
AMD DEVELOPER INSIDE TRACK

USING CPUID



This video features Randy Vanderheyden, a 17 year AMD veteran, talking about the best practices on how to use, and how not to use the CPUID instruction. He demonstrates how to find the processor core count with CPUID using the code provided in the [Processor and Core Enumeration Using CPUID](#) article (Thanks for the code Tracy Carver!). Also, for those of you using Visual Studio, there is very detailed information available on MSDN about the [CPUID Intrinsic](#).

TRANSCRIPTS

Randy Vanderheyden, MTS Software Engineer, ISV Engineering Support

Sharon: So, we are here with another episode of the AMD Developer Central Inside Track Video Series, which I'm shortening to ADIT vide series from now on because it is such a long name. We're sitting here with Randy Vanderheyden. Randy, why do you tell us a little bit about yourself, your title and how many years you have been here at AMD and what you do?

Randy: My name is Randy Vanderheyden and I've been here about 17 years, since the days of the 386. I'm a member of the technical staff in the ISV engineering team supporting our software partners.

Sharon: That's been a long ride here at AMD!

Randy: Indeed.

Sharon: So, I think you're very well qualified to tell us a little bit about CPUID. First of all what is it?

Randy: Ok. CPUID is an x86 or x64 instruction. It's been around, I think, since around the k6 days which is probably about 12 years or so. Now this instruction can be run in assembly language. There's also intrinsic for this function. If you are a Microsoft developer and you look for the CPUID intrinsic you will find it in the help system under Visual Studio. This instruction is usually used to find specific details about your hardware. It might be as simple as which instructions does the CPU that I'm running on support. The most common examples today are: Does it support SSE, SSE2 and so on. It can also be used to find a little more specific detail about your microprocessor such as cache sizes and more importantly how many cores are on this processor.

Sharon: Wow, so it gives you, really a lot of information that you can use. So when would you need to use something like CPUID?

Randy: Well there are a couple different times when developers use it. Quite often when they are writing an optimized code path when they want to use or generate, again, some kind of vectorized code like SSE2. They need to put in their code somewhere, some kind of CPUID instruction so that they know the processor they are currently running on supports it. If it doesn't support it then they usually have a more generic code path that the processor will execute on.

Now in more recent years, as we move towards more multithreaded code, developers can use CPUID to determine how many cores on the system the software is presently running on. This perhaps helps them decide how many threads, or how they want to split their code apart to run.

Sharon: That's a good point considering more and more people are having to go to parallel programming these days. So along that line, is there any way that CPUID could be used wrong, or is there anything to look out for? What advice would you give?

Randy: Sure. One of the things CPUID can return is what's called the vendor string or the vendorID and this is a plain text string which will show you who the manufacturer of the processor is. You can also get model numbers from CPUID and while this information may be useful, we don't encourage that you make specific decisions based on this information. For example if you know it is an AMD processor, and if you are familiar enough with the processors, then you might decide, 'I know that all these processors support SSE2'. We don't recommend doing this because you might be cutting out other manufacturers or even other models of AMD processors from executing this optimized code path. So use the specific feature bits that you are really trying to test, whether it is SSE2 or some other feature, just look at those bits and make the decision based on them. That way any other vendor's processor that you are running on will also take advantage of that code path.

Sharon: Great example. So, let's see it in action. Can you show us what it look like when you run a CPUID?

Randy: Sure. What I'm going to run is a sample program that is on our website, developer.amd.com. It's a good example of just basic enumeration of how many processors do I have and how many cores are on each processor.

Ok. This program is just called enum, e-u-n-m. And I'm going to run it with it giving a little bit of extra information. Which if you look at this program on the website you can see how to do that. I'm running this program on an engineering prototype that we have, on what we call internally, the "Magny-Cours" processor. This is a 12-core processor. And this particular box has 2 sockets. And what I'm showing is that this sample code can expand easily to that many processors. It's showing that I have a total of 24 processors, basically 2 sockets, ID0 and ID1. With each one with 12 cores. And up above here it is showing a bit more detailed information, such as the APIC Id and some of the other technical information that is available via CPUID. And by the way this is nowhere near all the information that you can glean from CPUID. Using both our architecture manuals, which is volume 3, or our CPUID Spec, you can read more about the specific bits that are there.

Sharon: Great. So we will have a link to this great article that Tracy Carver wrote that has some code samples right next to the video here and as well as the links to the architecture programmers manuals to so thanks for your time Randy this has been very valuable. There's a lot of various questions that pop up about CPUID on the forums and in our work with developers so I appreciate you taking the time to give us this in person talk about it.

Randy: Your welcome.

Sharon: Thanks!