Intel processor identification and CPUID instructions

Ali Heidari

Introduction

Why is processor identification important?

With the evolution of Intel architecture and the variety of models (from 8086 to Core i7 and Xeon), software must be able to identify the features available on the hardware.

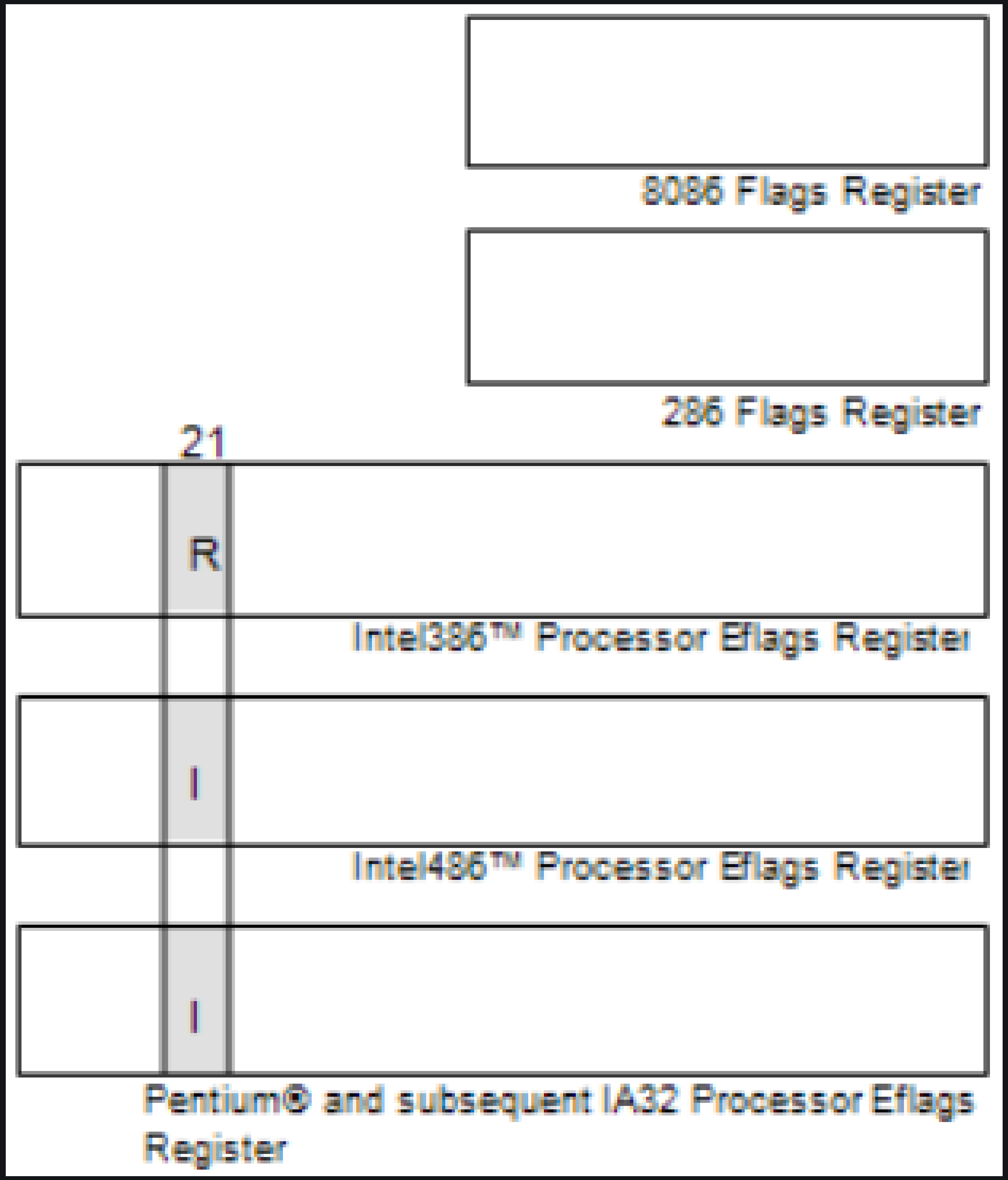
Purpose of the CPUID instruction

This instruction provides information about the processor signature and supported features so that software can adapt its performance to the hardware.

Evolution:

Initially, identification was based on architectural differences, but later the processor signature and then the CPUID instruction were added

Detecting the CPUID Instruction


8086 Flags Register
286 Flags Register
21
R
Intel386™ Processor Eflags Register
I
Intel486™ Processor Eflags Register
I
Pentium® and subsequent IA32 Processor Eflags Register

Challenge: Not all processors (especially very old models before the 486) support CPUID.

Detection method: Use the ID Flag (bit 21) in the EFLAGS register.

Algorithm: If software can change (toggle) the value of this bit, the processor supports the CPUID instruction.

output of the CPUID

Input (EAX): The output of the CPUID instruction is completely dependent on the value placed in the EAX register before execution.

Function classification:

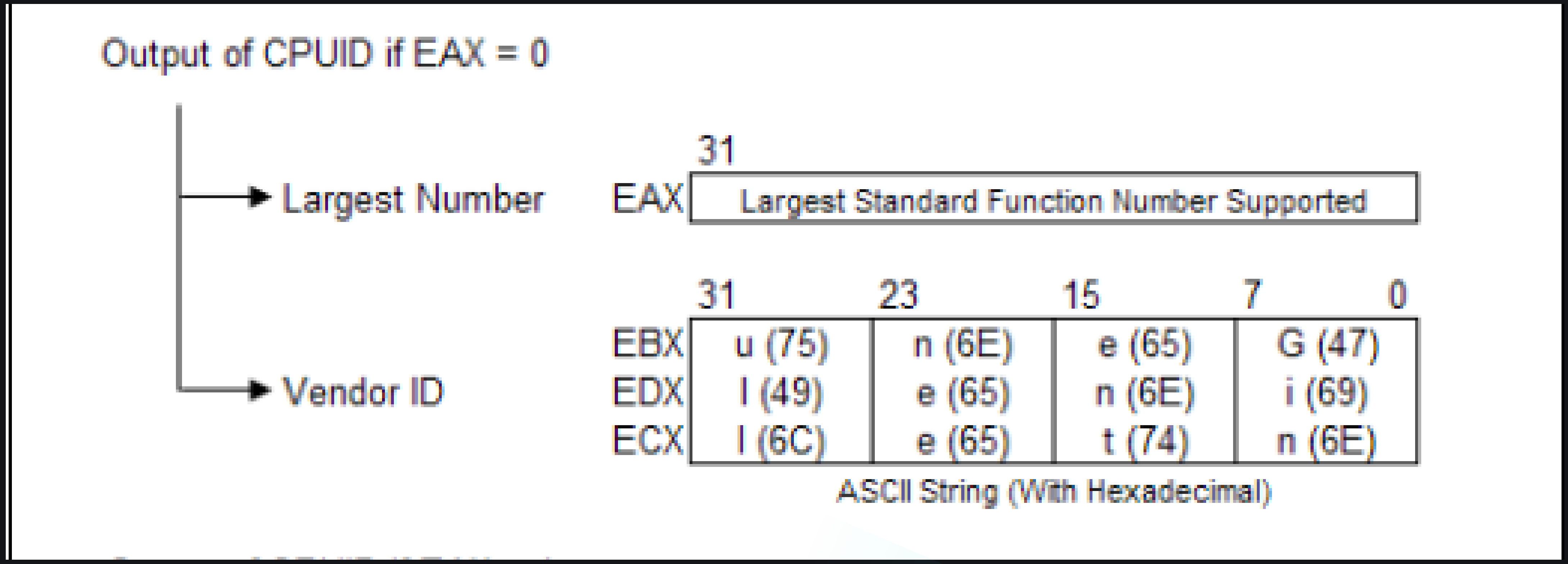
Standard Functions

Extended Functions

How to get the maximum function: By placing the value 0 in EAX, the maximum allowed input value is returned.

Vendor-ID and Largest Standard Function (Function 0)

```
mov EAX, 00h
CPUID
```



Output of CPUID if EAX = 0

Largest Number

	31
EAX	Largest Standard Function Number Supported

Vendor ID

	31	23	15	7 0
EBX	u (75)	n (6E)	e (65)	G (47)
EDX	I (49)	e (65)	n (6E)	i (69)
ECX	l (6C)	e (65)	t (74)	n (6E)

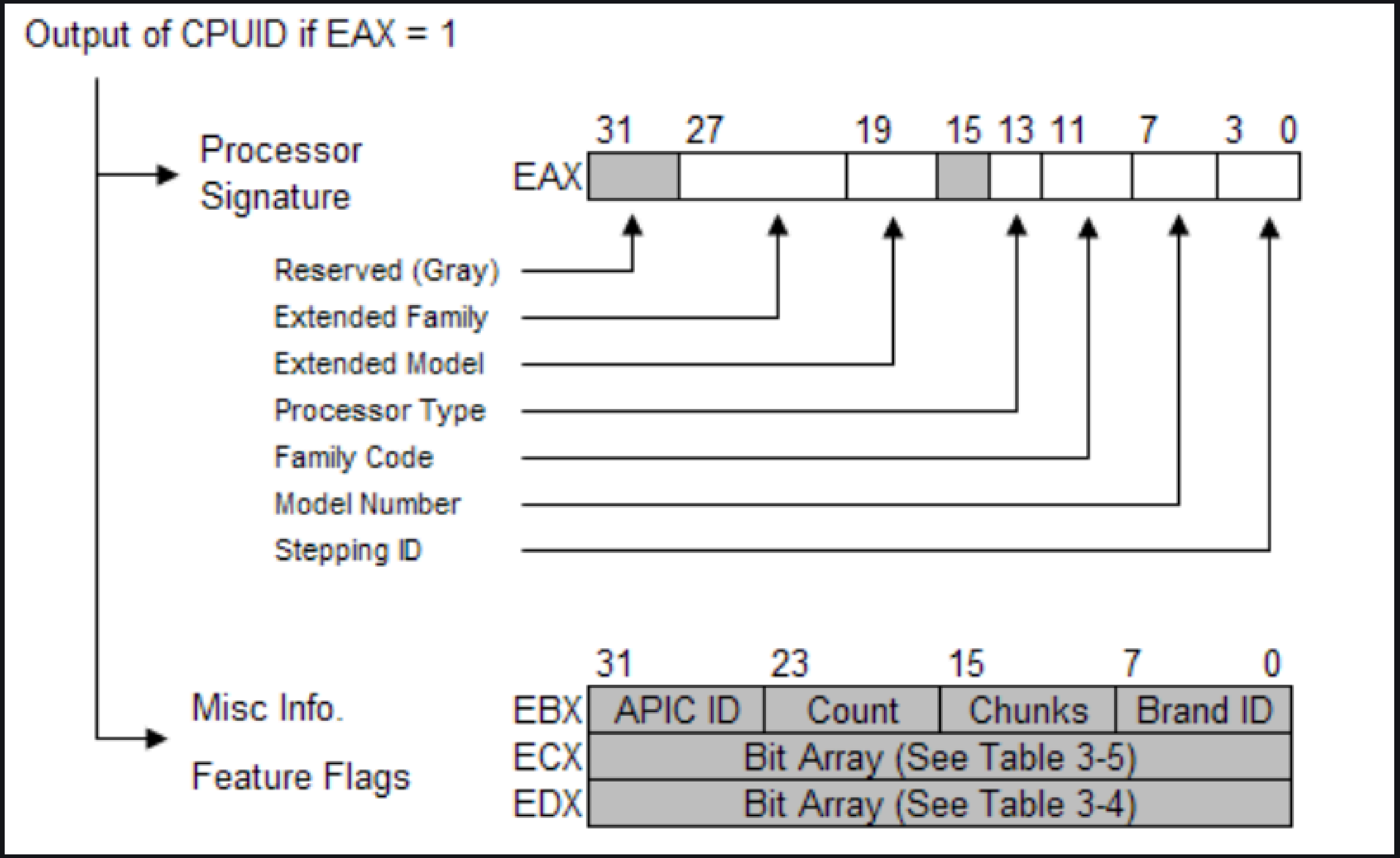
ASCII String (With Hexadecimal)

Returns the string "GenuineIntel" in the EBX, EDX, ECX registers, which guarantees an Intel processor.

