Simulation of CPUID of Intel Architecture

Grigory Rechistov* Name Surname†

March 15, 2014

Contents

1 Introduction

TODO Write me

1.1 Contributions

In this paper we make the following contributions.

1. Evaluate and compare existing means of processor features identification of different architectures.
2. Describe, implement and evaluate a structured solution to the simulation of CPUID instruction of Intel IA-32.

*Moscow Institute of Physics and Technology, grigory.rechistov@phystech.edu
†email@mail.com

2 Overview of Processor Identification

2.1 MIPS

This architecture processor identification with PRid register [1], which is the 15th register in Coprocessor 0. It contains 32 bits of information (Fig. 1), part of which is vendor–specific.

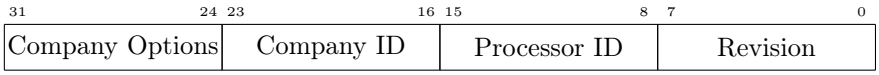

Figure 1: MIPS PRid register fields

TODO Check this
`http://hwdb.mipt.cc/MIPS_PRId_register`
`http://code.google.com/p/phantomuserland/source/browse/trunk/phantom/dev/mips/cpuid.c?r=1094`
`http://www.imgtec.com/mips/mips32-architecture.asp`

2.2 ARM

The popular RISC architecture provides certain means of a processor identification with CPUID base register [2]. It contains 32 bits of information (Fig. 2).
TODO Chehk this
`http://infocenter.arm.com/help/topic/com.arm.doc.dai0099c/DAI0099C_core_type_rev_id.pdf`
`http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0363e/ch01s11s01.html`
`http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0363e/Bcgdjadd.html`
`http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0388f/Babififh.html`
`http://forum.xda-developers.com/showthread.php?t=480226`

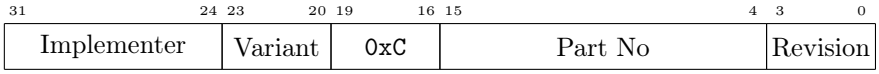

Figure 2: ARM CPUID base register fields

Examples of values [3]: Intel PXA272 (P535) – 0x69054117, Qualcomm MSM7200A — 0x4117b362.

2.3 IBM System z

IBM System z10 [4] provides STSI and STIDP instructions. TODO Expand

2.4 PowerPC

PowerPC [5] offers a 32-bit PVR register that contains just version and revision numbers.

2.5 Intel IA-64 (Itanium)

Intel Itanium™ [6] was designed later after the original 80x86 series and was meant to supplant it. A set of CPUID registers is used for identification purposes. The set's design somewhat resembles IA-32 CPUID instruction (discussed later) — its register numbers loosely resemble leaves of the latter. Five registers are guaranteed to be present, and bits 0–7 of CPUID [3] store the actual size of the register set (thus it is limited to 256 entries).

An example of Itanium CPUID registers content is given on Fig. 3.

TODO Collect it

Figure 3: Contents CPUID registers of Intel Itanium.

2.6 Intel IA-32 and Intel 64

The common PC architecture, starting from Intel Pentium and its clones, provides CPUID [7, 8] instruction.

Since its inception it has been extended numerous number of times.

On Figure 4 an output of all defined leaves and subleaves for an Intel processor is demonstrated. The resulting table contains 25 tuples of 4 values of 32 bit width each, total of more than 3 kb of information. Essentially a number of values can be deduced from it.

- Processor producer brand (leaf 0) and SKU naming (leaves 0x80000002–0x80000004).
- Availablity of ISA extensions such as 64 bit mode, SSEx, AVX, MOVBE etc.
- Cache configuration of all layers, both in legacy format (leaf 2) and in current format (leaf 4 with subleaves)
- Multi-processor configuration knowledge, such as availability of Intel HyperThreading, relative position inside CPU package (topology), presence of APIC (interrupt controller)
- Numerous architectural varaibles, such as addresses widths (leaf 0x80000008), availability of dynamic frequency scaling etc.

