White Paper

Using CPUID to Detect the presence of SSE 4.1 and SSE 4.2 Instruction Sets

Introduction

Several application notes have been written by Intel to assist customers with discerning which processor their application is running on and the features supported by a particular processor. This information may then be used to choose appropriate code paths for processor specific optimizations, or to selectively enable features based on processing power.

In this application note, a set of code sequences is shown to determine if the processor being queried supports the SSE 4.1 and SSE 4.2 instruction sets . The code in this application note was designed to run on Intel 64 Architecture processors running a 32 bit or 64 bit Windows or Linux Operating System. The code, as shown is designed to be compiled with the Intel compiler, although, only minor changes, would be required to compile the code on other compilers.

Table of Contents

At least two prior reference articles exist that cover or touch the CPUID topic. The two referenced for this application note are listed below.

App Note 485, "Intel® Processor Identification and the CPUID Instruction[i]" explains in depth how to distinguish the various Intel Architecture processors starting with the original 8086. Several customers have requested assistance with CPUID code sequences that will operate under more constrained cicrumstances and thus, can be simplified substantially compared to the general assumptions made in App Note 485.

Another Intel reference is an article titled "Intel® 64 Architecture Processor Topology Enumeration.[ii] " This article covers much more than CPUID. However, it contains code for the CPUID sequence that is much simpler for our usage, and so it is also listed as a reference.

It should be noted that the Intel(r) compiler also supports functionality that removes the burden of CPUID coding from the user and may be preferable. The Intel compiler provides the capability to automatically generate multiple code paths and generate the appropriate CPUID code sequence and runtime code path selection code on a per function basis. The user can specify which functions and should have specific code paths and for which target processors these specific code paths should be generated. The user may write the code for each code path or rely on the compilers auto vectorization capability. This topic is beyond the scope of this app note and is not covered further here.

Usage Guidelines

This code illustrates determining if a particular processor is an Intel processor which supports the SSE 4.1 and the SSE 4.2 instruction sets. operating under the following conditions:

1) The target processor must be a 32 bit capable processor. The presence of the CPUID instruction is determined by checking the ability to toggle bit 21 of the EFLAGS register as specified in the section "Detecting the CPUID Instruction" in Application note 485.

2) Running a 32 bit or 64 bit Windows* or Linux* operating system. (The general principles apply to other operating systems, but may required code modification in order for the code to compile and function correctly due to potential differences in the Application Binary Interfaces (ABIs) of other operationg systems.)

3) Compiled with the Intel® Compiler for the desired target.

4) Use of the –use-msasm switch with the Intel® compiler when the target is Linux. This switch allows the usage of Microsoft assembly syntax preventing the need to have different versions of source code for the two operating systems. This may not be generally possible because of the differences in ABI (application binary interface) between Linux and Windows but is a successful strategy when applicable. Note that the Intel® Compiler is also capable of compiling GNU style assembly code for window targets, though all assembly code in this application note is windows style.

Performance

This code is not in and of itself designed to be high performance. CPUID is not a fast executing instruction. Therefore, it should not be called on a regular basis to determine code path choices when more than one path based on optimization strategy is provided. Instead, this code should be called once at initialization time and the result stored and used to load the correct shared library, or set a global variable to check for code path determination.

Conclusion

The source code provided illustrates that it is fairly simple to determine whether a processor supports the SSE 4.1 and SSE 4.2 instruction set. The code can be easily modified to detect other features designated by other CPUID feature bits by referring to the Intel Software Developers Manual.

* Other names and brands may be claimed as the property of others.