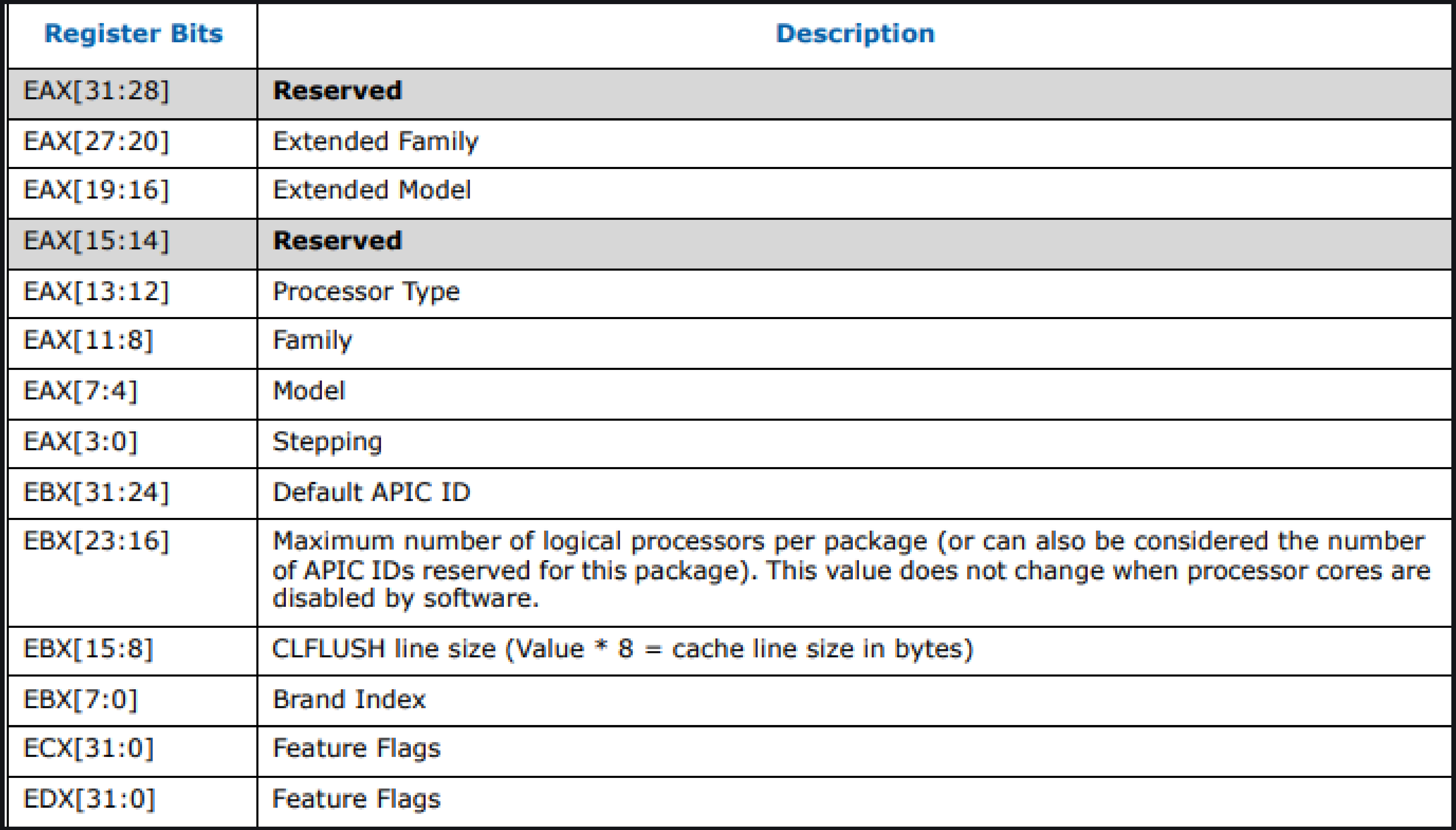
Feature Information (Function 01h):

The most important function for identification is that it gives the following information:

1. processor Signature
2. Feature Flag



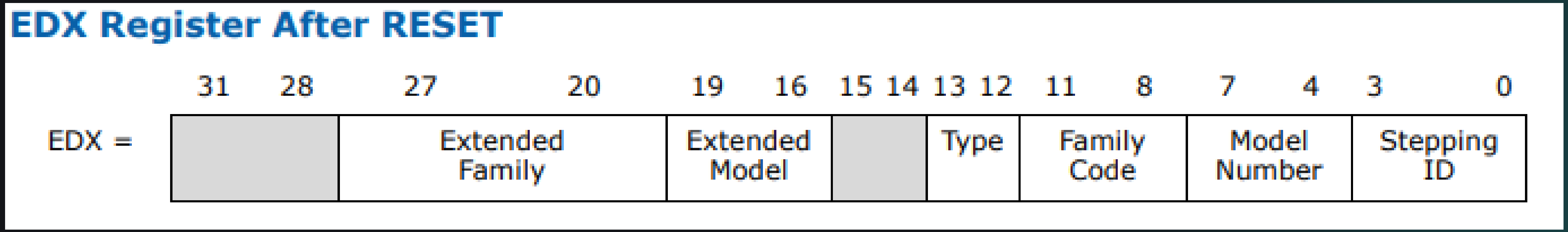
Output for input EAX = 01h



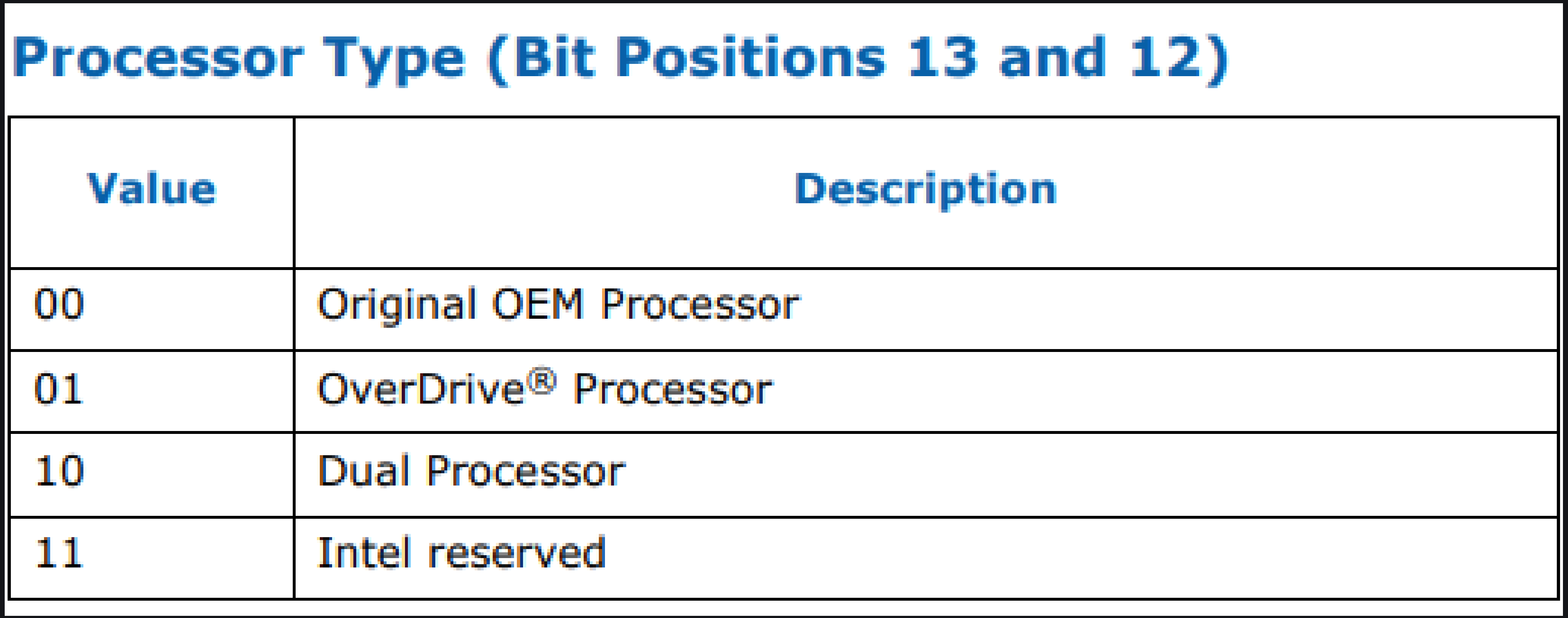
Register Bits	Description
EAX[31:28]	**Reserved**
EAX[27:20]	Extended Family
EAX[19:16]	Extended Model
EAX[15:14]	**Reserved**
EAX[13:12]	Processor Type
EAX[11:8]	Family
EAX[7:4]	Model
EAX[3:0]	Stepping
EBX[31:24]	Default APIC ID
EBX[23:16]	Maximum number of logical processors per package (or can also be considered the number of APIC IDs reserved for this package). This value does not change when processor cores are disabled by software.
EBX[15:8]	CLFLUSH line size (Value * 8 = cache line size in bytes)
EBX[7:0]	Brand Index
ECX[31:0]	Feature Flags
EDX[31:0]	Feature Flags

Processor Signature

Beginning with the Intel486 processor family, the EDX register contains the processor identification signature after RESET. The processor identification
signature is a 32-bit value. The processor signature is composed from eight different bit fields. The fields in gray represent reserved bits, and should be masked out when utilizing the processor signature. The remaining six fields form the processor identification signature.



NOTE: It is possible that in some cases the processor signature of multiple systems may be the same.

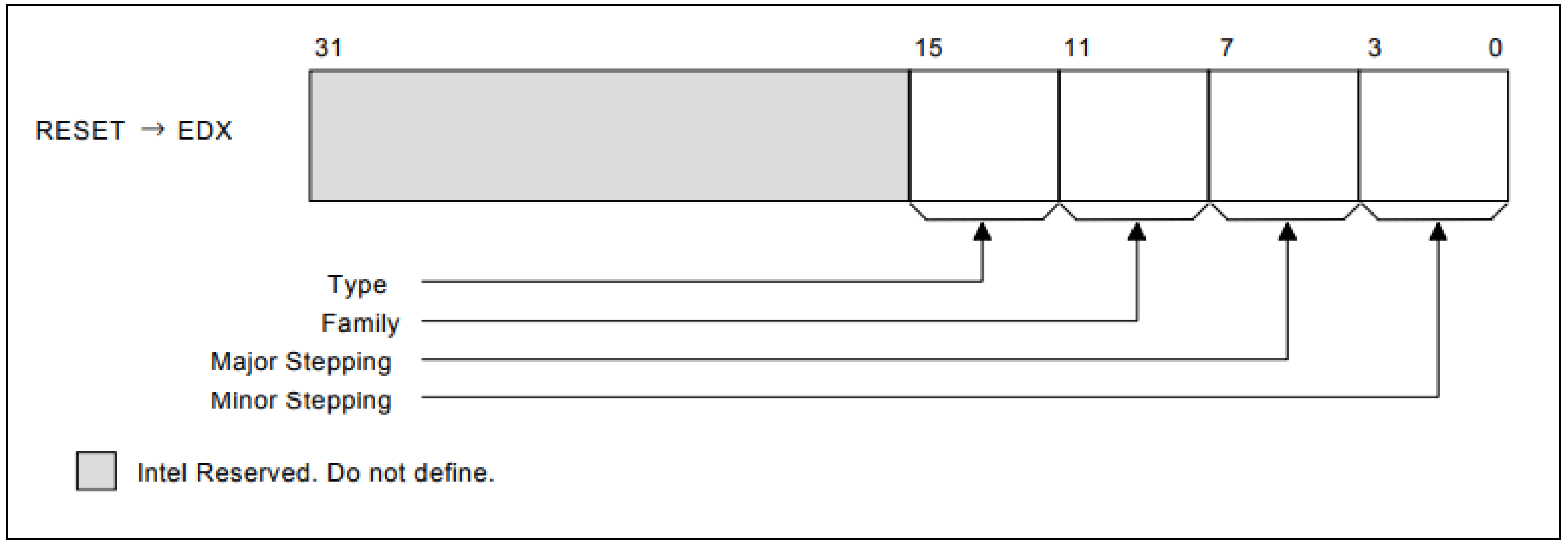


Processor Type (Bit Positions 13 and 12)

Value	Description
00	Original OEM Processor
01	OverDrive® Processor
10	Dual Processor
11	Intel reserved

shows the format of the processor signature for Intel386 processors. The Intel386 processor signature is different from the signature of other processors. provides the processor signatures of Intel386T processors.

Processor Signature Format on Intel386™ Processors



Composing the Family, Model and Stepping (FMS) values:

F = Extended Family + Family

F = CPUID (1) . EAX [27: 20] + CPUID (1) . EAX [11 : 8]

M = (Extended Model << 4) + Model

M = (CPUID (1) . EAX [19:16] << 4) + CPUID (1) .EAX [7: 4]

S = Stepping

S = CPUID (1) . EAX [3 : 0]

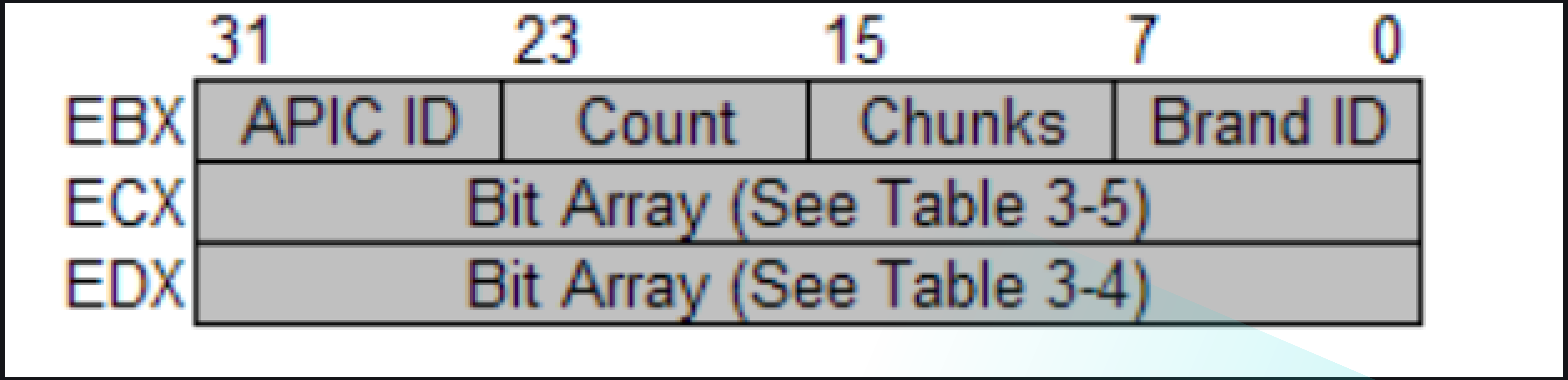
Output of the EBX when input EAX = 01h

Brand ID:

Chunks:

