プロセッサおよびオペレーティング・システムのストリーミング SIMD 拡張命令（Streaming SIMD Extensions)のサポートの判定

バージョン 1.1

1999 年 1 月

注文番号: 244413J-002

【輸出規制に関する告知と注意事項】

本資料に掲載されている製品のうち、外国為替および外国為替管理法に定める戦略物資等または役務に該当するものについては、輸出または再輸出する場合、同法に基づく日本政府の輸出許可が必要です。また、米国産品である当社製品は日本からの輸出または再輸出に際し、原則として米国政府の事前許可が必要です。

【資料内容に関する注意事項】

● 本ドキュメントの内容を予告なしに変更することがあります。

●インテルでは、この資料に掲載された内容について、市販製品に使用した場合の保証あるいは特別な目的に合うことの保証等は、いかなる場合についてもいたしかねます。また、このドキュメント内の誤りについても責任を負いかねる場合があります。

●インテルでは、インテル製品の内部回路以外の使用にて責任を負いません。また、外部回路の特許についても関知いたしません。

●本書の情報はインテル製品を使用できるようにする目的でのみ記載されています。

インテルは、製品について「取引条件」で提示されている場合を除き、インテル製品の販売や使用に関して、いかなる特許または著作権の侵害をも含み、あらゆる責任を負わないものとします。

● いかなる形および方法によっても、インテルの文書による許可なく、この資料の一部またはすべてを複写することは禁じられています。

本資料の内容についてのお問い合わせは、下記までご連絡下さい。

インテル株式会社　資料販売センター

〒305-8603　筑波学園郵便局　私書箱 115 号

Fax: 0120-478832

「予約」もしくは「未定義」と記された機能や命令は、実際には存在しないものであるか、存在する場合でもその説明は暫定的なものになります。これらの定義は将来変更される可能性があり、また、将来変更されたことにより生じる矛盾や互換性の欠如などに対してはインテルはいっさい責任を負いません。

Pentium® II と Pentium III プロセッサにはエラッタと呼ばれる設計上の欠陥やエラーが含まれることがあり、公表された仕様どおりの製品となっていないことがあります。現時点で判明しているエラッタについては、ご要望があればいつでも入手することが可能です。

*サードパーティのブランドおよび商品名の所用権は、それぞれ各社に帰属します。

目次

改訂履歴

改訂	改訂履歴	日付
1.1	リリース用の更新	1999年1月

参考資料

このアプリケーション・ノートでは次の資料を参考にした。この資料には、本書で取り上げた事項を理解するための有用な情報や背景情報が含まれている。

1. 『*インテル・プロセッサの識別と CPUID 命令*』インテル・アプリケーション・ノート AP-485、注文番号 241618J-012

1 はじめに

ストリーミング SIMD 拡張命令を利用するアプリケーションを開発する場合、アプリケーション開発者は、アプリケーションの開発に使用するシステムと、アプリケーション製品が実行されるシステムが、ストリーミング SIMD 拡張命令をサポートしているかどうか確認しなければならない。アプリケーションは、ストリーミング SIMD 拡張命令のサポートの有無についてプロセッサとオペレーティング・システムに問い合わせた後、必要に応じて、そのシステム用の適切なアルゴリズムまたは DLL を起動できる。この処理は、次の 4 段階で行われる。

1. プロセッサが CPUID 命令をサポートするインテル・プロセッサであることを確認する。
2. プロセッサがストリーミング SIMD 拡張命令をサポートしていることを確認する。
3. オペレーティング・システムが SIMD 浮動小数点ステート管理をサポートしているかどうか確認する。
4. オプションにより、オペレーティング・システムがマスクなしの例外をサポートしているかどうか判定する(アプリケーションがこのようなサポートを必要とする場合)。

本書では、上記の各手順について説明する。また、ここには、アプリケーション内でこの判定を実行する方法を示す C++の関数の例も記載されている。付録では、このコードを使用した簡単なプログラム例について説明する。