Source Code

/************ Beginning of source file sse41andsse42detection.cpp ********************/

```
/*      Copyright 2009 Intel Corporation
 *      sse41andsse42detection.cpp
 *      This file uses code first published by Intel as part of the processor enumeration
 *      article available on the internet at:
 *      http://software.intel.com/en-us/articles/intel-64-architecture-processor-topology-
*       enumeration/
 *      Some of the original code from cpu_topo.c
 *      has been removed, while other code has been added to illustrate the CPUID usage
 *      to determine if the processor supports the SSE 4.1 and SSE 4.2 instruction sets.
 *      The reference code provided in this file is for demonstration purpose only. It assumes
 *      the hardware topology configuration within a coherent domain does not change during
 *      the life of an OS session. If an OS support advanced features that can change
 *      hardware topology configurations, more sophisticated adaptation may be necessary
 *      to account for the hardware configuration change that might have added and reduced
 *      the number of logical processors being managed by the OS.
 *
 *      Users of this code should be aware that the provided code
 *      relies on CPUID instruction providing raw data reflecting the native hardware
 *      configuration. When an application runs inside a virtual machine hosted by a
 *      Virtual Machine Monitor (VMM), any CPUID instructions issued by an app (or a guest OS)
 *      are trapped by the VMM and it is the VMM's responsibility and decision to emulate
 *      CPUID return data to the virtual machines. When deploying topology enumeration code based
*       on CPUID inside a VM environment, the user must consult with the VMM vendor on how an VMM
 *      will emulate CPUID instruction relating to topology enumeration.
 *
 *      Original code written by Patrick Fay, Ronen Zohar and Shihjong Kuo[ii].
 *      Modified by Garrett Drysdale for current application note.
 */

#include "sse41andsse42detection.h"

#define SSE4_1_FLAG           0x080000
#define SSE4_2_FLAG           0x100000

int isSSE41andSSE42Supported (void)
{
        // returns 1 if is a Nehalem or later processor, 0 if prior to Nehalem

        CPUIDinfo Info;
        int rVal = 0;
        // The code first determines if the processor is an Intel Processor.  If it is, then
        // feature flags bit 19 (SSE 4.1) and 20 (SSE 4.2) in ECX after CPUID call with EAX = 0x1
        // are checked.
        // If both bits are 1 (indicating both SSE 4.1 and SSE 4.2 exist) then
        // the function returns 1
        const int CHECKBITS = SSE4_1_FLAG | SSE4_2_FLAG;

        if (isGenuineIntel() >= 1)
        {
                // execute CPUID with eax (leaf) = 1 to get feature bits,
                // subleaf doesn't matter so set it to zero
                get_cpuid_info(&Info, 0x1, 0x0);
                if ((Info.ECX & CHECKBITS) == CHECKBITS)
                {
                        rVal = 1;
                }
        }
        return(rVal);
}

int isGenuineIntel (void)
{
        // returns largest function # supported by CPUID if it is a Geniune Intel processor AND
it supports
        // the CPUID instruction, 0 if not
        CPUIDinfo Info;
        int rVal = 0;
        char procString[] = "GenuineIntel";
        if (isCPUIDsupported())
        {
                // execute CPUID with eax = 0, subleaf doesn't matter so set it to zero
                get_cpuid_info(&Info, 0x0, 0x0);
                if ((Info.EBX == ((int *)procString)[0]) && \
                        (Info.EDX == ((int *)procString)[1]) && (Info.ECX == ((int
                *)procString)[2]))
                {
                        rVal = Info.EAX;
                }
        }
        return(rVal);
}

#if (defined(__x86_64__) || defined(_M_X64))
// This code is assembly for 64 bit target OS.
// Assembly code must be compiled with the –use-msasm switch for Linux targets with the
// Intel compiler.
int isCPUIDsupported (void)
{
        // returns 1 if CPUID instruction supported on this processor, zero otherwise
        // This isn't necessary on 64 bit processors because all 64 bit processor support CPUID
        return((int) 1);
}

void get_cpuid_info (CPUIDinfo *Info, const unsigned int leaf, const unsigned int subleaf)
{
        // Stores CPUID return Info in the CPUIDinfo structure.
        // leaf and subleaf used as parameters to the CPUID instruction
        // parameters and register usage designed to be safe for both Windows and Linux