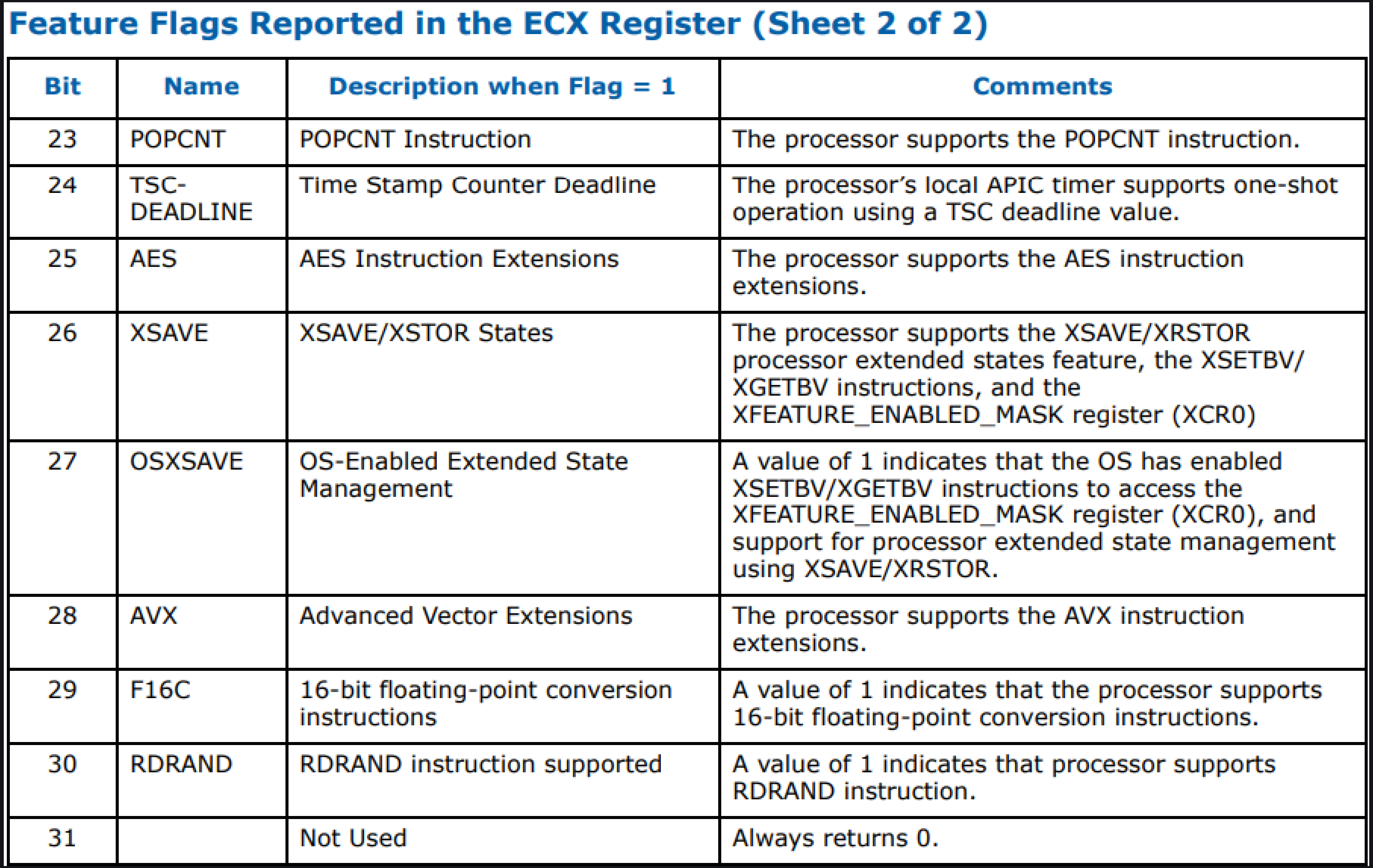
Count:

APIC ID:



	31	23	15	7 0
EBX	APIC ID	Count	Chunks	Brand ID
ECX	Bit Array (See Table 3-5)			
EDX	Bit Array (See Table 3-4)			

The SYSENTER Present (SEP) Feature bit (returned in EDX bit 11 after execution of CPUID Function 1) indicates support for SYSENTER/SYSEXIT instructions. An operating system that detects the presence of the SEP Feature bit must also qualify the processor family and model to ensure that the SYSENTER/SYSEXIT instructions are actually present



Feature Flags Reported in the ECX Register (Sheet 2 of 2)

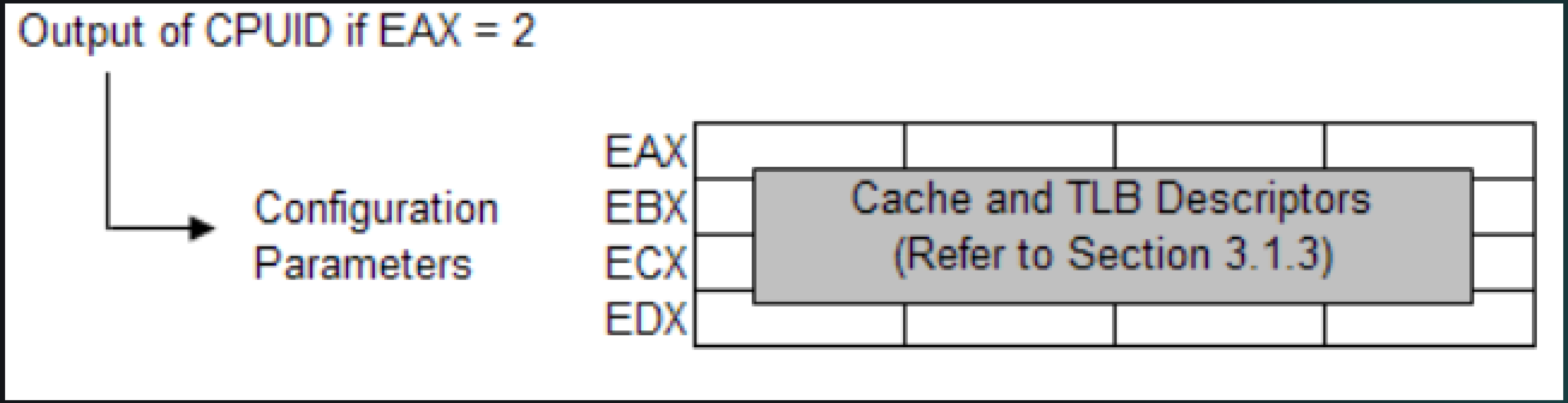
Bit	Name	Description when Flag = 1	Comments
23	POPCNT	POPCNT Instruction	The processor supports the POPCNT instruction.
24	TSC-DEADLINE	Time Stamp Counter Deadline	The processor's local APIC timer supports one-shot operation using a TSC deadline value.
25	AES	AES Instruction Extensions	The processor supports the AES instruction extensions.
26	XSAVE	XSAVE/XSTOR States	The processor supports the XSAVE/XRSTOR processor extended states feature, the XSETBV/XGETBV instructions, and the XFEATURE_ENABLED_MASK register (XCR0)
27	OSXSAVE	OS-Enabled Extended State Management	A value of 1 indicates that the OS has enabled XSETBV/XGETBV instructions to access the XFEATURE_ENABLED_MASK register (XCR0), and support for processor extended state management using XSAVE/XRSTOR.
28	AVX	Advanced Vector Extensions	The processor supports the AVX instruction extensions.
29	F16C	16-bit floating-point conversion instructions	A value of 1 indicates that the processor supports 16-bit floating-point conversion instructions.
30	RDRAND	RDRAND instruction supported	A value of 1 indicates that processor supports RDRAND instruction.
31		Not Used	Always returns 0.

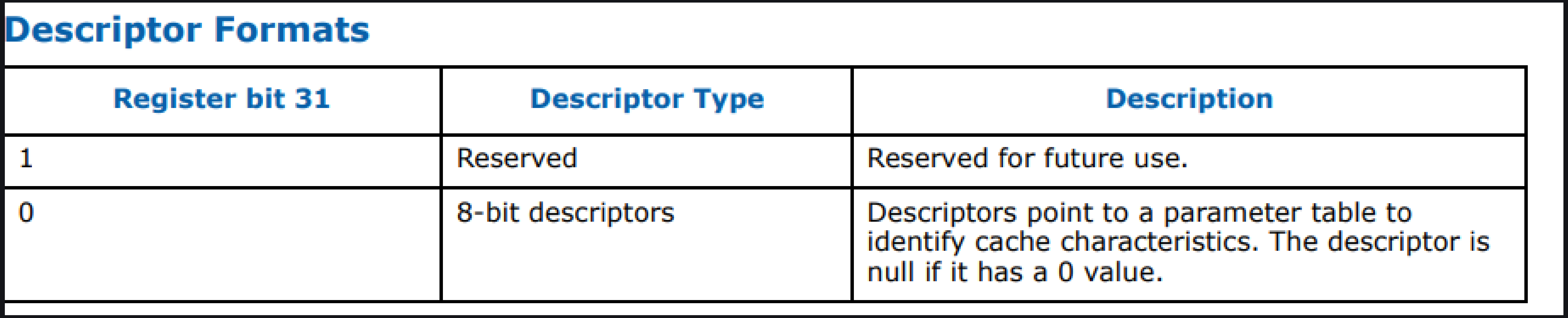
Cache Descriptors (Function 02h):

Provides cache and TLB information in the form of 8-bit descriptors.

The TLB is a very small, incredibly fast cache memory within the processor that is responsible for maintaining "last address translations."

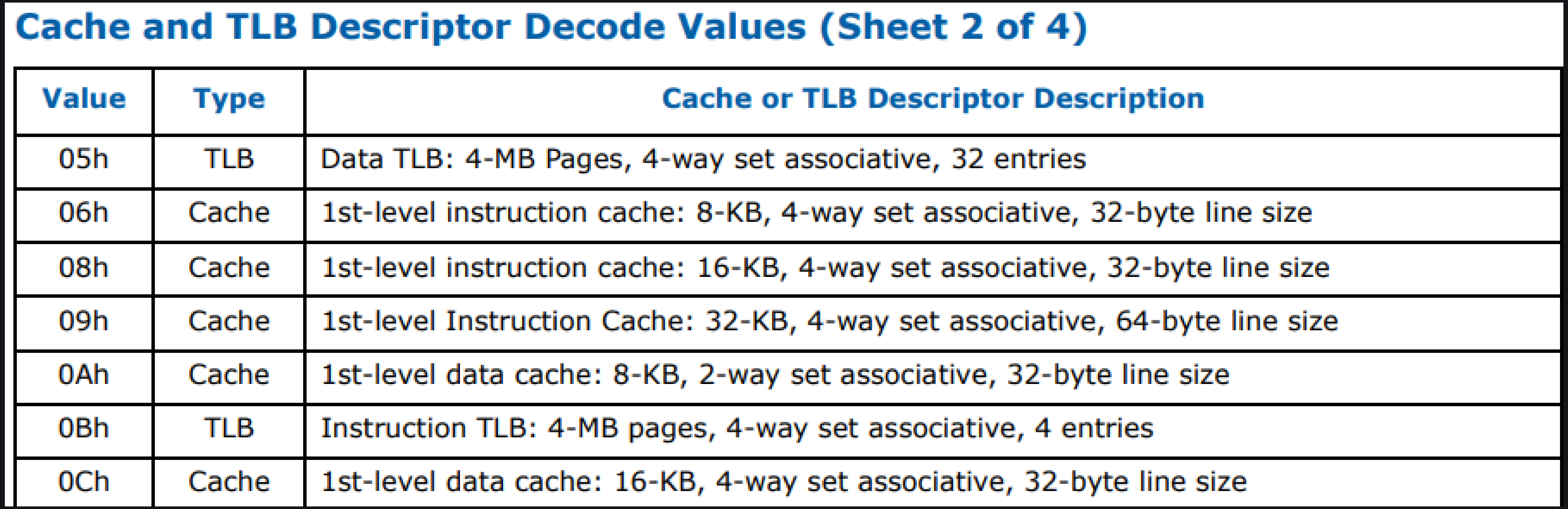
The lower 8 bits of the EAX register (AL) contain a value that identifies the number of times the CPUID must be executed in order to obtain a complete image of the processor's caching systems


Output of CPUID if EAX = 2
Configuration Parameters
EAX
EBX
ECX
EDX
Cache and TLB Descriptors (Refer to Section 3.1.3)



Descriptor Formats

Register bit 31	Descriptor Type	Description
1	Reserved	Reserved for future use.
0	8-bit descriptors	Descriptors point to a parameter table to identify cache characteristics. The descriptor is null if it has a 0 value.

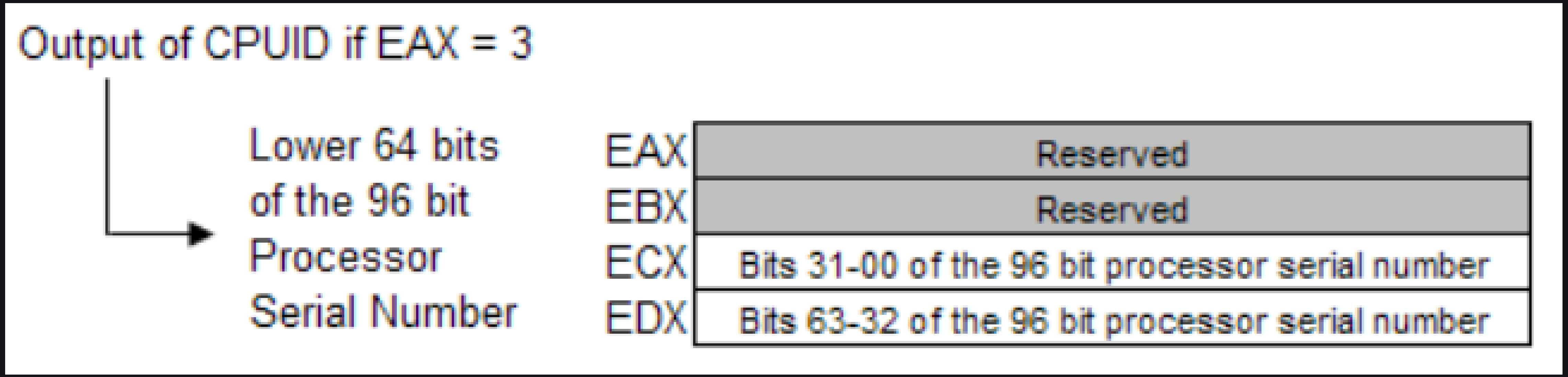


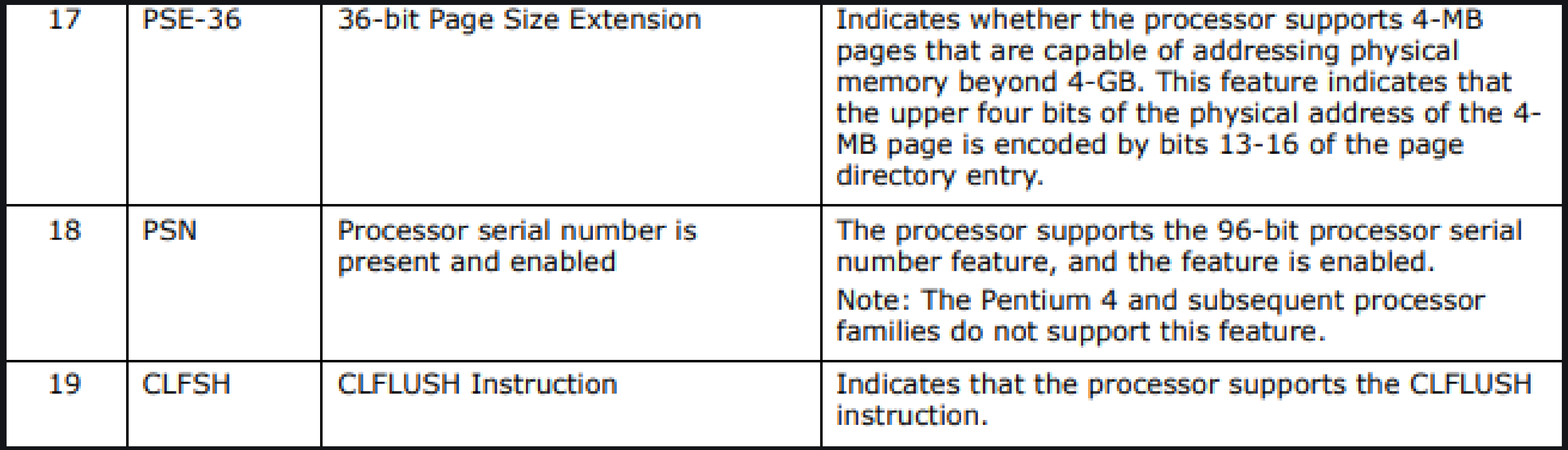
Cache and TLB Descriptor Decode Values (Sheet 2 of 4)

