

# **Workload Analysis**

**(Stage 3)**

# Compiler Directive (Pragma)

- 嘗試使用compiler directive (pragma)，協助compiler做更好的決定、獲得更好的優化效果。
- <https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/pragmas.html>

Intel® C++ Compiler Classic Developer Guide and Reference

## Development Reference Guides

Version: 2021.7      Last Updated: 09/08/2022      Public Content      [Download as PDF](#)

- Intel® C++ Compiler Classic Developer Guide and Reference
  - Introducing the Intel® C++ Compiler Classic
  - Compiler Setup
  - Compiler Reference
    - C/C++ Calling Conventions
    - Compiler Options
    - Floating-Point Operations
    - Attributes
    - Intrinsics

### Pragmas

Pragmas are directives that provide instructions to the compiler for use in specific cases. For example, you can use the `novector` pragma to specify that a loop should never be vectorized. The keyword `#pragma` is standard in the C++ language, but individual pragmas are machine-specific or operating system-specific, and vary by compiler.

Some pragmas provide the same functionality as compiler options. Pragmas override behavior specified by compiler options.

Some pragmas are available for both Intel® and non-Intel microprocessors but they may perform additional optimizations for Intel® microprocessors than they perform for non-Intel microprocessors. Refer to the individual pragma name for detailed description.

The Intel® C++ Compiler Classic pragmas are categorized as follows:

- [Intel-specific Pragmas](#) - pragmas developed or modified by Intel to work specifically with the Intel® C++ Compiler Classic

#### In This Topic

- Using Pragmas
- Individual Pragma Descriptions

# unroll / nounroll

- Tells the compiler to unroll or not to unroll a counted loop.

Use the *unroll* pragma for innermost loop unrolling:

```
1  void unroll(int a[], int b[], int c[], int d[]) {  
2      #pragma unroll(4)  
3      for (int i = 1; i < 100; i++) {  
4          b[i] = a[i] + 1;  
5          d[i] = c[i] + 1;  
6      }  
7  }
```

# 其他注意事項

- 盡量專注於某一個版本為基礎，做程式碼的分析與修改
- 嘗試使用SIMD的方式加速程式執行的效能
- OpenMP directive (OpenMP writing)
  - **SIMD intrinsic function**
  - **Function inlining**
  - **Loop unrolling**
  - A specialized function instead of the original version.  
(ex: Intel math kernel library, MKL)
  - Profile-guided optimization
  - Thread binding to CPU core
  - Others ...

# SIMD Intrinsic Functions

## Intel® Intrinsics Guide

Updated  
12/06/2021

Version  
3.6.1

### Instruction Set

☐ MMX☐ SSE☐ SSE2☐ SSE3☐ SSSE3☐ SSE4.1☐ SSE4.2☐ AVX☐ AVX2☐ FMA☐ AVX\_VNNI☐ AVX-512☐ KNC☐ AMX

The Intel® Intrinsics Guide contains reference information for Intel intrinsics, which provide access to Intel instructions such as Intel® Streaming SIMD Extensions (Intel® SSE), Intel® Advanced Vector Extensions (Intel® AVX), and Intel® Advanced Vector Extensions 2 (Intel® AVX2).

- For information about how Intel compilers handle intrinsics, view the [Intel® C++ Compiler Classic Developer Guide and Reference](#).
- For questions about Intel intrinsics, visit the [Intel® C++ Compiler](#) board.

```
void _mm_2intersect_epi32 (__m128i a, __m128i b, __mmask8* k1, __mmask8* k2)  vp2intersectd
void _mm256_2intersect_epi32 (__m256i a, __m256i b, __mmask8* k1, __mmask8* k2)  vp2intersectd
void _mm512_2intersect_epi32 (__m512i a, __m512i b, __mmask16* k1, __mmask16* k2)  vp2intersectd
void _mm_2intersect_epi64 (__m128i a, __m128i b, __mmask8* k1, __mmask8* k2)  vp2intersectq
void _mm256_2intersect_epi64 (__m256i a, __m256i b, __mmask8* k1, __mmask8* k2)  vp2intersectq
```

<https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html>

# 其他注意事項 (2)

- 若報告投影片有程式碼/圖表，請確認投影出來是清楚的，以利說明
- 請思考如何呈現你的數據，讓聽眾可以更清楚的了解你的想法與結果



# Presentation

- 口頭報告
  - workload program簡介
  - 嘗試修改的方法與結果 (結果好或壞不是最重要，最重要是需要知道為什麼好或為什麼不好)
  - 每組報告時間約10~15分鐘
  - Deadline: January 2, 2024 (面授，創新大樓 R322)
- 完整書面報告
  - Deadline: January 5, 2024 (WORD, PDF, Markdown)