        // Use the Intel compiler option -use-msasm when the target is Linux
        __asm
        {
                mov r10d, subleaf       ; arg2, subleaf (in R8 on WIN, in RDX on Linux)
                mov r8, Info            ; arg0, array addr (in RCX on WIN, in RDI on Linux)
                mov r9d, leaf           ; arg1, leaf (in RDX on WIN, in RSI on Linux)
                push rax
                push rbx
                push rcx
                push rdx
                mov eax, r9d
                mov ecx, r10d
                cpuid
                mov     DWORD PTR [r8], eax
                mov     DWORD PTR [r8+4], ebx
                mov     DWORD PTR [r8+8], ecx
                mov     DWORD PTR [r8+12], edx
                pop rdx
                pop rcx
                pop rbx
                pop rax
        }
}

#else   // 32 bit
//Note need to make sure -use-msasm switch is used with Intel compiler for Linux to get the
// ASM code to compile for both windows and linux with one version source

int isCPUIDsupported (void)
{
        // returns 1 if CPUID instruction supported on this processor, zero otherwise
        // This isn't necessary on 64 bit processors because all 64 bit Intel processors support
CPUID
        __asm
        {
                push ecx ; save ecx
                pushfd ; push original EFLAGS
                pop eax ; get original EFLAGS
                mov ecx, eax ; save original EFLAGS
                xor eax, 200000h ; flip bit 21 in EFLAGS
                push eax ; save new EFLAGS value on stack
                popfd ; replace current EFLAGS value
                pushfd ; get new EFLAGS
                pop eax ; store new EFLAGS in EAX
                xor eax, ecx ; Bit 21 of flags at 200000h will be 1 if CPUID exists
                shr eax, 21     ; Shift bit 21 bit 0 and return it
                push ecx
                popfd ; restore bit 21 in EFLAGS first
                pop ecx ; restore ecx
        }
}

//Note need to make sure -use-msasm switch is used with Intel compiler for Linux to get the
// ASM code to compile for both windows and linux with one version source
void get_cpuid_info (CPUIDinfo *Info, const unsigned int leaf, const unsigned int subleaf)
{
        // Stores CPUID return Info in the CPUIDinfo structure.
        // leaf and subleaf used as parameters to the CPUID instruction
        // parameters and registure usage designed to be safe for both Win and Linux
        // when using -use-msasm
        __asm
        {
                mov     edx, Info   ; addr of start of output array
                mov     eax, leaf  ; leaf
                mov     ecx, subleaf  ; subleaf
                push edi
                push ebx
                mov  edi, edx                      ; edi has output addr
                cpuid
                mov     DWORD PTR [edi], eax
                mov     DWORD PTR [edi+4], ebx
                mov     DWORD PTR [edi+8], ecx
                mov     DWORD PTR [edi+12], edx
                pop ebx
                pop edi
                ret
        }
}
#endif
/************ End of source file sse41andsse42detection.cpp *******************************/

/************ Beginning of source file sse41andsse42detection.h ***************************/
/*              Copyright 2008 Intel Corporation
 *      The source code contained or described herein and all documents related
 *      to the source code ("Material") are owned by Intel Corporation or
 *      its suppliers or licensors. Use of this material must comply with the
 *      rights and restrictions set forth in the accompnied license terms set
 *      forth in file "license.rtf".
 *
 *      Original code contained in cputopology.h.
 *      This file has been renamed to cpuid.h for this app note, code removed, and some
 *      code added.
 *
 *      This is the header file that contain type definitions
 *      and prototypes of functions in the file cpuid.cpp
 *      The source files can be compiled under 32-bit and 64-bit Windows and Linux.
 *
 *      Original code written by Patrick Fay and Shihjong Kuo
 *      Modified by Garrett Drysdale for this application note.
 */

typedef struct
{
        unsigned __int32 EAX,EBX,ECX,EDX;
} CPUIDinfo;

void get_cpuid_info (CPUIDinfo *, const unsigned int, const unsigned int);
int isCPUIDsupported (void);
int isGenuineIntel (void);
int isSSE41andSSE42Supported (void);
```

```
/************ End of source file sse41andsse42detection.h ****************************/
```