Value	Type	Cache or TLB Descriptor Description
05h	TLB	Data TLB: 4-MB Pages, 4-way set associative, 32 entries
06h	Cache	1st-level instruction cache: 8-KB, 4-way set associative, 32-byte line size
08h	Cache	1st-level instruction cache: 16-KB, 4-way set associative, 32-byte line size
09h	Cache	1st-level Instruction Cache: 32-KB, 4-way set associative, 64-byte line size
0Ah	Cache	1st-level data cache: 8-KB, 2-way set associative, 32-byte line size
0Bh	TLB	Instruction TLB: 4-MB pages, 4-way set associative, 4 entries
0Ch	Cache	1st-level data cache: 16-KB, 4-way set associative, 32-byte line size

Processor Serial Number (Function 03h)

Processor serial number (PSN) is available in Pentium III processor only. The value in this register is reserved in the Pentium 4 processor or later. On all models, use the PSN flag (returned using CPUID) to check for PSN support before accessing the feature.


Output of CPUID if EAX = 3
Lower 64 bits of the 96 bit Processor Serial Number
EAX
Reserved
EBX
Reserved
ECX
Bits 31-00 of the 96 bit processor serial number
EDX
Bits 63-32 of the 96 bit processor serial number



17	PSE-36	36-bit Page Size Extension	Indicates whether the processor supports 4-MB pages that are capable of addressing physical memory beyond 4-GB. This feature indicates that the upper four bits of the physical address of the 4-MB page is encoded by bits 13-16 of the page directory entry.
18	PSN	Processor serial number is present and enabled	The processor supports the 96-bit processor serial number feature, and the feature is enabled. Note: The Pentium 4 and subsequent processor families do not support this feature.
19	CLFSH	CLFLUSH Instruction	Indicates that the processor supports the CLFLUSH instruction.

Deterministic Cache Parameters (Function 04h)

When EAX is initialized to a value of 4, the CPUID instruction returns deterministic cache information in the EAX, EBX, ECX and EDX registers. This function requires ECX be initialized with an index which indicates which cache to return information about. The OS is expected to call this function (CPUID.4) with ECX = 0, 1, 2, until EAX[4:0] == 0, indicating no more caches. The order in which the caches are returned is not specified and may change at Intel's discretion.

Calculating the Cache Size:

Cache Size in Bytes = (Ways +1) x (Partitions +1) x (Line Size +1) x (Sets +1)

Cache Sharing Among Cores and Threads:

Software may wish to determine how many threads share a given cache and also which specific threads share a cache. The following sections explain a method to determine this for processors that support CPUID Function 04h and another method that can be used when CPUID function 0Bh is also available. Both of these methods
determine a cache mask width for each type of cache. Using this value and knowledge of the number of processors in the system, it is possible to determine how many processors share a specific type of cache. If the APIC IDs for every processor in the system is also known, it is possible to determine which processors share a cache.

What is the goal?

Suppose you have 4 logical processors (with APIC IDs 0, 1, 2, 3).
The level 1 cache is usually dedicated (each has its own cache).
The level 3 cache is usually shared between all.
The level 2 cache can be shared between any 2 cores.
This algorithm wants to find exactly this grouping.

1.Determining Cache Sharing Using Function 04h:

input:

- List of all APIC IDs: which we learned earlier how to get with CPUID (EAX=1).
- Cache sharing capacity: How many people can share this level of cache (e.g. L2) at most? This is what the CPUID (EAX=4) command tells us.

1.Ask the CPUID command how many threads this cache can hold.

2.Calculate how many bits are needed for this number (take the logarithm).

3."Remove" (zero) that number of bits from the right side of the APIC ID of all processors.

4.Now any processor whose remaining number is the same, that is, they use a single cache.

2.Determining Cache Sharing Using Function 0Bh

example:

Suppose our system has 4 processors:

CPU 0 (APIC ID: 00)

CPU 1 (APIC ID: 01)

CPU 2 (APIC ID: 10)

CPU 3 (APIC ID: 11)

And our cache is shared between all 2. So we need to ignore the right 1 bit to figure out the groups.

MONITOR / MWAIT Parameters (Function 05h):

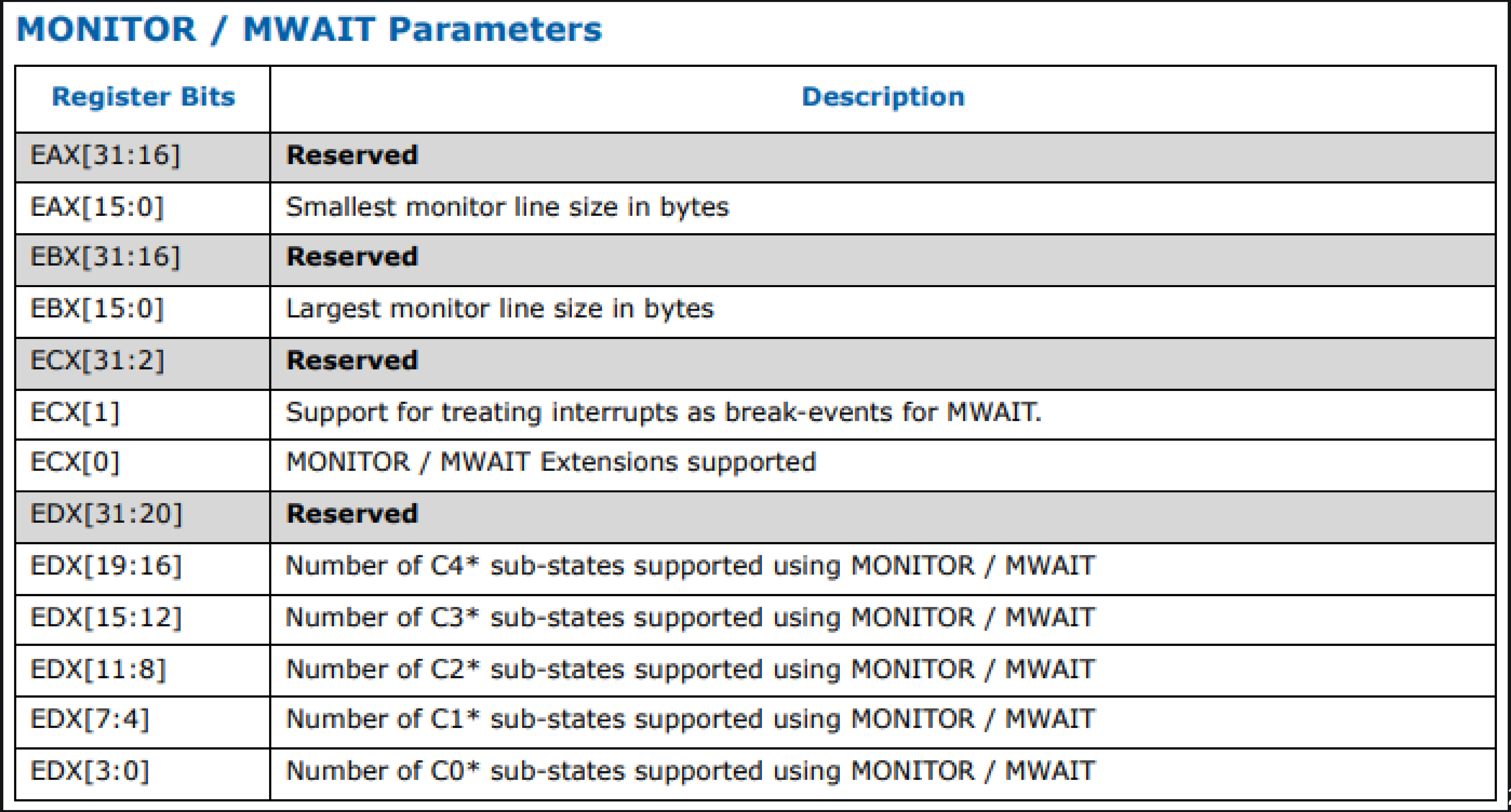
It's one of the smartest features of modern processors to save power and increase performance.

When EAX is initialized to a value of 5, the CPUID instruction returns MONITOR / MWAIT parameters in the EAX, EBX, ECX and EDX registers if the MONITOR and MWAIT instructions are supported by the processor.

Imagine you are a security guard waiting for a light in the room across from you to turn on so you can start your work. You have two options:

- The old method (Polling): Every few seconds, don't blink and ask "Is it on? No. Is it on? No..." (This takes a lot of energy and makes you tired).

- The smart method (MONITOR/MWAIT): You tell your nervous system: "I'm sleeping, but watch out for that light. Wake me up as soon as it turns on." (This means using almost zero energy while waiting).

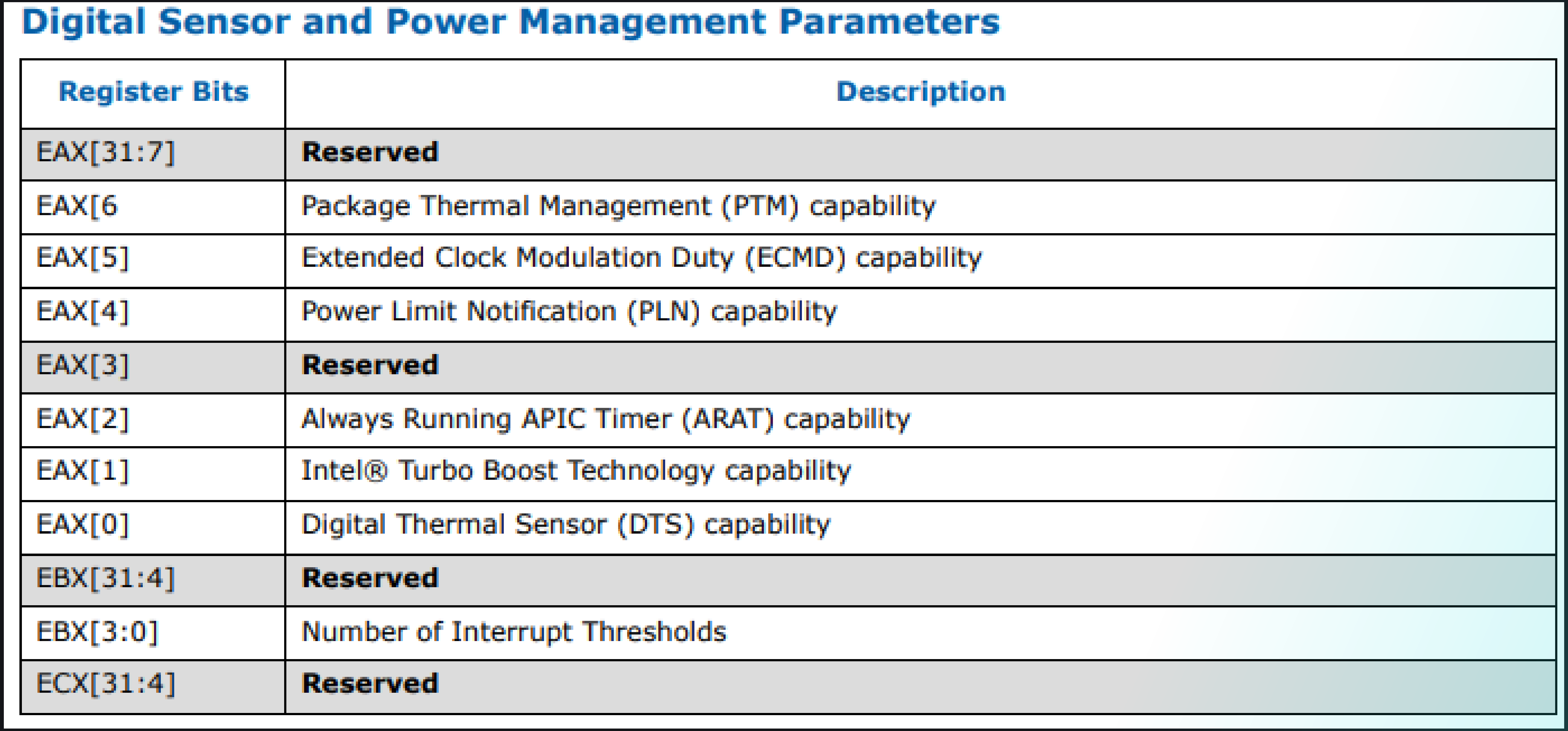


MONITOR / MWAIT Parameters

Register Bits	Description
EAX[31:16]	**Reserved**
EAX[15:0]	Smallest monitor line size in bytes
EBX[31:16]	**Reserved**
EBX[15:0]	Largest monitor line size in bytes
ECX[31:2]	**Reserved**
ECX[1]	Support for treating interrupts as break-events for MWAIT.
ECX[0]	MONITOR / MWAIT Extensions supported
EDX[31:20]	**Reserved**
EDX[19:16]	Number of C4* sub-states supported using MONITOR / MWAIT
EDX[15:12]	Number of C3* sub-states supported using MONITOR / MWAIT
EDX[11:8]	Number of C2* sub-states supported using MONITOR / MWAIT
EDX[7:4]	Number of C1* sub-states supported using MONITOR / MWAIT
EDX[3:0]	Number of C0* sub-states supported using MONITOR / MWAIT

Digital Thermal Sensor and Power Management Parameters (Function 06h):

When EAX is initialized to a value of 6, the CPUID instruction returns Digital Thermal Sensor and Power Management parameters in the EAX, EBX, ECX and EDX registers.

