I told the Microsoft Visual C++ compiler not to generate AVX instructions, but it did it anyway!

devblogs.microsoft.com/oldnewthing/20201026-00

October 26, 2020

Raymond Chen

A customer passed the `/arch:SSE2` flag to the Microsoft Visual C++ compiler, which means "Enable use of instructions available with SSE2-enabled CPUs." In particular, the customer did *not* pass the `/arch:SSE4` flag,[1] so they did not enable the use of SSE4 instructions.

And then they did this:

```
#include <mmintrin.h>

void something()
{
    __m128i v = _mm_load_si128(&mem);
    ... more SSE2 stuff ...
    v = _mm_insert_epi32(v, alpha, 3);
    ... more SSE2 stuff ...
}
```

The `_mm_insert_epi32()` intrinsic maps to the `PINSRD` instruction, which is an SSE4 instruction, not SSE2.

To the customer's surprise, this code not only compiled, it even ran! The customer wanted to know what is happening. Did the compiler convert the `_mm_insert_epi32()` into an equivalent series of SSE2 instructions?

No, the compiler didn't do that. You explicitly requested an SSE4 instruction, so the compiler honored your request. The `/arch:SSE2` flag tells the compiler not to use any instructions beyond SSE2 in its own code generation, say during autovectorization or optimized `memcpy`. But if you invoke it explicitly, then you get what you wrote.

I guess the option could be more accurately (and verbosely) named "Enable *automatic* use of instructions available with SSE2-enabled CPUs." Because what this controls is whether the compiler will use those instructions of its own volition.

The customer happened to test their program on a CPU that supported SSE4, so the instruction worked. If they had run it on a a CPU that supported SSE2 but not SSE4, it would have crashed.

The reason SSE4 intrinsics are still allowed even in SSE2 mode is that you might have identified some performance-sensitive operations and written two versions of the code, one that uses SSE2 intrinsics, and another that uses SSE4 intrinsics, choosing between the two at runtime based on a processor capability check.

The compiler won't generate any SSE4 instructions on its own, so your code is safe on SSE2 systems. When you detect an SSE4 system, you can explicitly call the SSE4 code paths.

1 As commenter Danielix Klimax noted, there is no actual `/arch:SSE4` option. Please interpret the remark in the spirit it was intended. ("The custom did not pass any flags that would enable SSE4 instructions.")

Raymond Chen

Follow