Leaf	Subleaf	EAX	EBX	ECX	EDX
0	0	0xd	0x756e6547	0x6c65746e	0x49656e69
0x1	0	0x306a9	0x6100800	0x7f9ae3bf	0xbfebfbff
0x2	0	0x76035a01	0xf0b0ff	0	0xca0000
0x4	0	0x1c004121	0x1c0003f	0x3f	0
0x4	0x1	0x1c004122	0x1c0003f	0x3f	0
0x4	0x2	0x1c004143	0x1c0003f	0x1ff	0
0x4	0x3	0x1c03c163	0x2c0003f	0x1fff	0x6
0x5	0	0x40	0x40	0x3	0x1120
0x6	0	0x77	0x2	0x9	0
0x7	0	0	0x281	0	0
0xa	0	0x7300803	0	0	0x603
0xb	0	0x1	0x1	0x100	0x6
0xb	0x1	0x4	0x4	0x201	0x6
0xb	0x2	0	0	0x2	0x6
0xb	0x3	0	0	0x3	0x6
0xd	0	0x7	0x340	0x340	0
0xd	0x1	0x1	0	0	0
0xd	0x2	0x100	0x240	0	0
0x80000000	0	0x80000008	0	0	0
0x80000001	0	0	0	0x1	0x28100800
0x80000002	0	0x20202020	0x20202020	0x65746e49	0x2952286c
0x80000003	0	0x726f4320	0x4d542865	0x35692029	0x3534332d
0x80000004	0	0x50432030	0x20402055	0x30312e33	0x7a4847
0x80000006	0	0	0	0x1006040	0
0x80000007	0	0	0	0	0x100
0x80000008	0	0x3024	0	0	0

Figure 4: Output of CPUID instruction obtained for Intel®Core™i5-3450 using `ggg-cpuid` [9]

3 What is so hard about Intel CPUID

A complete definition of CPUID in [7] takes about 40 pages.

There is a number of complications that have resulted from long uncontrolled expansion of the CPUID .

Multiple Vendors Influence Until recently, there were numerous CPU vendors that offered processors compatible with Intel architecture, including IBM, Cyrix, VIA, Centaur, Transmeta etc. By 2014, only two companies remain — Intel itself and AMD. Competition lead to the numerous small and subtle but essential differences in implementation. Until CPUID was introduced,

A robust identification of processor brand and model required surprisingly intricate methods [10].

Elements adressing To inspect a value of a particular

- Leaves
- Subleaves
- Registers
- Bit range

Non-constant values Firmware is able to suppress certain features indicated by CPUID by manipulating bits of model specific register (MSR) IA32_MISC_ENABLE. For example: TODO NX, Leaf3, 1GB pages

Topology-varaible elements Finally, it should be noted that, besides EAX, EBX, ECX, EDX, one more register may be affected by CPUID , namely IA32_SIGNATURE TODO .

4 Existing Approaches to CPUID Simulation

What is required from a CPUID model.

- Be accurate TODO
- Be configurable TODO

4.1 Bochs

4.2 Xen

4.3 Qemu

4.4 Simics

5 The Structured Approach

The described approach
It has the following advantages

- Uses natural unit of configuration state — a field.
- In the meantime, it allows its users to operate in terms of 32 bit leaves values which are more convenient and compact in many cases.
-

6 Evaluation

6.1 Compatibility with existing code

“legacy”

6.2 Extensions

TODO Field flags, such as “hidden”.

7 Conclusions

In this paper we described our approach to simulation of a single but complex processor instruction CPUID .

8 Acknowledgements

References

[1] "MIPS32 architecture", 2014.

[2] ARM Limited, *CPUID Base Register*, 2008.

[3] intruders, "ARM CPUID - processor model detection", 2009.

[4] Per Fremstad, Wolfgang Fries, Marian Gasparovic, Parwez Hamid, Brian Hatfield, Dick Jorna, Fernando Nogal, and Karl-Erik Stenfors, *IBM System z10 Enterprise Class Technical Guide*, IBM, November 2009.

[5] IBM, *PowerPC® Microprocessor Family: The Programming Environments Manual for 64-bit Microprocessors Version 3.0*, July 2005.

[6] Intel Corporation, *Intel® Itanium® Architecture Software Developer's Manual*, 2010.

[7] Intel Corporation, *Intel® 64 and IA-32 Architectures Software Developer's Manual. Volumes 1–3*, 2012.

[8] Advanced Micro Devices, *AMD64 Architecture Programmer's Manual Volume 1: Application Programming*, 2012.

[9] Grigory Rechistov, "ggg-cpuid, a program to print cpuid", 2014.

[10] Robert R. Collins, "CPUID algorithm wars", *Dr. Dobbs*, November 1996.