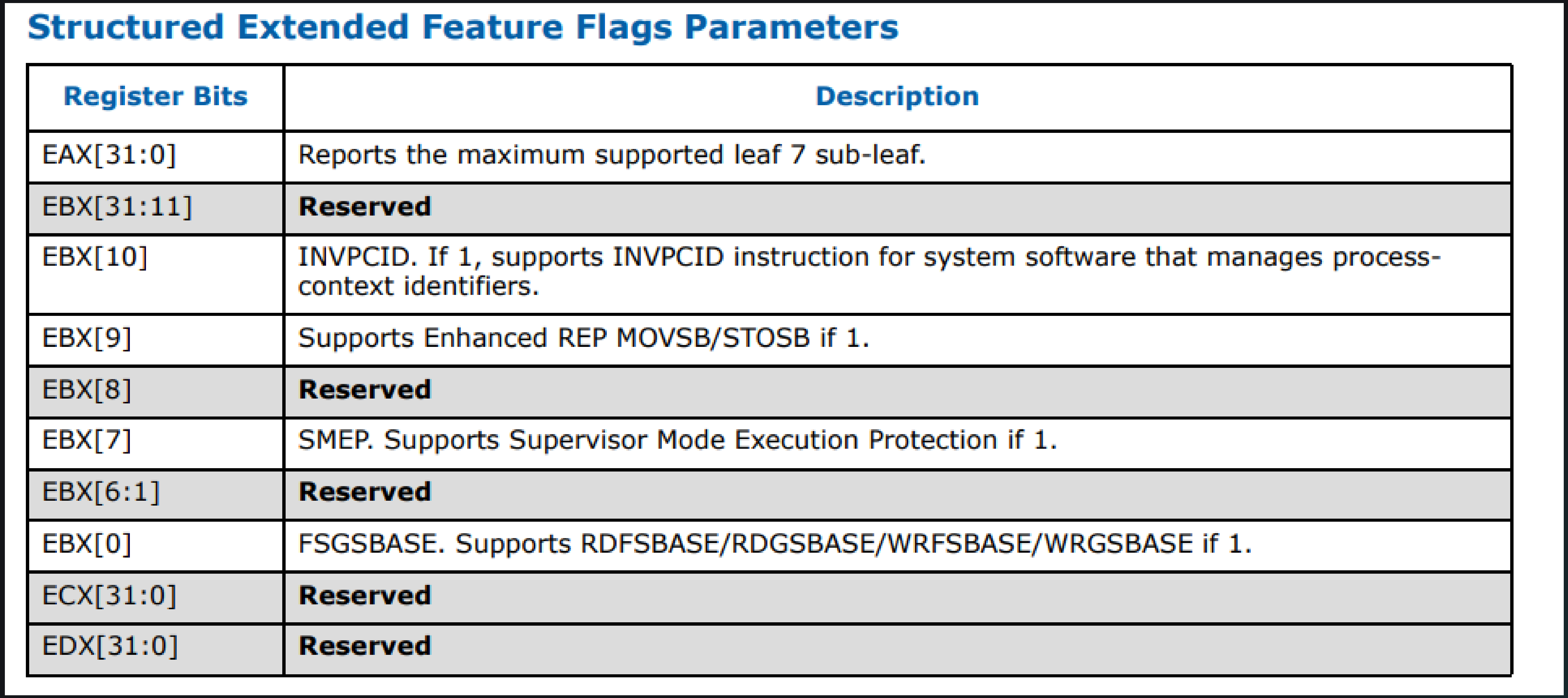


Digital Sensor and Power Management Parameters

Register Bits	Description
EAX[31:7]	**Reserved**
EAX[6	Package Thermal Management (PTM) capability
EAX[5]	Extended Clock Modulation Duty (ECMD) capability
EAX[4]	Power Limit Notification (PLN) capability
EAX[3]	**Reserved**
EAX[2]	Always Running APIC Timer (ARAT) capability
EAX[1]	Intel® Turbo Boost Technology capability
EAX[0]	Digital Thermal Sensor (DTS) capability
EBX[31:4]	**Reserved**
EBX[3:0]	Number of Interrupt Thresholds
ECX[31:4]	**Reserved**

Structured Extended Feature Flags Enumeration (Function07h):

In older architectures, all the processor features were contained in the same EAX=1 function. But as technology advanced, the number of features became so large that they ran out of bits! That's why they created the 07h function, which is like a "multi-page file". You flip pages by changing ECX (which is called a Sub-leaf here)



Structured Extended Feature Flags Parameters

Register Bits	Description
EAX[31:0]	Reports the maximum supported leaf 7 sub-leaf.
EBX[31:11]	**Reserved**
EBX[10]	INVPCID. If 1, supports INVPCID instruction for system software that manages process-context identifiers.
EBX[9]	Supports Enhanced REP MOVSB/STOSB if 1.
EBX[8]	**Reserved**
EBX[7]	SMEP. Supports Supervisor Mode Execution Protection if 1.
EBX[6:1]	**Reserved**
EBX[0]	FSGSBASE. Supports RDFSBASE/RDGSBASE/WRFSBASE/WRGSBASE if 1.
ECX[31:0]	**Reserved**
EDX[31:0]	**Reserved**

Reserved (Function 08h)

This function is reserved

Direct Cache Access (DCA) Parameters (Function 09h)

This is a very advanced hardware feature that is mostly used in servers.

What is the problem? When a high-speed network card receives data, it writes it to RAM (main memory). Then the processor has to read it from RAM and bring it into its cache (cache) to process it. This round trip is time-consuming.

DCA Solution: This technology allows the network card to inject data directly into the processor cache! (without waiting for the processor to request it)

Architectural Performance Monitor Features (Function 0Ah)

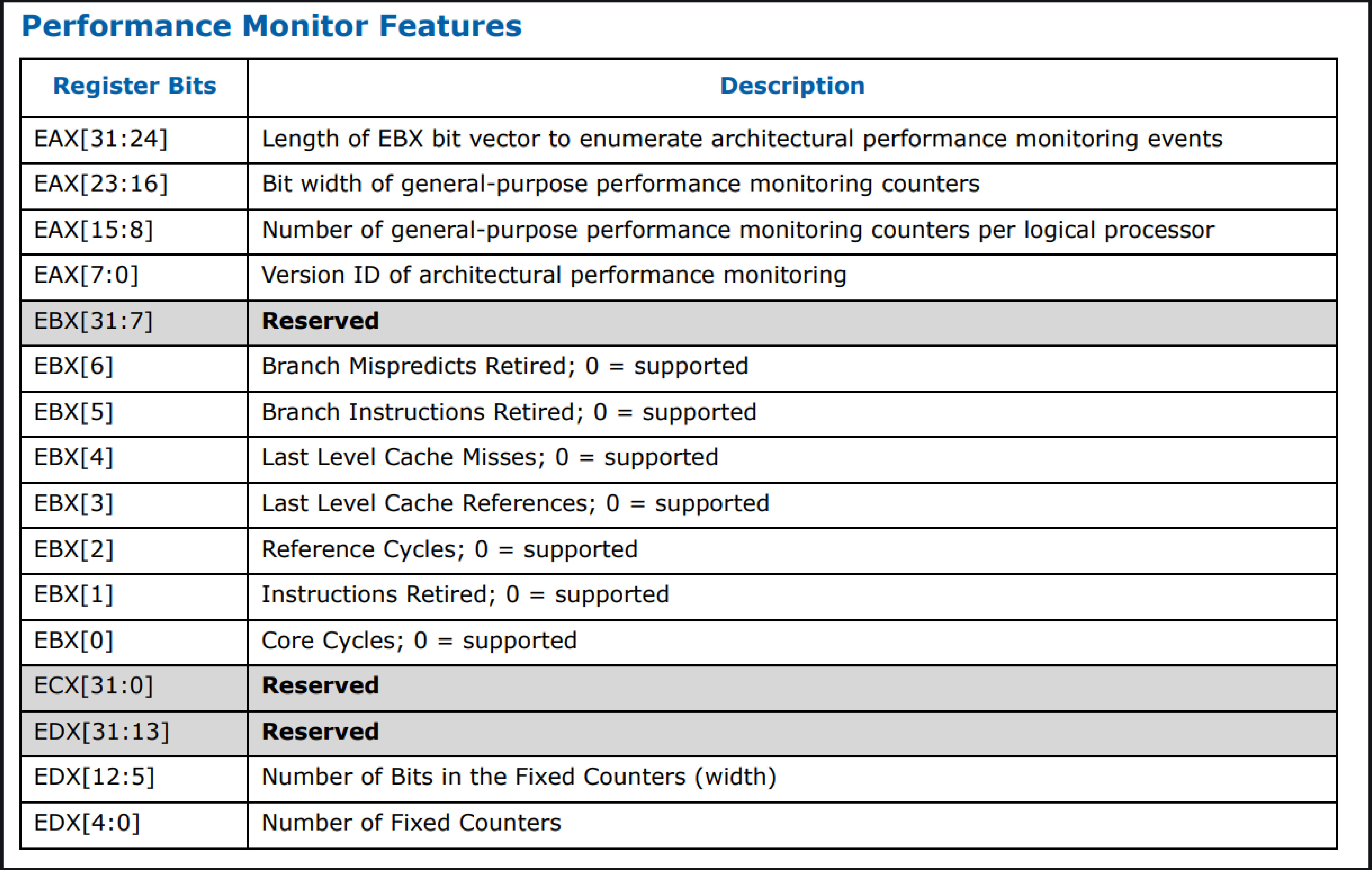
When CPUID executes with EAX set to 0Ah, the processor returns information about support for architectural performance monitoring capabilities. Architectural performance monitoring is supported if the version ID is greater than 0.

This page is essentially a "feature catalog."

It tells you that if you set EAX to 0A, I (the processor) will tell you which of these measurement tools are available in my hardware so that they can be used to debug programs.

- How many instructions were executed?
- How many times did the processor have to fetch information from RAM because it was not in the cache?

- How many times did the branch prediction fail?



Performance Monitor Features

Register Bits	Description
EAX[31:24]	Length of EBX bit vector to enumerate architectural performance monitoring events
EAX[23:16]	Bit width of general-purpose performance monitoring counters
EAX[15:8]	Number of general-purpose performance monitoring counters per logical processor
EAX[7:0]	Version ID of architectural performance monitoring
EBX[31:7]	**Reserved**
EBX[6]	Branch Mispredicts Retired; 0 = supported
EBX[5]	Branch Instructions Retired; 0 = supported
EBX[4]	Last Level Cache Misses; 0 = supported
EBX[3]	Last Level Cache References; 0 = supported
EBX[2]	Reference Cycles; 0 = supported
EBX[1]	Instructions Retired; 0 = supported
EBX[0]	Core Cycles; 0 = supported
ECX[31:0]	**Reserved**
EDX[31:13]	**Reserved**
EDX[12:5]	Number of Bits in the Fixed Counters (width)
EDX[4:0]	Number of Fixed Counters

x2APIC Features / Processor Topology (Function 0Bh):

This function allows the operating system (OS) or BIOS to understand exactly what the processor structure is (how many physical cores it has, how many threads it has, and how their architecture relates to each other).

Unlike the previous simple functions that were called only once, this function must be executed in a loop, similar to function 4 (cache).

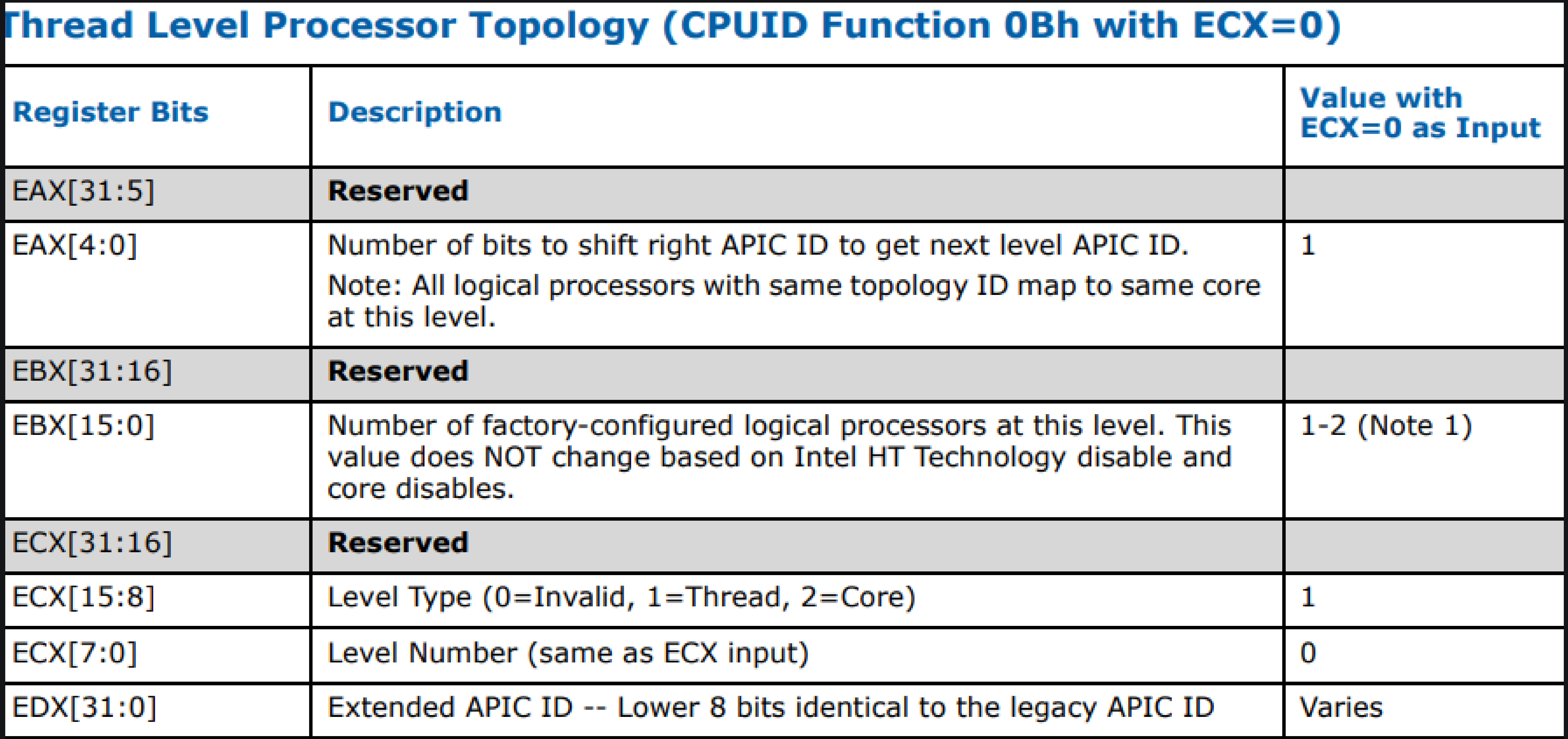
Fixed input: EAX = 0Bh.

