon(tm)"; }
            }
            else {
                  if (EFF bit 19 == 1  && FSB_speed <= 266)
                              { cpuid_name_string = "AMD Duron(tm) MP"; }
                  else { cpuid_name_string = "AMD Duron(tm)"; }
             }
            }
      else if (cpu == Model 8) {
            if (L2size >= 256KB) {
                  if (EFF bit 19 == 1 && FSB_speed >= 333) {
                        cpuid_name_string = "AMD Sempron(tm) " +
                                                          modelnum_sempron; }
                  else { cpuid_name_string = "AMD Athlon(tm) XP " + modelnum; }
            }
            else {
                  if (EFF bit 19 == 1  && FSB_speed <= 266) {
                          cpuid_name_string = "AMD Duron(tm) MP"; }
                  else { cpuid_name_string = "AMD Duron(tm)"; }
             }
            }

       else if (cpu == Model 10) {  // Desktop Model 10 processor identification

         if (EFF bit 19 == 1 && FSB_speed >= 333) { //check for Sempron Model 10
            if (L2size == 512KB) {    // Model 10 with 512KB L2
               cpuid_name_string = 'AMD Sempron(tm) ' + modelnum_sempron512k;
            }
            else {              // Model 10 with 256 KB L2 cache
                cpuid_name_string = 'AMD Sempron(tm) ' + modelnum_sempron;
            }
         }
         else {
           if (L2size == 512KB) { // AMD Athlon Model 10
             if (EFF bit 19 == 1 && FSB_speed <= 266) { //check for MP part
```

```
                cpuid_name_string = 'AMD Athlon(tm) MP ' + modelnum512k;
              }
              else {
                cpuid_name_string = 'AMD Athlon(tm) XP ' + modelnum512k;
              }
            }
          else {  // AMD Athlon XP Model 10 with 256 KB cache
              cpuid_name_string = 'AMD Athlon(tm) XP ' + modelnum256k;
          }
        }

    }

      else if (cpu == Model 3 or 7) { cpuid_name_string = "AMD Duron(tm)"; }
      else { cpuid_name_string = "AMD Athlon(tm)"; } // covers Models 1, 2, and 4
}
```

Appendix A Model Number Mappings

Table 14 and 15 summarize the information necessary to construct the name string and model number for all AMD Athlon™ and AMD Athlon 4, processors Models 6, Models 8 and 10 and AMD Sempron™ processors Models 8 and 10. Neither the AMD Athlon processor Model 4 nor AMD Duron™ processors (all models) ever add a model number to the namestring. The use of the model number may have certain restrictions when applied to other AMD processors.

See the detailed discussion in Chapter 3 “Programming the Processor Name String” for details on using the information in these tables.

A.1 Model Numbers for AMD Athlon™ Desktop and MP Processors

Model numbers are *not* affixed to the name strings for any desktop or MP (server/workstation) operating at frequencies below 1300 MHz except for the AMD Athlon MP processor Model 6 for low-power, which operates at 1200 MHz and 1400 MHz.

A.2 Model Numbers for Mobile AMD Athlon™ Processors

Model numbers are *not* affixed to the name string of any mobile AMD Athlon processors operating at frequencies below 1300 MHz, except for the mobile AMD Athlon processor Model 8, which displays the model numbers for 950 MHz, 1000 MHz, 1100 MHz, and 1200 MHz and higher frequencies.

Table 14. Model Number Mappings for Desktop and Mobile AMD Athlon™ XP and AMD Athlon™ MP Processors

L2 Cache (Kbytes)	FSB	Frequency (MHz)	Model Number
256	200	1000	1200+
		1100	1300+
		1200	1400+
		1300	1500+
		1400	1600+
		1500	1800+
		1600	1900+
	266FSB	1200	1400+
		1333	1500+
		1400	1600+
		1467	1700+
		1533	1800+
		1600	1900+
		1667	2000+
		1733	2100+
		1800	2200+
		2000	2400+
		2133	2600+
	333FSB	2083	2600+
		2167	2700+
		2250	2800+
	400FSB	2200	3100+

Table 14. Model Number Mappings for Desktop and Mobile AMD Athlon™ XP and AMD Athlon™ MP Processors

L2 Cache (Kbytes)	FSB	Frequency (MHz)	Model Number
512	200FSB	1300	1700+
		1400	1800+
	266FSB	1400	1800+
		1467	1900+
		1533	2000+
		1600	2100+
		1667	2200+
		1800	2400+
		1867	2500+
		2000	2600+
		2133	2800+
	333FSB	1833	2500+
		1917	2600+
		2083	2800+
		2167	3000+
		2333	3200+
	400FSB	2000	2900+
		2100	3000+
		2200	3200+

A.3 Model Number Mappings for AMD Sempron™ Processors Models 8 and 10

Because of the somewhat idiosyncratic nature of the construction of name strings for AMD Sempron processors, a separate table of model number mappings for these processors is provided below. For details on construction the namestrings for AMD Sempron processors. See "Name Strings for AMD Sempron™ Processors" on page 41.

Table 15. AMD AMD Sempron™ Processors Model 8 and Model 10

Cache Size (Kbytes)	FSB	Frequency (MHz)	Model Number
256	333FSB	1500	2200+
		1583	2300+
		1667	2400+
		1750	2500+
		1833	2600+
		2000	2800+
512	333FSB	2000	3000+