6.1 Using intrinsics

Compiler intrinsics are special functions whose implementations are known to the compiler. They enable you to easily incorporate domain-specific operations in C and C++ source code without resorting to complex implementations in assembly language.

The C and C++ languages are suited to a wide variety of tasks but they do not provide built-in support for specific areas of application, for example *Digital Signal Processing* (DSP).

Within a given application domain, there is usually a range of domain-specific operations that have to be performed frequently. However, if specific hardware support is available, then these operations can often be implemented more efficiently using the hardware support than in C or C++. A typical example is the saturated add of two 32-bit signed two's complement integers, commonly used in DSP programming. The following example shows a C implementation of a saturated add operation:

```
#include <limits.h>
int L_add(const int a, const int b)
{
    int c;
    c = a + b;
    if (((a ^ b) & INT_MIN) == 0)
    {
        if ((c ^ a) & INT_MIN)
        {
            c = (a < 0) ? INT_MIN : INT_MAX;
        }
    }
    return c;
}
```

Using compiler intrinsics, you can achieve more complete coverage of target architecture instructions than you would from the instruction selection of the compiler.

An intrinsic function has the appearance of a function call in C or C++, but is replaced during compilation by a specific sequence of low-level instructions. The following example shows how to access the `__qadd` saturated add intrinsic:

```
#include <arm_acle.h>    /* Include ACLE intrinsics */

int foo(int a, int b)
{
  return __qadd(a, b);  /* Saturated add of a and b */
}
```

Using compiler intrinsics offers a number of performance benefits:

- The low-level instructions substituted for an intrinsic are either as efficient or more efficient than corresponding implementations in C or C++. This results in both reduced instruction and cycle counts. To implement the intrinsic, the compiler automatically generates the best sequence of instructions for the specified target architecture. For example, the `__qadd` intrinsic maps directly to the A32 assembly language instruction `qadd`:

  ```
  QADD r0, r0, r1    /* Assuming r0 = a, r1 = b on entry */
  ```

- More information is given to the compiler than the underlying C and C++ language is able to convey. This enables the compiler to perform optimizations and to generate instruction sequences that it cannot otherwise perform.

These performance benefits can be significant for real-time processing applications. However, care is required because the use of intrinsics can decrease code portability.

Note

If you need to use SVE and SVE2 intrinsics, see *7.2 Using SVE and SVE2 intrinsics directly in your C code* on page 7-116 for more information.

Related information

Compiler-specific intrinsics