Variable input (index): ECX. This register plays the role of "Level Number".

ECX = 0: First level (lowest level - usually Thread level).
ECX = 1: Second level (usually Core level).
ECX = n: Higher levels (up to physical package level).

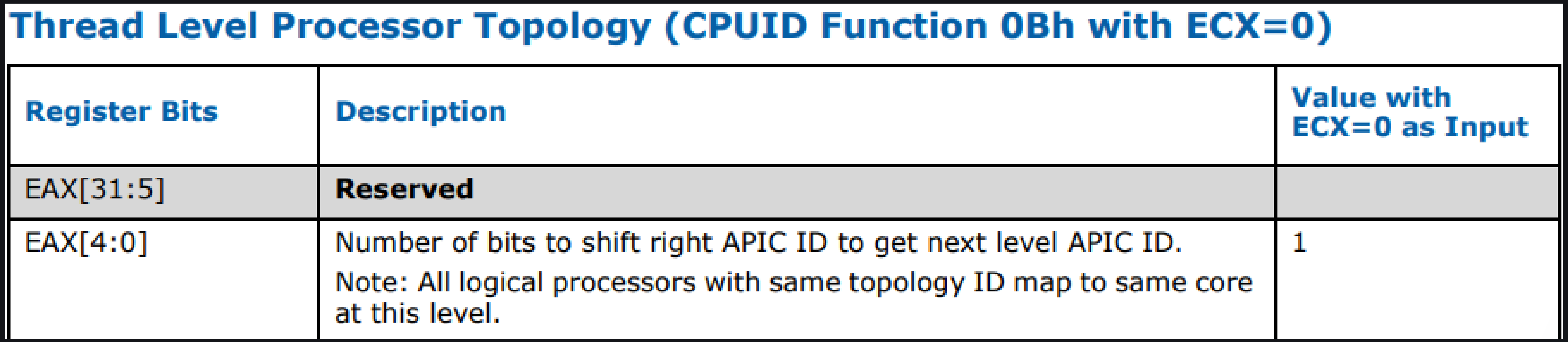
Exit condition: The software should increment ECX by one and execute the instruction until the processor returns EAX = 0 and EBX = 0 at the output. This means "no more levels" and the loop ends.

CPUID Function 0Bh with ECX = 0



Thread Level Processor Topology (CPUID Function 0Bh with ECX=0)

Register Bits	**Description**	**Value with ECX=0 as Input**
EAX[31:5]	**Reserved**	
EAX[4:0]	Number of bits to shift right APIC ID to get next level APIC ID. Note: All logical processors with same topology ID map to same core at this level.	1
EBX[31:16]	**Reserved**	
EBX[15:0]	Number of factory-configured logical processors at this level. This value does NOT change based on Intel HT Technology disable and core disables.	1-2 (Note 1)
ECX[31:16]	**Reserved**	
ECX[15:8]	Level Type (0=Invalid, 1=Thread, 2=Core)	1
ECX[7:0]	Level Number (same as ECX input)	0
EDX[31:0]	Extended APIC ID -- Lower 8 bits identical to the legacy APIC ID	Varies



Thread Level Processor Topology (CPUID Function 0Bh with ECX=0)

Register Bits	Description	Value with ECX=0 as Input
EAX[31:5]	**Reserved**	
EAX[4:0]	Number of bits to shift right APIC ID to get next level APIC ID. Note: All logical processors with same topology ID map to same core at this level.	1

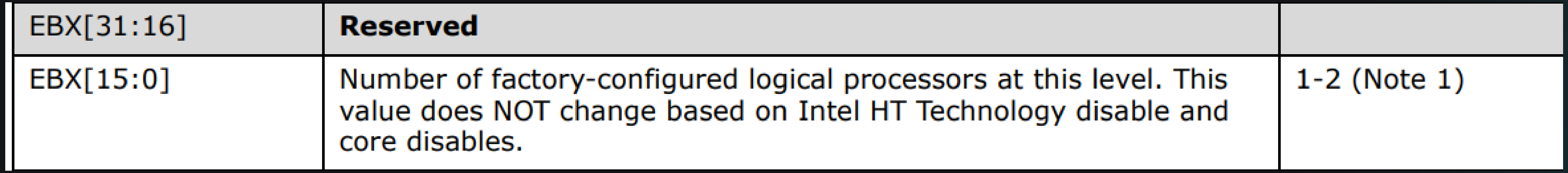
Suppose Intel were to make devices in the future that contained 6 core in each package:

Calculating EAX (shift bits):

How many bits do we need to address 6 unique items?

We need to allocate 3 bits

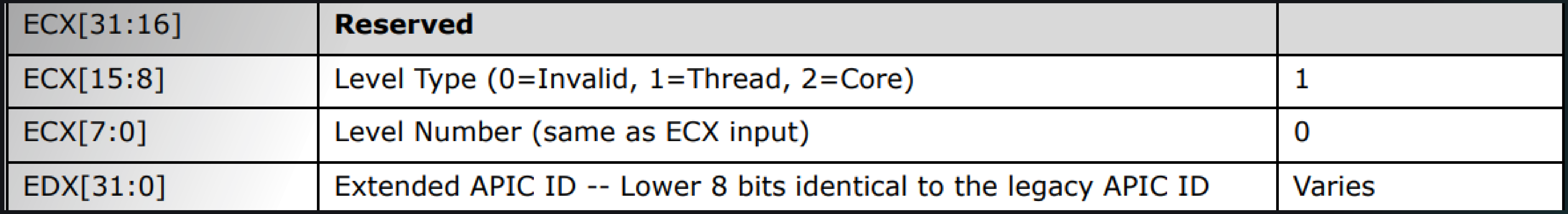
so EAX = 3



EBX[31:16]	**Reserved**	
EBX[15:0]	Number of factory-configured logical processors at this level. This value does NOT change based on Intel HT Technology disable and core disables.	1-2 (Note 1)

EBX = 6

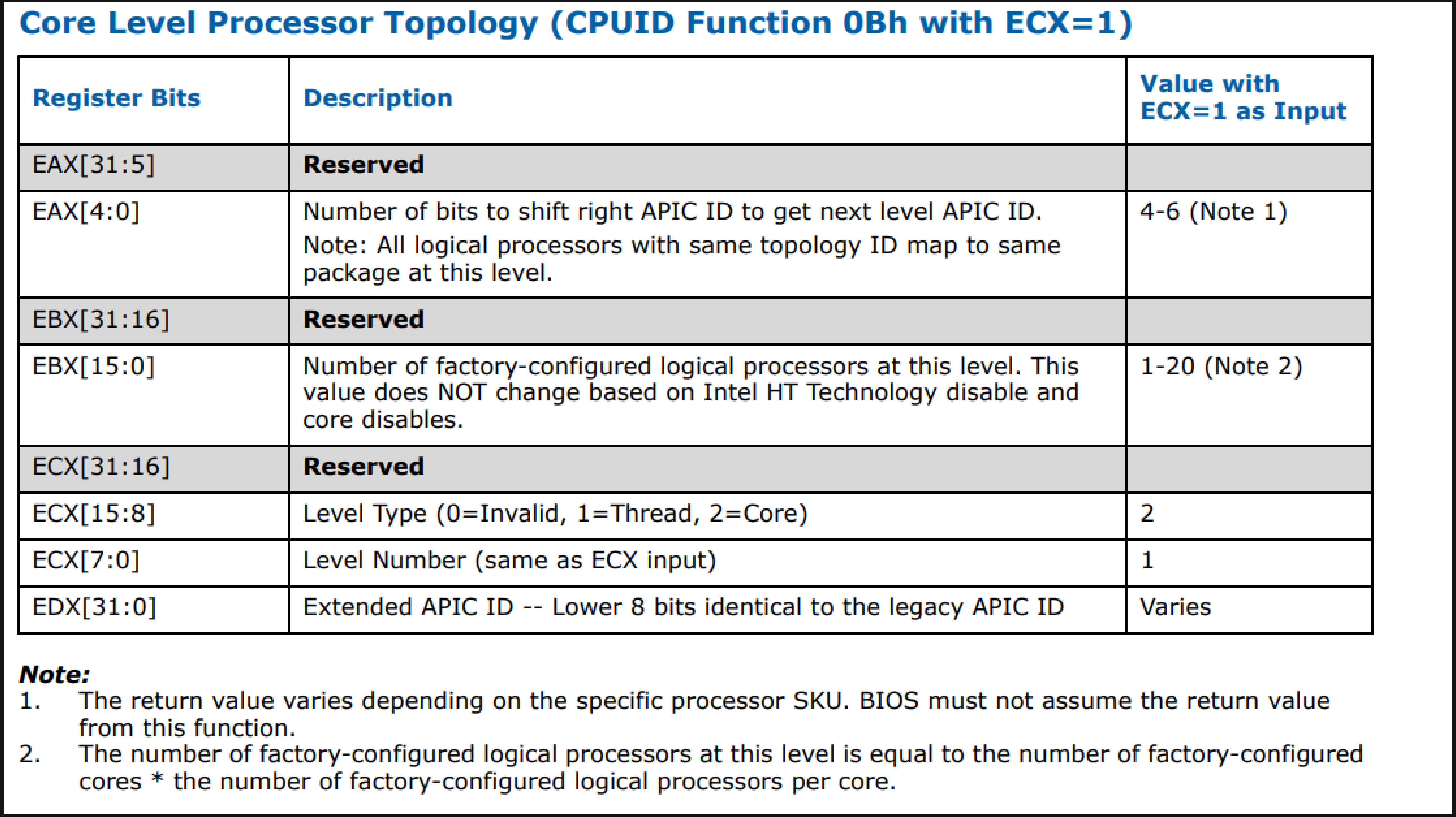
What is the ECX & EDX:



ECX[31:16]	**Reserved**	
ECX[15:8]	Level Type (0=Invalid, 1=Thread, 2=Core)	1
ECX[7:0]	Level Number (same as ECX input)	0
EDX[31:0]	Extended APIC ID -- Lower 8 bits identical to the legacy APIC ID	Varies

EDX: This 32-bit number is the unique ID of the Logical Processor, or "thread," that is executing this instruction.

CPUID Function 0Bh with ECX = 1



Core Level Processor Topology (CPUID Function 0Bh with ECX=1)

Register Bits	Description	Value with ECX=1 as Input
EAX[31:5]	**Reserved**	
EAX[4:0]	Number of bits to shift right APIC ID to get next level APIC ID. Note: All logical processors with same topology ID map to same package at this level.	4-6 (Note 1)
EBX[31:16]	**Reserved**	
EBX[15:0]	Number of factory-configured logical processors at this level. This value does NOT change based on Intel HT Technology disable and core disables.	1-20 (Note 2)
ECX[31:16]	**Reserved**	
ECX[15:8]	Level Type (0=Invalid, 1=Thread, 2=Core)	2
ECX[7:0]	Level Number (same as ECX input)	1
EDX[31:0]	Extended APIC ID -- Lower 8 bits identical to the legacy APIC ID	Varies

Note:
1. The return value varies depending on the specific processor SKU. BIOS must not assume the return value from this function.
2. The number of factory-configured logical processors at this level is equal to the number of factory-configured cores * the number of factory-configured logical processors per core.

EAX register: Shift bits
Bits 0-4: This number tells the operating system how many bits of the APIC identifier to shift right to reach the "next level identifier" (usually the package or physical processor level).

Value: Usually a number between 4 and 6 (depending on the number of cores in the particular processor).

Usage: The operating system uses this number to know which logical processors belong to a physical package. All processors that shift their identifiers to the same number are in the same package.

EBX register: Total number of logical processors. Important: This value is based on the "factory configuration" and will not change even if you disable hyperthreading in the BIOS or turn off a core.

EDX register: throughout all stages of this loop execution (whether ECX=0 or ECX=1), always returns a constant number: "The unique identifier of the processor executing the instruction."