2 プロセッサによるストリーミングSIMD拡張命令のサポートの有無の判定

ストリーミング SIMD 拡張命令をサポートしていないインテル・プロセッサ上で、アプリケーションがストリーミング SIMD 拡張命令を実行しようとすると、無効オペコード例外が発生する。これを避けるために、アプリケーションは、インテル・アーキテクチャ機能フラグを確認して、プロセッサがストリーミング SIMD 拡張命令をサポートしているかどうか判定できる。アプリケーションは、CPUID 命令の結果を検討して(この命令は、EDX レジスタに機能フラグを返す)、どのバージョンのコードがシステムのプロセッサに最もよく適合するかを決定できる。

機能フラグのビット 25 が 1 にセットされていれば、そのプロセッサはストリーミング SIMD 拡張命令をサポートしている。次のコードは、機能フラグを EDX にロードし、その結果をテストしてサポートの有無を判定する。『インテル・プロセッサの識別と CPUID 命令』(インテル・アプリケーション・ノート AP-485, 資料番号 241618J-012)の説明に従って、CPUID 命令をサポートするインテル・プロセッサかどうかを、最初にアプリケーションに確認させることをお勧めする。

```
Mov    eax, 1          ; request the processor feature flags from CPUID
CPUID                  ; CPUID loads the feature flags into edx
Test   edx, 02000000h  ; test bit 25 for Streaming SIMD Extensions existence
                       ; if 1, Streaming SIMD Extensions is supported
```

次のコードは、インライン・アセンブリ内でこのアセンブリ・シーケンスを C++関数の一部として使用する。このコードは、最初に、構造化例外処理(try/except 節)を使用して、CPUID 命令が利用可能であることと、コードがインテル・プロセッサ上で実行されていることを確認する。プロセッサが CPUID 命令をサポートしていない場合は、無効コード例外(Windows* 95, 98, および NT 上では STATUS_ILLEGAL_INSTRUCTION 例外コード)が発生し、except 節で処理される。CPUID が利用可能であれば、このコードは、CPUID のメーカ識別文字列をチェックして"GenuineIntel(インテル純正)"プロセッサであるかどうかを確認した後、CPUID 命令を実行してプロセッサの機能に関する情報を取得する。プロセッサの機能の判定についての詳細は、アプリケーション・ノート AP-485 を参照されたい。

```
bool StreamingSIMDExtensionsHWSupport() {
   bool HWSupport = false;
   char brand[12];
   unsigned *str = (unsigned *) brand;

   // Does the processor have CPUID, and is it "GenuineIntel"?
   __try {
      _asm{
         mov     eax, 0      //First, check to make sure this is an Intel processor
         cpuid               //by getting the processor information string with CPUID
         mov     str, ebx    //  ebx contains "Genu"
         mov     str+4, edx  //  edx contains "ineI"
         mov     str+8, ecx  //  ecx contains "ntel"  -- "GenuineIntel"
      }
   }
   __except(EXCEPTION_EXECUTE_HANDLER) {
      if (_exception_code() == STATUS_ILLEGAL_INSTRUCTION) {
          cout << endl << "****CPUID is not enabled****" << endl;
          return (false);
      }
      return (false); // If we get here, an unexpected exception occurred.
   }
   // Now make sure the processor is "GenuineIntel".
   if (!strncmp(brand, "GenuineIntel", 12)) {
      cout << endl << "****This is not an Intel processor!****" << endl;
      return (false);
   }
   // And finally, check the CPUID for Streaming SIMD Extensions support.
   _asm{
      mov     eax, 1         // Put a "1" in eax to tell CPUID to get the feature bits
      cpuid                  // Perform CPUID (puts processor feature info into EDX)
      test    edx, 02000000h // Test bit 25, for Streaming SIMD Extensions existence.
      Jz       NotFound      // If not set, jump over the next instruction (No Streaming SIMD
      Mov     [HWSupport],1  // Extensions).  Set return value to 1 to indicate,
                             // that the processor does support Streaming SIMD Extensions.
   NotFound:
   }
   return (HWSupport);
}
```

