ddcpuid User Manual

Exploring your x86 micro-processor

First Edition

Table of Contents

1 Introduction

ddcpuid is a simple x86 information tool that works best with Intel and AMD micro-processors.

It solely relies on the CPUID instruction.

By default, ddcpuid shows basic information, like the CPU vendor string, processor brand string, instruction extensions, cache, and processor technologies.

1.1 History

The project started in June 26, 2016, in D. It was created as a mean of exploration into x86.

On February 1st, 2017, ddcpuid got 64-bit compilation support.

On December 11, 2017, ddcpuid was rewritten to be compiled with D’s betterC feature.

On June 15, 2018, cache information was added.

1.2 References

This project is based on information from *Intel® 64 and IA-32 Architectures Software Developer’s Manual* (May 2018) and *AMD64 Architecture Programmer’s Manual Volume 3: General-Purpose and System Instructions* (May 2018).

2 Command Line Interface

The help screen can be accessed via `-h` or –`help`.

Some options can be combined: `-ro` is the same as `-r -o`.

2.1 Options

`-d`

Advanced mode. Includes advanced information that would be useful for an engineer

`-r`

Show raw CPUID data in a table

`-o`

Override CPUID leaves to 20H and 8000_0020H respectively, only useful with `-r`

`-v`, `--version`

Show version screen and exit

`-h`, `--help`

Show help screen and exit

3 Basic Mode

By default, ddcpuid shows information in basic mode, which includes:

- Vendor string;
- Brand string (model);
- Identifier;
- Extensions (e.g. MMX, SSE2, etc.);
- Cache information for one physical core;
- And featured technologies.

3.1 Identifier Calculation

On Intel processors, processor identifiers are calculated according to the reference manual (Page 3-206 Vol. 2A, Figure 3-6):

```
if (BaseFamily != 0)
     Family = BaseFamily;
else
     Family = ExtendedFamily + BaseFamily;

if (BaseFamily == 6 || BaseFamily == 0)
     Model = (ExtendedModel << 4) + BaseModel;
else
     Model = BaseModel;
```

On AMD processors, processor identifiers are calculated according to the reference manual (Page 603, §E.3.2, §CPUID Fn0000_0001_EAX):

```
if (BaseFamily < 0xF) {
     Family = BaseFamily;
     Model = BaseModel;
} else {
     Family = ExtendedFamily + BaseFamily;
     Model = (ExtendedModel << 4) + BaseModel;
}
```

3.2 Supported Processor Features

3.2.1 Intel Processors

- Enhanced SpeedStep(R) Technology
- TurboBoost

3.2.2 AMD Processors

- Core Performance Boost

4 Advanced Mode

Advanced mode, accessed via `-d`, is a mode that displays information that may hopefully be useful to engineers, which additionally includes:

- Highest processor leaf and extended leaf;
- Processor type;
- FPU features;
- APCI and APIC features;
- Virtualization features;
- Memory features;
- Debugging features;
- Brand index
- And miscellaneous features.

Features are explained in §Processor Features.

4.1 Advanced Identifier Display

In advanced mode, ddcpuid shows the processor identifier like so (in hexadecimal):

```
Family [BaseFamily:ExendedFamily] Model [BaseModel:ExtendedModel]
Stepping
```

Identifier calculation is explained in §Identifier Calculation.

4.2 Processor Type

Processor type is a deprecated feature in Intel processors and includes the following values:

- Original OEM Processor
- Intel OverDrive Processor
- Dual Processor
- Reserved (Intel)

All recent Intel processors show Original OEM Processor.

5 Processor Features

This section attempts to explain some of the more advanced processor features.

5.1 FPU Features

Most recent x86 processors include what is called a Floating Point Unit. A processor unit designed to handle decimal values, sometimes known as float numbers, following the IEEE 754 standard.

5.1.1 16-bit conversion

If set, 16-bit conversion is available, such as converting 32-bit float numbers to 16-bit numbers and vice versa.

5.2 APCI features

The Advanced Configuration and Power Interface was introduced to help power and thermal management through the operating system.

If set, the APCI feature is present.

5.2.1 APIC

The Advanced Programmable Interrupt Controller is an updated standard from Intel, from the older PIC standard. It is used to redirect interrupts effectively.

If set, an APIC is present.

Initial ID refers to the logical core that the program is being run on. On a modern operating system, this will be random.

Max ID refers to the maximum ID that the program can run under. Note that Intel processor usually have this value doubled to the logical core count.

5.2.2 Thermal Monitor

A thermal monitor monitors the processor temperature through its sensor.

If set, a thermal monitor is available.

5.3 Virtualization Features

5.3.1 Virtual 8086 Mode Enhancements

A number of enhancements were added within the Pentium architecture to the virtual 8086 mode, including virtual interrupts.

If set, the virtual 8086 mode enhancements are available.

5.4 Memory features

5.4.1 Memory Type Range Registers

A set of additional control registers to fine-tune memory regions that should be cached by the processor.

If set, these specific control registers are available.

5.4.2 Page Size Extension

Traditionally, memory pages were 4 KiB in size, which usually ended up cluttering the translation lookaside buffer.

Introduced in the Pentium processor, the page size extension let user programs request bigger memory pages.

If set, the processor can support memory pages larger than 4 KiB.

5.4.3 36-bit Page Size Extension

The 36-bit page size extension is an extension from 32-bit memory page addressing, allowing to address from 4 GiB to 64 GiB of memory.

If set, the processor may address up to at least 64 GiB of memory.

5.4.4 1 GB Pages support

This feature allows the processor to page 1 GiB memory spaces.

If set, the processor can initiate 1 GiB memory pages.