References

i App Note 485, “Intel® Processor Identification and the CPUID Instruction” can be found at http://www.intel.com/Assets/PDF/appnote/241618.pdf.

ii Intel article titled ”Intel® 64 Architecture Processor Topology Enumeration” can be found at http://software.intel.com/en-us/articles/intel-64-architecture-processor-topology-enumeration/.

License Agreement

Intel® Source Code License Agreement

This license governs use of the accompanying software. By installing or copying all or any part of the software components in this package, you (“you” or “Licensee”) agree to the terms of this agreement. Do not install or copy the software until you have carefully read and agreed to the following terms and conditions. If you do not agree to the terms of this agreement, promptly return the software to Intel Corporation (“Intel”).

1. **Definitions:**

   A. “Materials” are defined as the software (including the Redistributables and Source as defined herein), documentation, and other materials, including any updates and upgrade thereto, that are provided to you under this Agreement.

   B. ”Redistributables” are the binary files listed in the ”redist.txt” file that is included in the Materials or are otherwise clearly identified as redistributable files by Intel.

   C. “Source” is the source code file(s) that: (i) demonstrate(s) certain functions for particular purposes; (ii) are identified as source code; and (iii) are provided hereunder in source code form.

   D. “Intel’s Licensed Patent Claims” means those claims of Intel’s patents that (a) are infringed by the Source or Redistributables, alone and not in combination, in their unmodified form, as furnished by Intel to Licensee and (b) Intel has the right to license.

2. **License Grant:** Subject to all of the terms and conditions of this Agreement:

   A. Intel grants to you a non-exclusive, non-assignable, copyright license to use the Material for your internal development purposes only.

   B. Intel grants to you a non-exclusive, non-assignable copyright license to reproduce the Source, prepare derivative works of the Source and distribute the Source or any derivative works thereof that you create, as part of the product or application you develop using the Materials.

C. Intel grants to you a non-exclusive, non-assignable copyright license to distribute the Redistributables in binary form, or any portions thereof, as part of the product or application you develop using the Materials.

D. Intel grants Licensee a non-transferable, non-exclusive, worldwide, non-sublicenseable license under Intel's Licensed Patent Claims to make, use, sell, and import the Source and the Redistributables.

3. **Conditions and Limitations:**

A. This license does not grant you any rights to use Intel's name, logo or trademarks.

B. Title to the Materials and all copies thereof remain with Intel. The Materials are copyrighted and are protected by United States copyright laws. You will not remove any copyright notice from the Materials. You agree to prevent any unauthorized copying of the Materials. Except as expressly provided herein, Intel does not grant any express or implied right to you under Intel patents, copyrights, trademarks, or trade secret information.

C. You may NOT: (i) use or copy the Materials except as provided in this Agreement; (ii) rent or lease the Materials to any third party; (iii) assign this Agreement or transfer the Materials without the express written consent of Intel; (iv) modify, adapt, or translate the Materials in whole or in part except as provided in this Agreement; (v) reverse engineer, decompile, or disassemble the Materials not provided to you in source code form; or (vii) distribute, sublicense or transfer the source code form of any components of the Materials and derivatives thereof to any third party except as provided in this Agreement.

4. **No Warranty:**

THE MATERIALS ARE PROVIDED "AS IS". INTEL DISCLAIMS ALL EXPRESS OR IMPLIED WARRANTIES WITH RESPECT TO THEM, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR ANY PARTICULAR PURPOSE.

5. LIMITATION OF LIABILITY: NEITHER INTEL NOR ITS SUPPLIERS SHALL BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF INTEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

6. USER SUBMISSIONS: You agree that any material, information or other communication, including all data, images, sounds, text, and other things embodied therein, you transmit or post to an Intel website or provide to Intel under this Agreement will be considered non-confidential ("Communications"). Intel will have no confidentiality obligations with respect to the Communications. You agree that Intel and its designees will be free to copy, modify, create derivative works, publicly display, disclose, distribute, license and sublicense through multiple tiers of distribution and licensees, incorporate and otherwise use the Communications, including derivative works thereto, for any and all commercial or non-commercial purposes

7. **TERMINATION OF THIS LICENSE**: This Agreement becomes effective on the date you accept this Agreement and will continue until terminated as provided for in this Agreement. Intel may terminate this license at any time if you are in breach of any of its terms and conditions. Upon termination, you will immediately return to Intel or destroy the Materials and all copies thereof.

8. **U.S. GOVERNMENT RESTRICTED RIGHTS**: The Materials are provided with "RESTRICTED RIGHTS". Use, duplication or disclosure by the Government is subject to restrictions set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor. Use of the Materials by the Government constitutes acknowledgment of Intel's rights in them.

9. **APPLICABLE LAWS**: Any claim arising under or relating to this Agreement shall be governed by the internal substantive laws of the State of Delaware, without regard to principles of conflict of laws. You may not export the Materials in violation of applicable export laws.

[ii] HPatrick.j.fay@intel.com
HShihjong.kuo@intel.com
HGarrett.t.drysdale@intel.com
Ronen.zohar@intel.com