コード例 1: CPUID **を使用したストリーミング** SIMD **拡張命令のサポートの判定**

3 オペレーティング・システムによるストリーミングSIMD拡張命令のサポートの有無の判定

ストリーミング SIMD 拡張命令は、新しいプロセッサ・ステート(厳密には、ストリーミング SIMD 拡張命令浮動小数点レジスタ、制御レジスタ、およびステータス・レジスタ)を導入するため、システムのオペレーティング・システムは、さまざまな段階でこのステートの保存と復元を管理できなければならない。また、オペレーティング・システムは、必要に応じて、マスクなしの SIMD 浮動小数点例外の例外処理をサポートしていなければならない。ここでは、使用中のオペレーティング・システムがこれらの機能をサポートしているかどうか判定する方法を説明する。

3.1 FXSAVEおよびFXRSTORのサポート

Deschutes ベースの Pentium® II プロセッサ(および Pentium® III プロセッサ)には、FXSAVE と FXRSTOR の 2 つのステート管理命令が追加された。これらの命令は、浮動小数点ステート、MMX®テクノロジ・ステート、および SIMD 浮動小数点ステートをメモリに保存する。オペレーティング・システムは、これらの命令を使用して、アプリケーションのステートが正確に保存され、復元されるようにする。オペレーティング・システムの SIMD 浮動小数点ステート管理が正しくイネーブルになっている場合は、制御レジスタ 4(CR4)のビット 9(OSFXSR ビット)が 1 にセットされる。このビットは、次の条件が満たされる場合に、オペレーティング・システムによってセットされる。

- プロセッサが FXSAVE/FXRSTOR 命令をサポートしている(CPUID.FXSR ビットがセットされている)。
- オペレーティング・システムが、FXSAVE/FXRSTOR 命令を使用した SIMD 浮動小数点ステート管理をサポートしている。

次の 2 つの条件が満たされる場合、アプリケーションは、拡張されたレジスタ・セット(新しいステート)を使用するストリーミング SIMD 拡張命令を正常に実行できる。

- 浮動小数点エミュレーションがディスエーブルになっている(CR0.EM = 0)。この条件は、ストリーミング SIMD 拡張命令にも MMX®テクノロジにも必要である。
- オペレーティング・システムが、FXSAVE/FXRSTOR を使用してステートを管理する(CR4.OSFXSR = 1)。

オペレーティング・システムがこの条件を満たすかどうか確認する場合、最も簡単な方法は、次の例に示すように、単にストリーミング SIMD 拡張命令を使用した命令を実行し、C++の構造化例外処理を使用して無効オペコード割り込みを捕捉することである。このような間接的な判定方法が必要になるのは、制御レジスタに対するアクセスはオペレーティング・システムの特権 ring0 にのみ許されるため、ユーザ・アプリケーションは CR0.EM ビットと CR4.OSFXSR ビットを直接確認できないからである。次の例は、C++の構造化例外処理を使用して、オペレーティング・システムでの SIMD 命令のサポートを確認する方法を示している。

```
bool StreamingSIMDExtensionOSSupport()
{
   __try
   {
      _asm    xorps xmm0, xmm0   //Execute an instruction using Streaming
                                 //SIMD Extensions to see if support exists.
   }
   //
   // Catch any exception.  If an Invalid Opcode exception (ILLEGAL_INSTRUCTION)
   // occurs, and you have already checked the XMM bit in CPUID, Streaming SIMD
   // Extensions are not supported by the OS.
   //
   __except(EXCEPTION_EXECUTE_HANDLER) {
     if (_exception_code() == STATUS_ILLEGAL_INSTRUCTION) {
        return (false);
     }
     // If we get here, an unknown exception occurred; investigation needed.
     return (false);
   }
   return (true);
}
```

コード例 2:　OS **によるストリーミング** SIMD **拡張命令のサポート**

アプリケーションは、オペレーティング・システムがストリーミング SIMD **拡張命令をサポートしているかどうか確認する前に、コード例** 1 **で説明した方法で、プロセッサがストリーミング** SIMD **拡張命令をサポートしていることを確認する必要がある。ストリーミング** SIMD **拡張命令をサポートしていないプロセッサ上で** SIMD **命令を実行した場合にも、無効オペコード例外が発生する。上の例では、プロセッサによるサポートは確認済みと見なしている。したがって、すべての無効オペコード例外は、**OS **が浮動小数点ステート管理をサポートしていないことが原因で発生したものと見なされる。**