5.4.5 Page Attribute Table

The page attribute table, succeeding from MTRR, allows users to control attributes on a per-page basis. Like if memory pages should be cached.

If set, the processor supports the page attribute table feature.

5.4.6 Page Global Bit

The global bit in a page table is used to prevent the TLB from updating the address in cache if CR3 is reset.

If set, indicates support for the global bit in the page table.

5.4.7 Direct Cache Access

Direct cache access is an I/O technology that permits other devices to directly place data into the processor’s cache.

If set, direct cache access is available for other devices to use.

5.5 Debugging Features

5.5.1 Machine Check Architecture

Mechanism that enables hardware error reporting to the operating system.

If set, machine check architecture is available.

5.5.2 Machine Check Exception

On hardware errors, the processor may throw a machine-check exception, which may include errors about system buses errors, ECC errors, parity errors, etc.

This feature does not define the model-specific implementations of machine-check error logging, reporting, and processor shutdowns.

If set, the processor supports machine-check exceptions.

5.5.3 Debugging Extensions

Defines support for I/O breakpoints.

If set, the processor supports I/O breakpoints.

5.5.4 Debug Store

A debug store is a processor feature where the processor is able to write debugging information to a memory buffer.

If set, the processor may use the debug store.

5.5.5 Debug Store CPL

The processor may supported the extensions to the debug store feature allowing for branch messages storage qualified by the CPL (Ring).

If set, indicates support for debug store CPL message branching.

5.5.6 64-bit DS Area

Allows the use of 64-bit addresses in the debug store area.

If set, the debug store area can be used using 64-bit addresses.

5.5.7 Perfmon And Debug Capability

Indicates performance and debug feature indication in IA32_PERF_CAPABILITIES (MSR).

If set, IA32_PERF_CAPABILITIES feature is available.

5.5.8 IA32_DEBUG_INTERFACE

Firmware may use IA32_DEBUG_INTERFACE (MSR) for silicon debugging.

If set, silicon debugging is available.

5.6 Other Features

5.6.1 Brand Index

Before the processor brand string, the brand index was used instead. For example, a brand index of 04H indicated the Intel(R) Pentium(R) III processor.

If clear, the processor does not support the brand index identification.

Any values higher than 18H and higher are reserved (Intel).

5.6.2 L1 Context ID

The L1 data cache may be adapted to two modes: adaptive and shared. The L1 context ID serves as setting the L1 data context strategy.

If set, the L1 Context ID feature is available.

5.6.3 xTPR Update Control

Intel has yet to disclose information about xTPR Update Control.

If set, the processor may change IA32_MISC_ENABLE[23].

5.6.4 Process-Context Architecture

Enables up to 4,096 processes to be created an managed via a unique ID (PCID) via the processor.

If set, PCID is available.

5.6.5 Processor Serial Number

Introduced only in the Pentium III, the processor serial number is supposed to be a unique serial number per processor packages.

If set, the processor has a serial number.

5.6.6 Self Snoop

Management of conflicting memory types may be done by performing a snoop of its own cache structure for transactions issued to the bus, thus self snooping.

If set, memory self snooping is available.

5.6.7 Pending Break Enable

In interrupt handling, the processor may use FERR# and PBE# pins in a stop-clock state (STPCLK#) that an interrupt in pending and that the processor should return to normal.

If set, supports FERR# and PBE# pins when in a stop-clock state.

5.6.8 Supervisor Mode Execution Protection

Instruction fetches from user-mode addresses may be restricted in privileged addresses.

If set, supervisor mode execution protection is available (CR4.SMEP).

5.6.9 Bit manipulation groups

Defines groups of recent bit manipulation instructions, such as ANDN, MOVBE, PDEP, etc. as BMI1 and BMI2.

ddcpuid may display either, if supported.

6 Raw CPUID information

The `-r` option in ddcpuid prints a table of raw CPUID data, useful for debugging. Sub-leaf information is not provided.

The `-o` option overrides default leaves obtained by the processor and may be useful in exploring data further than the supported CPUID leaf.

Example (Intel Core i7-3770):

```
| Leaf     | EAX      | EBX      | ECX      | EDX      |
|----------|----------|----------|----------|----------|
|        0 |        D | 756E6547 | 6C65746E | 49656E69 |
|        1 |    306A9 |  5100800 | 7FBAE3FF | BFEBFBFF |
|        2 | 76035A01 |   F0B2FF |        0 |   CA0000 |
|        3 |        0 |        0 |        0 |        0 |
|        4 | 1C004121 |  1C0003F |       3F |        0 |
|        5 |       40 |       40 |        3 |     1120 |
|        6 |       77 |        2 |        9 |        0 |
|        7 |        0 |      281 |        0 |        0 |
|        8 |        0 |        0 |        0 |        0 |
|        9 |        0 |        0 |        0 |        0 |
|        A |  7300403 |        0 |        0 |      603 |
|        B |        1 |        2 |      100 |        3 |
|        C |        0 |        0 |        0 |        0 |
|        D |        7 |      340 |      340 |        0 |
| 80000000 | 80000008 |        0 |        0 |        0 |
| 80000001 |        0 |        0 |        1 | 28100800 |
| 80000002 | 20202020 | 20202020 | 65746E49 | 2952286C |
| 80000003 | 726F4320 | 4D542865 | 37692029 | 3737332D |
| 80000004 | 50432030 | 20402055 | 30342E33 |   7A4847 |
| 80000005 |        0 |        0 |        0 |        0 |
| 80000006 |        0 |        0 |  1006040 |        0 |
| 80000007 |        0 |        0 |        0 |      100 |
| 80000008 |     3024 |        0 |        0 |        0 |
```