CPUID Function 0Bh with ECX >= 2

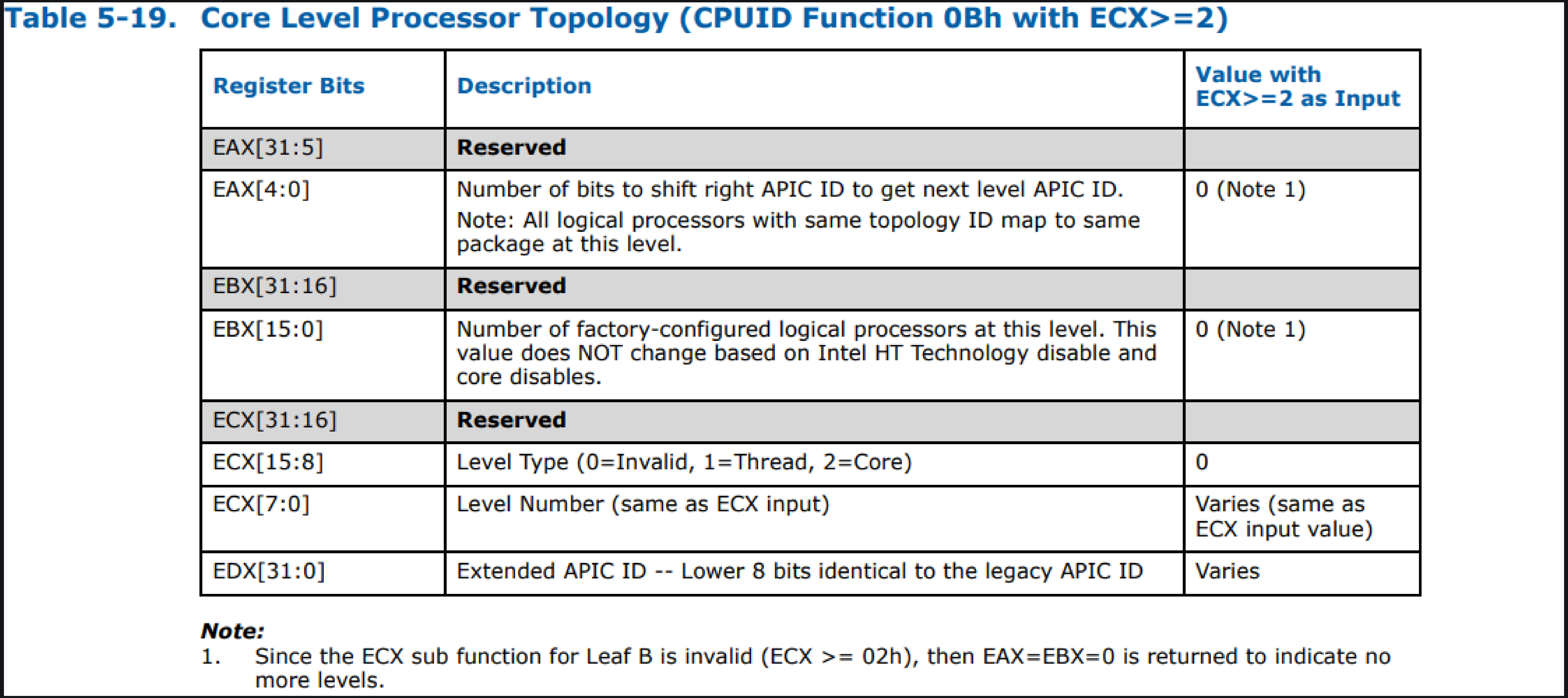


Table 5-19. Core Level Processor Topology (CPUID Function 0Bh with ECX>=2)

Register Bits	Description	Value with ECX>=2 as Input
EAX[31:5]	**Reserved**	
EAX[4:0]	Number of bits to shift right APIC ID to get next level APIC ID. Note: All logical processors with same topology ID map to same package at this level.	0 (Note 1)
EBX[31:16]	**Reserved**	
EBX[15:0]	Number of factory-configured logical processors at this level. This value does NOT change based on Intel HT Technology disable and core disables.	0 (Note 1)
ECX[31:16]	**Reserved**	
ECX[15:8]	Level Type (0=Invalid, 1=Thread, 2=Core)	0
ECX[7:0]	Level Number (same as ECX input)	Varies (same as ECX input value)
EDX[31:0]	Extended APIC ID -- Lower 8 bits identical to the legacy APIC ID	Varies

Note:

1. Since the ECX sub function for Leaf B is invalid (ECX >= 02h), then EAX=EBX=0 is returned to indicate no more levels.

EAX Register: Stop Signal
Bits 0-4: The value of this field is 0.

Meaning: When the shift bits are set to zero, there is no higher level to go to. This is the first signal to the operating system to stop the loop.

EBX Register: No Processors
.Bits 0-15: The value of this field is also 0.

.Meaning: The number of logical processors in this level is zero. This makes sense because there is no level, so there are no processors

ECX Register: Invalid Type
Bits 8-15 (Level Type): This field returns the value 0.

Meaning: In the Intel standard, type 0 means Invalid. This is the final confirmation that level 2 (and higher) is not defined in the current architecture.

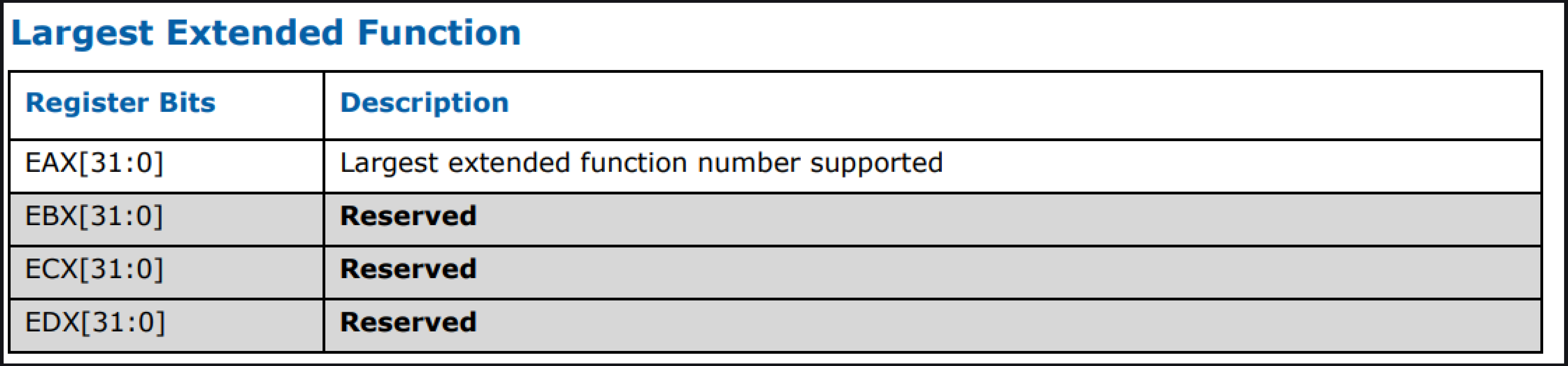
Bits 0-7 (Level Number): Returns the same number you gave in the ECX input (for example, 2).

EDX register: throughout all stages of this loop execution (whether ECX=0 or ECX=1), always returns a constant number: "The unique identifier of the processor executing the instruction."

Extended CPUID Functions

Largest Extended Function # (Function 80000000h):

When EAX is initialized to a value of 80000000h, the CPUID instruction returns the largest extended function number supported by the processor in register EAX.



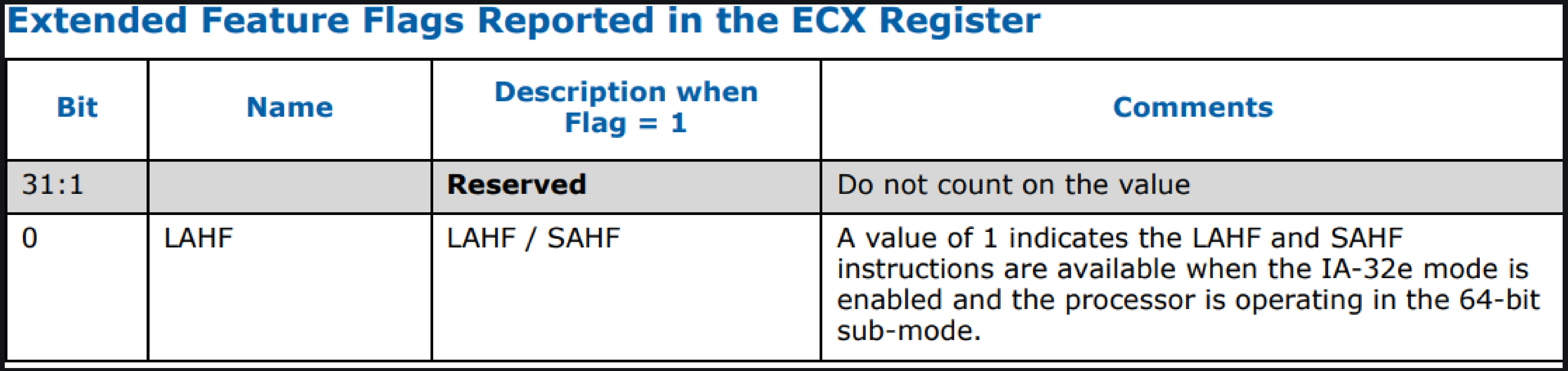
Largest Extended Function

Register Bits	Description
EAX[31:0]	Largest extended function number supported
EBX[31:0]	**Reserved**
ECX[31:0]	**Reserved**
EDX[31:0]	**Reserved**

Extended Feature Bits (Function 80000001h):

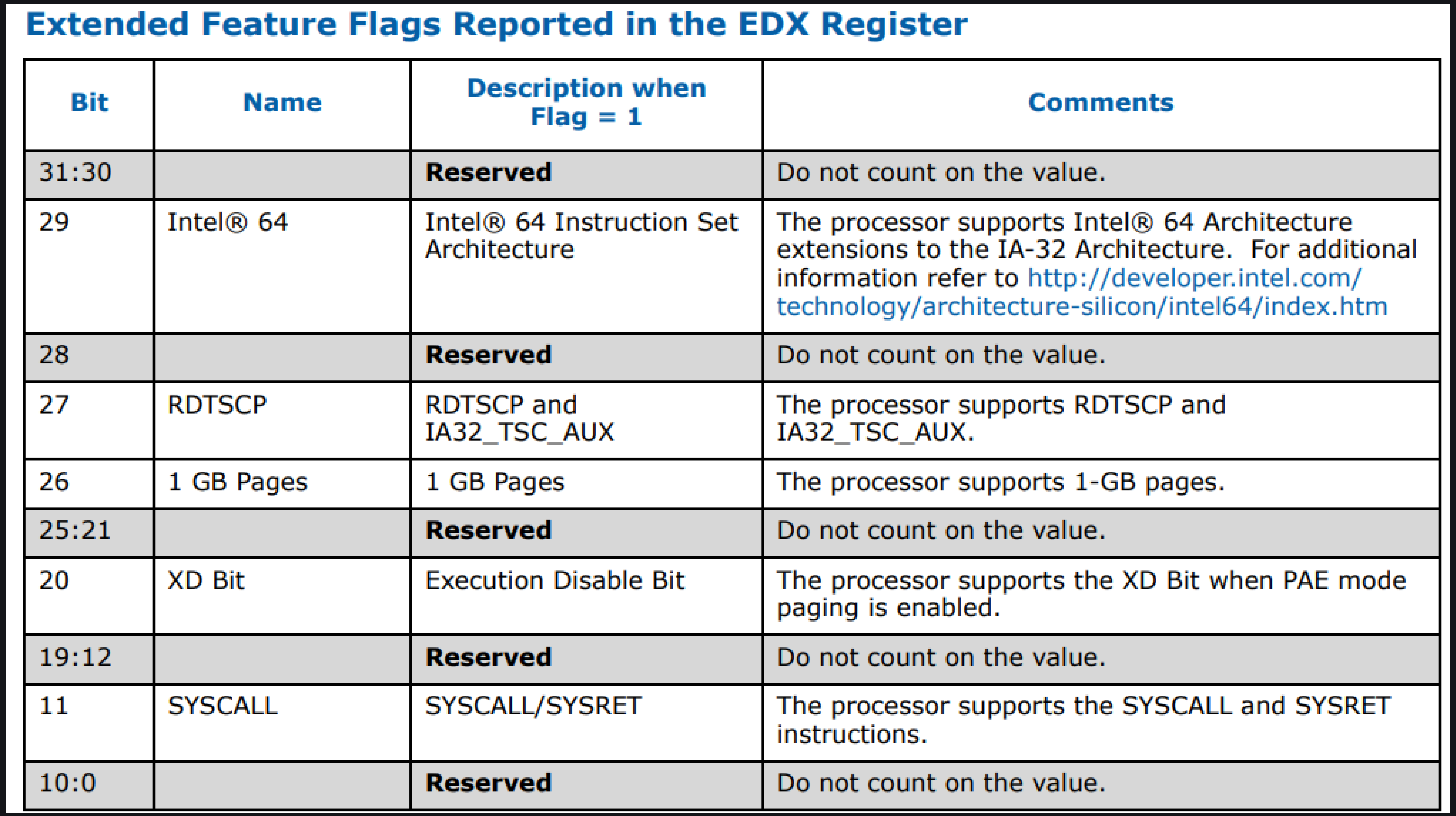
When the EAX register contains a value of 80000001h, the CPUID instruction loads the EDX register with the extended feature flags. The feature flags (when a Flag = 1) indicate what extended features the processor supports.

By using CPUID feature flags to determine processor features, software can detect and avoid incompatibilities introduced by the addition or removal of processor features.



Extended Feature Flags Reported in the ECX Register

Bit	Name	Description when Flag = 1	Comments
31:1		**Reserved**	Do not count on the value
0	LAHF	LAHF / SAHF	A value of 1 indicates the LAHF and SAHF instructions are available when the IA-32e mode is enabled and the processor is operating in the 64-bit sub-mode.



Extended Feature Flags Reported in the EDX Register

Bit	Name	Description when Flag = 1	Comments
31:30		**Reserved**	Do not count on the value.
29	Intel® 64	Intel® 64 Instruction Set Architecture	The processor supports Intel® 64 Architecture extensions to the IA-32 Architecture. For additional information refer to http://developer.intel.com/technology/architecture-silicon/intel64/index.htm
28		**Reserved**	Do not count on the value.
27	RDTSCP	RDTSCP and IA32_TSC_AUX	The processor supports RDTSCP and IA32_TSC_AUX.
26	1 GB Pages	1 GB Pages	The processor supports 1-GB pages.
25:21		**Reserved**	Do not count on the value.
20	XD Bit	Execution Disable Bit	The processor supports the XD Bit when PAE mode paging is enabled.
19:12		**Reserved**	Do not count on the value.
11	SYSCALL	SYSCALL/SYSRET	The processor supports the SYSCALL and SYSRET instructions.
10:0		**Reserved**	Do not count on the value.