3.2　オペレーティング・システムによるマスクなしの例外のサポート

ここで説明する、オペレーティング・システムがストリーミング SIMD **拡張命令をサポートしているかどうかの最後のチェックは、オプションである。アプリケーションがマスクなしの例外を処理しなければならない場合にのみ、このチェックの実行をお勧めする。例外マスクの解除は、通常は、製品版でないコード内でのみ、あるいはアプリケーションのデバッグ時に例外状態の発生時点を確認する場合にのみ行われる。例外が発生したとき、その例外がマスクされていれば、プロセッサは、**IEEE **規格に基づいて、計算のための「妥当な」値を自動的に生成する。**

プロセッサがストリーミング SIMD **拡張命令アプリケーションのマスクなしの例外状態を検出した場合は、例外を発生させた命令の終了直後に、ソフトウェア例外ハンドラが起動される。ストリーミング** SIMD **拡張命令のマスクなしの例外がイネーブルになっていない場合は(制御レジスタ** 4, **ビット** 10 - OSXMMEXCPT)、**マスクなしの例外が発生すると、無効オペコード・フォルトが発生する。しかし、オペレーティング・システムがストリーミング** SIMD **拡張命令のマスクなしの例外処理をサポートしている場合は(**OSXMMEXCEPT **ビットが** 1)、**ストリーミング** SIMD **拡張命令例外が発生する。この場合は、オペレーティング・システムはフォルトの原因究明に役立つ情報を提供できる。(ストリーミング** SIMD **拡張命令例外の実際の例外コード名は、** STATUS_FLOAT_MULTIPLE_TRAPS **および** STATUS_FLOAT_MULTIPLE_FAULTS **である。それとともに、**Windows® NT 5.0 **のベータ・バージョン** 2 **およびそれ以降では、**Windows **の** C++**構造化例外処理用の**_exception_code()**ルーチンが整数値**

を生成する。本書の作成の時点では、Windows® NT 5.0 ベータ・バージョン 2 およびそれ以降だけが、この機能をサポートしている。)

オペレーティング・システムがストリーミング SIMD 拡張命令の例外処理をサポートしているかどうかは、例 2 のコードで説明したのと同じ方法で判定できる。つまり、例外を発生させるコード・シーケンスを実行し、例外を捕捉すればよい。ただし、この場合は、プロセッサとオペレーティング・システムがデフォルトにより数値例外をマスクするため、例外を発生させる前に、その例外のマスクを解除する必要がある。

次の例では、ゼロを分母とするストリーミング SIMD 拡張命令のパックド除算を実行した結果、ゼロによる除算の例外が発生する。オペレーティング・システムでマスクなしの例外がイネーブルになっていない場合は、無効オペコード例外が発生する。オペレーティング・システムでマスクなしの例外がイネーブルになっている場合は、ストリーミング SIMD 拡張命令例外が発生する。この場合は、Windows®がサポートしている`_exception_info()`ルーチンを `except` 節で使用すれば、状況を検討して適切な処置をとるための情報が得られる。ただし、本書にはこのルーチンの使用方法は記載していない。構造化例外処理についての詳細は、Microsoft Developer's Studio のヘルプ情報を参照されたい。

```
bool StreamingSIMDExtensionOSException()
{
   bool StreamingSIMDExtensionOSExcept = true;
   unsigned long csr;
   float x[4] = {1.0f, 2.0f, 3.0f, 4.0f};

    __try
    {
       _asm{
           stmxcsr [csr]      //Get the MXCSR (Streaming SIMD Extensions control register)
           and     [csr], 0FFFFFDFFh // Set the divide-by-zero mask bit (bit 9) to 0 to
           ldmxcsr [csr]      // unmask this exception, then reload the MXCSR.

           Xorps   xmm0, xmm0 //Fill the denominator register with 0's
           movups  xmm1,[x]   //Do a divide by zero using Streaming SIMD Extensions
           divps   xmm1, xmm0 //  packed divide.  The interrupt handler should be invoked.
        }
    }
    //
    // Catch any exception that occurs. The exception handler here needs to take different
    // actions depending on the OS.  If you have determined that your processor and OS
    // indeed support Streaming SIMD Extensions, an ILLEGAL_INSTRUCTION exception
    // indicates that unmasked SIMD floating-point exceptions are not supported.
    //
    __except(EXCEPTION_EXECUTE_HANDLER) {
       unsigned long code = _exception_code();
       if (code == STATUS_ILLEGAL_INSTRUCTION) {
           cout << endl << "****OS did not handle the exception!****" << endl;
           StreamingSIMDExtensionOSExcept = false;
       }
       else {
          cout << endl << "****OS handled the exception.****" << endl;
          StreamingSIMDExtensionOSExcept = true;
          // But you would need to have special handling code here to decipher what
          // exception really occurred and why!
       }
    }
    return (StreamingSIMDExtensionOSExcept);
}
```

コード例 3:　**マスクなしの** SIMD **浮動小数点例外のサポートの有無の判定**

4 まとめ

アプリケーションを実行するシステム上で、すべてのストリーミング SIMD 拡張命令がサポートされているかどうかを確認するには、プロセッサとオペレーティング・システムの両方の機能を確認する必要がある。この処理は、次の 4 段階で行われる。

1) プロセッサが CPUID 命令をサポートするインテル・プロセッサであることを確認する。
2) CPUID 機能フラグをチェックして、プロセッサがストリーミング SIMD 拡張命令をサポートしていることを確認する。
3) オペレーティング・システムが SIMD 浮動小数点ステート管理をサポートしているかどうか判定する。
4) オペレーティング・システムがマスクなしの SIMD 浮動小数点例外をサポートしているかどうか判定する(アプリケーションがこのようなサポートを必要とする場合)。

上の条件がすべて満たされていれば、そのシステム上ですべてのストリーミング SIMD 拡張命令がイネーブルになっている。

付録: CpuOSIdプログラム

VTune 4.0 CD-ROM に入っている Samples/CpuOSId/CpuOSId.cpp のプログラム例は、上記の 4 つの条件(純正インテル・プロセッサである、プロセッサがストリーミング SIMD 拡張命令をサポートしている、オペレーティング・システムが SIMD 浮動小数点ステート管理をサポートしている、オペレーティング・システムがストリーミング SIMD 拡張命令のマスクなしの例外をサポートしている)を確認するプログラムである。

このプログラム例は、次のシステム上でテスト済みである。

- Pentium® Pro および Pentium II プロセッサと Windows® NT 4.0 Service Pack 3
- Pentium®プロセッサと Windows 95
- ストリーミング SIMD 拡張命令をサポートするプロセッサと Windows 98
- ストリーミング SIMD 拡張命令をサポートするプロセッサと Windows NT 5.0 ベータ・バージョン 2.0
- ストリーミング SIMD 拡張命令をサポートするプロセッサと Windows NT 4.0 Service Pack 4
- ストリーミング SIMD 拡張命令をサポートするプロセッサと Windows 95

このプログラムは、本書で説明した関数の例を使用して、次の 3 つの質問のそれぞれに"YES!!!"または"No."で答える。

```
"Does your processor support Streaming SIMD Extensions? "
(プロセッサはストリーミングSIMD拡張命令をサポートしているか?)

"Does the OS support save/restore for SIMD floating-point state? "
(OSはSIMD浮動小数点ステートの保存/復元をサポートしているか?)

"Does the OS support SIMD floating-point exceptions? "
(OSはSIMD浮動小数点例外をサポートしているか?)
```

Windows NT 5.0 のベータ・バージョン 2 およびそれ以上を搭載し、ストリーミング SIMD 拡張命令をサポートするシステムだけが、上の 3 つの質問すべてに"YES!!!"と答える。ストリーミング SIMD 拡張命令をサポートする Windows 98 搭載のシステムは、マスクなしの例外をサポートしない。