The most significant bit in this table(bit 29):
If this bit is 1, your processor is 64-bit (x64 / x86-64).
Usage: This bit tells the operating system that it can boot in "Long Mode". If this bit is 0, the processor is only 32-bit (which is no longer the case with modern processors).

bit 27: Support for RDTSCP instructions
Usage: This command is used to read the "Time-Stamp Counter" but unlike the regular RDTSC, it ensures that the previous commands have been fully executed and also returns the processor ID. This is very crucial for accurate benchmarking.

bit 26:Meaning: Support for 1 GB memory pages.

bit 20:This is a very important security feature. It allows the operating system to mark parts of memory (such as the Stack or Heap) as "non-executable". This prevents many virus attacks and Buffer Overflows.

bit 11:Support for SYSCALL and SYSRET commands.
This is the fastest way to go from User Mode to Kernel Mode on 64-bit systems.

Processor Brand String (Function 80000002h, 80000003h,80000004h)

Functions 80000002h, 80000003h, and 80000004h each return up to 16 ASCII bytes of the processor name in the EAX, EBX, ECX, and EDX registers. The processor name s constructed by concatenating each 16-byte ASCII string returned by the three functions. The processor name is right justified with leading space characters. It is returned in little-endian format and NULL terminated. The processor name can be a maximum of 48 bytes including the NULL terminator character. In addition to the processor name, these functions return the maximum supported speed of the processor in ASCII.

Building the Processor Brand String

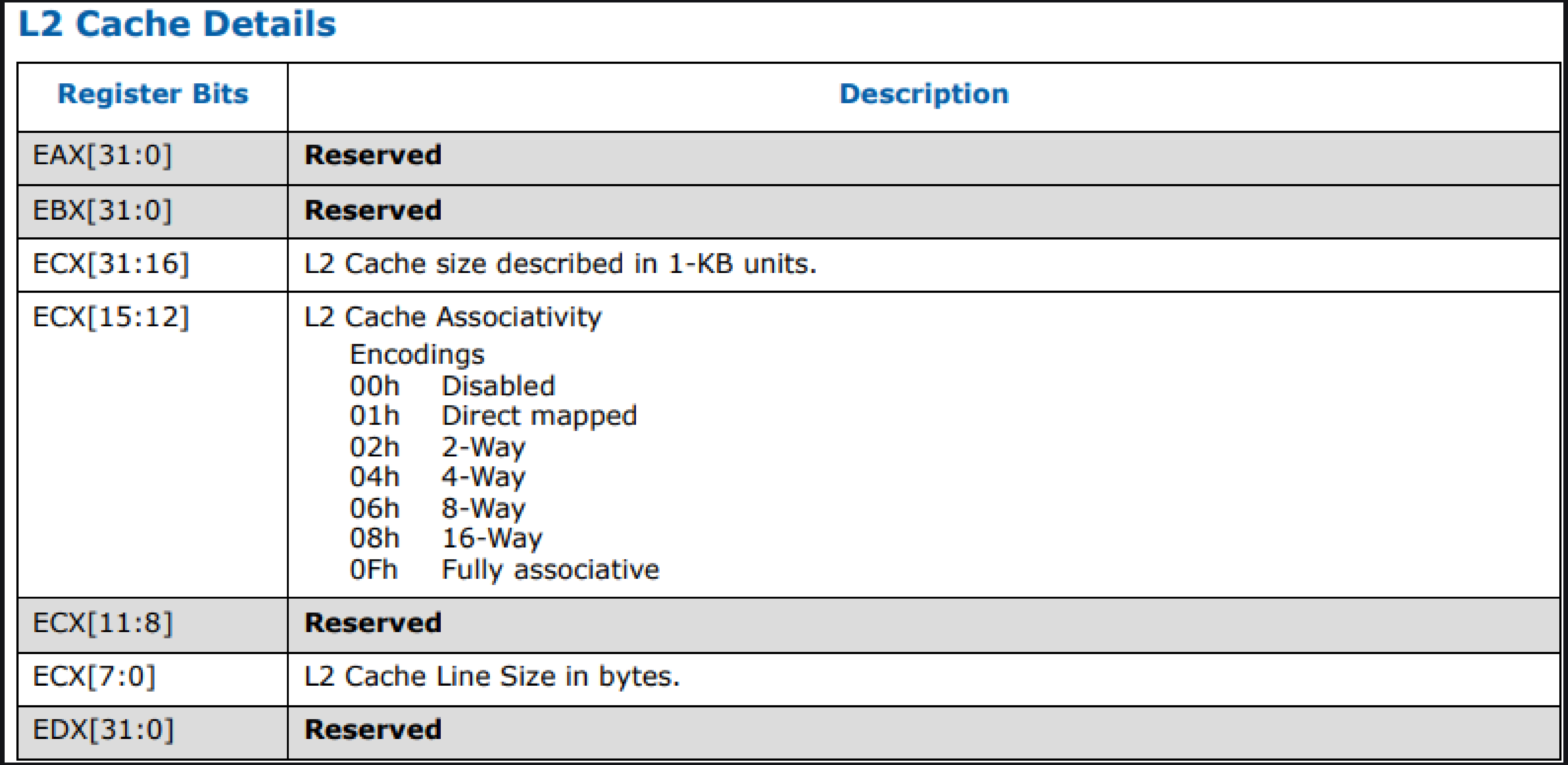
BIOS must reserve enough space in a byte array to concatenate the three 16 byte ASCII strings that comprise the processor name. BIOS must execute each function in sequence. After sequentially executing each CPUID Brand String function, BIOS must concatenate EAX, EBX, ECX, and EDX to create the resulting Processor Brand String.

Reserved (Function 80000005h)

This function is reserved.

Extended L2 Cache Features (Function 80000006h)

Functions 80000006h returns details of the L2 cache in the ECX register. The details returned are the line size, associativity, and the cache size described in 1024-byte units.

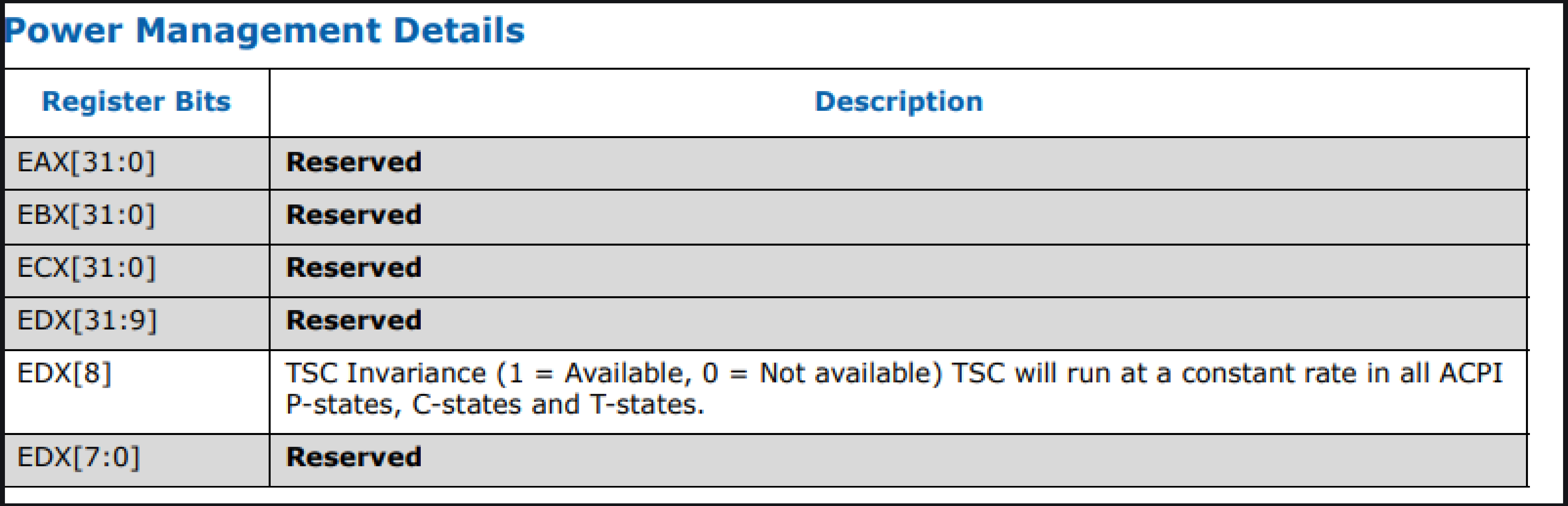


L2 Cache Details

Register Bits	Description
EAX[31:0]	**Reserved**
EBX[31:0]	**Reserved**
ECX[31:16]	L2 Cache size described in 1-KB units.
ECX[15:12]	L2 Cache Associativity Encodings 00h Disabled 01h Direct mapped 02h 2-Way 04h 4-Way 06h 8-Way 08h 16-Way 0Fh Fully associative
ECX[11:8]	**Reserved**
ECX[7:0]	L2 Cache Line Size in bytes.
EDX[31:0]	**Reserved**

Advanced Power Management (Function 80000007h)

In the Core i7 and future processor generations, the TSC will continue to run in the deepest C-states. Therefore, the TSC will run at a constant rate in all ACPI P-, C -. and T- states. Support for this feature is indicated by CPUID.(EAX=80000007h):EDX[8]. On processors with invariant TSC support, the OS may use the TSC for wall clock timer services (instead of ACPI or HPET timers). TSC reads are much more efficient and do not incur the overhead associated with a ring transition or access to a platform resource.



Power Management Details

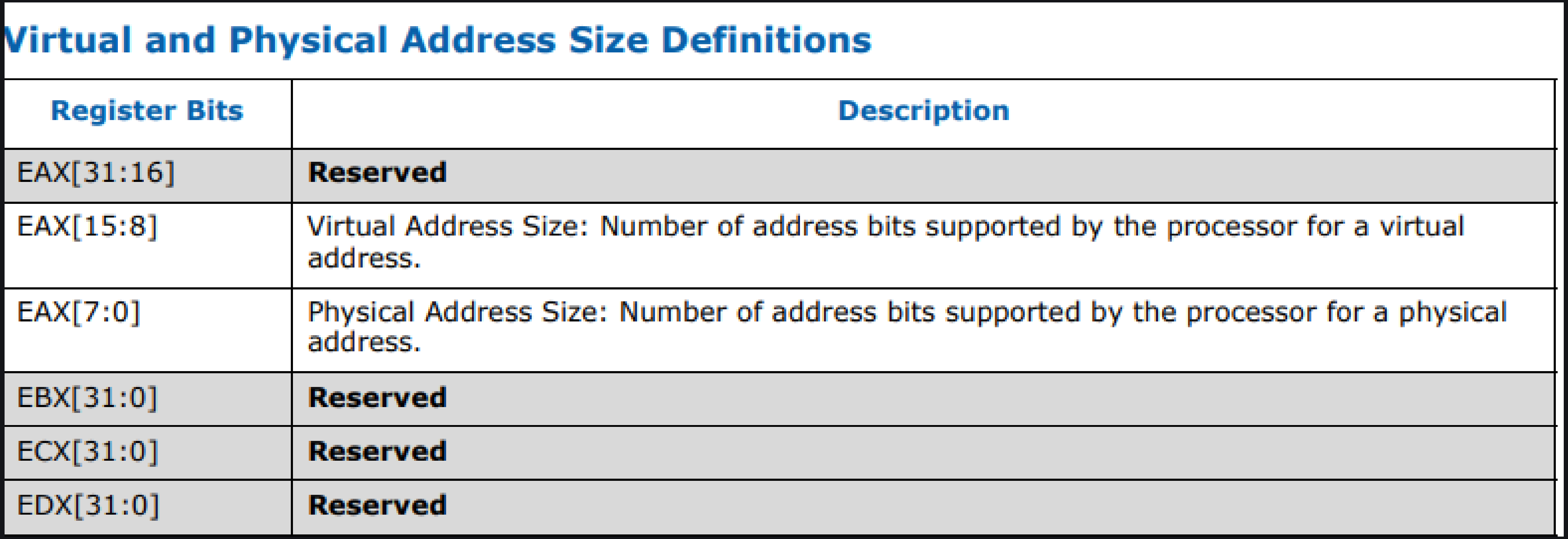
Register Bits	Description
EAX[31:0]	**Reserved**
EBX[31:0]	**Reserved**
ECX[31:0]	**Reserved**
EDX[31:9]	**Reserved**
EDX[8]	TSC Invariance (1 = Available, 0 = Not available) TSC will run at a constant rate in all ACPI P-states, C-states and T-states.
EDX[7:0]	**Reserved**

TSC: It is like a super accurate stopwatch for measuring the passage of time. The audio and video sync depends on the TSC.

Virtual and Physical Address Sizes (Function 80000008h)

On the Core Solo, Core Duo, Core2 Duo processor families, when EAX is initialized to a value of 80000008h, the CPUID instruction will return the supported virtual and physical address sizes in EAX. Values in other general registers are reserved. This information is useful for BIOS to determine processor support for Intel® 64 Instruction Set Architecture(Intel® 64).

If this function is supported, the Physical Address Size returned in EAX[7:0] should be used to determine the number of bits to configure MTRRn_PhysMask values with. Software must determine the MTRR PhysMask for each execution thread based on this function and not assume all execution threads in a platform have the same number of physical address bits.



Virtual and Physical Address Size Definitions

Register Bits	Description
EAX[31:16]	**Reserved**
EAX[15:8]	Virtual Address Size: Number of address bits supported by the processor for a virtual address.
EAX[7:0]	Physical Address Size: Number of address bits supported by the processor for a physical address.
EBX[31:0]	**Reserved**
ECX[31:0]	**Reserved**
EDX[31:0]	**Reserved**

ginyard

conclusion

CPUID instruction: The main tool for extracting hardware information. The program must first verify that the processor supports this instruction by checking the ID flag in the EFLAGS register.

Standard and extended functions: By changing the input to the EAX register, various information is returned, including the string "GenuineIntel", the processor signature (family, model, stepping), and cache details.

Feature Flags check: The document emphasizes that software should not guess capabilities based on the processor model, but should check specific bits for features such as SSE, AVX, or FPU.

Brand identification: Explains how to obtain the Brand ID (a numeric code for older models) and Brand String (a text string with the full processor name and nominal frequency).

Specific applications: Includes methods for detecting DAZ mode (for computing efficiency) and calculating the actual operating frequency of the processor thank you